MATHEMATISCHES FORSCHUNGSINSTITUT OBERWOLFACH

Tagungsbericht 30/1997

Effiziente Algorithmen

3.8. – 9.8.1997

This years Oberwolfach meeting on efficient algorithms was organized by Zvi Galil (New York) and Kurt Mehlhorn (Saarbrücken). There were 33 participants coming from Canada (1), France (1), Germany (20), Italy (3), Switzerland (1), and USA (7). There were 32 presentations, each presentation taking between between 30 and 60 minutes. Wednesdays evening there was a session were open problems were presented and discussed. The list of problems discussed can be found at the end of the report.

The presentations covered a wide range of topics, ranging from pure theoretical talks to program demonstrations, the later using the presentation facilities offered by the institute. Several presentations had experimental results supplementing the theoretical work presented. Many talks presented recent work related to computational geometry or combinatorial optimization, but various other topics were covered too as given by the list below.

- Approximation algorithms
- Combinatorial optimization
- Computational geometry
- Computational molecular biology
- Dynamic algorithms
- Exact computations
- External memory algorithms
- Graph algorithms
- Implementations
- Network algorithms
- Network flow algorithms
- On-line algorithms
- Program demonstrations
- Searching problems
- String algorithms

Of social events Wednesday afternoon contained a four hour hiking tour and Thursday evening was devoted to wine drinking. The complete program of the meeting can be found on the following pages.

Schedule of the meeting

Monday

9.00 – 9.10	*Kurt Mehlhorn*: Opening Remarks
9.10 – 9.55	*Torben Hagerup*: Dynamic Algorithms for Graphs of Bounded Tree-width
10.05 – 10.40	*Giuseppe F. Italiano*: Splitting and Merging Techniques for Order Decomposable Problems, with Applications
10.55 – 11.35	*Paolo Ferragina*: On Sorting Strings in External Memory
11.45 – 12.30	*Ernst W. Mayr*: On Polynomial Ideals, Their Complexity, and Applications
16.00 – 16.30	*Friedhelm Meyer auf der Heide*: Dynamic Data Structures for Realtime Management of Large Geometric Scenes
16.40 – 17.15	*Raimund Seidel*: Planar Point Location Close to the Information Theoretic Lower Bound
17.25 – 17.55	*Karsten Weihe*: Minimum Quadrangulations of Meshes
18.00 – 18.30	*Sven Schuierer*: Efficient Robot Self-Localization in Simple Polygon

Tuesday

9.00 – 10.00	*Andrew V. Goldberg*: Beyond the Flow Decomposition Barrier
10.15 – 10.55	*Alberto Apostolico*: Annotated Statistical Indices for Sequence Analysis
11.05 – 11.50	*Berthold Vöcking*: Exploiting Locality for Data Management in Systems of Limited Bandwidth
12.00 – 12.30	*Hervé Brönnimann*: Computing Exact Geometric Predicates Using Modular Arithmetic with Single Precision
16.00 – 16.35	*Hans-Peter Lenhof*: Algorithms for the Protein Docking Problem
16.45 – 17.15	*Bernard Chazelle*: Deterministic Minimum Spanning Trees
17.25 – 17.55	*David Alberts*: On a Software Library of Dynamic Graph Algorithms
18.00 – 18.30	*Stefano Leonardi*: On-line Randomized Edge-Disjoint Paths

Wednesday

9.00 – 10.00	*Martin Grötschel*: Combinatorial Optimization Problems in Telecommunication
10.15 – 10.55	*Thomas Erlebach*: Algorithms for Call Scheduling and Wavelength Allocation
11.05 – 11.50	*Michael Kaufmann*: On the Computation of Rectilinear Steiner Minimum Trees
12.00 – 12.30	*Gerth S. Brodal*: Finger Search Trees with Constant Insertion Time
13.30 – 17.30	Hiking tour
20.00 –	Open Problems Session

Gefördert durch DFG Deutsche Forschungsgemeinschaft

Thursday

9.00 – 9.40	*Pankaj Agarwal*: The Discrete 2-Center Problem
9.50 – 10.30	*David Eppstein*: Fast Hierarchical Clustering and Other Applications of Dynamic Closest Pairs
10.40 – 11.10	*Stefan Näher*: GraphWin, A LEDA Data Type for Visualizing and Manipulating Graphs
11.20 – 11.50	*Joseph Cheriyan*: Approximating Minimum-Size k-Node Connected Spanning Subgraphs
12.00 – 12.30	*Volker Kaibel*: Polyhedral Combinatorics of the Quadratic Assignment Problem
16.00 – 16.45	*Harald Lauer*: GraVis, A Dynamically Extensible Platform for Interactive Graph Algorithms
16.55 – 17.35	*Martin Farach*: Optimal Suffix Tree Construction with Large Alphabets
17.50 – 18.30	*Stefan Tschöke*: Exact and Heuristic Algorithms for the Fleet Assignment Problem
20.30 –	Wine Evening

Friday

9.00 – 9.40	*Bernd Gärtner*: Linear Programming with Exact Arithmetic
9.50 – 10.30	*Stefan Felsner*: Arrangements of Pseudolines: Higher Bruhat Orders, Triangles and k-Sets
10.40 – 11.15	*Naveen Garg*: A Polylogarithmic Approximation Algorithm for the Group Steiner Tree Problem
11.25 – 12.00	*Kurt Mehlhorn*: Shortest Path and Connected Components in Secondary Memory

Abstracts

Dynamic Algorithms for Graphs of Bounded Treewidth

Torben Hagerup

The formalism of monadic second-order (MS) logic has been very successful in unifying a large number of algorithms for graphs of bounded treewidth. We extend the elegant framework of MS logic from static problems to dynamic problems, in which queries about MS properties of a graph of bounded treewidth are interspersed with updates of vertex and edge labels. This allows us to unify and occasionally strengthen a number of scattered previous results obtained in an ad-hoc manner and to enable solutions to a wide range of additional problems to be derived automatically.

As an auxiliary result of independent interest, we dynamize a data structure of Chazelle for answering queries about sums of labels along paths in a tree with edges labeled by elements of a semigroup.

Splitting and Merging Techniques for Order Decomposable Problems, with Applications

Giuseppe F. Italiano
(joint work with Roberto Grossi)

Let S be a set whose items are sorted with respect to $d > 1$ total orders $\prec_1, \ldots, \prec_d$, and which is subject to dynamic operations, such as insertions of a single item, deletions of a single item, split and concatenate operations performed according to any chosen order $\prec_i$ $(1 \leq i \leq d)$. This generalizes to dimension $d > 1$ the notion of concatenable data structures, such as the 2–3–trees, which support splits and concatenates under a single total order. The main contribution of this paper is a general and novel technique for solving order decomposable problems on S, which yields new and efficient concatenable data structures for dimension $d > 1$. By using our technique we maintain S with the following time bounds: $\mathcal{O}(\log p)$ for the insertion or the deletion of a single item, where p is the number of items currently in S; $\mathcal{O}(p^{1-1/d})$ for splits and concatenates along any order, and for rectangular range queries. The space required is $\mathcal{O}(p)$.

