Dagstuhl Seminar

No. 00371, Report No. 285

10.09 - 15.09.2000

Experimental Algorithmics

Rudolf Fleischer (Hong Kong U. of Science and Technology)

Bernard Moret (U. of New Mexico, Albuquerque, USA)

Erik Meineche Schmidt (Aarhus U., Denmark)

Contents

1 Summary

In recent years, many areas of theoretical computer science have experienced a shift to more applied research. Clear evidence of this fact is the surge of experimental papers in areas which used to be completely satisfied with a thorough theoretical analysis of the problems and algorithms. One reason for this is that people have learned that a purely theoretical analysis of an algorithm is often insufficient for practical purposes because

- Many practical problems belong to "difficult" problem classes (NP-complete problems or worse), so asymptotic worst-case bounds do not give a satisfactory answer to the question whether an algorithm is "useful" or not.
- More often then not, practical algorithms build on various heuristics where no tight bounds even exist.

So people have started to implement and test their algorithms besides of doing the usual theoretical analysis. Unfortunately, it is often not clear what these experiments actually tell us. Is an algorithm good just because it seems to behave well on random instances or on some benchmark test set? There is no sound basis for experimental algorithmics which could give us answers to questions like

- What are relevant experiments?
- What can be learned from experiments?
- What is a good benchmark test set?

and finally

- What is a good experimental paper?

The aim of this seminar was to bring together two groups, theoreticians and practitioners, to discuss these problems and start establishing a methodology of experimental algorithmics. In all, 45 researchers with affiliations in Austria, Canada, Denmark, France, Germany, Great Britain, Greece, Hong Kong, Italy, the Netherlands, Spain, and the USA participated in the meeting. Three invited keynote speakers, Jon Bentley, David S. Johnson, and Kurt Mehlhorn, gave one-hour position talks. The remaining 26 presentations given by participants of the meeting covered a wide range of topics in experimental algorithmics. The abstracts of most of these presentations are contained in this seminar report. Two evenings were reserved for discussions on specific topics, a summary of the outcome of the discussions is included at the end of this report.
As usual, Schloß Dagstuhl proved to be an excellent place to hold a great meeting, so we would not only like to thank the participants of the seminar for making

this a very successful event but also the Dagstuhl staff for providing a friendly and stimulating working environment.

Rudolf Fleischer
Bernard Moret
Erik Meineche Schmidt

2 Program

Monday, September 11

9:00 Rudolf Fleischer: Introduction

9:30 Philippe Flajolet: The unreasonable efficiency of asymptotic analysis

10:00 Jordi Petit i Silvestre: Random geometric graphs and layout problems

10:30 **Coffee Break**

11:00 David Johnson: Pet peeves and pitfalls in the experimental analysis of algorithms

12:15 **Lunch**

14:30 Erik Demaine and Ian Munro: Experience in adaptive set intersections

15:00 Sven Krumke: Simulation studies for online problems

15:30 **Coffee Break**

16:00 Peter Sanders: Teaching algorithm engineering

16:30 Walter Tichy: An empirical analysis of Delta algorithms

18:00 **Dinner**

Tuesday, September 12

9:00 Catherine McGeoch: Hard problems in experimental algorithmics

9:30 Mike Fellows: Parameterized complexity and the design and analysis of heuristic algorithms

10:00 Giri Narasimhan: Computing geometric MSTs and t-spanners via well-separated pair decomposition

10:30 **Coffee Break**

11:00 Jon Bentley: Cache-conscious algorithms and data structures

12:15 **Lunch**

14:30 Roberto Grossi: Engineering linked data structures for strings

15:00 Christoph Meinel: WWW.BDD-Portal.org — A platform of cooperative research on BDD-algorithms and heuristics

15:30 **Coffee Break**

16:00 Andrew Goldberg: Competitive auctions and digital goods

16:30 Karsten Weihe: Towards a solution to a complex scheduling problem

18:00 **Dinner**

20:00 **Special Session**: Teaching, general questions, open problems

Wednesday, September 13

9:00 Ben Juurlink: Performance relevant issues for parallel computation models

9:30 David Bader: Parallel algorithms for uniform shared-memory

10:00 Paul Spirakis: Experiments for distributed algorithms

10:30 **Coffee Break**

11:00 Kurt Mehlhorn: Three topics in experimental algorithmics

12:15 **Lunch**

13:30 Hike to a nice place with cake, coffee, tea, ...

18:00 **Dinner**

20:00 **Special Session**: Discussion on terminology. And answer the following questions:

- What are relevant experiments?
- What can be learned by experiments?
- What is a good benchmark test?
- What is a good experimental paper?
- When is implementation and testing a necessary part of algorithm development?

Thursday, September 14

9:30 Martin Dietzfelbinger: Proof versus experiment in hashing

10:00 Josep Díaz: Generating perfect matchings uniformly

10:30 **Coffee Break**

11:00 Bernard Moret: A case study in algorithmic engineering

12:15 **Lunch**

14:30 Richard Ladner: Cache performance of algorithms

15:00 Michael Bender: Cache-oblivious algorithms and data structures

15:30 **Coffee Break**

16:00 Norbert Zeh: Simulating BSP-algorithms in external memory

16:30 Tomasz Radzik: Experimental evaluation of network flow algorithms

18:00 **Dinner**

Friday, September 15

9:00 Irene Finocchi: Experiments on hierarchical decompositions of trees

9:30 Matthias Müller-Hannemann: A case study in experiments with algorithms: Hexahedral mesh generation for the simulation of the human mandible

10:00 David Johnson: The TSP challenge

10:30 **Coffee Break**

11:00 Concluding discussion

12:15 **Lunch**

18:00 **Dinner**

3 Abstracts

The unreasonable efficiency of asymptotic analysis

Philippe Flajolet

The main point here is that asymptotic analysis (in the sense of classical mathematics) is often a wonderfully accurate tool. Thus, I take exception to claims that it is usually imprecise, or that is only applies to extraordinarily large values of problem size (n).
Several problems are considered from the design and analysis where results of asymptotic analysis are matched against either "real" data or against simulation data. The confrontation is rewarding. Deviations are commonly to be assigned to oversimplistic models of data rather than to the analytic process as such. Sometimes even (e.g., in hashing or some cryptographic problems), the answers are wonderfully accurate. Examples reviewed include: (i) directory size in extendible hashing (originally "A very large data base problem"); (ii) behaviour of DES chip under iteration (a problem from cryptography); (iii) the ternary tree protocol (an efficient way to resolve contention in networks that is embedded into the IEEE 802.14 norm); (iv) mergesort and fractal fluctuations; (v) the ternary search trees of Bentley and Sedgewick (confronted to textual data); (vi) the HAKMEM algorithm for obtaining orientation of triangles in a robust and efficient way and its generalization to sorting numbers by continuous fraction (F+Valleé). In the latter case, the mathematical analysis even reveals unexpected connections with the Riemann hypothesis (!).
For many, the author claims that analysis of algorithms is fun, that experimenting is equally fun, and the confrontation between the theoretical and the practical viewpoints is rewarding.

