MATHEMATISCHES FORSCHUNGSINSTITUT OBERWOLFACH

Tagungsbericht 40/1989

Effiziente Algorithmen

17.9. bis 23.9.1989

The Oberwolfach Conference on "Efficient Algorithms" was organized by K. Mehlhorn (Saarbrücken), W.J. Paul (Saarbrücken), and H. Walter (Darmstadt). The participants came from the following countries: France, FRG, GDR, Israel, Netherlands, Sweden, and USA. The goal of the meeting was to bring together researchers working on the design, analysis and implementation of efficient algorithms and data structures. There were given 24 lectures concerning the following fields:

- computational geometry
- data structures
- graph algorithms
- numerical algorithms
- probabilistic methods
- parallel and distributed computation

Vortragsauszüge

H. ALT

Approximation of Polygonal Shapes

We consider the problem of approximating planar figures which are bounded by polygonal curves by simpler figures using methods from computational geometry. The simpler figures are, e.g., axes-parallel rectangles, arbitrary rectangles, circles, or k-gons where k is some constant. As measure for the quality of the approximation we use the area of the symmetric difference of both figures, δ_S, or the Hausdorff-metric, δ_H. We

obtain the following results, where the figure to be approximated is a convex n-gon P: With respect to δ_S the axes-parallel rectangle optimally approximating P can be constructed in $O(\log^3 n)$ time. With respect to δ_H, an approximating rectangle for P, whose Hausdorff-distance differs from the optimal only by a constant factor, can be constructed in $O(n)$ time. A k-gon, which is "pseudo-optimal" int this sense, can be found in $O(n^2 \log n)$ time using a result by Melkman/O'Rourke. This is joint work with J.B. Blömer, M. Godan, and H. Wagener.

B. BECKER

Computations over Finite Monoids and their Test Complexity

We consider the test pattern generation problem fo circuits, which perform computations over finite monoids. Two important computations, "expression evaluation" and "parallel prefix computation" are considered. The relations between algebraic properties of a monoid and its test complexity (with respect to these problems) is studied. The test complexity of a monoid with respect to a problem is measured by the number of tests needed to check the best testable circuit (in a certain computational model) solving the problem. It is shown for both types of computations that the set of all finite monoides partitions into exactly three classes with constant, logarithmic and linear test complexity, respectively. These classes are characterized using algebraic properties. For example "groups" are exactly the monoids with constant test complexity. For each class we provide circuits with optimal test sets and efficient methods, which decide the membership problem for a given finite monoid M, i.e. determine the test complexity of M.

T. BETH

Galois Engineering for Algorithms

The principles of algorithm engineering based on specification and modeling abstract data types are being investigated from the point of view of modern algebra. In several practical applications in areas like digital signal processing, Fourier analysis and polynomial arithmetic the role of group theoretic modeling is demonstrated using representation theoretic methods to derive efficient algorithms. Following Jonssen (1981) and Engeler (1981) the concept of a Galois theory of problems is introduced. We show that for a homogeneous class of problems the inverse Galois correspondence

of finding a problem suited to a automorphism subgroup can be solved. Examples of applications are the (retrospectively) automatic generation of practically all known FFT-algorithms. As a new result of algorithm engineering we present the automatic derivation procedure for the so-called Fast Hartley Transform algorithm.

R. COLE

Conjectures for Splay Trees: recent advances

The splay tree is a data structure, devised by Sleator and Tarjan, which implements a dictionary and is efficient in an amortized sense. For instance, n accesses (searches, inserts, deletes) to a splay tree, initially of n nodes, take time $O((m+n)\log(m+n))$. Various conjectures, due to Sleatro and to Tarjan, are reviewed, including:

(i) The Dynamic Optimality Conjecture: The splay tree is as efficient, up to a constant factor, in an amortized sense, as any other algorithm which manipulates a binary search tree by means of rotations.