We provide several applications of our technique. First, we present new multidimensional data structures implementing two–dimensional priority queues, two–dimensional search trees, and concatenable interval trees: these data structures allow us to improve many previously known results on decomposable problems under split and concatenate operations, such as membership query, minimum–weight item, range query, convex hulls and Voronoi diagrams. Second, and perhaps a bit surprisingly, we reduce some dynamic graph problems to order decomposable problems. Finally, we show how to make our technique for decomposable problems suitable for efficient external memory implementation.

On Sorting Strings in External Memory

Paolo Ferragina

(joint work with L. Arge, R. Grossi and J. Vitter)

In this talk we investigate for the first time the I/O complexity of the problem of sorting a set of strings in external memory, which is a fundamental component of many large-scale text applications. In the standard unit-cost RAM comparison model, the complexity of sorting K strings of total length N is $\Theta(K \log_2 K + N)$. By analogy, in the external memory model, where the internal memory has size M and the block transfer size is B, it would be natural to guess that the I/O complexity of sorting strings is $\Theta(K/B * \log_{M/B} K/B + N/B)$, but the known algorithms do not come even close to achieving this bound. Our results show, somewhat counterintuitively, that the I/O complexity of string sorting depends upon the length of the strings relative to the block size. We obtain improved algorithms and in several cases lower bounds that match their I/O bounds. We also develop more practical algorithms without assuming the comparison model and discuss their performance.

On Polynomial Ideals, Their Complexity, and Applications

Ernst W. Mayr

We first consider binomial ideals over the rationals in the unknowns $x_1, \ldots, x_n$. It is known that Gröbner bases for such ideals are again binomial and obey a doubly exponential degree bound. We use this bound and another doubly exponential degree bound (due to Herrman/26) for the word problem for finitely presented commutative semigroups to derive an exponential space bound for the following problems:

1. the subword reachability problem in finitely presented commutative semigroups;
2. the problem of computing the minimal elements of an equivalence class in a finitely presented commutative semigroup;
3. the problem of computing the periods of an equivalence class in a finitely presented commutative semigroup;
4. the equivalence problem for finitely presented commutative semigroups;
5. the problem of computing the reduced Gröbner bases for a binomial ideal.

We then discuss the complexity of computing normal forms and reduced Gröbner bases for general ideals (with coefficients in the rationals or finite fields). Here we show that both problems are complete for EXPSPACE, whereas Buchberger's algorithm requires (in the worst case) space doubly exponential in the input size.

Dynamic Data Structures for Realtime Management of Large Geometric Scenes

Friedhelm Meyer auf der Heide
(joint work with Matthias Fischer and Willy B. Strothmann)

We present a data structure problem which describes the requirements of a simple variant of fully dynamic walk-through animation: We assume the scene to consist of unit size ball in two or three dimensional space. The scene may be arbitrarily large and has to be stored in secondary memory. We allow a visitor to walk in the scene and a modeller to update the scene by inserting or deleting balls. The data structure has to present all balls within distance t (t is specified by the speed of the graphic hardware) to the current visitor's posn, 20 times per second. The updates also have to be executed in real time, i.e., in time independent of the size of the scene. We present a data structure that fulfills these requirements. Preliminary experiments also indicate that it is efficient in practice.

Planar Point Location Close to the Information Theoretic Lower Bound

Raimund Seidel
(joint work with U. Adamy)

We show that point location queries in a planar subdivision of size n can be answered in the worst case using at most $\log_2 n + \sqrt{8 \log_2 n} + \mathcal{O}(1)$ steps, where a step is a comparison of the query point against a line. Such a bound can even be achieved if only $\mathcal{O}(n)$ space is allowed. We also show that on a very realistic model of computation point location queries must take in the worst case at least $\log_2 n + \sqrt{2 \log_2 n} - \mathcal{O}(1)$ steps.

Added note: During the course of the workshop both the upper and the lower bound were improved. $Q(n)$, the worst case query complexity in a subdivision of size n was shown to satisfy

$$\log_2 n + 2\sqrt{\log_2 n} - (1/2)\log_2 \log_2 n - 2 \leq Q(n) \leq \log_2 n + 2\sqrt{\log_2 n} + (1/2)\log_2 \log_2 n + 2\,,$$

thus narrowing the gap between upper and lower bound to $\log \log n + \mathcal{O}(1)$.

Minimum Quadrangulations of Meshes

Karsten Weihe
(joint work with Matthias Müller-Hannemann)

In this talk, a mesh is a finite set of polygons in the three-dimensional space, which are not necessarily plane and together approximate a two-dimensional manifold (or a finite set of manifolds, which must not intersect, but may be incident at so-called folding edges). The problem is to refine such a mesh by decomposing the polygons such that the refined mesh solely consists of conformant quadrilaterals and the number of quadrilaterals is minimum. In that, conformant means that any two non-disjoint quadrilaterals either share a single corner or a whole side.

This problem is NP-hard. Here we present approximation algorithms. More specifically, constant factor 4 can be achieved in linear time; if there are no folding edges, factor 3 can be achieved in linear time and factor 28/15 in $\mathcal{O}(mn \log n)$.

Efficient Robot Self-Localization in Simple Polygon

Sven Schuierer

Localization is the process of determining an unknown starting position on a given map. It is an important problem for autonomous mobile robots and has applications in numerous areas ranging from aerial photography to autonomous vehicle exploration. In this paper we present a new fast implementation of a simple strategy for a robot to localize inside a simple polygon. The only information available to the robot is given by its visual sensors. We assume that in this way the robot has access to its local visibility polygon.

The simple strategy we consider repeatedly goes to the closest point at which the robot is able to eliminate at least one of the possible positions it may be located at. Our implementation of this strategy runs in time $\mathcal{O}(kn \log n)$ and space $\mathcal{O}(kn)$ where n is number of vertices of the polygon and k the number of possible robot positions at the beginning.

Beyond the Flow Decomposition Barrier

Andrew V. Goldberg

(joint work with Satish Rao)

The maximum flow problem is a classical optimization problem that has been intensely studied because of its numerous applications. For a network with n vertices and m arcs, $\mathcal{O}(nm)$ is a natural bound for maximum flow algorithms: The size of an explicit flow decomposition gives a matching lower bound. This lower bound does not apply if the decomposition is not needed. No previous maximum flow algorithm, however, runs in $\mathcal{O}(nm)$ time. In the unit capacity case, the decomposition size is $\mathcal{O}(m)$ and the problem can be solved in $\mathcal{O}(\min(n^{2/3}, m^{1/2})m)$ time [Karzanov, Even & Tarjan].

We present an algorithm that significantly improves upon the flow decomposition bound unless the input capacities are huge. Our algorithm runs in $\mathcal{O}(\min(n^{2/3}, m^{1/2})m \log \frac{n^2}{m} \log U)$ time, assuming the capacities are integers between 1 and U. This bound bridges the gap between the unit capacity case and the case of arbitrary integral capacities. The algorithm is based on a new approach to the maximum flow problem.