Lower bounds for the minimum linear arrangement problem

Jordi Petit i Silvestre

Given an undirected graph $G = (V, E)$ with $|V| = n$, a *layout* of G is a one-to-one function $\varphi : V \to \{1, \ldots, n\}$. The *minimum linear arrangement* problem (MinLA) is a layout problem formulated as follows: Given a graph $G = (V, E)$, find a layout φ of G that minimizes

$$\textsc{la}(G, \varphi) = \sum_{uv \in E} |\varphi(u) - \varphi(v)|.$$

In this talk I concentrated on the analysis of lower bounding techniques for the MinLA problem on sparse graphs. Experiments evidence that there exists a large gap between the best costs of the solutions obtained with several approximation heuristics (local search variations, spectral methods and successive augmentation algorithms) and the best known lower bounds. The analytical results on uniform random graphs and on random geometric graphs, as well as the graphs for which their optimal solution is known, suggest that upper bounding heuristics might be much closer to the optimal values than the lower bounds obtained with several published methods. In this talk was presented the so-called *mesh method*, a new lower bounding method particularly appropriated for input graphs arising in finite element methods. This method is based on the search of maximal square meshes as a subgraph of the input graph. The experimental results show that this method provides better lower bounds but these are still far from the best upper bounds.
The conclusion I draw is that it is not only hard—in the practical sense—to get good upper bounds the MinLA problem, but that it also difficult to get good estimates for the lower bounds.

Pet peeves and pitfalls in the experimental analysis of algorithms

David S. Johnson

A "pet peeve" is a personal annoyance, something that I find particularly misguided in papers or experimental practice. A "pitfall" is a temptation or practice that can lead an experimentalist into a large waste of time/computation. This talk, based on a paper I am writing for the proceedings of the 5th DIMACS Implementation Challenge entitled "A Theoretician's Guide to the Experimental Analysis Algorithms," is based on lessons learned (and biases reinforced) over the course of more than a decade of experimentation, survey paper writing, refereeing, and lively discussions with other researchers. I organize the discussion in terms of 10 fundamental principles:

1. Perform newsworthy experiments
2. Tie your paper to the literature
3. Use instance testbeds that can support general conclusions
4. Use efficient and effective experimental designs
5. Use reasonably efficient implementations

6. Ensure reproducibility
7. Ensure comparability (e.g. of running times)
8. Report the full story
9. Draw well-justified conclusions and look for explanations
10. Present your data in informative ways

Overall, six pitfalls and 32 pet peeves are currently covered in the paper, but as a result of discussions at this workshop, these lists will no doubt grow.

Experience in adaptive set intersections

Erik D. Demaine and J. Ian Munro

(Joint work with Alex López-Ortiz of the University of New Brunswick).
We explore the use of adaptive algorithms whose running time depends upon, and takes advantage of, the particular instance at hand. In SODA 2000, we developed a theory behind such algorithms and lower bounds, under a comparison model. In particular, we developed an adaptive algorithm for intersecting k sorted sets whose performance is within a constant factor of the optimal for the given instance, under a reasonable model of computation. It can be shown that the adaptive technique gives a modest improvement over standard techniques on uniformly random input sets.
This talk deals with our experience on "burstier" data. We describe some preliminary experiments comparing our adaptive approach with the standard method and with lower bounds on the given instances. The intent is to estimate how much an adaptive approach can improve performance on "real" data, in this case a portion of the web. The results are indeed encouraging.

Simulation studies for online-problems

Sven O. Krumke

Research supported by the German Science Foundation (DFG), grant 883/5-3
authors: Jörg Rambau, Martin Grötschel, Dietrich Hauptmeier
The pallet transportation system at the Herlitz PBS AG in Falkensee near Berlin (the major European office supply provider) consists of truck loading stations,

pallet sorters, conveyor belts (horizontal transportation), elevators (vertical transportation), registration and inspection stations, and an automatic storage system. The flow of pallets from the automatic storage system to the truck loading stations is controlled by a central computer unit. The overall goal is to control the system in such a way that there is a quick and congestion free flow of pallets which satisfies all restrictions. Since the pallets that have to be transported during one day of production are not known in advance, decisions have to be made *online* without any knowledge of the future.
The above described situation can be mathematically modeled by an "Online-Dial-a-Ride-Problem": In the problem OnlineDarp objects are to be transported between points in a given metric space X with the property that for every pair of points $(x, y) \in X$ there is a path $p : [0, 1] \to X$ in X with $p(0) = x$ and $p(1) = y$ of length $d(x, y)$. A request consists of the objects to be transported and the corresponding source and target of the transportation request. The requests arrive online (i.e., a request becomes known to an online algorithm at its release time) and must be handled by a server which starts and ends its work at a designated origin and which moves along the paths in X. The server picks up and drops objects at their starts and destinations. It is assumed that neither the release time of the last request nor the number of requests is known in advance to an online-algorithm.
A feasible solution to an instance of the OnlineDarp is a schedule of moves (i.e., a sequence of consecutive moves in X together with their starting times) in X so that every request is served and that no request is picked up before its release time. The goal of OnlineDarp is to find a feasible solution with "minimal cost", where the notion of "cost" depends on the objective function used.
We investigate the problem OnlineDarp with respect to competitive analysis for the objective of minimizing the total completion time (makespan) and provide a best possible 2-competitive algorithm. However, for the task of minimizing the *maximal (average) waiting time* or the *maximal (average) flow time* there can be no algorithm with constant competitive ratio. Hence, all algorithms are equally bad from a competitive analysis point of view.
We report on simulation experiments for OnlineDarp, where we simulated various algorithms for the basic elevator control problem in varying load situations. In particular, we focused on the performance with respect to minimizing the average and the maximal flow times, resp.