(ii) The Dynamic Finger Conjecture: Let i_j be the j-th item accessed in a sequence of m searches, where i_j denotes the rank of the item in sorted order. Let $d_j = |i_j - i_{j-1}| + 1$, for $1 \leq j \leq m$, where i_0 is the item originally at the root of the splay tree. Then the sequence of m searches take time $O(m+n+\sum_{j=1}^{m}\log d_j)$. Cole's bound of $O(m+n+\sum_{j=1}^{m}[\log d_j + \log\log n\alpha(n)] + n\log\log n)$ is reported.

(iii) The Scanning Theorem [Tarjan]. If the n items of a splay tree are accessed in left to right order, the time taken for the accesses is $O(n)$. A new, simple proof due to Sundar is described; it provides a slightly tighter bound than Tarjan's proof.

(iv) The Deque Conjecture. Suppose a deque is implemented using a splay tree (in the natural way). Then n operations on a deque, initially of n items, take time $O(m+n)$. Sundar's bound of $O((m+n)\alpha(m+n))$ is reported.

P. FLAJOLET

Automatic Analysis of Algorithms

The problem is to find some well defined class of algorithms and data structures whose analysis in the average case is "decidable". From combinatorial analysis we know a

catalog of combinatorial constructions that have translations into generating functions. We can extend those translation mechanism to certain program construction. In this way one obtains a (fairly!) well defined class of programs whose specification can be "compiled" directly and automatically into functional equations over generating functions. We can then try to categorize those class of functional equations, the ultimate problem being to extract asymptotic forms of coefficients several routs are open (Darboux's method, Haymann-Harnis-Schoenfeld saddle point methods, Flajolet-Odlyzko for singularity analysis). A mixed strategy based on saddle point methods and singularity analysis has been implemented in the computer algebra system Maple: It relies on an implementation of Hardy scales (in "orders of infinity") and makes it possible to extract a "non-null" class of problems that are solvable automatically. Extensions to implicitly defined functions and singular differential systems seem possible. They suggest the possibility of developing a theory of "statistical combinations". (Based on foint work with B. Salvy and P. Zimmermann.)

T. HAGERUP

A Randomized Algorithm for Network Flow

We propose a new algorithm for the problem of computing a maximum flow in a directed network. The new algorithm, called the PLED or Prudent Linking Excess Diminishing algorithm, is randomized and achieves an expected running time of $O(nm + n^2 \log^3 n)$ on networks with n vertices and m edges. For networks that are not very sparse, it compares favorably with the best known deterministic and strongly polynomial algorithm (Goldberg/Tarjan), whose running time is $O(nm \log(n^2/m))$. The algorithm combines the excess scaling technique of Ahuja and Orlin with the use of the dynamic tree data structure, and several new ideas are introduced in its analysis. In particular, the running time is shown to be related to the total number of so-called PTR events, and the number of PTR events is demonstrated to be sufficiently small with high probability if all adjacency lists of the network are randomly permuted regularly during the execution. (Joint work with J. Cheriyan.)

J. HASTAD

A simple lower bound for the depth of monotone circuits computing clique

We prove an $\Omega(\sqrt{k})$ lower bound for the depth of a monotone circuit recognizing whether a graph contains a subgraph which is the complete graph on k vertices. The

bound holds for $k \leq n^{2/3}$ where n is the number of nodes in the graph. This result was already known since it follows from the lower bounds for the size of such circuits by Alon and Boppana. However our proof is much simpler.