Annotated Statistical Indices for Sequence Analysis

Alberto Apostolico

(joint work with F. P. Preparata and, respectively, M. E. Bock and X. Xuyan)

We discuss the frequently encountered task of identifying words that are, by some statistical measure, typical or anomalous in the context of larger sequences.

Tables for storing the number of occurrences in a string of substrings of (or up to) a given length are routinely computed in applications. Actually, clever methods are available to compute and organize the counts of occurrences of *all* substrings of a given string. The corresponding tables take up the tree-like structure of a special kind of digital search index or *trie*.

Once the index itself is built, it makes sense to annotate its entries with the expected values and variances that may be associated with them under one or more probabilistic models. One such process of annotation is addressed in this talk.

Gefördert durch DFG Deutsche Forschungsgemeinschaft

We derive formulae expressing the expected values and variances for substring occurrences, in the hypothesis of a generative process governed by independent, identically distributed random variables. The formulae are then re-structured in a way that is more conducive to efficient computation, in the sense that the expected values and variances of all prefixes of a string can be computed optimally in overall linear time, whence the entire index annotation can be carried out in quadratic time. The heart of the construction exploits the structure of the set of periods of a string, in conjunction with a classical implement of fast string searching known as the "failure function".

Exploiting Locality for Data Management in Systems of Limited Bandwidth

Berthold Vöcking
(joint work with B. M. Maggs, F. Meyer auf der Heide, and M. Westermann)

Large parallel and distributed systems, including massively parallel processor systems (MPPs) and networks of workstations (NOWs), are usually connected by a network of limited bandwidth. In this paper, we consider the problem of placing and accessing shared objects in such systems. Our focus is on reducing the bandwidth bottleneck, i.e., the congestion, as much as possible by exploiting locality in the pattern of read and write accesses to the objects.

Most previous work in this area investigates either hashing or caching based strategies. Hashing distributes the objects uniformly among the processors giving up the locality of the application. Caching exploits locality by minimizing the distances to the accessed objects, which, however, can produce bottlenecks in the network.

We present an approach that combines hashing and caching techniques. We introduce static and dynamic strategies. For the static strategies, we assume that frequencies of read and write accesses for all processor-object pairs are given in advance. For the dynamic strategies, we assume no knowledge about the access pattern. We show that our strategies achieve optimal or close-to-optimal congestion for the most relevant classes of bandwidth limited networks, e.g., trees, meshes and clustered networks.

Computing Exact Geometric Predicates Using Modular Arithmetic with Single Precision

Hervé Brönnimann
(joint work with Ioannis Z. Emiris, Sylvain Pion and Victor Y. Pan)

We propose an efficient method that determines the sign of a multivariate polynomial expression with integer coefficients. This is a central operation on which the robustness of many geometric algorithms depends. Our method relies on modular computations, for which comparisons are usually thought to require multiprecision. Our novel technique of *recursive relaxation of the moduli* enables us to carry out sign determination and comparisons by using only floating point computations in single precision. This leads us to propose a hybrid symbolic-numeric approach to exact arithmetic. The method is highly parallelizable and is the fastest of all known multiprecision methods from a complexity point of view. As an application, we show how to compute a few geometric predicates that reduce to matrix determinants and we discuss implementation efficiency, which can be enhanced by arithmetic filters. We substantiate these claims by experimental results and comparisons to other existing approaches. Our method can be used to generate robust and efficient implementations of geometric algorithms (convex hulls, Delaunay triangulations, arrangements) and numerical computer algebra (algebraic representation of curves and points, symbolic perturbation, Sturm sequences and multivariate resultants).

Algorithms for the Protein Docking Problem

Hans-Peter Lenhof

We have developed and implemented a parallel distributed algorithm for the rigid-body protein docking problem. The algorithm is based on a new fitness function for evaluating the surface matching of a given conformation. The fitness function is defined as the weighted sum of two contact measures, the *geometric contact measure* and the *chemical contact measure*. The geometric contact measure measures the "size" of the contact area of two molecules. It is a potential function that counts the "van der Waals contacts" between the atoms of the two molecules (the algorithm does not compute the Lennard-Jones potential). The chemical contact measure is also based on the "van der Waals contacts" principle: We consider all atom pairs that have a "van der Waals" contact, but instead of adding a constant for each pair (a, b) we add a "chemical weight" that depends on the atom pair (a, b). We tested our docking algorithm with "real world" docking examples and compared the results of our docking algorithm with the results of the best known algorithms. In 32 of 35 test examples the best conformation with respect to the fitness function was an approximation of the real conformation.

Deterministic Minimum Spanning Trees

Bernard Chazelle

I will discuss a deterministic algorithm for computing a minimum spanning tree of a weighted graph. Its complexity is $O(m\alpha \log \alpha + n)$, where m, n, and α are, respectively, the number of edges, the number of vertices, and the functional inverse of Ackermann's function. No numeric assumptions are made on the edge weights.

On a Software Library of Dynamic Graph Algorithms

David Alberts

We present a project for assembling a library of implementations of dynamic graph algorithms and related tools. It aims at bridging the gap between theoretically interesting algorithms and usable software. The library is based on LEDA and written in C++. It will become available as a LEDA Extension Package (LEP).

In the talk we concentrate on some of the practical problems arising in the design and implementation of the library, particularly in keeping the consistency among several graph data structures working on the same dynamically changing graph. For more information and a list of all participating sites and people see http://www.informatik.uni-halle.de/ alberts/lepdga.html.

On-line Randomized Edge-Disjoint Paths

Stefano Leonardi

(joint work with Alberto Marchetti-Spaccamela, Alessio Presciutti and Adi Rosèn)

We consider the on-line version of the on-line edge-disjoint paths problem on trees and meshes. Previous work gave randomized on-line algorithms for these problems, and proved that they have optimal *competitive ratios*. However, these algorithms can obtain very low profit with high probability.

We investigate the question of devising for these problems on-line competitive algorithms that also guarantee a "good" solution with "good" probability. We give a new family of randomized algorithms with optimal (up to constant factors) competitive ratios, and provably "good" probability to get a profit close to the expectation. We complement these results by providing bounds on the probability, of any optimally-competitive randomized on-line algorithm for the problems we consider, to get a profit close to the expectation.

This work is also a first study of how well can the benefit of a randomized on-line algorithm be concentrated around its expectation.

Combinatorial Optimization Problems in Telecommunication

Martin Grötschel

Deregulation has resulted in a worldwide boom in the telecommunication industry. Competition leads to strict cost and quality management, which, in turn, offers interesting perspectives for mathematics.

In this talk I describe several fundamental operational and design problems coming up in telecommunication that can be modeled mathematically and yield large-scale combinatorial optimization problems. I focus on the design of low-cost networks that survive certain failure situations and on versions of the frequency assignment problem for mobile phone systems. I explain the mathematical models, the theory developed for their solution, and I report on computational results with data from practice.

