Universität Ulm
Fakultät für Informatik

17. Workshop über Komplexitätstheorie, effiziente Algorithmen und Datenstrukturen am 26. Mai 1992 in Ulm

Programm und Abstracts

Nr. 92-03

Ulmer Informatik-Berichte

Mai 1992

Programm des 17. Workshop über Komplexitätstheorie, effiziente Algorithmen und Datenstrukturen
26. Mai 1992, Universität Ulm

9:45 *Ernst W. Mayr, Ralph Werchner (Frankfurt)*: Optimales Routing von Klammern auf dem Hyperwürfel

10:05 *Volker Heun (Frankfurt)*: Ein Hyperwürfel-Algorithmus zur Einbettung binärer Bäume in den Hyperwürfel

10:15 *Christian Schindelhauer (Darmstadt)*: Hierarchien in Average-Case-Komplexitätsklassen

10:35 *Andreas Brandstädt (Duisburg), Heinz-Jürgen Voss (Dresden)*: Short disjoint cycles in cubic graphs

10:45 Pause

11:00 *Ingo Althöfer, Torsten Grotendiek (Bielefeld)*: Random Walk-Entscheidungsmodelle, und zur Durchschnittskomplexität von "Read-Once Formula"

11:20 *Svetlana Anoulova, Paul Fischer, Stefan Pölt, Hans Ulrich Simon (Dortmund)*: Approximation von Bayesschen Entscheidungen

11:40 *Steffen Lange (Leipzig), Thomas Zeugmann (Darmstadt)*: Charakterisierungen monoton erlernbarer Familien rekursiver Sprachen

12.00 Gründungsversammlung der GI-Fachgruppe "Komplexität"

Mittagspause

13:30 *Albrecht Hoene, Arfst Nickelsen (Berlin)*: Counting, Selecting, and Sorting by Query-Bounded Machines

13:50 *Claudia Leopold (Berlin)*: A fast sort using parallelism within memory

14:10 *Detlef Sieling, Ingo Wegener (Dortmund)*: Erweiterte BDD's — Datenstrukturen für Boolesche Funktionen im Spektrum zwischen BDD's und BP1

14:30 *Franz Höfting, Egon Wanke (Paderborn)*: Minimum cost paths in periodic graphs

14:50 Pause

15:10 *Friedhelm Hinz (Trier)*: Basisklauseln für Lineare Resolution

15:20 *Andreas Goerdt (Paderborn)*: A threshold for unsatisfiability

15:30 *Maria Breuing (Paderborn)*: Dynamische Einbettung von Bäumen in DeBruijn-Netzwerke

15:50 *Jordan Gergov, Christoph Meinel (Trier)*: Branching Programme als Datenstruktur für den Schaltkreisentwurf

16:10 Pause

16:30 *Bernd Schmeltz (Darmstadt)*: Gestörte Schaltkreise sublogarithmischer Tiefe

16:50 *Andreas Jakoby (Darmstadt)*: The Complexity of Scheduling Problems with Communication Delays

17:10 *Maciej Liśkiewicz (Darmstadt, Wrocławski)*: On relationship between time and reversal complexity measure

Optimales Routing von Klammern auf dem Hyperwürfel

Ernst W. Mayr Ralph Werchner
Fachbereich Informatik
J.W. Goethe-Universität, Frankfurt am Main

Wir betrachten Probleme auf Prozessornetzen. Ein Prozessornetz besteht aus einer Menge von Prozessoren mit lokalen Speichern (RAM's) und einigen bidirektionalen Verbindungsleitungen zwischen den Prozessoren. In einem Schritt kann ein Prozessor entweder einen RAM-Berechnungsschritt ausführen, ein Datenelement zu einem direkten Nachbarprozessor senden oder eines empfangen.

Bei der Implementierung paralleler Algorithmen auf Prozessornetzen sind in der Regel zusätzlich routing Probleme zu lösen. Unter einem routing Problem verstehen wir die Aufgabe, einige Datenelemente jeweils von ihrem Prozessor (Startadresse) zu einem anderen Prozessor (Zieladresse) zu schicken. Jeder Prozessor darf dabei nur höchstens einmal als Startadresse und höchstens einmal als Zieladresse auftreten.

Hier soll speziell als Prozessornetz der d-dimensionalen Hyperwürfel betrachtet werden. Die Anzahl der Prozessoren darin ist $p = 2^d$. Es ist bekannt, daß darauf jedes routing in $2 \log p$ Schritten ausgeführt werden kann, wenn man nicht die Zeit zum Berechnen geeigneter Wege für die Datenelemente mitzählt. Aber es ist noch offen, ob es einen Algorithmus gibt, der jedes routing in logarithmischer Zeit ausführen kann. Der beste deterministische routing Algorithmus benötigt $O(\log p(\log\log p)^2)$ Schritte [CP90].

Es gibt aber für eingeschränkte Klassen von routings Algorithmen mit logarithmischer Laufzeit. Dies sind z.B. monotone routings [NS81], bei denen sich die Reihenfolge der Datenelemente nicht ändert, oder die linearen Permutationen [RB91], bei denen sich die Zieladressen durch eine affin lineare Funktion in $GF(2)^d$ aus den Startadressen ergibt. Wir geben einen routing Algorithmus mit logarithmischer Laufzeit für eine neue Klasse von routings an. Dies sind routings, die sich durch die Struktur eines korrektes Klammerwortes ergeben. Nehmen wir an, einige der Prozessoren enthalten eine Klammer, und die Folge dieser Klammern ergibt ein korrektes Klammerwort. Dann definiert dieses Klammerwort ein routing Problem, in dem für jedes Klammerpaar ein Datenelement von der öffnenden Klammer zum Prozessor mit der entsprechenden schließenden Klammen geschickt werden soll. Ein routing, das sich so beschreiben läßt, heiße ein *Klammerrouting*.