M. JÜNGER

Convex Hulls and Euclidean Matching

In the first part of the talk I outline our algorithm for finding the convex hull of a set A of points in the Eucledean plane. The idea is to apply any convex hull algorithm CH to a small subset $S \subseteq A$ such that S contains $O(\log n)$ points and $conv(S) = conv(A)$ with high probability. If the n points are independently and uniformly distributed in the unit square the average case analysis shows that the method has linear average case time complexity provided that CH has polynomial worst case time complexity. In the unit disc, the small set $S \subseteq A$ contains $O(\sqrt[3]{n})$ points and we get linear average case time complexity f CH has $O(n^r)$ worst case time complexity, where $r < 3$. In the second part of the talk I present an $O(n \log n)$ approximation algorithm for perfect matching og n points in the Eucledean plane. The matching is determined by a recursive decomposition technique using a minimum length spanning tree of the n points. At the same time, the algorithm computes a lower bound for the total length of an Eucledean perfect matching by finding a feasible solution to the linear programming dual of the perfect matching problem.

M. KARPINSKY

Fast Parallel Algorithms for GF[2]-Interpolation and Counting

We design fast parallel algorithms for interpolation of boolean functions f in their RSE-representation (t-terms Galois polynomials in GF[2][$x_1, \ldots, x_n$]). We study also computational difficulty of computing the number of solutions (or satisfying assignments) of f. The interpolation algorithm works over a "slight" field extension GF[$2^{\lceil 2\log(nt)+3 \rceil}$] and runs in $O(\log^3(nt))$ parallel time and $O(n^2t^5\log^2(nt))$ processors. This gives the first polynomial time (and NC^3-algorithm for the boolean RSE-conversion problem (conversion of any boolean function, given by arbitrary boolean circuit such that $|RSE(f)| \leq t$), into the equivalent RSE-formula. We give also a fast parallel algorithm (NC^2) for counting the number of satisfying assignments (solutions) of quadratic polynomials over $GF[2]$. This can be generalized to arbitrary finite fields and used for determining NC-simulations of (XOR, AND)-depth-2 AND-Fan-In-2 randomized circuits.

B. KORTE

Polynomial Time Algorithms for Convex Shellings in the Plane

Convex shellings are combinatorial structures which are special cases of antimatroids, which again are a subclass of greedoids. In an earlier paper we could give a polyhedral characterization for all but one antimatroids. The exception is convex shellings. However, h we can show that the validity problem for convex shellings can be decided by a polynomial time algorithm. Using a linear ordering of the candidates for a recurrence we could demonstrate an $O(n^3)$ resp. $O(n^5)$ algorithm. This question is related to the famous happy-end problem of Erdös and Szekeres [1953]. Unfortunately we were not able so far to improve the lower bound of the Erdös-Szekeres problem. This is joint work with Laszlo Lovasz.

J. VAN LEEUWEN

Structuring NC algorithms

The interconnection pattern in a (boolean) NC circuit can differ from layer to layer arbitrarily. From a practical viewpoint one would want to restrict to bounded-degree single-stage networks. This leads to consider SNC^k $(k \geq 1)$: the class of problems solvable by networks that are $O(\log^k n)$ iterates of a bounded-degree single-stage (boolean) network. Clearly $NC^k \subseteq SNC^{k+1}$, but direct simulations also affect the width of a circuit. It seems that the increase in width is not needed when the extra logarithmic factor in depth is allowed, and the extra factor in width can remain small $(\sim w^\epsilon)$ if it is not. Some well known NC problems (for example, n-bit binary addition) can be solved by structured circuits without requiring the extra factors in size and depth, but it is open whether this can be achieved for all NC problems.

E. MAYR

Parallel Approximation Algorithms

We study the parallel complexity of approximating solutions to some P-complete as well as NP-complete problems. We exhibit typical representatives for various types of

NC-approximation schemes, among them list scheduling (strongly NC approximation scheme), the high degree-subgraph problem (approximation up to a factor of 2 unless $P = NC$), and various discretization based problems which are highly inefficient in practice (bin-packing, makespan optimization, subset sum).