Some experiences with teaching algorithm engineering

Peter Sanders

(Joint work with Kurt Mehlhorn).

We report experiences with an undergraduate course on algorithms which includes algorithm engineering material. The general content is quite conventional. Foundations, techniques (randomization, divide-and-conquer, greedy, ...) data structures, sorting and related problems, and simple graph algorithms.
New for us was a more intensive coupling to practical issues regarding applications, elegant interfaces, documentation, correctness and program checking, and implementation techniques. The machine model used is based on Knuth's new MMIX and imperative programming languages. Effects due to caches, and secondary memory are mentioned as a refinement. Most algorithms are first analyzed mathematically and then evaluated and compared experimentally. Programming assignments not only ask for implementation but also for systematic comparison of algorithms.

An empirical analysis of Delta algorithms

Walter Tichy

(Joint work with James Hunt and Phong Vo).
Delta algorithms compress data by encoding one file in terms of another. They are useful for storing multiple versions, updating software, displaying and merging changes, and other applications. This talk presents the performance parameters of several delta algorithms, using a benchmark of over 1300 pairs of files taken from two successive releases of GNU software. Results indicate that modern delta algortihms based on Ziv-Lempel techniques achieve significantly higher compression rates and better speeds than diff, a popular, but older delta compressor based on an algorithm for computing longest common subsequences.
We use this study to illustrate desirable properties of benchmarks for comparing algorithms and what can be expected from experimenting with algorithms.

Hard problems in experimental algorithmics

Catherine McGeoch

There are many hard problems in experimental algorithms. Here we consider techniques and strategies in situations where the experiment is *computationally* too hard, whether because of large problem spaces, slow programs, or data with high variance. These techniques are grouped in three categories: Focus it, Fake it, Finesse it. *Focus it* involves reducing the scope of the experiment until you are able to make some progress. I use an example from a Markov M.C. graph coloring sampling algorithm. *Fake it* means to exploit the experimental environment to

achieve efficiencies that would not be possible in application. Examples from tests of LRU style algorithms and self-organizing data structures. *Finesse it* means to change the question and find something easier to test, that sheds light on the original question.

Parameterized complexity and the design of heuristic algorithms

Mike Fellows

Part of the challenge in designing useful heuristic algorithms for realistic applications of hard computational problems consists in coming to grips with the fact that real world distributions of problem instances are not random, but frequently involve extra structure. Parameterized complexity provides a framework in which the contribution of the extra structure (the parameter) to problem complexity can be assessed. The talk will survey the basic ideas and results of parameterized complexity, and a variety of connections to the design of heuristics for hard problems.

Computing minimum spanning trees and t-spanners using well-separated pair decompositions

Giri Narasimhan

The well-separated pair decomposition is a practical tool for designing efficient geometric algorithms. This talk discusses experiments with using these decompositions to compute minimum spanning trees, t-spanners and stretch factors of geometric networks. Theoretically, the well-separated pair decomposition can be computed in $O(n \log n)$ time and $O(n)$ space for points in d-dimensional Euclidean space. The resulting implementation for computing minimum spanning trees is very efficient for $d > 2$. Work is underway to improve the performance of our implementations for computing t-spanners and stretch factors. For $d = 2$, methods using Delaunay triangulations seem to be faster than methods that use the well-separated pair decomposition. For very high-dimensional space, the constants involved (especially for the space requirements) make the well-separated pair decomposition less practical.

Cache-conscious algorithms and data structures

Jon Bentley

Which data structure is faster for representing sorted sequences, arrays or lists? Experiments show that the answer depends on the relative costs of accessing memory, and usually changes between the Level-1 cache (where lists are faster) and RAM (where arrays are faster). A simple program generates a cost model that allows one to answer such questions easily. We apply the insights of the model to problems in set representation, sorting, searching, strings, and computational geometry. Techniques such as data compression and improving memory access patterns reduce the run time of real programs by factors ranging from 2 to 16.

Engineering linked data structures for strings

Roberto Grossi

A vast repertoire of basic search data structures are currently available to scientists and computer programmers to maintain a set of keys arbitrarily long, such as strings, multidimensional points, multiple-precision numbers, and multi-key data. These data structures require sophisticated techniques to circumvent the problem that each comparison requires more than constant time. In this talk, we present some practical data structures to efficiently search and sort strings and long keys in a simple way. We show how to obtain them by augmenting several basic textbook implementations of searching data structures so that they can operate on long keys. Despite the simplicity and generality of this method, our experimental study on searching and storing long keys shows that it is quite competitive with several optimized and tuned implementations currently available in the literature.

WWW.BDD-Portal.org — A platform of cooperative research on BDD-algorithms and heuristics

Christoph Meinel

To address the needs of the BDD research community, (binary decision diagrams) we have created a specialized portal site for this area. The specific problem with BDD's is that it is usually impossible to evaluate the performance of a new BDD algorithm without actually trying it out. On the other hand, these algorithms

are very sensitive to the environment they are executed in, consume a lot of resources when run and are difficult to include in existing software packages. So comparison of new methods with existing ones is generally problematic. To address this problem our portal includes a system for the online "publication" of BDD-functionality by allowing the use of tools that contain BDD methods via a WWW interface. The system is mainly aimed at facilitating the process of determining whether a BDD method is suitable for a specific application and of allowing easy comparison between methods. It is open to all researchers who wish to publicize their results in this way. (co-work with A. Wagner, H. Sack, C. Stangier)

Competitive auctions and digital goods

Andrew Goldberg

(Joint work with Jason Hartline (U. Washington and InterTrust) and Andrew Wright (InterTrust)).
Suppose a pay-per-view TV company gets exclusive rights for a live broadcast of a Tyson-Hollyfield rematch. How much should the company charge to maximize its revenue? Usually companies run marketing studies to determine how much their customers are willing to pay, and then set the best (optimal) price. We investigate an alternative of auctioning the viewing rights.
We study a class of single round, sealed bid auctions for items in unlimited supply, such as digital goods or TV broadcasts. We focus on auctions that are truthful and competitive. Truthful auctions encourage consumers to bid their utility; competitive auctions yield revenue that is within a constant factor of the revenue for optimal fixed pricing. We give a bound on how far the revenue for optimal fixed pricing can be from the total market utility. We show that several randomized auctions are truthful and competitive under certain assumptions, and that no truthful deterministic auction is competitive. We also show a somewhat surprising result: no truthful auction, even a multi-price auction, can outperform optimal fixed pricing. We present simulation results which confirm that our auctions compare favorably to fixed pricing: they often come very close to optimal fixed pricing and outperform fixed pricing with a 25for which we also get truthful and competitive auctions.
In this talk we emphasize our experimental results.