Wir geben einen Algorithmus an, der jedes Klammerrouting, das durch ein entsprechendes Klammerwort gegeben ist, in logarithmischer Zeit ausführt. Dazu wird das routing Problem zunächst auf das Problem reduziert, jedes der gegebenen Klammerpaare – samt zugehöriger Datenelemente – zu einem beliebigen gemeinsamen Treffpunkt zu schicken. Und das letztere Problem wird mit einem divide-and-conquer Algorithmus gelöst.

Anwendungen unseres routing Verfahrens ergeben sich für das Wortproblem von Dyck-Sprachen, Transformationen zwischen verschiedenen Darstellungen arithmetischer Ausdrücke und für die Auswertung arithmetischer Ausdrücke.

Literatur

[CP90] R. Cypher and C.G. Plaxton. Deterministic sorting in nearly logarithmic time on the hypercube and related computers. *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, 193–203, 1990.

[NS81] D. Nassimi and S. Sahni. Data broadcasting in SIMD computers. *IEEE Transactions on Computers*, C-30:101–107, 1981.

[RB91] C.S. Raghavendra and R.V. Boppana. On self-routing in Benes and shuffle-exchange networks. *IEEE Transactions on Computers*, C-40:1057–1064, 1991.

Ein Hyperwürfel-Algorithmus zur Einbettung binärer Bäume in den Hyperwürfel

Volker Heun
Fachbereich Informatik (20)
Johann Wolfgang Goethe-Universität
Frankfurt am Main

Da binäre Bäume eine grundlegende Datenstruktur darstellen, ist man daran interessiert diese auch auf dem Hyperwürfel effizient einsetzen zu können. Aus den Arbeiten von [BCLR86] und [MS88] ist bekannt, daß beliebige binäre Bäume der Größe n in den optimalen Hyperwürfel mit konstanter Dilation injektiv eingebettet werden können. Hierbei ist der optimale Hyperwürfel der nächstgrößere, also $\lceil \log(n) \rceil$-dimensionale, Hyperwürfel, wobei n die Anzahl der Knoten im einzubettenden Graphen ist. Ausgehend von der Arbeit von [BCLR86] soll nun gezeigt werden, daß man einen beliebigen binären Baum mit Dilation 9 in den optimalen Hyperwürfel in Zeit $O(\mathrm{T}_{\mathrm{Sort}}(n) \cdot \log(n))$ im Hyperwürfel injektiv einbetten kann. Dabei ist $\mathrm{T}_{\mathrm{Sort}}(n)$ die Zeit, die für das Sortieren von n Elementen im n-dimensionalen Hyperwürfel benötigt wird. Nach dem bisher besten bekannten Sortieralgorithmus im Hyperwürfel von [CP90] gilt: $\mathrm{T}_{\mathrm{Sort}}(n) = O(log(n)(loglog(n))^2)$.

Literatur

[BCLR86] Bhatt, S., Chung, F., Leighton, T., Rosenberg, T.: Optimal Simulations of Tree Machines, *Proceedings of the 27th Annual Symposium on Foundations of Computer Science 1986*, 274–282

[CP90] Cypher, R., Plaxton, G.: Deterministic Sorting in nearly logarithmic time on the hypercube and related computers, *Proceedings of the 21nd Annual Symposium on Theory of Computing*, 193–203.

[MS88] Monien, B., Sudborough, I. H.: Simulating Binary Trees on Hypercubes, *VLSI Algorithms and Architectures, Proccedings of the 3rd Aegean Workshop on Computing 1988, LNCS 319*, 170–180.

Hierarchien in Average-Case-Komplexitätsklassen

Christian Schindelhauer[3]

Technische Hochschule Darmstadt
Fb Informatik Institut für Theoretische Informatik

Bei der Betrachtung des Average-Case-Laufzeitverhalten von Algorithmen treten drei grundsätzliche Probleme auf:

- Legt man den Erwartungswert des Laufzeitverhaltens eines Algorithmus zugrunde, gibt es polynomial zeitbeschränkte Algorithmen und polynomial zeitbeschränkte Simulationen (oder Reduktionen), welche kombiniert, exponentiell viel Zeit beanspruchen.
- Betrachtet man jeweils eine Gewichtung μ_n für jede Eingabelänge n, degeneriert jedes Average-Case-Maß für dünne Gewichtungen zu Worst-Case-Untersuchungen. Dünn heißt, daß für jede Eingabelänge nur sehr wenige gewichtete Eingaben existieren.
- Betrachtet man Gewichtungsfunktionen μ für alle Eingaben, so bestimmen herkömmliche Maße eine zu kleine Zeitschranke $T' \leq o(T)$ für Algorithmen, welche auf Eingabe x grundsätzlich Rechenzeit $T(|x|)$ benötigen.

Die ersten beiden Probleme werden mit der Definition *average on polynomial* von Leonid Levin [1] hinreichend gelöst. In diesem Vortrag wird ein strengeres Average-Case-Maß sAv vorgestellt, welches alle drei Probleme löst. In dieser Definition spielt der Rang ($rank_\mu(x)$) einer Eingabe x bezüglich einer Wahrscheinlichkeitsverteilung μ eine wichtige Rolle. Dieser ist definiert als die Anzahl aller Eingaben, welche eine höhere Wahrscheinlichkeit besitzen.

Mit Hilfe von sAv können sinnvolle Komplexitätsklassen **sAvTime** definiert werden. Ferner wird die Menge **V-rankable** aller Gewichtungen μ benötigt, deren Rangfunktion $\text{rank}_\mu(x)$ in Zeit $V(|x|)$ berechnet werden kann.