Algorithms for Call Scheduling and Wavelength Allocation

Thomas Erlebach

(joint work with K. Jansen, C. Kaklamanis and P. Persiano)

Call scheduling means assigning starting times and paths to connection requests (calls) in a communication network. Each call specifies its bandwidth requirement and its duration. The sum of bandwidth requirements of simultaneously active calls using the same link must not exceed the capacity of that link. The goal is to complete all calls within the shortest possible time, i.e., to minimize the makespan. In the case of unit bandwidths and unit durations, this problem is equivalent to wavelength allocation in all-optical networks, where each call must be assigned a path and a wavelength such that calls using the same link are assigned different wavelengths. A minimum makespan schedule corresponds to a wavelength allocation with a minimum number of different wavelengths.

After a short survey of known results regarding off-line and on-line approximation algorithms for call scheduling and wavelength allocation (both problems are $\mathcal{NP}$-hard in most settings), we will focus on a wavelength allocation algorithm that assigns wavelengths to directed calls in trees using at most $\frac{5}{3}L$ wavelengths, where L is the maximum link load and a lower bound on the optimum number of wavelengths. The most important component of this algorithm is a subroutine for edge-coloring a bipartite graph in which the colors on certain edges have been fixed beforehand. We show how to modify the original presentation of the algorithm in order to solve this constrained bipartite edge-coloring problem in the same time bounds as the unconstrained version.

On the Computation of Rectilinear Steiner Minimum Trees

Michael Kaufmann
(joint work with Uli Fößmeier)

In this talk, we report on our experiments for the computation of rectilinear Steiner minimum trees.

After sketching the two main approaches, dynamic programming and branch and bound, we consider the concept of full components as the input of the two algorithms, and demonstrate the importance of a considerable reduction of the number of full components.

We discuss several heuristics for the reduction and try some predictions for the expected running times for different problem sizes.

Finger Search Trees with Constant Insertion Time

Gerth S. Brodal

We consider the problem of implementing finger search trees on the pointer machine, i.e., how to maintain a sorted list such that searching for an element x, starting the search at any arbitrary element f in the list, only requires time logarithmic in the distance between x and f in the list.

We present the first pointer based implementation of finger search trees allowing new elements to be inserted at any arbitrary position in the list in worst case constant time. Previously the best known insertion time on the pointer machine was $\mathcal{O}(\log^* n)$, where n is the total length of the list. On a unit-cost RAM a constant insertion time has been achieved by Dietz and Raman by using standard techniques of packing small problem sizes into a constant number of machine words.

Deletion of a list element is supported in $\mathcal{O}(\log^* n)$ time, which matches the previous best bounds. Our data structure requires linear space.

The Discrete 2-Center Problem

Pankaj Agarwal

Let P be a set of points in the plane. We wish to cover P by two congruent disks of the smallest possible radius and centered at two points of P. We present an $\mathcal{O}(n^{4/3}\log^5 n)$-time algorithm for this problem. This is the first subquadratic algorithm for this problem.

Fast Hierarchical Clustering and Other Applications of Dynamic Closest Pairs

David Eppstein
(some of the work is joint with Jeff Erickson)

We develop data structures for dynamic closest pair problems with arbitrary (not necessarily geometric) distance functions, based on a technique previously used by the author for Euclidean closest pairs. We show how to insert and delete objects from an n-object set, maintaining the closest pair, in $O(n \log^2 n)$ time per update and $O(n)$ space. With quadratic space, we can instead use a quadtree-like structure to achieve an optimal time bound, $O(n)$ per update. We apply these data structures to hierarchical clustering, greedy matching, TSP heuristics, collision detection, and straight skeleton construction, and discuss other potential applications in machine learning, Gröbner bases, and local improvement algorithms for partition and placement problems. Experiments show our new methods to be faster in practice than previously used heuristics.

GraphWin, A LEDA Data Type for Visualizing and Manipulating Graphs

Stefan Näher

The main goal of the new LEDA data type *GraphWin* is to offer a simple and efficient interactive tool for graph visualization and manipulation within LEDA's comfortable graph environment. For this purpose GraphWin combines the two types *graph* and *window* and forms a bridge between the various graph data types and algorithms on one side and the graphics interface of LEDA on the other side. The implementation of GraphWin is based on an *observer design pattern* for separating the graphical representation from the underlying graph data structure. GraphWin can easily be used in programs for constructing, displaying and manipulating graphs and for animating and debugging graph algorithms. We discuss some of the most important features of GraphWin.

- Simple User Interface
  The user interface of GraphWin was designed to be as simple and intuitive as possible. For instance, the user can easily create or move nodes and edges with the left mouse button and delete nodes and edges with the right button. An optional and customizable set of menues and buttons at the top of the window gives access to graph generators, modifiers, basic algorithms, embeddings, setup and file dialogs.
- Generators, Modifiers, and Tests
  Graphwin offers a collection of graph generators, modifiers and tests. The generators include functions for constructing random, planar, complete, bipartite, grid graphs, ... Graph modifiers change existing graphs (e.g., by removing or adding a certain set of edges) to fit in one of these categories.
- Basic Algorithms and Embeddings
  The standard menu includes a choice of fundamental algorithms (topological sorting, depth first search, breadth first search, connected components, transitive closure, st-numbering, ...) and basic embedding algorithms.

- Parameterized Graphs
  Graphwin can display and manipulate data associated with the nodes and edges of LEDA's parameterized graph type *GRAPH<vtype, etype>*. When a graph window is opened for a graph G, say of type *GRAPH<string, double>*, it can label every node v with the associated string $G[v]$ and every edge e with the associated floating point number $G[e]$.
- Customization and Extensibility
  Most of the actions of GraphWin can be customized by defining call-back functions. So the user can define what happens if a node or edge is selected, moved, or deleted. This is very useful in the case that an additional data structure has to be maintained. For example, in the crossing reduction application, the different levels of the hierarchy are implemented by arrays. When dragging a node over another node (during a GraphWin edit operation) its position in the corresponding array has to be changed. It is also possible to restrict the set of possible modifications.

Approximating Minimum-Size k-Node Connected Spanning Subgraphs

Joseph Cheriyan
(joint work with Ramki Thurimella)

An approximation algorithm is given for the NP-hard problem of finding a k-node connected spanning subgraph G' of a given graph $G = (V, E)$ such that G' has the minimum number of edges. The algorithm achieves an approximation guarantee of $1 + \frac{1}{k}$ and runs in time $\mathcal{O}(k|E|^2)$.

Polyhedral Combinatorics of the Quadratic Assignment Problem

Volker Kaibel

Many classical $\mathcal{NP}$-hard combinatorial optimization problems, like, e.g., the *traveling salesman problem*, the *stable set problem*, or the *max cut problem*, have been investigated from polyhedral points of view quite extensively. The structural insight gained this way lead to algorithms that can often solve instances of these problems very "efficiently". For example "real world" instances of the traveling salesman problem with several thousands cities can be often solved to optimality within several hours of CPU time. Although the *quadratic assignment problem (QAP)* is one of the most famous $\mathcal{NP}$-hard combinatorial optimization problems, only a few results concerning the polytope that is naturally associated to the problem have been known so far.