Towards a solution to a complex scheduling problem – Reasoning about on–going work

Karsten Weihe

If one is faced with a new algorithmic real–world problem, chances are high that the theoretical (and practical) literature is only of little help. The problem may belong to a well–studied class, however, the concrete version at hand is often completely new. The algorithmic approaches developed for similar algorithmic problems need not be adaptable, and even if so, it is not clear which of them are promising and which are not. On the other hand, the deadlines often do not suffice to evaluate a variety of approaches rigorously. (read "scientifically").
In such a case, the best an algorithmician can do is using his/her own intuition. In this talk, I will try to verbalize the intuitive reasoning behind a recent applied work of mine: production planning in steel plants.

Performance relevant issues for parallel computation models

Ben Juurlink

In the last 10 years, an overwhelming number of parallel computation models have been proposed. However, the question of which aspects of parallel coputer systems should be included in the "right" model and which should be excluded still remains largely unresolved. In this talk I will first describe some desirable properties that the "right" parallel computation model should possess. After that, I will attempt to find answers to the following questions. First, should the "right" parallel computation model reward sending large messages? And second, should the model have a "capacity constraint" (as in the LogP model)?

Using PRAM algorithms on a uniform memory access shared-memory architecture

David A. Bader

Shared-memory architectures with uniform memory access come very close to the PRAM, the theoretical model of parallel computing, and stand in sharp contrast to the common cluster approach. While PRAM algorithms have been devised for a large variety of combinatorial problems in graphs, strings, and geometry,

none has been run successfully on a conventional parallel machine – the irregular nature of the computations cause a tremendous communication load and make the parallel machine run much more slowly than a simple workstation. Our preliminary results indicate that true shared-memory architectures, such as the Sun Enterprise 10000, run these same PRAM algorithms quite efficiently, showing effective speedups and thus opening up the possibility of leveraging over 20 years of research in PRAM algorithms for practical applications that require complex combinatorial optimization (such as sequence alignment and tree reconstruction problems that arise in genomics and bioinformatics).

Experiments for distributed algorithms

Paul G. Spirakis

In this work we examine the characteristics of experiments that concern distributed algorithms. Distributed algorithms are designed for environments that are usually asynchronous, fault-prone and supporting communication (via shared memory or messages). Experiments for such algorithms have to model such fault-prone environments. There is need for simulators since experiments require scalability and are hard to execute on real networks. The algorithmic languages for the design of such algorithms require new types and classes (process, message, link, delays etc). A major difficulty is the correct representation of arbitrary asynchrony, or in general the creation of worst case instances for such experiments. In our work we describe two methods for creation of worst case instances (a game theoretic and a method of simulating lower bounds proofs) via a case study of a distributed pursuit - evation algorithm. A special part of the work addresses mobile computing algorithms, especially mobile control ones (routing, leader election, counting...). Such experiments must describe, in addition, the motion profiles of stations. We especially study the use of random walks as a suitable assumption for motions description of stations whose motions are not described by the algorithms themselves. We find that such walks, in some cases, can lead to nice analysis which can be verified and even strenthened by the experiments. We indicate this via a study of a new method for routing in ad hoc mobile nets. We also describe the main features of the Distributed Algorithms Platform, a platform and a library designed in our group, suitable for such experiments.

Three topics in experimental algorithmics

Kurt Mehlhorn

I discussed three topics in algorithm engineering.

1. The implementation of an $O(nm \log n)$ algorithm for weighted matchings in general graphs (joint work with Guido Schäfer). The implementation is (at least) competitive with other weighted matching codes. (`www.mpi-sb.mpg.de/~mehlhorn/ftp/WAE00.ps.gz`)

2. The overhead for using LEDA in the implementation of graph algorithms (joint work with Stefan Näher). Recent improvements in basic graph data type of LEDA allow the implementation of graph algorithms which mach the performance of hand-coded C-implementations as a detailed comparison with Goldberg's maxflow code shows. (`http://www.informatik.uni-trier.de/~naeher/maxflow.html`)

3. The difficulties in implementing geometric algorithms (joint work with Stefan Schirra, Stefan Näher, Christoph Burnikel, Rudi Fleischer, Stefan Funke).

Proofs versus experiments in hashing

Martin Dietzfelbinger

(Joint work with Torben Hagerup).
A minimal perfect hash function h maps a set S of n keys one-to-one to $\{0, \ldots, n-1\}$. We study methods for finding minimal perfect hash functions that can be evaluated with very few arithmetic operations and one table lookup in a table $d[0..m-1]$ of integers in $\{0, \ldots, n-1\}$. In 1999, R. Pagh described a construction method that yields such a function and is guaranteed to work in expected linear time if $m \geq (2+\varepsilon)n$, for $\varepsilon > 0$ arbitrary. We give a more involved construction ("undo-one algorithm") that works if $m \geq (1+\varepsilon)n$ for any constant $\varepsilon > 0$. We discuss possible ways of implementing these methods and (preliminary) experimental results. The experiments indicate that the critical range for m, where the algorithms don't work anymore, lies far below n, even below $0.5n$. Here the proofs of correctness for Pagh's algorithm and for ours do not work anymore. New theoretical questions are suggested by the experiments, especially which structural properties are responsible for the algorithms to work for m in this small range.