Für polynomiales T wird in [2] gezeigt, daß

$$\text{sAvTime}(T,\ T \cdot \text{EXL}\text{-rankable}) = DTime(T).$$

Zusätzlich gelingt es in diesem Maß auch Hierachiesätze zu zeigen, denn für $T_1 \leq o(T_2)$ und für alle V gilt

$$\text{sAvTime}(T_1,\ V\text{-rankable}) \subset \text{sAvTime}(T_2,\ V\text{-rankable}).$$

[1] L. Levin, *Average Case Complete Problems*, SIAM J. of Computing 15, 1986, 285-286.

[2] C. Schindelhauer, *Neue Average Case Komplexitätsklassen*, Diplomarbeit, Technische Hochschule Darmstadt, 1991.

[3] E-mail: schindel@iti.informatik.th-darmstadt.de

Short disjoint cycles in cubic graphs

by Andreas Brandstädt
FB 11 - Mathematik /FG Informatik
Uni-GH Duisburg PF 10 15 03, 4100 Duisburg 1
and Heinz-Jürgen Voss
Pädagogische Hochschule Dresden
PF 365 , 8060 Dresden

In this note we show that cubic graphs G=(V,E) with |V|=n,|E|=m, have $\Omega(n/\log n)$ pairwise disjoint cycles of length $O(\log n)$.
Furthermore such collections of cycles can be determined efficiently within $O(n^2 \cdot m)$ steps using a modification of BFS.
This result can be generalized into several directions:
a) cycles can be replaced by "useful subgraphs" (i.e. subgraphs with more edges than vertices)
b) not only cubic graphs but also graphs with lower degree bound 3 and constant upper degree bound can be considered.
The investigation was motivated by a question of B. Monien concerning the bisection problem of transputer networks.

Random Walk-Entscheidungsmodelle und zur Durchschnittskomplexität von "Read-Once Formula"

Ingo Althöfer und Torsten Grotendiek
Fakultät für Mathematik
Universität Bielefeld
Postfach 8640
W–4800 Bielefeld 1

(i) Gegeben ist die Menge $\{0,1,\ldots,n\}$ und darauf 3 Partikel. Wird ein Partikel, das sich auf Position i befindet, angestoßen, so geht es jeweils mit Wahrscheinlichkeit $\frac{1}{2}$ nach $(i-1)$ und nach $(i+1)$. An den Rändern 0 und n werden die Partikel absorbiert.
Gefragt ist nach der optimalen "Anstoß–Strategie", wenn in jeder elementaren Zeiteinheit genau ein Partikel angestoßen werden darf und man möglichst schnell wissen will, an welchem Rand die Mehrheit der Partikel absorbiert wird.

(ii) Sei $f : \{0,1\}^n \to \{0,1\}$ monoton und $\# f^{-1}(0) = \# f^{-1}(1) = 2^{n-1}$.
Für $x \in \{0,1\}^n$ sei

$$d(x) = \# \left\{ y \in \{0,1\}^n \;\middle|\; \begin{array}{l} f(x) \neq f(y) \text{ und} \\ \text{Hamming–Abstand } (x,y) = 1 \end{array} \right\}$$

und $D = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} d(x)$ der zugehörige Durchschnittswert.

Satz: Dann muß man im Durchschnitt mindestens D^2 viele Komponenten x_i eines zufällig gegebenen $x = (x_1,\ldots,x_n)$ erfragen, um $f(x)$ zu kennen.

Im Vortrag werden auch die Bezüge zwischen (i) und (ii) deutlich gemacht.

Referencen

I. Althöfer, On pathology in game tree and other recursion tree models, Habilitationsschrift, Bielefeld 1991.
T. Grotendiek, Random Walk–Entscheidungsmodelle, Diplomarbeit, in Vorbereitung.

Approximation von Bayesschen Entscheidungen.

Svetlana Anoulova, Paul Fischer, Stefan Pölt, Hans Ulrich Simon
Lehrstuhl Informatik II, Universität Dortmund
Postfach 500 500, D-4600 Dortmund 50

Wir untersuchen Bayessche Klassifikationsprobleme aus der Sicht der Valiantschen Lerntheorie. Dabei ist das Klassifikationsproblem durch Boolesche Merkmalsvektoren gegeben. Wir erhalten Information in Form von *Beispielen*, d.h. in Form von klassifizierten Merkmalsvektoren. Unser Ziel ist es, aus diesen Beispielen effizient und mit hoher Zuverlässigkeit ein fast optimales Klassifikationsschema zu lernen. Ein klassischer Ansatz in der Mustererkennung benutzt dazu empirische Schätzungen der Bayesschen Diskriminantenfunktion. Wir analysieren diesen Ansatz für zwei Klassen von Verteilungen, die in der Mustererkennung eine Rolle spielen: die Bahadur-Lazarsfeld-Verteilungen k-ter Ordnung und die Chow-Expansionen k-ter Ordnung. In beiden Fällen können wir polynomielle obere Schranken für die Stichprobengröße zeigen.

Charakterisierungen monoton erlernbarer Familien rekursiver Sprachen

Steffen Lange
TH Leipzig
FB Mathematik und Informatik
PF 66
O–7030 Leipzig
lange@guuglz.uucp
Thomas Zeugmann
TH Darmstadt
Institut für Theoretische Informatik
Alexanderstr. 10
W–6100 Darmstadt
zeugmann@iti.informatik.th-darmstadt.de

Abstrakt Im Vortrag beschäftigen wir uns mit streng–monotonen, monotonen und schwach–monotonen Algorithmen zum Erlernen rekursiver Sprachen aus positiven sowie aus positiven und negativen Beispielen. Die drei Monotoniebegriffe widerspiegeln unterschiedliche Formalisierungen der natürlichen Forderung an den Lernalgorithmus, im Verlaufe des Lernprozesses nur dann neue Hypothesen zu produzieren, wenn diese besser als die alten sind.