K. MEHLHORN

On the Complexity of a Game Related to the Dictionary Problem

We consider a game on trees which is related to the dictionary problem. There are two players A and B which take turns. Player A models the user of the dictionary and player B models the implementation of it. At his turn, player A modifies the tree by adding new leaves and player B modifies the tree by replacing subtrees. The cost of an insertion is the depth of the new leaf and the cost of an update is the size of the subtree replaced. The goal of player A is to maximize cost and the goal of B is to minimize it. We show that there is a strategy for player A, which forces a cost of $\Omega(n \log\log n)$ for an n-game, i.e. a game consisting of n turns of both players, and that there is a strategy for player B, which keeps the cost in $O(n \log\log n)$. (Joint work with S. Näher and M. Rauch)

F. MEYER AUF DER HEIDE

How to Distribute a Dictionary in a Complete Network

A dictionary is a data structure which supports the instructions lookup, insertion, and deletion. We implement a dictionary on a complete network of some number p of processors. Each processor can be fed with instructions. This distributed dictionary is based on a randomized hashing strategy. It is the first known optimal distributed dictionary, i.e., we achieve an $O(n/p)$ expected time bound for executing n instructions, n/p per processor, and expected constant lookup time. Our novel hash functions for distributing the data items among the processors can be evaluated in constant time, but, with high probability, they behave like random functions on the up to n keys currently considered in the operations. This property is basic for the analysis of the algorithm. (Joint work with M. Dietzfelbinger.)

B. MONIEN

On the Influence of Rounds Necessary to Disseminate Information

We study how efficiently information can be spread in a communication network and investigate the number of rounds necessary to spread all the pieces of information to all processors. We use the "telegraph communication mode", when in each round each processor is active only via one of its links and the communication is one-way, i.e., each processor can either transmit or receive , but not both. We prove upper and lower bounds. The two bounds are related to Fibonacci numbers and differ by, at most, an additive constant of 1. Our lower bound technique uses elments from matrix theory, specifically matrix norms. These results show, for the first time, that in the two-way mode information can be distributed faster than in the one-way mode. This is joint work with S. Even.

S.M. MÜLLER

The Influence of Startup-Time on the Performance of Parallel Computers in Numerical Applications

We first looked into the parallelization of numerical algorithms (basic numerics, ODE's and PDE's) with regard to the influence of startup-time on the parallel runtime. Normally the startup-time is very large. One startup takes as long as 500 - 300 floating point operations. The computing model is p processors with private memory, crossbar network, single program multiple data mode. We have seen that 3 methods will be enough to parallelize many numerical algorithms. **1. Simulation Method:** In theory many processors can be used but to get a better efficiency one uses less processors than possible. Each processor has to simulate some theoretical ones. **2. Separation Method:** Strongly sequential algorithms must be separated into nearly independent subproblems in a mathematical way using spanning sets or renumbering of the variables. **3. Pipelining Method** is used for strongly sequential but iterative algorithms. One iteration is divided into some stages. Stages of different iterations will be overlapped. # processors << # iterations. With regard to the startup-time we get the result: If granularity h is large, $h = 100S$ we get good efficiencies $eff = 70\%$. One also gets a condition for the size of memory: $G = 10^5$, $S = 10^3$ then memory must be larger than 8 Mbyte. Then we regarded the **Triangle Method**, which is used to save startup-time in te N-S-E-W kind (after each iteration the borders must be exchanged with those of the direct neighbours). It is sometimes better to compute as many values of different time-stages as one can do without transfer and than transfer data.

S. NÄHER

LEDA, a Library of Efficient Data Types and Algorithms

LEDA is a library of efficient data types and algorithms. At present, its strength is graph algorithms and related data structures. The computational geometry part is evolving. The main features of the library are

1) A clear separation between (abstract) data types and the data structures used to implement them: The data types currently available are stack, queue, list, set, dictionary, ordered sequence, priority queue, directed graph and undirected graph. The most difficult decision, which we faced, was the treatment of positions and pointers. For the efficiency of many data structures it is crucial that some operations on the data structure take positions in (= pointers into) the data structure as arguments. We have chosen an item concept to cast the notion of position into an abstract form.