We give an overview of new polyhedral results on the QAP that we obtained exploiting different kinds of projection based techniques. These results include answers to the basic questions for the dimensions, the affine hulls, and the "trivial facets" not only of the "natural" QAP-polytope, but also of several variants of it, which are associated especially to "symmetric" or "sparse" instances, or to instances with "less objects than locations". Moreover, we present the first large class of facets for these polytopes, and show that the corresponding inequalities can be utilized for cutting plane procedures very effectively.

GraVis, A Dynamically Extensible Platform for Interactive Graph Algorithms

Harald Lauer

Interactive layout techniques are gaining in significance both in research and practical applications. GraVis offers the functionality necessary for the effective development and employment of dynamic layout algorithms. GraVis is also a complete and powerful interactive graph visualization system, supporting the integration into practical and research applications. These features are derived from an object-oriented design, realizing dynamic extensibility and the flexibility to integrate new concepts like multi-user/groupware support.

We will present the design of GraVis, its features and describe the realization of the extension mechanism. Finally, a demonstration of the current version of GraVis will give an impression of its intuitive user interface, as well as the overall efficiency and usability.

Optimal Suffix Tree Construction with Large Alphabets

Martin Farach

The suffix tree of a string is the fundamental data structure of combinatorial pattern matching. Weiner, who introduced the data structure, gave an $\mathcal{O}(n)$ time algorithm algorithm for building the suffix tree of an n character string drawn from a constant size alphabet. In the comparison model, there is a trivial $\Omega(n \log n)$ time lower bound based on sorting, and Weiner's algorithm matches this bound trivially. Since Weiner's paper, the main open question has been how to deal with integer alphabets. There is no super-linear lower bound, and the fastest known algorithm was the $\mathcal{O}(n \log n)$ time comparison based algorithm. We settle this open problem by closing the gap: we build suffix trees in linear time for integer alphabet.

Exact and Heuristic Algorithms for the Fleet Assignment Problem

Stefan Tschöke

The fleet assignment problem is one of a series of optimization problems occuring in airline industry operations, beginning with market modelling and flight scheduling followed by fleet assignment, crew pairing and crew rostering. The fleet assignment has usually to be done six month before day of operation and is planned on a weekly basis. It is not unusual that a large internationally operating airline offers more than 100.000 possible itineraries on 10.000 legs a week working with more than 200 aircrafts of 20 different subtypes (fleet) on 150 airports.

It can be shown that a fleet assignment with more than two different given types of aircraft (fleet) is NP-complete. There are a lot of hard and soft constraints. A fleet assignment is only valid if the aircrafts are assigned on round-trips. Additional restrictions are the passenger capacity, range of the aircrafts, maintenance periods, crew restrictions, take-off and landing time-slots etc.

Most of the known approaches of the fleet assignment are based on integer programming models of the problem and are using LP relaxations and column generation methods. To reduce the size of the huge LPs or IPs fleetings are often solved only on daily basis. These approaches usually minimize the operational costs but are not maximizing the profit. We developed such an exact approach for solving smaller instances and and for upper bounds on the profit.

Our heuristic approach has two phases, firstly generating rotational elements and secondly assigning an aircraft type to every rotational element guaranteeing rotations for the whole planning period. We maximize the profit instead of minimizing costs. The difference is that number of passengers on a certain leg is not independent of the passenger capacity, i.e., if passengers are rejected (the number of estimated passengers is higher than the aircraft capacity) this has impact on the revenue of other legs because passengers will look for alternative flights (spill-off and recover model). We tested our algorithm on real world data (8000 legs, 250 aircrafts, 200.000 itinararies) provided by a large german airline.

To be part of an interactive decision support system, the fleet assignment has to be solved not only once in six months but many times. Therefore we also parallelized our algorithm and could reduce the computational times significantly.

Linear Programming with Exact Arithmetic

Bernd Gärtner

We describe a new exact-arithmetic approach to linear programming when the number of variables n is much larger than the number of constraints m (or vice versa). The algorithm is an implementation of the simplex method which combines exact (multiple precision) arithmetic with inexact (floating point) arithmetic, where the number of exact arithmetic operations is small and usually bounded by a function of $\min(n, m)$. Combining this with a "partial pricing" scheme (based on a result by Clarkson which is particularly tuned for the problems under consideration, we obtain a correct and practically efficient C++ algorithm that even competes with the inexact state-of-the-art solver CPLEX for small values of $\min(n, m)$.

Arrangements of Pseudolines: Higher Bruhat Orders, Triangles and k-Sets

Stefan Felsner

(joint work with Helmut Alt, Klaus Kriegel and Helmut Weil)

In computational geometry a good understanding of arrangements often aids discovery and analysis of efficient algorithms. We provide a combinatorial framework for the study of simple Euclidean arrangements of pseudolines. They correspond bijectively to functions $a : \binom{[n]}{3} \rightarrow \{+,-\}$ obeying a monotonicity condition on every 4-element subset of $[n]$. These functions are the elements of the higher Bruhat order $B(n, 2)$. The Bruhat order $B(n, k)$ represents a class of arrangements in $\mathbf{R}^k$. The minimum degree in $B(n, 2)$ is the minimal number of triangles in simple arrangements. We show that simple arrangements have at least $n-2$ triangles while non-simple arrangements can have as few as $2n/3$ triangles but not less. Finally, we give a surprising new proof for the old $n^{3/2}$ bound for the complexity of the middle level of an arrangement.

Gefördert durch DFG Deutsche Forschungsgemeinschaft

A Polylogarithmic Approximation Algorithm for the Group Steiner Tree Problem

Naveen Garg
(joint work with Goran Konjevod and R. Ravi)

The group Steiner tree problem is a generalization of the Steiner tree problem where we are given several subsets (groups) of vertices in a weighted graph, and the goal is to find a minimum-weight connected subgraph containing at least one vertex from each group. The problem was introduced by Reich and Widmayer and finds applications in VLSI design. The group Steiner tree problem generalizes the set cover problem, and hence a logarithmic approximation factor is the best possible unless P=NP.

We give a randomized $\mathcal{O}(\log^4 n)$-approximation algorithm for the group Steiner tree problem. The best previous performance guarantee was $(1 + \frac{\ln k}{2})\sqrt{k}$, where k is the number of groups (Bateman, Helvig, Robins and Zelikovsky) We use the result of Bartal on probabilistic approximation of finite metric spaces by tree metrics to reduce the problem to one in a tree metric. To find a solution on a tree, we use a variant of randomized rounding.