Es werden vier Charakterisierungssätze vorgestellt, die es gestatten, monotone Inferenz rekursiver Sprachen unter einem unifizierenden Blickwinkel zu betrachten. Da schwach-monotone Erkennung mit konservativer Erkennung zusammenfällt gelingt somit die Lösung des in Angluin (1980) offen gebliebenen Problems der Charakterisierung von Lernalgorithmen, die aus positiven Beispielen lernen und dabei niemals eine Hypothese produzieren, die eine Sprache beschreibt, die die zu erlernende Sprache echt umfaßt.Darüberhinaus liefern die angegebenen Charakterisierungssätze vertiefte Einsichten in die Art und Weise, wie Inferenzalgorithmen arbeiten können.

Der 2. teil des Vortrages beschäftigt sich mit monotoner Erkennung durch iterativ arbeitene Lernalgorithmen, die aus der Sicht möglicher Anwendungen von besonderem Interesse sind, da sie den in allen praktischen Berechnungen zu beachtenen Restriktionen bzgl. verfügbaren Speicherplatz Rechnung tragen.

Literatur

Angluin, D., Inductive Inference of Formal Languages from Positive Data, Information and Control 45, 1980, 117 - 135.

Counting, Selecting, and Sorting by Query-Bounded Machines

Albrecht Hoene und *Arfst Nickelsen*[1]

Die Anzahl der Fragen, die eine polynomielle Orakelmaschine benötigt, um gewisse Probleme zu lösen, hat sich in letzter Zeit als Komplexitätsmaß etabliert. Der allgemeine Ansatz aus [AG88,Bei87] stellt die Frage: Gegeben eine Menge A und $k \geq 1$, wieviel Fragen sind nötig um das k-fache *Membership*-Problem zu lösen? Das heißt, wieviele Fragen muß eine polynomielle Maschine an ein beliebiges Orakel stellen, um für einen beliebigen Eingabevektor $(x_1, \ldots, x_k)$ herauszufinden, welche Elemente in der Menge A sind.

Wir untersuchen Zähl-, Selektier- und Sortierfunktionen und ihre Frage-Komplexität. Das Zählproblem für eine Menge A und $k \geq 1$ besteht darin, bei Eingabe $(x_1, \ldots, x_k)$ die Anzahl der Elemente in $\{x_1, \ldots, x_k\} \cap A$ zu bestimmen. Beim Selektierproblem geht es darum, aus gegebenem $(x_1, \ldots, x_k)$ ein Element aus A zu bestimmen, falls eines existiert, und beim Sortierproblem soll die Eingabe $(x_1, \ldots, x_k)$ so sortiert werden, daß für den Ausgabevektor $(y_1, \ldots, y_k)$ gilt: $i < j \ \& \ y_i \in A \Rightarrow y_j \in A$. Während das Zählen von Worten mit einer bestimmten Eigenschaft ein wichtige Rolle in vielen Beweisen spielt, handelt es sich bei den anderen beiden Problemen um Verallgemeinerungen von P-Selektivität. Wir fragen jeweils, wieviele Fragen eine polynomielle Maschine an ein beliebiges Orakel stellen muß, um eine Funktion zu berechnen, die das gegebene Problem löst.

Unsere Ergebnisse liefern optimale Schranken für den allgemeinen Fall der genannten Probleme ($\lceil \log k + 1 \rceil$ für Zählen, $\lceil \log k \rceil$ für Selektieren und $\lceil \log (\lceil \frac{k}{2} \rceil) \rceil$ für Sortieren). Darüber hinaus zeigen wir, daß sich für NP-vollständige Mengen beim Zählen und Selektieren nur dann eine Frage sparen läßt, wenn P=NP gilt. Wir verallgemeinern damit ein Ergebnis aus [AG88]. Außerdem zeigen wir, daß Mengen, bei denen sich eine Frage einsparen läßt, polynomiell große Schaltkreise haben. Daraus folgt, daß solche Mengen in der dritten Stufe der extended low Hierarchie liegen. Wir zeigen, daß dieses Ergebnis nicht auf die zweite Stufe verbessert werden kann, und erhalten als Korollar die Lösung einer Frage, die in [ABG90] offengelassen wurde.

Schließlich vergleichen wir die eingeführten Begriffe mit denen der Cheatability und P-Superterseness. Wir zeigen, daß sich bei Mengen, die cheatable sind, alle oben genannten Probleme unterhalb der angegebenen Schranken lösen lassen. Darüber hinaus zeigen wir, daß Mengen, die in diesem Sinne leichte Zähl- oder Selektierfunktionen haben, nicht P-superterse sind. Zuletzt zeigen wir, daß das Sortierproblem in einem bestimmten Sinn genauso schwierig wie das Membership-Problem ist.

Literatur:

[A&90] A. Amir, R. Beigel, and W. Gasarch. Some connections between bounded query classes and non-uniform complexity. In *Proceedings 5th Structure in Complexity Theory Conference*, pages 232–243. IEEE Computer Society Press, 1990.

[AG88] A. Amir and W. Gasarch. Polynomial terse sets. *Information and Computation*, 77:37–55, 1988.

[Bei87] R. Beigel. A structural theorem that depends quantitavely on the complexity of SAT. In *Proceedings 2nd Structure in Complexity Theory Conference*, pages 28–32. IEEE Computer Society Press, 1987.