2) Type parameters: Most of our data types have type parameters. For example, a dictionary has a key type K and an information type I and a specific dictionary type is obtained by setting, say, K to int and I to real.

3) A comfortable data type graph: The data type graph allows to write programs for graph problems in a form close to the way the algorithms are usually presented in text books.

4) Ease of use: LEDA is written in C++ All data types and algorithms are precompiled modules which can be linked with application programs.

(Joint work with K. Mehlhorn)

J. B. ORLIN:

Two Algorithms for Network Optimization

We first considered the minimum cost flow problem on a network $G = (N, A)$ with $n = |N|$ nodes and $m = |A|$ arcs. We developed an algorithm that solves the minimum cost flow problem as a sequence of $O(n \log n)$ shortest path problems. This is the best known "strongly polynomial algorithm" for the minimum cost flow problem. (An algorithm is strongly polynomial if it is polynomial in the dimension of the problem and not in the number of bits of precision for the data.) We next considered the maximum

flow problem on a network $G = (N, A)$ in which U is the largest (integer) capacity of an arc. We gave an $O(n^2 \log(m/n) \log U)$ time parallel PRAM algorithm using m/n parallel processors. For parallel algorithms with a small number of processors this is nearly the best time bound. (The best time bound is due to the author with Ahuja and Tarjan.)

J. REIF

Optimal Size Integer Division Circuits

Division is a fundamental problem for arithmetic and algebraic computation. We contribute an improved method of high order iterative formulas. Theses techniques yield boolean circuits (of bounded fan-in) for integer division that have size $O(M(n))$ and depth $O(\log n \log\log n)$ where $M(n)$ is the size complexity of $O(\log n)$ depth integer multiplication circuits ($M(n)$ is known to be $O(n \log n \log\log n)$). Previously, no one has been able to devise a division circuit with size $O(n \log^c n)$ for any c, and simultaneous depth less than $O(\log^2 n)$. Our circuits are log space uniform, that is, they can be constructed by a Turing Machine in space $O(\log n)$.

R. REISCHUK

Parallel Algorithms for Graphs

Our goal is to design divide-and-conquer strategies to solve various kinds of combinatorial problems on graphs efficiently. This task splits into two parts: the decomposition of graphs and the combining of partial solutions on components of a graph into a global solution. To find separators of a graph we study the k-flow problem: given a graph G with special nodes s and t, find k node disjoint paths or a separating set of size at most $k-1$ between s and t. A simple $O(k \log n)$ time parallel algorithm is presented. We then discuss how all separating sets of a given size can be found efficiently in parallel . From this one can derive an NC^2 algorithm to completely decompose a graph into k - connected components. We have constructed a suitable algebraic framework to express optimization problems on graphs. Applying this on can derive fast parallel algorithms for a large number of graph problems for all graphs that have a decomposition into k-connected components of at most logarithmic size (in the number of total nodes) for small values of k.

L. RUDOLPH

A Complexity Theory of Efficient Parallel Algorithms

We outline a theory of parallel algorithms that emphasizes two crucial aspects of parallel computations: speedup and efficiency. We define six classes; One of particular interest, EP, is of algorithms that achieve a polynomial speedup with constant efficiency. The relations between these classes are examined. They are very robust over a wide range of models of parallel computation. For example, the class EP is invariant across the PRAM models of CRCW, CREW, EREW, CRCW + Read-modify-write and complete networks of processors. This is proved by simulations. A more restricted model can simulate a less restrictive model by using $P/\log P$ simulating processors for the P simulated processors. Each processor must simulate $\log P$ processors. During this time, the processors can integer sort the referenced addresses and therefore avoid any memory conflicts.