Shortest Path and Connected Components in Secondary Memory

Kurt Mehlhorn
(joint work with Andreas Crauser and Ulrich Meyer)

In this report we investigate the I/O complexity of computing the single source shortest path on a input graph with non-negative edge weights and finding the connected components of a undirected graph. We present an algorithm that uses $\mathcal{O}(\frac{N}{D} + \frac{m}{DB} \log_{S/B} \frac{m}{B})$ I/O with high probability for a large class of random graphs, where n, m are the number of nodes respectively the number of edges of the graph, S is the size of available internal memory, B is the size of block transfer and D is the number of independent parallel disks; D is constrained to be $\mathcal{O}(\sqrt{n/d})$. For the problem of computing connected components of undirected graphs we introduce a randomized algorithm that computes the connected components in $\mathcal{O}(\frac{m}{B} \log_{S/B} \frac{m}{B})$ expected I/O. The proposed algorithm can be used to compute the biconnected components of an undirected graph with the same number of I/Os.

Open problems

1. **Nonoverlapping occurrences of substrings** (*Alberto Apostolico*).

   It is trivial to count the number of occurrences of each substring of a given string: just build a suffix tree, and label each node with the number of leaves descending from it. Each substring corresponds to a path in the suffix tree starting at the root and ending either at a tree node or along an edge; if the path ends at a node the number of occurrences is just the label of that node, and otherwise it is the label found by continuing the path until it reaches a node.

   However, the problem becomes more interesting if one wishes to count the maximum number of *nonoverlapping* occurrences of a substring. At the workshop, Apostolico presented an algorithm that similarly places labels on a suffix tree, so that this number of nonoverlapping occurrences can again be found by following a path from the root, continuing the path until it reaches a node, and returning the label at the node. However, this can require the insertion of additional degree-one nodes into the suffix tree; for instance, "aabaaba" has two non-overlapping occurrences of "aab" but only one occurrence of "aaba", despite the fact that both substrings correspond to paths ending on the same edge of the suffix tree. In Apostolico's data structure, this edge needs to be divided by a new node at the position corresponding to string "aab", with label two.

   The first question is, how quickly can this modified data structure be constructed? Apostolico's algorithm takes time $\mathcal{O}(n \log^2 n)$, using many complicated data structures, but perhaps this can be improved.

   Second, how many extra nodes are required in this data structure? A closely related question is how many distinct squares there can be in a string – a *square* is just a word of the form ww for some word w. Each extra node comes from a square, but some squares may not give rise to extra nodes (either because the corresponding node already exists in the original suffix tree, or because there is a maximum cardinality set of copies of w that does not include both copies in the square). One can find strings with many squares using the *Fibonacci words* (start with two single-character strings then repeatedly concatenate the previous two strings) but most of the squares in these words are not distinct strings.

2. **Space-efficient text indexing** (*Paolo Ferragina*).

   With a suffix tree, one can find all occurrences of pattern p in an n-character text in time $\mathcal{O}(|p| + k)$, with space (measured as the amount needed in excess of the pattern and text themselves) $\mathcal{O}(n)$. With the Knuth-Morris-Pratt algorithm, one can find all occurrences in time $\mathcal{O}(n + k)$, with space $\mathcal{O}(|p|)$. Is there a data structure which is both space- and time-efficient (search time $\mathcal{O}(|p| + k)$, space $o(n)$)?

   Martin Farach noted the existence of an algorithm with search time $\mathcal{O}(|p| \log n)$ and space $\mathcal{O}(n/ \log n)$. However, the time-space product remains the same.

3. Batched LCA queries on a pointer machine (*Martin Farach*).

It is known that one can preprocess a tree in $\mathcal{O}(n)$ time so that lowest common ancestor (LCA) queries can be performed in constant time each, by any of several methods; however these methods are heavily dependant on the ability to perform random memory access and can not be used on pointer machines. In fact, there is an $\Omega(\log \log n)$ lower bound on the time per LCA query in the pointer machine model. However, even on a pointer machine, for large enough values of k, one can perform a batch of k queries in $\mathcal{O}(k)$ time; in particular this is true if $k = \Omega(n^2)$. What is the smallest k for which this is true?

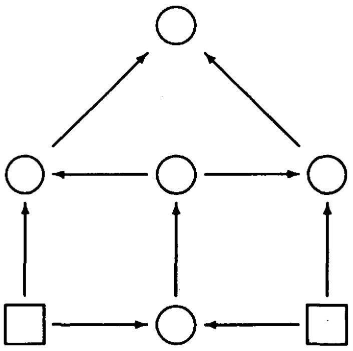
4. Batched dynamic graph algorithms (*Pino Italiano*).

Most dynamic graph algorithms we know of maintain some property P on a graph G subject to individual updates. What about batched updates? E.g., for the minimum spanning tree we can handle batch size 1 in $\mathcal{O}(n^{1/3} \log n)$ time per update [Henzinger and King, ICALP 1997] or batch size m in constant or close to constant time per update (just use the static algorithm); can we say anything nontrivial about intermediate batch sizes?

5. Steiner branchings (*Naveen Garg*).

Suppose we have a digraph G rooted at vertex r, such that all other vertices have at least k edge-disjoint paths to r. Then a famous theorem of Edmonds states that we can find k edge-disjoint branchings in G, spanning the vertices of G and rooted at r.

But now, suppose this requirement on the existence of k disjoint paths does not hold for all vertices of G, but only for certain designated terminals. If the number of terminals is one, we can still find k edge-disjoint "Steiner branchings" (paths in this case). But if there are two terminals, such branchings might not exist: the squares in the following figure can not be connected by two branchings to the topmost vertex, even though they each have two disjoint paths to that vertex.



Conjecture: In any digraph with designated terminals, such that each terminal has k edge-disjoint paths to a common root r, there exist k branchings connecting the terminals to r, such that each edge of G is used in $\mathcal{O}(\log n)$ branchings.

A constructive proof of this conjecture would improve by a logarithmic factor the $\mathcal{O}(\log^4 n)$ approximation algorithm for group Steiner trees presented by Garg at the workshop.

6. **Connectivity augmentation** (*Joseph Cheriyan*).

   Suppose we are given a k-vertex-connected graph and wish to add the minimum number of edges to make it $(k+1)$-vertex-connected. Is this in P? Is it NP-complete? NP-completeness is not even known for augmenting k-connected graphs to be $(k+\ell)$-connected, even when k and ℓ may be part of the input.

7. **Tree scheduling with random durations** (*Ernst Mayr*).

   Suppose we have a set of tasks to be performed, with precedence constraints in the form of a tree directed towards the root, and p identical processors on which to perform the tasks. If all tasks have the same runtime, there is a simple solution which finds an optimal schedule: sort all tasks by their distance to the root of the tree, and when each processor becomes free assign it the next task in the sorted order.

   But, what if we change the model so that task execution times are independent identically distributed exponential random variables? A scheduling algorithm in this context means a well defined procedure for selecting the next task to assign to a free processor; it is never worthwhile to delay assigning a task to a processor. The optimal strategy would be one that minimizes the expected total runtime of the set of tasks. If $p = 1$ or $p = 2$ the sorting algorithm still works, but not for $p = 3$. Question: how can we solve this scheduling problem for $p \geq 3$?