[1]Technische Universität Berlin, Fachbereich Informatik, FR 6–2, Franklinstr. 28–29, 1000 Berlin 10

A Fast Sort using Parallelism within Memory

Claudia Leopold, Humboldt-Universität zu Berlin

We model the internal structure of memory by a tree, where nodes represent memory modules (like cache, main memory, disks), and edges represent busses between. The modules have the smaller access times, capacities and block sizes the nearer they are to the root. The parameters are identical for modules at the same depth. All busses may in parallel transmit blocks of data, when conflicts are excluded. So we reflect the hierarchical structure of memory at one hand, and possible sources of parallelism at the other.

We give a deterministic sorting algorithm based on Greed-Sort ([2]). Its running time is shown to be optimal up to a constant factor. In particular, a lower bound on I/O is met for each pair of adjacent memory levels, and the majority of busses is kept working in parallel most of the time. Our tight bound yields the number of parallel modules necessary at each hierarchy level to overcome the I/O bottlenecks of sorting.

The algorithm also applies to the less general models UMH and P-UMH ([1], [2]). In $\text{UMH}_{1/(k+1)}$, it has a running time of $O(N \log N)$, hence giving a positive answer to the question for the existence of such an algorithm posed in [2]. In $\text{P-UMH}_{1/(k+1)}$, the randomized $O\left(\frac{N}{P} \log N \log\left(\frac{\log N}{\log P}\right)\right)$ algorithm ([2]) is improved to a deterministic algorithm running in optimal time $O\left(\frac{N}{P} \log N\right)$.

References

[1] A.Alpern, L.Carter, E.Feig: *Uniform Memory Hierarchies,* in Proc. 31th IEEE Symp. on Foundations of Computer Science, 1990, 600-608

[2] M.H.Nodine, J.S.Vitter: *Large-Scale Sorting in Parallel Memories,* in Proc. 3rd Int. Symp. on Parallel Algorithms and Architectures, 1991

Erweiterte BDD's — Datenstrukturen für Boolesche Funktionen im Spektrum zwischen BDD's und BP1

Detlef Sieling Ingo Wegener
Universität Dortmund
Lehrstuhl Informatik II
Postfach 50 05 00
W-4600 Dortmund 50

Eine Datenstruktur für Boolesche Funktionen soll eine kompakte Darstellung einer möglichst großen Klasse von Booleschen Funktionen und die effiziente Ausführung von Operationen wie den Test auf Gleichheit zweier Funktionen, den Test auf Erfüllbarkeit, die Berechnung der Anzahl der erfüllenden Belegungen und die Verknüpfung zweier Funktionen mit einer Operation $o \in B_2$ ermöglichen.

Eine häufig verwendete Datenstruktur sind *(Ordered) Binary Decision Diagrams (BDD's)*, das sind read-once Branching Programme, bei denen eine Variablenordnung gegeben ist, d. h. wenn auf einem Berechnungspfad die Variable x_i vor x_j abgefragt wird, muß dieses auch für alle anderen Berechnungspfade, auf denen x_i und x_j vorkommen, gelten. Die Größe der BDD's einer Funktion ist häufig entscheidend von der gewählten Variablenordnung abhängig. Das führt dazu, daß Funktionen, bei denen für verschiedene Subfunktionen verschiedene Variablenordnungen günstig sind, u. U. für jede Variablenordnung nur exponentielle BDD's haben.

Wir wollen das Konzept der BDD's so erweitern, daß es auch für derartige Funktionen Diagramme mit polynomieller Größe gibt. Es soll erlaubt sein, daß es von den bereits getesteten Variablen abhängt, welche Variable als nächste getestet wird. Dadurch kann für jede Subfunktion eine günstige Reihenfolge gewählt werden. Die Einschränkung, daß jede Variable nur einmal getestet werden darf, soll jedoch bestehen bleiben. Weil es für allgemeine read-once Branching Programme ein NP-hartes Problem ist, aus den Diagrammen für f_1 und f_2 ein Diagramm für $f_1 \circ f_2$ zu berechnen, ist es nötig, daß die Prozedur, die die nächste abzufragende Variable bestimmt, für alle BDD's in einer Rechnung gleich ist.

Wir zeigen für diese Datenstruktur:

- Es gibt einen effizienten Reduktionsalgorithmus, und für jede Boolesche Funktion gilt, daß die reduzierten Diagramme bis auf Isomorphie eindeutig sind, wenn die Prozedur, die die nächste abzufragende Variable bestimmt, fest ist.
- Es gibt effiziente Algorithmen für die Berechnung der Datenstruktur für $f_1 \circ f_2$ aus den Datenstrukturen für f_1 und f_2, für die Tests auf Erfüllbarkeit und Gleichheit und die Berechnung der Anzahl erfüllender Belegungen.
- Es gibt Funktionen, für die BDD's bei jeder Variablenordnung exponentielle Größe haben, während die Größe der erweiterten BDD's nur linear in der Anzahl der Variablen ist.

Minimum Cost Paths in Periodic Graphs

Franz Höfting* and Egon Wanke*

Abstract

We consider directed graphs with d-dimensional integer vector weights and rational cost values associated with the edges. We analyze the complexity of the following problem: Given two vertices and a d-dimensional target vector $m \in \mathbb{Z}^d$, find a minimum cost path between the two given vertices such that the vector sum of all edges in the path equals the given target vector m. This MINIMUM COST m-PATH problem is equivalent to the minimum cost path problem in periodic graphs. As subproblems the reachability of the target vertex with displacement m and the problem of the unboundedness of the solution have to be solved. We show that the general version of the MINIMUM COST m-PATH problem is NP-complete under various restrictions, but solvable in pseudo polynomial time if the dimension of the vector weights is bounded by a constant. That is, it is solvable in polynomial time with respect to the input and the largest integer in the vector weights associated with the edges.