A. SCHÖNHAGE:

The Division of Complex Numbers

The standard reduction of complex division to real operations by

$$\frac{(a+ib)\cdot w}{(c+id)\cdot w} = \frac{A+iB}{u+iv} \qquad (1)$$

with the choice of $w = c - id$ (thus $v = 0$) requires 8 real mults/divs. With $w = 1 - i \cdot d/c$ (assuming $0 \le d \le c$, w.r.) this factor relating real and complex division can be reduced to 6. Lickteig showed that this 6 is optimal in the algebraic model, but for implementing multi-precision software (processing binary rationals of N bits up to errors 2^N), more favorable methods do exist. **Asymptotical bounds** (for $N \longrightarrow \infty$), based on fast integer multiplication $mod(2^N + 1)$ in time $\le \mu(N) \sim c_0 \cdot N \cdot \lg N \cdot \lg\lg N$, are like $\le 10\mu(N)$ for complex division versus $\le 5\mu(N)$ for real division. **For numbers of moderate length** (n 32-bit words, $n \le 300$, say), elementary real mul or div take $\sim 1/2n^2$ inner loops of the form "word * word + word $\longrightarrow$ word". Compared to this, complex division is possible by $\sim 2n^2$ loops (giving the factor of 4 instead of 6). The decisive idea is to use (1) with $w = 1 - i\Theta$, $\Theta = f/2^3 2 \approx d/c$ with a one word integer f, so that v becomes smaller than u by about one word, and then to perform ordinary division steps by the leading word of u to find the word of the real and the imaginary part of the quotient **alternatingly**.

E. UPFAL

Time-Randomness Tradeoff For Oblivious Routing

Three parameters characterize the performance of a probabilistic algorithm T, the run-time of the algorithm; Q, the probability that the algorithm fails to complete the computation in the first T steps and R, the amount of randomness used by the algorithm measured by the entropy of its random source. We present a tight tradeoff between these three parameters for the problem of oblivious routing on N-vertex bounded degree networks. We prove a $(1-Q)\log N/T - \log Q - O(1)$ lower bound for the entropy of a random source of any oblivious packet routing algorithm that routes an arbitrary permutation in T steps with probability $1-Q$. We show that this lower bound is almost optimal by proving the existence, for every $e^{\epsilon}\log N \leq T \leq \sqrt{N}$, of an oblivious algorithm that terminates in T steps with probability $1-Q$ and uses $(1-Q+o(1))\log N/T - \log Q$ independent random bits.

U. VISHKIN

Deterministic Sampling for fast Pattern Matching

Consider the following three-stage strategy for recognizing patterns in larger scenes: *Mimic randomization deterministically*: Sample several positions of the pattern. *Search for sample*: Find all occurrences of the sample in the scene. *Verify*: For each occurrence of the sample, verify occurrence of the full pattern. We show how to apply this intuitive strategy for matching patterns in strings: Given any pattern, we select (deterministically) a sample of its positions, whose size is at most logarithmic. Trivial processing of occurrences of the sample positions in the text sparse verification. This approach enables to perform the text analysis in $O(\log^* n)$ time and optimal speed-up on a PRAM. This improves on the previous fastest optimal speed-up result. It also leads to a new linear time serial algorithm for string matching. We expect the approach to be applicable for pragmatic pattern recognition problems. In some sense our algorithms are based on degenerate forms of computation, such as AND and OR of a large number of bits. However, traditional machine designs do not take advantage of such degeneracies, and usual complexity measures do not even enable to reflect them. This led us to conclude the paper with some speculative thoughts on desirable capabilities that would enhance computing machinery for some pattern recognition applications.