   Andrew Goldberg asked whether it is easier if the tree has constant degree. Apparently the problem remains open for that case.

8. **Hard examples for matching** (*Andrew Goldberg*).

   This problem comes from Goldberg's experiments on bipartite matching, with different variations of augmenting path, augment-relabel, and push-relabel algorithms. He couldn't come up with problems that match the known worst-case performance bounds on push-relabel methods, without controlling the order in which the program scanned vertex adjacency lists. Do these worst case bounds still hold for graphs with randomly permuted adjacency lists?

9. **Graph partitioning** (*Berthold Vöcking*).

   Define the *flux* of a graph G to be the minimum (over all vertex partitions $(A, V(G)-A)$) of the number of edges crossing the partition divided by the number of vertices in A. (There was some argument about whether A might or might not be required to be the smaller of the two subsets in the partition.) One can form a hierarchical partition of G by repeatedly choosing flux-minimizing cuts; label each node v of this

Gefördert durch
DFG Deutsche Forschungsgemeinschaft

hierarchy by the flux α_v of its cut, and by another number β_v. This β_v is defined as the larger of two numbers, each coming from one of the two subsets in the partition at v: the subset A_v gives the number $b/|A_v|$ where b is the number of edges leaving A_v in the whole graph (not just in the subgraph which we partitioned at v).

If we are given a labeled hierarchical partition T like this, we can achieve competitive ratio $\max_{v \in T} \beta_v/\alpha_v \log n$ for the optical network path coloring problem Vöcking talked about at the workshop. This problem also has applications in Gaussian elimination.

However, choosing non-optimal flux cuts may produce a better competitive ratio, by giving better values of b_v. If we always choose cuts with flux at most $(1+\epsilon)$ times the minimum flux, then the same competitive ratio applies, multiplied by a $(1+\epsilon)$ factor. How small can we make this competitive ratio?

10. Parametric MST and combinations of halfspaces (*David Eppstein*).

Suppose we are given a graph with edge weights that are linearly varying functions of some parameter. For different parameter values, this graph will have different minimum spanning trees. Known bounds on the number of distinct trees arising in this way are $\Omega(m\alpha(n))$ [Eppstein, STOC '95] and $O(mn^{1/3})$ [Dey, FOCS '97]. One can compute the set of all such trees in time $O(mn \log n)$ [Fernández-Baca et al., SWAT '96]. We'd like to tighten these bounds.

One approach to this problem is through a form of *constructive solid geometry* (a method of forming shapes by unions and intersections of simpler shapes). Start with n shapes, each of which is the halfspace below some line, and then repeatedly replace pairs of shapes by single shapes, either their union or their intersection. All shapes formed in this way will be bounded by monotone paths through the arrangement of the initial lines; the question is how many vertices such a path can have. A solution with k vertices could be converted to a linearly weighted two-terminal series parallel graph with $\Omega(k)$ distinct minimum spanning trees, so a solution with $k = \omega(n\alpha(n))$ would improve the known parametric MST bounds.

11. Planar domination (*Alberto Apostolico*).

Given n points in a plane, point (a,b) *dominates* point (c,d) if $a \geq c$ and $b \geq d$. This defines a partial order. By Dilworth's theorem, the longest chain in this partial order has length equal to the minimum number of antichains in any partition of the points into antichains. It is known how to compute such an antichain decomposition sequentially in optimal $O(n \log n)$ time. What about in parallel, with n processors on a PRAM? (It can be done with n^2 processors in polylog time.)

12. Planar domination continued (*Pankaj Agarwal*).

By Dilworth's theorem again, one can find either a chain or an antichain of length $\Omega(\sqrt{n})$, and repeatedly removing such a chain or antichain produces a decomposition of the point set into $O(\sqrt{n})$ chains and antichains. How quickly can we compute such a chain? The known bound is $O(n^{3/2})$, which can be achieved by repeatedly removing the longest chain or antichain.

13. Fat-lattice polytopes (*Raimund Seidel*).

If we are given a d-dimensional polytope P, let $f_i(P)$ denote the number of i-dimensional faces of P, so $f_0(P)$ is the number of vertices and $f_{d-1}(P)$ is the number of facets. By convention we let $f_{-1}(P) = f_d(P) = 1$. Further define $\bar{f}(P) = \sum f_i(P)$, so, e.g., for a d-simplex $\bar{f}(P) = 2^{d+1}$. Simple polytopes have $\bar{f}(P) = O(f_0(P))$ and simplicial polytopes have $\bar{f}(P) = O(f_{d-1}(P))$. For what d can there be a family of d-polytopes with $\bar{f}(P) = \omega(f_0(P) + f_{d-1}(P))$?

There are examples for $d = 5$ with $\bar{f}(P) = \Omega((f_0(P) + f_{d-1}(P))^2)$, and in higher dimensions with $\bar{f}(P) = \Omega((f_0(P) + f_{d-1}(P))^{\lfloor (d+1)/3 \rfloor})$ (unpublished work by Amenta, Bern, Eppstein, and Erickson). For $d = 3$ no such fat-lattice polytope can exist (a consequence of Euler's formula $\sum(-1)^i f_i(P) = 0$). So, the remaining open question is whether fat-lattice polytopes exist for $d = 4$.

The motivation for this is in the analysis of convex hull and facet enumeration algorithms. We'd like to have time bounds for such algorithms that are proportional to the total input and output size, $f_0(P) + f_{d-1}(P)$, but known algorithms take time proportional to $\bar{f}(P)$.

14. Branching-lattice polytopes (*Hervé Brönnimann*).

Given a polytope P, define a directed acyclic graph, the vertices of which are faces of P, with an arc between the vertices representing an i-dimensional face and an $(i+1)$-dimensional face if those two faces are incident. For consistency with the notation $f_i(P)$ above, include a single "(-1)-dimensional face" incident to all vertices and a single d-dimensional face (the polytope itself) incident to all facets. Let $\ell_i(P)$ denote the number of paths of length i in this graph, so that $\ell_1(P)$ is the number of arcs and $\ell_{d+1}(P)$ is the number of maximal paths. In a d-dimensional simplex, $\ell_1(P) = (d+1)2^d$ and $\ell_{d+1}(P) = (d+1)!$. All $\ell_i(P)$ are bounded by $O(f_0(P)^{\lfloor d/2 \rfloor})$. Simple polytopes have $\ell_i(P) = O(f_0(P))$, and simplicial polytopes have $\ell_i(P) = O(f_{d-1}(P))$. For $d \le 4$, $\ell_1(P) = O(\ell_{d+1}(P))$. Does there exist a dimension d and a family of d-dimensional polytopes for which $\ell_{d+1}(P) = \omega(\ell_1(P))$?