An open problem for which experimental algorithms may help

Josep Díaz

The problem we consider in the almost uniform generation of perfect matchings. A problem known to be P-complete. Using the well known techniques of the Markov chain it is possible to sample in $O(n^5)$ a matching with an almost uniform distribution, where n is the number of nodes of the graph.
In a previous paper, we gave an alternative using a a quadratic system, which starting from a initial distribution Π_0 of the matchings it evolves step by step, converging to the uniform distribution. In spite that the system can be easily parallelized with a polynomial number of processors, we have been unable to obtain a polylogarithmic mixing times. Preliminary and rough empirical experience, seems to indicate that indeed our non-linear system could sample matchings with an almost uniform distribution in sub-polynomial time. However, the experiments done are too limited to give any conclusive evidence.
We posse as an open challenge, to design a set of experiments to validate the open conjecture that indeed the non-linear system from [Díaz, Serna, and Spirakis 1999], can quickly sample matchings with an almost uniform distribution.

A new implementation and detailed study of breakpoint analysis

Bernard Moret

Phylogenies derived from gene order data may prove crucial in answering some fundamental open questions in biomolecular evolution. Yet very few techniques are available for such phylogenetic reconstructions. One method is *breakpoint analysis*, developed by Blanchette and Sankoff for solving the "breakpoint phylogeny." Our earlier studies confirmed the usefulness of this approach, but also found that `BPAnalysis`, the implementation developed by Sankoff and Blanchette, was too slow to use on all but very small datasets. We report here on a reimplementation of `BPAnalysis` using the principles of algorithmic engineering. Our faster (by 2 to 3 orders of magnitude) and flexible implementation allowed us to conduct studies on the characteristics of breakpoint analysis, in terms of running time, quality, and robustness, as well as to analyze datasets that had so far been considered out of reach. We report on these findings and also discuss future directions for our new implementation.

Program instrumentation to understand cache performance

Richard E. Ladner

Cache performance is important to the overall running time of programs. Cache misses can cost 100 cycles or more on modern processors. Trace-driven cache simulation through program instrumentation can give valuable information about the cache performance of a program. This information can be used to improve program design by reducing cache misses. Trace-driven cache simulation can also be used to validate cache models and analyses of algorithms. The general instrumentation tool ATOM (Eutace and Srivatava, 1994) is described. It is shown how ATOM can be used to implement a trace-driven cache simulator which gives accurate measurements of cache misses. Several examples are given where trace-driven cache simulation is used to validate a simple direct mapped cache model and the analyses of several basic programs. This work was done with Jim Fix, Anthony LaMarca, and Ed Hong.

Cache-oblivious algorithms and data structures

Michael A. Bender

(Joint work with Erik Demaine and Martin Farach-Colton).
We present methods for constructing cache-oblivious algorithms and data structures. Cache-oblivious algorithms run efficiently on a hierarchical memory, even though they completely avoid any memory-specific parameterization (such as cache block sizes or access times). We first review the regular and static problems in the literature. Then we show new methods for designing memory efficient cache-oblivious algorithms for dynamic and irregular problems, such as data structures. We show how to design a cache-oblivious B-tree, which matches the performance of the standard B-tree. Then we outline future directions in this area.

Simulating BSP-algorithms in external memory

Norbert Zeh

Abstract not available.

Experimantal evaluation of algorithms for the generalised network flow problem

Tomasz Radzik

The generalised network flow problem is to maximise the net flow into a specified sink node in a network with gain-loss factors associated with edges. In practice, computation of solutions for instances of this problem is almost always done using general-purpose linear programming codes, but this may change because a number of specialized combinatorial generalised-flow algorithms have been recently proposed. To complement the known theoretical analyses of these algorithms, we develop their implementations and investigate their practical performance. We include in our study the excess-scaling algorithm proposed by Goldfarb, Jin and Orlin, and the gain-scaling "push-relabel" algorithm proposed by Tardos and Wayne. We compare the performance of our implementations of these algorithms with implementations of the straightforward highest-gain path-augmentation algorithms.

Experiments on hierarchical decompositions of trees

Irene Finocchi

(Joint work with R. Petreschi).
Hierarchical decompositions of graphs, used together with navigation techniques, are a useful tool for drawing large graphs. Such decompositions can be represented by means of a data structure called *hierarchy tree*, whose leaves coincide with the vertices of the graph: visualizing suitable sets of nodes of the hierarchy tree (clusters) makes it possible to considerably reduce the dimension of the drawing. In the talk we address the problem of computing hierarchical decompositions of trees and forests. We present efficient algorithms for doing so and we experimentally study their performances with respect to several structural properties of the hierarchy tree which are useful from a graph drawing perspective.

A case study in experiments with algorithms: Hexahedral mesh generation for the simulation of the human mandible

Matthias Müller-Hannemann

Hexahedral mesh generation is of fundamental importance for the numerical analysis and simulation of partial differential equations by means of the finite element method (FEM) in many fields of applications. We describe experience with experiments for a combinatorial algorithm using the concepts of shellings of cell complexes, weighted perfect b-matching, and efficient algorithms on planar graphs to construct high-quality hexahedral meshes. We use the case study of an interesting application from biomechanics and medicine (the simulation of the human mandible after a bite on a "hard nut") to discuss the shortcomings of previous work and several obstacles which had to be overcome to get a successful experiment. Our experiment comprises the full meshing process from the initial raw data (computed tomography data) to the evaluation and interpretation of the simulation results. We point out the importance of flexibility as a major design principle for algorithms which are to be applied in a real world setting. The simulation work has been done jointly with Cornelia Kober (TU Munich).

The TSP challenge

David S. Johnson

This talk summarizes the 8th DIMACS Implementation Challenge: The Traveling Salesman Problem (url: `http://www.research.att.com/~dsj/chtsp/`). The TSP is probably the most-studied optimization problem of all time, and often the first problem that newcomers to the field (or visitors from other domains) attack. Consequently, it has one of the most fractionated literatures in the field, with many papers written in apparent ignorance of what has been done before. One goal of this Challenge is to create a reproducible picture of the state of the art in the area of TSP heuristics (their effectiveness, their robustness, their scalability, etc.), so that future algorithm designers can quickly tell on their own how their approaches compare with already existing TSP heuristics. To this end we are identifying a standard set of benchmark instances and generators (so that quality of tours can be compared on the same instances), as well as a benchmark implementation of a well-known TSP heuristic (the greedy or "multi-fragment" algorithm), so that the effect of machine speed on reported running time can (roughly) be normalized away.