*Fachbereich Mathematik/Informatik, Uni-GH Paderborn, Warburger Str. 100, W-4790 Paderborn, e-mail: fh@uni-paderborn.de, egon@uni-paderborn.de

Basisklauseln für lineare Resolution

Friedhelm Hinz, Universität Trier

Verschiedene Einschränkungen des aussagenlogischen Resolutionskalküls sind als vollständig bekannt, uns interessiert hier die lineare Resolution. Vollständigkeit heißt hier zunächst, daß jede widerspruchsvolle Klauselmenge eine Ableitung der leeren Klausel durch lineare Resolution gestattet. Darüber hinaus ist bekannt, daß für eine solche Ableitung jede Klausel als Basisklausel gewählt werden kann, deren Streichung zur Widerspruchsfreiheit der Klauselmenge führen würde. Wir verschärfen dieses Resultat dahingehend, daß jede Klausel als Basisklausel gewählt werden kann, die überhaupt in einer (nicht notwendig linearen) Ableitung der leeren Klausel verwendet wird.

A threshold for unsatisfiability

Andreas Goerdt
FB 17 Mathematik/ Informatik
Universität-GH- Paderborn
Postfach 1621
Warburger Straße 100
D-W-4790 Paderborn
Germany

Abstract

We show the following threshold property of satisfiability of propositional formulas in 2-CNF: If $C = 1 + \varepsilon$ where $\varepsilon > 0$ is fixed, then almost all formulas in 2-CNF with $C * n$ clauses over n variables are unsatisfiable. If $C = 1 - \varepsilon$ then almost all such formulas are satisfiable. Here "almost all" simply means that the probability with respect to the uniform distribution of instances considered here tends to n as $n \to \infty$.

Due to the close relationship between satisfiability of formulas in 2-CNF and graph theoretic properties it is not surprising that our proof uses techniques from the theory of random graphs.

Dynamische Einbettung von Bäumen in DeBruijn-Netzwerke

Maria Breuing
Fachbereich Mathematik/Informatik
Universität – Gesamthochschule Paderborn

Diese Arbeit stellt Ergebnisse zur dynamischen Einbettung beliebiger Bäume mit M Knoten in DeBruijn-Netzwerke und verwandte Netzwerke mit N Prozessoren vor.

Zunächst werden binäre Bäume und DeBruijn-Netzwerke, dann d-äre Bäume und Varianten von DeBruijn-Netzwerken mit Ausgangsgrad d betrachtet.
In beiden Fällen liefert ein einfacher probabilistischer Algorithmus eine dynamische Einbettung, die mit hoher Wahrscheinlichkeit bei Kantenstreckung 1 eine maximale Knotenlast von $O(M/N + \log^2 N)$ erzielt.
Insbesondere wird mit hoher Wahrscheinlichkeit eine gleichmäßige Lastverteilung garantiert, falls $M \geq N \log^2 N$ ist.

Der verwendete Algorithmus arbeitet folgendermaßen:
Sei $u = u_0 \dots u_i$ ein Knoten im Level i eines Baumes T, der bereits auf einen Prozessor w im Netzwerk abgebildet wurde, und sei $u' = u_0 \dots u_i x$ ein Sohn von u. ($u_0, \cdots, u_i, x \in \{0, 1\}$, falls T binär; $u_0, \cdots, u_i, x \in \{0, \cdots, d-1\}$, falls T d-är.)
Wähle für u zufällig eine Permutation Π_u aus der Menge aller 2- bzw. d-stelligen Permutationen aus.
Der Wert $\Pi_u(x)$ entscheidet dann, auf welchen Nachbarn von w der Knoten u' im Netzwerk abgebildet wird.

Im Gegensatz zur Einbettung in Butterfly-Netzwerke (vgl. [1]) ist keine künstliche Streckung des einzubettenden Baumes mehr nötig, um die oben genannte maximale Knotenlast zu erhalten.

Für die Einbettung d-ärer Bäume in (Standard-)DeBruijn-Netzwerke ergibt sich bei Kantenstreckung $\log d$ ebenfalls mit hoher Wahrscheinlichkeit eine maximale Knotenlast von $O(M/N + \log^2 N)$.

Literatur:

[1] T. Leighton, M. Newman, A. G. Ranade, E.Schwabe;
Dynamic Tree Embeddings in Butterflies and Hypercubes.
ACM symposium on parallel algorithms and architectures, 1989 Santa Fé.

Branching Programme als Datenstruktur für den Schaltkreisentwurf

Jordan Gergov, Christoph Meinel
FB IV - Theoretische Informatik
Universität Trier
D–5500 Trier

Der Wunsch und die technischen Möglichkeiten, immer komplexere Schaltfunktionen zu implementieren, setzt die Fähigkeit voraus, algorithmisch mit diesen Schaltfunktionen umzugehen. Das erfordert eine rechnerinterne Darstellung, die einerseits möglichst kompakt ist und andererseits eine effiziente Ausführung diverser Analyse- und Bearbeitungsalgorithmen unterstützt. In jüngster Zeit finden unter diesem Gesichtspunkt die aus der Komplexitätstheorie bekannten eingeschränkten Branching–Programm–Modelle besondere Aufmerksamkeit.

Im Vortrag wird die Komplexität von einigen wichtigen Analyse– und Manipulationsproblemen im Schaltkreisentwurf für eingeschränkte Branching–Programm–Typen untersucht. Insbesondere geht es dabei um

- das Erfüllbarkeitsproblem und den Tautologietest,
- das Äquivalenzproblem und
- diverse Syntheseprobleme.