Author of the report: Gerth Stølting Brodal

List of participants

Agarwal, Pankaj	pankaj@cs.duke.edu
Alberts, David	alberts@informatik.uni-halle.de
Apostolico, Alberto	axa@cs.purdue.edu
Arora, Sanjeev	arora@cs.princeton.edu
Brönnimann, Hervé	Herve.Bronnimann@sophia.inria.fr
Brodal, Gerth S.	brodal@mpi-sb.mpg.de
Chazelle, Bernard	chazelle@cs.princeton.edu
Cheriyan, Joseph	jcheriyan@dragon.uwaterloo.ca
Eppstein, David	eppstein@ics.uci.edu
Erlebach, Thomas	erlebach@informatik.tu-muenchen.de
Farach, Martin	farach@lucent.com
Felsner, Stefan	felsner@inf.fu-berlin.de
Ferragina, Paolo	ferragin@di.unipi.it
Gärtner, Bernd	gaertner@inf.ethz.ch
Garg, Naveen	naveen@mpi-sb.mpg.de
Goldberg, Andrew V.	avg@research.nj.nec.com
Grötschel, Martin	groetschel@zib.de
Hagerup, Torben	torben@mpi-sb.mpg.de
Italiano, Giuseppe F.	italiano@dsi.unive.it
Kaibel, Volker	kaibel@informatik.uni-koeln.de
Kaufmann, Michael	mk@informatik.uni-tuebingen.de
Lauer, Harald	lauer@informatik.uni-tuebingen.de
Lenhof, Hans-Peter	len@mpi-sb.mpg.de
Leonardi, Stafano	leon@dis.uniroma1.it
Mayr, Ernst W.	mayr@informatik.tu-muenchen.de
Mehlhorn, Kurt	mehlhorn@mpi-sb.mpg.de
Meyer auf der Heide, Friedhelm	fmadh@uni-paderborn.de
Näher, Stefan	naeher@informatik.uni-halle.de
Schuierer, Sven	schuiere@informatik.uni-freiburg.de
Seidel, Raimund	seidel@cs.uni-sb.de
Tschöke, Stefan	sts@uni-paderborn.de
Vöcking, Berthold	voecking@uni-paderborn.de
Weihe, Karsten	karsten.weihe@uni-konstanz.de

Tagungsteilnehmer

Prof.Dr. Pankaj Agarwal
Department of Computer Science
Duke University
Box 90129

Durham , NC 27708-0129
USA

Dr. David Alberts
Fachb. Mathematik u. Informatik
Martin-Luther-Universität
Halle-Wittenberg
Kurt-Mothes-Str. 1

06120 Halle

Prof.Dr. Alberto Apostolico
Dipartimento di Elettronica e
Informatica
Universita di Padova
Via Gradenigo 6/A

I-35131 Padova

Dr. Sanjeev Arora
Department of Computer Science
Princeton University

Princeton , NJ 08544
USA

Prof.Dr. Gerth Brodal
Max-Planck-Institut für Informatik
Geb. 46
Im Stadtwald

66123 Saarbrücken

Prof.Dr. Herve Brönnimann
INRIA/project PRISME
BP 93
2004 Route des Lucioles

F-06902 Sophia Antipolis Cedex

Prof.Dr. Bernard Chazelle
Department of Computer Science
Princeton University

Princeton , NJ 08544
USA

Prof.Dr. Joseph Cheriyan
Department of Combinatorics and
Optimization
University of Waterloo

Waterloo , Ont. N2L 3G1
CANADA

Prof.Dr. David Eppstein
Dept. of Information and Computer
Sciences
University of California at Irvine

Irvine , CA 92697
USA

Thomas Erlebach
Institut für Informatik
TU München

80290 München

Gefördert durch
DFG Deutsche Forschungsgemeinschaft

Prof.Dr. Martin Farach
Department of Computer Sciences
Rutgers University

Piscataway , NJ 08855
USA

Dr. Stefan Felsner
Institut für Informatik (WE 3)
Freie Universität Berlin
Takustr. 9

14195 Berlin

Prof.Dr. Paolo Ferragina
Max-Planck-Institut für Informatik
Geb. 46
Im Stadtwald

66123 Saarbrücken

Prof.Dr. Naveen Garg
Max-Planck-Institut f. Informatik
Im Stadtwald

66123 Saarbrücken

Bernd Gärtner
Institut für Informatik
ETH-Zürich
ETH-Zentrum

CH-8092 Zürich

Prof.Dr. Andrew V. Goldberg
NEC Research Institute
4 Independence Way

Princeton , NJ 08540
USA

Prof.Dr. Martin Grötschel
Konrad-Zuse-Zentrum für
Informationstechnik Berlin (ZIB)
Takustr. 7

14195 Berlin

Dr. Torben Hagerup
Max-Planck-Institut für Informatik
Geb. 46
Im Stadtwald

66123 Saarbrücken

Prof.Dr. Giuseppe F. Italiano
Dipt. di Matematica Applicata e
Informatica
Universita "Ca Foscari" di Venezia
via Torino 155

I-30173 Venezia Mestre

Dr. Volker Kaibel
Institut für Informatik
Universität zu Köln
Pohligstr. 1

50969 Köln

Dr. Michael Kaufmann
Fakultät für Informatik
Universität Tübingen
Sand 13

72076 Tübingen

Harald Lauer
WSI
Universität Tübingen
Sand 13

72076 Tübingen

Dr. Hans-Peter Lenhof
Max-Planck-Institut für Informatik
Geb. 46
Im Stadtwald

66123 Saarbrücken

Prof.Dr. Stefano Leonardi
Max-Planck-Institut für Informatik
Geb. 46
Im Stadtwald

66123 Saarbrücken

Prof.Dr. Ernst W. Mayr
Institut für Informatik
TU München

80290 München

Prof.Dr. Kurt Mehlhorn
Max-Planck-Institut für Informatik
Geb. 46
Im Stadtwald

66123 Saarbrücken

Prof.Dr. Friedhelm Meyer auf der Heide
Heinz-Nixdorf Institut &
FB Mathematik - Informatik
Universität Paderborn
Fürstenallee 11

33102 Paderborn

Dr. Stefan Näher
Institut für Numerische Mathematik
Fachbereich Mathematik u.Informatik
Universität Halle-Wittenberg
Weinbergweg 17

06120 Halle

Dr. Sven Schuierer
Institut für Informatik
Universität Freiburg
Am Flughafen 17

79110 Freiburg

Prof.Dr. Raimund Seidel
Fachbereich Informatik
Universität Saarbrücken

66041 Saarbrücken

Stefan Tschöke
Heinz-Nixdorf Institut &
FB Mathematik - Informatik
Universität Paderborn
Fürstenallee 11

33102 Paderborn

Berthold Vöcking
FB 17: Mathematik/Informatik
Universität Paderborn
Warburger Str. 100

33098 Paderborn

Karsten Weihe
Fakultät für Mathematik und Informatik
Universität Konstanz
D 188

78457 Konstanz

Gefördert durch
DFG Deutsche Forschungsgemeinschaft