A second goal is to enable current researchers to compare their codes with each other, in hopes of identifying the more effective of the recent algorithmic innovations that have been proposed, both for fast heuristics and for attempts at improving on the classical Lin-Kernighan algorithm. Although many implementations do not include the most sophisticated speed-up tricks and thus may not be able to compete on speed with highly tuned alternatives, we may be able to gain insight into the effectiveness of various algorithmic ideas by comparing codes with similar levels of internal optimization. It will also be interesting to compare the running times of the best current optimization codes with those of the more complicated heuristics on instances that both can handle.

4 Discussion Sessions

Special Session: Teaching, general questions, open problems

The discussion was centered on the use of experimental algorithmics in the classroom: why use it, how to use it, when to use it, and how well does it work?
Kurt Mehlhorn mentioned that working on LEDA showed him that the hardest issues he faced in developing LEDA, ostensibly "just implementing algorithms," were in fact issues never addressed in the typical algorithms class: checking, specification, numerical accuracy, etc. This made him question the relevance of the more traditional, pencil-and-paper style of class. He found the he could teach a class with a strong experimental component without compromising the intellectual content of the course – keeping comparable intellectual depth and introducing real theoretical concepts.
Bernard Moret, focusing only on the junior-level class he has taught for 7 years with an experimental term project, mentioned that the practical aspects are the motivating factor for the students. US students, at least, are very concrete and need to confront real examples of the need for checking (get buggy software off the web), specification (get software off the web that does not do what it says it does), and, most of all efficiency (nothing like seeing one piece of code take 2 seconds while the other takes 2 hours, as in, say, Prim's algorithm with priority queues vs. Prim's algorithm with simple lists). He also stressed the need to reuse existing components (software off the Web, LEDA), so as to allow the students to focus on algorithmic rather than programming issues.
Richard Ladner demonstrated a rather different course, geared towards professional MS students, which did not involve experimental work by the students, but included experimental demos as well as a a strong application content. The delivery vehicle includes online lectures (video), animations, etc.
Christos Zaroliagis argued for a course that would teach Algorithm Engineering directly (something both Kurt M. and Bernard M. argued against) to students who have already taken a more conventional algorithms or data structures course. In his course, he used the LEDA text and library; he found the students particularly enjoyed the visualization component of experiments.
Giri Naramsimhan used LEDA in a project-oriented graduate algorithms class and reported good results.
Cathy McGeoch discussed several classes she has taught over the years, including a class on experimental analysis, data structure classes where the class is broken up into teams implementing and testing various solutions, and classes where someone comes in from industry or government to pose a real problem to the class, which then has to formalize it and design and implement a solution.
Jon Bentley listed several tools or tricks that he has used in teaching and found particularly helpful: mini programming contests (including the design of worst-

case examples), term papers, case studies, assigning the final exam questions on the first day of class, and asking the students to produce the most elegant code they could.
Michael Bender and **Lars Arge** both reported being able to attract students to more conventional theory; Lars also discussed the second-year graduate project at Duke, which has been very successful in the area of algorithms; he stressed the importance of being an "honest theoretician", i.e., of accompanying algorithm presentations with an honest estimate of the likely practical use of same.
Ian Munro asked about the move from implementations/algorithms that work well for medium instances to algorithms that work well for very large ones.
Erik Meineche-Schmidt asked about the role of experimental analysis per se, which did not seem to feature prominently in most of the classes described; **Bernard Moret** mentioned that his heuristics class, a graduate class taught for the last 16 years, does that systematically, but that he does not do the same in his junior class.

Questions:
Here are the tallies on the 6 short questions:

1. Do you agree that incorporating experimental work in the first course on algorithms/data structures is a good idea?

   `yes / for the most part / no opinion / not really / not at all`
   `18/14/1/2/0  or 32/1/2`

2. Do you agree that this experimental aspect should focus on algorithmic questions, as opposed to programming, and thus should rely on libraries (e.g., LEDA) rather than on the students' own coding of the algorithms?

   `yes / for the most part / no opinion / not really / not at all`
   `4/12/4/14/1  or 16/4/15`

3. Do you agree that the following devices should find a place within classes on algorithms:

   - programming contests

     `yes / for the most part / no opinion / not really / not at all`
     `15/9/4/5/2  or 24/4/7`

   - algorithm visualization

     `yes / for the most part / no opinion / not really / not at all`
     `16/3/7/4/2  or 19/7/6`

- term projects

  yes / for the most part / no opinion / not really / not at all
  19/8/4/4/0 or 27/4/4

- specific programming assignments (e.g., implement RB trees)

  yes / for the most part / no opinion / not really / not at all
  17/9/2/2/4 or 26/2/6

4. Do you agree that some experimental components should be part of every course on algorithms (beginning, intermediate, advanced)?

   yes / for the most part / no opinion / not really / not at all
   15/8/2/7/3 or 23/2/10

5. Do you agree that most undergraduate students would benefit more from being exposed to experimental methodology and carrying out an experimental project than from being exposed to additional theoretical material?

   yes / for the most part / no opinion / not really / not at all
   15/7/5/8/0 or 22/5/8

6. Do you think that current textbooks are adequate to support an experimental component in the classroom?

   yes / for the most part / no opinion / not really / not at all
   0/0/0/2/33 or 0/0/35

5 List of Participants - Dagstuhl Seminar 00371 (08.04.99; 17:37)

Date: 10.09.2000 - 15.09.2000
Title: Experimental Algorithms
webpage: http://www.dagstuhl.de/DATA/Participants/00371.html

Lars Arge
Duke University
Computer Science Dept.
D205 LSRC, Box 90129
NC 27708-0129 Durham
USA
phone: +1-919-660-6557
fax: +1-919-660-6519
e-mail: large@cs.duke.edu
url: http://www.cs.duke.edu/~large

David Bader
University of New Mexico
Dept. of Electrical and Computer Engineering

NM 87131 Albuquerque
USA
phone: +1-505-277-6724
fax: +1-505-277-1439
e-mail: dbader@eece.unm.edu
url: http://www.eece.unm.edu/~dbader

Michael Bender
SUNY at Stony Brook
Dept. of Computer Science

NY 11794-4400 Stony Brook
USA
phone: +1-632-632-7835
fax: +1-632-632-8334
e-mail: bender@cs.sunysb.edu
url: http://www.cs.sunysb.edu/~bender