Gestörte Schaltkreise sublogarithmischer Tiefe

Bernd Schmeltz
Institut f. Theoretische Informatik
TH Darmstadt

Folgende Eigenschaften haben (im Gegensatz zu Schaltkreisen mit beschränktem Fanin), einen wesentlichen Einfluß auf die Möglichkeit, zuverlässige Schaltkreise mit unbeschränktem Fanin und kleiner Tiefe zu konstruieren:

- die Menge der Funktionen, die von einem einzelnen Gatter berechnet werden können,
- die Frage, ob die genauen Fehlerwahrscheinlichkeiten bekannt sind.

Entsprechend den von einzelnen Gattern berechenbaren Funktionen unterscheiden wir

- UND/ODER-Schaltkreise,
- gewöhnliche Schwellwert-Schaltkreise,
- gewichtete Schwellwert-Schaltkreise.

Wir betrachten zwei Arten von Zuverlässigkeit. Bei *schwacher Zuverlässigkeit* brauchen wir (mit hoher Wahrscheinlichkeit) eine korrekte Ausgabe, falls die individuellen Fehlerwahrscheinlichkeiten der Komponenten des Schaltkreises jeweils gleich ϵ sind; bei *starker Zuverlässigkeit* muß das für jede Wahl von individuellen Fehlerwahrscheinlichkeiten aus $[0, \epsilon]$ gelten.

Schwach zuverlässige Schwellwert-Schalkreise können bei gleicher Tiefe auf Kosten einer polynomiellen Zunahme von Größe und Fanin konstruiert werden.

Im Gegensatz dazu können zuverlässige UND/ODER-Schaltkreise und stark zuverlässige Schwellwert-Schaltkreise der Tiefe D höchstens $\mathrm{EXP}(O(D \log D))$ Eingabebits berücksichtigen. Dies bedeutet insbesondere, daß *keine zuverlässigen Schaltkreisfamilien konstanter Tiefe konstruiert werden können.* Wir zeigen dies für den Fall, daß Fehler in den Drähten des Schaltkreises auftreten. Bei Fehlern in den Gattern liegen die Dinge komplizierter; hier ist der Beweis bisher nur für UND/ODER-Schaltkreise und monotone Schwellwert-Schaltkreise gelungen.

The Complexity of Scheduling Problems with Communication Delays

Andreas Jakoby
Technische Hochschule Darmstadt [2]

A directed acyclic graph can be used to represent the dependencies among elementary computation steps in a given algorithm. We will call such a graph a **process graph**. Let us assume that each node of the process graph has the same execution time which will be taken as the unit. If one wants to implement an algorithm as fast as possible on a synchronous multiprocessor architecture, one simply has to schedule the nodes of its process graph on the available set of processors. In general, there will be a delay due to the necessary communication between processors. Typically, a single communication step exceeds the time of an elementary computation step by far. If processor P_a has computed an intermediate result z in the t-th step and this is needed by some other processor P_b as input for its computation one cannot assume that z is available for P_b already at step $t+1$. This leads to the following scheduling problem.

A **process graph** (G,δ) is a DAG $G=(V,A)$ with a delay function $\delta : A \rightarrow \mathbb{N}$. Let (G,δ) be a process graph and $P=\{P_1,P_2,\ldots\}$ a set of processors. A **schedule** S for (G,δ) and P is a set of executions $S \subset V \times P \times \mathbb{N}$, where processor P_i executes **task** x at time t such that the following conditions hold:

1. For each task $x \in V$ there is at least one processor P_i and a time t such that P_i executes x at t.
2. Each P_i does not execute different tasks x, y at the same time t.
3. If x does not correspond to a source in G and x is executed by processor P_i at time t and if y is a direct predecessor of x then either
   (a) y is executed by P_i at time $t-1$ or
   (b) y is executed by some other processor P_j at time $t-1-\delta(y,x)$.

Let $S_i := \{(x,t)|(x,P_i,t) \in S\}$ be the **schedule** S restricted to processor P_i and let $T(S_i) := 1+\max\{t|\ \exists x \in V : (x,t) \in S_i\}$ the finishing time of P_i. Let $S_i(t)$ denote the task which is executed by P_i at time t. The **time span** or **length** of S is given by $T(S) := \max_{P_i \in P} T(S_i)$. A schedule S is called **optimal** for (G,δ) and P, if $T(S) \leq T(S')$ for all schedules S' for (G,δ) and P. Let $T_{opt}((G,\delta),P) := \min\{T(S)|S$ is a schedule for (G,δ) and $P\}$ be the optimal schedule length for a process graph (G,δ) and a set of processors P.

In the following we will consider the case of maximal parallelism, where the set of processors P is chosen arbitrarily large. Therefore, we simply denote the minimal scheduling time by $T_{opt}(G,\delta)$. We distinguish the following variants of this scheduling problem w.r.t. the type of communication delays used:

DELAY SCHEDULING (DS):
Given a process graph (G,δ) and a deadline $T^* \in \mathbb{N}$, decide:
Is there a schedule S with $T(S) \leq T^*$, i.e. $T_{opt}(G,\delta) \leq T^*$?

UNIFORM DELAY SCHEDULING (UDS):
DS restricted to inputs (G,δ) with $\delta : A \rightarrow \tau \in \mathbb{N}$ where τ may depend on the number of vertices of the chosen graph G.

[2] Institut für Theoretische Informatik, Alexanderstraße 10, D-W-6100 Darmstadt, Germany
email: reischuk@iti.informatik.th-darmstadt.de

CONSTANT DELAY SCHEDULING (CDS):
DS restricted to inputs (G, δ) with $\delta : A \leftarrow c \in \mathbb{N}$ where c is a fixed constant independent of the chosen graph G.