Berichterstatter: S. NÄHER

Tagungsteilnehmer

Prof. Dr. H. Alt
Institut für Mathematik III
der Freien Universität Berlin
Arnimallee 2-6

1000 Berlin 33

Dr. B. Becker
Fachbereich 10 - Informatik
Universität des Saarlandes
Im Stadtwald 15

6600 Saarbrücken 11

Prof. Dr. Th. Beth
Institut für Algorithmen und
Kognitive Systeme
Universität Karlsruhe
Haid-und-Neu-Str. 7

7500 Karlsruhe 1

Prof. Dr. R. Cole
Courant Institute of
Mathematical Sciences
New York University
251, Mercer Street

New York , N. Y. 10012
USA

Prof. Dr. P. Flajolet
INRIA
Bat. 8, Domaine de Voluceau-
Rocquencourt
B. P. 105

F-78150 Le Chesnay Cedex

Dr. T. Hagerup
Fachbereich 10 - Informatik
Universität des Saarlandes
Im Stadtwald 15

6600 Saarbrücken 11

Prof. Dr. J. Hastad
Dept. of Numerical Analysis and
Computing Science
Royal Institute of Technology
Lindstedtsvägen 25

S-100 44 Stockholm

Dr. M. Jünger
Mathematisches Institut
der Universität Augsburg
Memminger Str. 6

8900 Augsburg

Dr. H. Jung
Sektion Mathematik der
Humboldt-Universität Berlin
Postfach 1297
Unter den Linden 6

DDR-1086 Berlin

Prof. Dr. M. Karpinski
Institut für Informatik
Universität Bonn
Römerstraße 164

5300 Bonn 1

Prof. Dr. B. Korte
Institut für ökonometrie und
Operations Research
der Universität Bonn
Nassestr. 2

5300 Bonn 1

Prof. Dr. B. Monien
Fachbereich Mathematik/Informatik
der Universität Paderborn
Postfach 1621
Warburger Str. 100

4790 Paderborn

Prof. Dr. J. van Leeuwen
Department of Computer Science
University of Utrecht
P. O. Box 80, 089
Paudalaan 14

NL-3508 TB Utrecht

S. M. Müller
Lehrstuhl Professor Paul
FB 10 Informatik
Universität des Saarlandes
Im Stadtwald 15

6600 Saarbrücken 11

Prof. Dr. E. W. Mayr
Fachbereich 20 Informatik
Universität
Postfach 11 19 32

6000 Frankfurt 11

Dr. S. Näher
Fachbereich 10 - Informatik
Universität des Saarlandes
Im Stadtwald 15

6600 Saarbrücken 11

Prof. Ph.D. K. Mehlhorn
Fachbereich 10 - Informatik
Universität des Saarlandes
Im Stadtwald 15

6600 Saarbrücken 11

Prof. Dr. J. B. Orlin
Sloan School of Management
MIT E53-357

Cambridge , MA 02139
USA

Prof. Dr. F. Meyer auf der Heide
Fachbereich Mathematik/Informatik
der Universität Paderborn
Postfach 1621
Warburger Str. 100

4790 Paderborn

Prof. Dr. W. J. Paul
Fachbereich 10 - Informatik
Universität des Saarlandes
Im Stadtwald 15

6600 Saarbrücken 11

Prof. Dr. J. H. Reif
Department of Computer Science
Duke University

Durham , NC 27706
USA

Prof. Dr. E. Upfal
IBM Research Department K 53
650 Harry Road

San Jose , CA 95120-6099
USA

Dr. R. Reischuk
Institut für Theoretische Informatik
Technische Hochschule Darmstadt
Alexanderstr. 10

6100 Darmstadt

Prof. Dr. U. Vishkin
Institute for Advanced Computer Studies
University of Maryland

College Park , MD 20742-3251
USA

Prof. Dr. L. Rudolph
Institute of Mathematics and Computer Science
The Hebrew University
Givat-Ram

91904 Jerusalem
ISRAEL

Prof. Dr. E. Welzl
Institut für Mathematik III
der Freien Universität Berlin
Arnimallee 2-6

1000 Berlin 33

Prof. Dr. A. Schönhage
Institut für Informatik
Universität Bonn
Römerstraße 164

5300 Bonn 1