Jon Bentley
Bell Labs
Avaya Communication
Room 2C-317, 600 Mountain Avenue
NJ 07974 Murray Hill
USA
phone:

fax: +1-908-582-5857
e-mail: jlb@research.bell-labs.com
url: http://cm.bell-labs.com/

Gerth Stolting Brodal
Aarhus University
BRICS - Dept. of Computer Science
Ny Munkegade, BG 540
DK-8000 Aarhus
DK
phone: +45-89 42 34 72
fax: +45-89 42 32 55
e-mail: gerth@brics.dk
url: http://www.brics.dk/~gerth/

Erik Demaine
University of Waterloo
Dept. of Computer Science
200 University Avenue West
ON-N2L 3G1 Waterloo
CDN
phone: +1-519-725-5939
fax: +1-519-885-1208
e-mail: eddemain@uwaterloo.ca
url: http://daisy.uwaterloo.ca/~eddemain/

Martin Demaine
University of Waterloo
Dept. of Computer Science
200 University Avenue West
ON-N2L 3G1 Waterloo
CDN
phone: +1-519-725-5939
fax: +1-519-885-1208
e-mail: mldemaine@uwaterloo.ca
url:

Camil Demetrescu
Università di Roma "La Sapienza"
Dipartimento di Scienze dellInformazione
Via Salaria 113
I-00198 Roma
I
phone: +39-06-4991-8442
fax: +39-06-8530-0849
e-mail: demetres@dis.uniroma1.it
url: http://www.dis.uniroma1.it/~demetres/

Josep Díaz
Universitat Politecnica de Catalunya
Dept. L.S.I.
Moduc C-5, Jordi Girona 1-3
E-08034 Barcelona
E
phone:
fax: +-0034-3-401-7014
e-mail: diaz@lsi.upc.es
url: http://www.lsi.upc.es/~diaz

Martin Dietzfelbinger
Technische Universität Ilmenau
Fakultät für Informatik und Automatisierung
Postfach 100565
D-98684 Ilmenau
D
phone: +49-3677-69-2656
fax: +49-3677-69-1237
e-mail: martin.dietzfelbinger@theoinf.tu-ilmenau.de
url: http://eiche.theoinf.tu-ilmenau.de/person/dietzd.html

Mike Fellows
University of Victoria
Dept. of Computer Science
P.O. Box 3055
BC-V8W 3P6 Victoria
CDN
phone: +1-250-721-7299
fax: +1-250-721-7292
e-mail: mfellows@csr.uvic.ca
url: http://www.csr.uvic.ca/~mfellows

Irene Finocchi
Università di Roma "La Sapienza"
Dipartimento di Scienze dellInformazione
3rd floor, Via Salaria 113
I-00198 Roma
I
phone: +39-06-49918430
fax: +39-06-8541842
e-mail: finocchi@dsi.uniroma1.it
url: http://www.dsi.uniroma1.it/~finocchi/

Philippe Flajolet
INRIA Rocquencourt

Domaine de Voluceau
B.P. 105
F-78153 Le Chesnay
F
phone: +33-1-39 63 56 26
fax: +33-1-39 63 55 96
e-mail: philippe.flajolet@inria.fr
url: http://algo.inria.fr/flajolet/

Rudolf Fleischer
Hong Kong University of Science & Technology
Dept. of Computer Science
Clear Water Bay, Kowloon
Hong Kong
HK
phone: +852-2398-8770
fax: +852-2358-1477
e-mail: rudolf@cs.ust.hk
url: http://www.cs.ust.hk/~rudolf

Andrew V. Goldberg
Inter-Trust Techn.
STAR Labs
4750 Patrick Henry Drive
CA-95054 Santa Clara
USA
phone: +1-408-855-0117
fax:
e-mail: goldberg@intertrust.com
url: http://www.star-lab.com/goldberg/

Roberto Grossi
Università degli Studi di Pisa
Dipartimento di Informatica
Room O-II-11, Corso Italia 40
I-56125 Pisa
I
phone: +38-050-887-293
fax: +39-050-887-226
e-mail: grossi@di.unipi.it
url: http://www.di.unipi.it/~grossi/

Torben Hagerup
Universität Frankfurt
FB 20 Informatik, Robert-Mayer-Str. 11-15
PF 11 19 32
D-60054 Frankfurt

D
phone: +49-697-9828155
fax:
e-mail: hagerup@informatik.uni-frankfurt.de
url: http://www.informatik.uni-frankfurt.de/~hagerup/

David S. Johnson
AT&T Labs Research
180 Park Avenue
P.O. Box 971
NJ 07932-0971 Florham Park
USA
phone: +1-973-360-8440
fax: +1-973-360-8178
e-mail: dsj@research.att.com
url: http://www.research.att.com/~dsj/

Ben Juurlink
Delft University of Technology
Information Technology and Systems
HB 15.270, Mekelweg 4
NL-2628 Delft
NL
phone: +31-15-278-1572
fax: +31-15-278-4898
e-mail: B.H.H.Juurlink@its.tudelft.nl
url: http://cardit.et.tudelft.nl/~benj

Sven O. Krumke
Konrad-Zuse-Zentrum für Informationstechnik
Computing Science
Optimization, Takustr. 7
D-14195 Berlin
D
phone: +49-30-84185-246
fax: +49-30-84185-269
e-mail: krumke@zib.de
url: http://www.zib.de/krumke/

Richard Ladner
University of Washington
Dept. of Computer Science & Engineering
114 Sieg Hall, Box 352350
WA 98195-2350 Seattle
USA
phone: +1-206-543-9347
fax: +1-206-543-2969

e-mail: ladner@cs.washington.edu
url: http://www.cs.washington.edu/homes/ladner

Catherine McGeoch
Amherst College
Dept. of Mathematics and Computer Science

MA 01002-5002 Amherst
USA
phone: +1-413-542-79 13
fax: +1-413-542-2550
e-mail: ccm@cs.amherst.edu
url: http://www.cs.amherst.edu/~ccm

Kurt Mehlhorn
MPI für Informatik

Stuhlsatzenhausenweg 85
D-66123 Saarbrücken
D
phone: +49-681-9325 100
fax: +49-681-9325 199
e-mail: mehlhorn@mpi-sb.mpg.de
url: http://www.mpi-sb.mpg.de/~mehlhorn