The UDS problem was defined by Papadimitriou and Yannakakis in [PY88] and shown to be $\mathcal{NP}$-complete.

We show that the delay scheduling problem remains difficult, even for trees with a very simple structure. A nontrivial strategy to compute an optimal schedule efficiently is only known for completely symmetric trees. The following table gives an overview on the complexity in the different situations.

	DAGs	binary trees	complete trees
CDS	$\mathcal{P}$ [JKS89]	$\mathcal{P}$	$\mathcal{P}$
UDS	$\mathcal{NP}$-complete[PY88]	$\mathcal{NP}$-complete[JR92]	$\mathcal{P}$ [J91,JR92]
DS	||	||	$\mathcal{NP}$-complete[J91,JR92]

References

[J91] A. Jakoby, Prozess-Prozessor-Allokationsprobleme für Baumförmige Datenflussgraphen, Diplomarbeit, Technische Hochschule Darmstadt, 1991

[JKS89] H. Jung, L. Kirousis, P. Spirakis, Lower Bounds and Efficient Algorithms for Multiprocessor Scheduling of DAGs With Communication Delays, Proc. 1. SPAA, 1989, 254 - 264

[JR92] A. Jakoby, R. Reischuk, The Complexity of Scheduling Problems with Communication Delay for Trees, to appear in the proceedings of SWAT'92

[PY88] C. Papadimitriou and M. Yannakakis, Towards an Architecture-Independent Analysis of Parallel Algorithms, Proc. 20. STOC, 1988, 510-513, see also SIAM J. Comput. 19, 1990, 322-328

On Relationship Between Time and Reversal Complexity Measure *

Maciej Liśkiewicz
Institut für Theoretische Informatik, TH Darmstadt
and
Instytut Informatyki, Uniwersytet Wrocławski, Poland

Along with time and space, the number of reversals made by tape heads during a Turing machine (TM) computation is an important and widely used complexity measure. The reversal complexity was introduced by Hartmanis (*J. Comput. System Sc.*, 2 (1968) 117-135) and Fischer (*J. Comput. System Sc.*, 2 (1968) 136-147) and then it was intensively studied by many authors. Recently it has been showed that the reversal complexity on multitape deterministic TMs is intimately connected with parallel time.

We investigate a relationship between reversals and deterministic time. It is obvious that a TM working in time $T(n)$ makes at most $R(n) = T(n)$ reversals. Recently Chen and Yap (*SIAM J. Comput.*, 20 (1991), 622-638) have showed that, for any reversal constructible function $S(n) \geq \log n$,

$$DSPACE(S) \subseteq DREVERSAL(S), \tag{1}$$

where $DREVERSAL(S)$ denote the class of languages recognized by deterministic TMs working within $S(n)$ reversals. It is not difficult to prove that (1) holds also for functions $S(n) \geq \log n$ which are not reversal constructible (note, that for such functions Chen and Yap show that $DSPACE(S)$ is included in $DREVERSAL(S^2)$). Now using (1) and the well known result of Hopcroft, Paul and Valiant relating time and space one can show that any deterministic multitape TM of time complexity $T(n)$ can be simulated by a multitape machine within $R(n) = T(n)/\log T(n)$ reversals. Our main result is that the function $R(n)$ can be reduced to $\sqrt{T(n)}$.

THEOREM. *Let $T(n) \geq n$ be an arbitrary function. Then*

$$DTIME(T) \subseteq DREVERSAL(\sqrt{T}).$$

The problem if the inclusion (1) is proper remains open. Our result relating time to reversal seems to confirm a conjecture that $R(n)$ reversal bounded machines are more powerful than the machines working in space $R(n)$.

*This research was supported by the Alexander-von-Humboldt-Stiftung

Liste der bisher erschienenen Ulmer Informatik-Berichte:

91-01 KER-I KO, P. ORPONEN, U. SCHÖNING, O. WATANABE:
Instance Complexity.

91-02 K. GLADITZ, H. FASSBENDER, H. VOGLER:
Compiler-Based Implementation of Syntax-Directed Functional Programming.

91-03 ALFONS GESER:
Relative Termination.

91-04 JOHANNES KÖBLER, UWE SCHÖNING, JACOBO TORAN:
Graph Isomorphism is low for PP.

91-05 JOHANNES KÖBLER, THOMAS THIERAUF:
Complexity Restricted Advice Functions.

91-06 UWE SCHÖNING:
Recent Highlights in Structural Complexity Theory.

91-07 FREDERIC GREEN, JOHANNES KÖBLER, JACOBO TORAN:
The Power of the Middle Bit.

91-08 V. ARVIND, Y. HAN, L. HEMACHANDRA, J. KÖBLER, A. LOZANO,
M. MUNDHENK, M. OGIWARA, U. SCHÖNING, R. SILVESTRI, T. THIERAUF:
Reductions to Sets of Low Information Content.

92-01 VIKRAMAN ARVIND, JOHANNES KÖBLER, MARTIN MUNDHENK:
Bounded Truth-Table and Conjunctive Reductions to Sparse and Tally Sets.

92-02 THOMAS NOLL, HEIKO VOGLER:
Top-down Parsing with Simultaneous Evaluation of Noncircular Atrribute Grammars

92-03 PROGRAMM AND ABSTRACTS:
17. Workshop über Komplexitätstheorie, effiziente Algorithmen und Datenstrukturen
am 26. Mai 1992 in Ulm

Ulmer Informatik-Berichte

ISSN 0939-5091

Herausgeber: Fakultät für Informatik

Universität Ulm, Oberer Eselsberg, W-7900 Ulm