Christoph Meinel
Universität Trier
FB IV - Informatik
Universitätsring 15
D-54286 Trier
D
phone: +49-651-201-2728
fax: +49-651-201-3954
e-mail: meinel@uni-trier.de
url: http://www.informatik.uni-trier.de/~meinel

Friedhelm Meyer auf der Heide
Universität Paderborn
Heinz Nixdorf Institut, FB Mathematik/Informatik
Fürstenallee 11
D-33102 Paderborn
D
phone: +49-5251-60-6480
fax: +49-5251-60-6482
e-mail: fmadh@uni-paderborn.de
url: http://www.uni-paderborn.de/cs/fmadh.html

Bernard Moret
University of New Mexico
Dept. of Computer Science

NM 87131 Albuquerque
USA
phone: +1-505-277-5699
fax: +1-505-277-6927
e-mail: moret@cs.unm.edu
url: http://www.cs.umn.edu/~moret/

Ian Munro
University of Waterloo
Dept. of Computer Science
200 University Avenue West
ON-N2L 3G1 Waterloo
CDN
phone: +1-519-888-45 67 ext. 44 33
fax: +1-519-885-12 08
e-mail: imunro@uwaterloo.ca
url: http://algonquin.uwaterloo.ca/~imunro/

Matthias Müller-Hannemann
TU Berlin
FB 3 Mathematik, MA 6-1
Straße des 17. Juni 136
D-10623 Berlin
D
phone: +49-30-314-25181
fax: +49-30-314-25191
e-mail: mhannema@math.tu-berlin.de
url: http://www.math.tu-berlin.de/~mhannema/

Giri Narasimhan
The University of Memphis
Dept. of Computer Science

TN 38152-3240 Memphis
USA
phone:
fax: +1-901-678-2480
e-mail: giri@msci.memphis.edu
url: http://www.msci.memphis.edu/~giri/index.html

Stefan Näher
Universität Trier

FB IV - Informatik
Universitätsring 15
D-54286 Trier
D
phone: +49 651 201-3275
fax: +49 651 201-3856
e-mail: naeher@informatik.uni-trier.de
url: http://www.informatik.uni-trier.de/~naeher/

Michael Paterson
University of Warwick
Dept. of Computer Science

CV4 7AL Coventry
GB
phone: +44-2476-523194
fax: msp@dcs.warwick.ac.uk
e-mail: http://www.dcs.warwick.ac.uk
url:

Jordi Petit i Silvestre
Universitat Politecnica de Catalunya
Dept. L.S.I.
Jordi Girona 1-3
E-08034 Barcelona
E
phone: +34-934-017014
fax: jpetit@lsi.upc.es
e-mail: http://www.lsi.upc.es/ jpetit
url:

Cindy Phillips
Sandia National Laboratories
MS 1110
P.O. Box 5800
NM 87185-110 Albuquerque
USA
phone: +1-505-845-7296
fax: +1-505-845-7442
e-mail: caphill@mp.sandia.gov
url: http://www.cs.sandia.gov/~caphill

Tomasz Radzik
King's College
Dept. of Computer Science
Strand
WC2R 2LS London

GB
phone:
fax: +44-171-848-2851
e-mail: radzik@dcs.kcl.ac.uk
url: http://www.dcs.kcl.ac.uk/staff/radzik

Peter Sanders
MPI für Informatik

Stuhlsatzenhausenweg 85
D-66123 Saarbrücken
D
phone: +49 681 9325 115
fax: +49 681 9325 199
e-mail: sanders@mpi-sb.mpg.de
url: http://www.mpi-sb.mpg.de/~sanders/

Erik Meineche Schmidt
Aarhus University
BRICS - Dept. of Computer Science
Ny Munkegade, BG 540
DK-8000 Aarhus
DK
phone:
fax: +45-89 42 32 55
e-mail: ems@daimi.au.dk
url:

Paul G. Spirakis
Computer Technology Institute
3rd Floor, Riga Fereon 61
P.O. Box 1122
GR-26221 Patras
GR
phone: +30-61-225073, 220112
fax: +30-61-222096
e-mail: spirakis@cti.gr
url: http://www.cti.gr/structure

Walter F. Tichy
Universität Karlsruhe
Inst. für Programmstrukturen und Datenorganisation
Am Zirkel 2, Postfach 6980
D-76128 Karlsruhe
D
phone: +49-721-608-3934
fax: +49-721-608-7343

e-mail: tichy@ira.uka.de
url: http://wwwipd.ira.uka.de/~tichy

Berthold Vöcking
MPI für Informatik
AG1
Stuhlsatzenhausenweg 85
D-66123 Saarbrücken
D
phone:
fax: +49 +681 +9325 199
e-mail: voecking@mpi-sb.mpg.de
url: http://www.mpi-sb.mpg.de/~voecking/

Dorothea Wagner
Universität Konstanz
FB Informatik & Informationswissenschaft
D 188
D-78457 Konstanz
D
phone: +49-7531-88-2893
fax: +49-7531-88-3577
e-mail: Dorothea.Wagner@uni-konstanz.de
url: http://www.fmi.uni-konstanz.de/~wagner/

Karsten Weihe
Universität Bonn
Forschungsinstitut für Diskrete Mathematik
Lennestr. 2
D-53113 Bonn
D
phone: +49-179-2441246
fax: +49-228-738771
e-mail: weihek@acm.org
url: http://www.or.uni-bonn.de/card-weihe.eng.html

Gerhard Woeginger
TU Graz
Institut für Mathematik
Steyrergasse 30
A-8010 Graz
A
phone: +43-316-873-53 57
fax: +43-316-873-53 69
e-mail: gwoegi@opt.math.tu-graz.ac.at
url: http://www.opt.math.tu-graz.ac.at/woe/

Christos Zaroliagis
University of Patras
Dept. of Comp. Eng. and Informatics

GR-26500 Patras
GR
phone: +30-61-960-309
fax: +30-61-960-374
e-mail: zaro@ceid.upatras.gr
url: http://www.ceid.upatras.gr/faculty/zaro

Norbert Zeh
Carleton University
School of Computer Science
Room 5345, 1125 Colonel By Drive
ON-K1S 5B6 Ottawa
CDN
phone: +1-613-520-2600
fax: +1-613-520-4334
e-mail: nzeh@scs.carleton.ca
url: http://fusion.scs.carleton.ca/~nzeh/