The Fifth Dagstuhl Seminar on Data Structures

February 27 – March 3, 2000

organized by
Susanne Albers, Ian Munro and Peter Widmayer

The area of Data Structures continues to be an important and vibrant aspect of computer science. The topic is an essential component in the algorithmic solution of many problems. Although data structures have been studied for more than forty years, there is still a large research community working on exciting and challenging problems. The Fifth Dagstuhl Seminar on Data Structures was attended by over 50 people from 10 different countries; and so, was substantially larger than previous meetings. Attendees included many young researchers, e.g. there were 10 Ph.D. students. A third of the participants were attending the seminar for the first time, bringing new ideas and points of view.
The workshop presentations addressed classical data structuring problems as well as new problems arising in important applications. Interesting results were presented on classical issues such as search trees, priority queues, hashing and predecessor problems. Moreover, there were contributions on graph problems such as DFS and shortest paths computations. Several talks were concerned with geometric problems such as range searching and convex hull computations. There was an increase in the number of presentations on external memory algorithms, relative to previous meetings. Last but not least, there were several contributions investigating data structure problems in specific application areas such as Parallel and Distributed Computing, Computational Biology and Database Systems.
In an open problem session many challenging research problems were proposed. Indeed solutions to two of the open problems posed in this session were presented later in the week. As usual, most participants used the seminar to develop ongoing research collaborations and to form new ones.
Thanks: The abstracts of the 40 talks were compiled by Ulrich Meyer. The organizers would like to thank all participants and especially the team of Schloß Dagstuhl for helping to make the workshop a success. The warm atmosphere of the Schloß, as always, supported discussions and the informal exchange of ideas!

Improved algorithms for finding level ancestors in dynamic trees

Stephen Alstrup, The IT University of Copenhagen, Denmark

Given a node x at depth d in a rooted tree $\mathit{LevelAncestor}(x, i)$ returns the ancestor to x in depth $d - i$. We show how to maintain a tree under addition of new leafs such that updates and level ancestor queries is performed in worst case constant time. Given a forest of trees with n nodes where edges can be added, m queries and updates take $O(m\alpha(m, n))$ time. This solves two open problems (P.F. Dietz, Finding level-ancestors in dynamic trees, LNCS, 519:32-40, 1991). In a tree with node weights we can report the node with minimum weight on a path in the same time as $\mathit{LevelAncestor}$. Previously such results were known only for special cases (e.g., R.E. Tarjan. Applications of path compression on balances trees. J.ACM, 26(4):690-715, 1979.).
Joint work with Jacob Holm.

Sorting, searching, and priority queues with exponential search trees

Arne Andersson, Uppsala University

We introduce a novel technique for converting static polynomial space search structures for ordered sets into fully-dynamic linear space data structures. Based on this we present optimal bounds for dynamic integer searching, including finger search, and exponentially improved bounds for priority queues.

I/O-Efficient Dynamic Planar Point Location

Lars Arge, Duke University

We present the first provably I/O-efficient dynamic data structure for point location in a general planar subdivision. Our structure uses $O(N/B)$ disk blocks to store a subdivision of size N, where B is the disk block size. Queries can be answered in $O(\log_B^2 N)$ I/Os in the worst-case, and insertions and deletions can be performed in $O(\log_B^2 N)$ and $O(\log_B N)$ I/Os amortized, respectively. Previously, an I/O-efficient dynamic point location structure was only known for monotone subdivisions.

How to deal with Incomplete Imprecise Information

Hannah Bast, Max-Planck Institut für Informatik, Saarbrücken

We consider the problem of processing a given number of tasks on a given number of processors as quickly as possible when the processing times of the tasks are variable and not known in advance. The tasks are assigned to the processors in *chunks* consisting of several tasks at a time, and the difficulty lies in finding the optimal tradeoff between the processors' load balance, which is favoured by having small chunks, and the total scheduling overhead, which will be the lower the fewer chunks there are. What is symptomatic for this kind of problem is that traditional theoretical approaches are inappropriate to explain the phenomenology that is observed in practice. Namely, assuming complete knowledge of the input, in this case the tasks' processing times, is not realistic. Instead doing away with this information completely, that is, considering online-algorithms, inhibits high performance. Probabilistic models can address the issue of incompleteness of information, but are problematic because of their inability to describe imprecise knowlegde, and hence their tendency to create an artifical regularity in the modelled behaviour. We present a novel approach that is based on two concepts: an *estimator*, which represents an algorithm's a priori knowledge, and a *deviation*, which measures the distance of the actual behaviour from this estimator and is not known until after the event. In this very general framework, we provide an upper bound applicable for every conceivable parameter setting, as well as lower bounds showing that no algorithm can do significantly better than the ones we propose. These results, on the one hand, clarify how the complexity of the considered problem depends on the completeness and the preciseness of a priori knowledge. On the other hand, they imply optimal bounds for a whole variety of specific settings, including such based on pobabilistic models.

On maximum weighted matching in general graphs

Norbert Blum, Universität Bonn

We reduce the problem of finding an augmenting path in a general graph to a reachability problem in a directed, bipartite graph and show that a slight modification of depth-first search leads to an algorithm for finding such paths. Furthermore, a variant of Edmonds' primal-dual method for the maximum weighted matching problem in general graphs is described in such a manner that no knowledge of duality theory or linear programming is needed.

Online Routing in Triangulations

Prosenjit Bose, School of Computer Science, Carleton University

We consider online routing algorithms for routing between the vertices of embedded planar straight line graphs. Our results include (1) two deterministic memoryless routing algorithms, one that works for all Delaunay triangulations and the other that works for all regular triangulations, (2) a randomized memoryless algorithm that works for all triangulations, (3) an $O(1)$ memory algorithm that works for all convex subdivisions, (4) an $O(1)$ memory algorithm that approximates the shortest path in Delaunay triangulations, and (5) theoretical and experimental results on the competitiveness of these algorithms.
This is joint work with Pat Morin.

Dynamic Planar Convex Hull

Gerth Stølting Brodal, BRICS, University of Aarhus

The dynamic maintenance of the convex hull of a set of points in the plane is one of the most important problems in computational geometry. We present a data structure supporting point insertions in $O(\log n \cdot \log\log\log n)$ amortized time, point deletions in $O(\log n \cdot \log\log n)$ amortized time, and various queries about the convex hull in $O(\log n)$ worst-case time. The data structure requires $O(n)$ space. Queries supported are: find the extreme point on the convex hull in a given direction; report whether a given line intersects the convex hull; report if a given point is contained in the interior of the convex hull; find the two points adjacent to a point on the convex hull; and given an exterior point find the two tangent points on the convex hull from the point. The data structure improves the previously best update time due to Chan by a factor $\log^{\varepsilon} n / \log\log n$. A component of the data structure is a new data structure for the semi-dynamic case where only deletions are allowed after preprocessing. Given a lexicographically sorted set of n points the preprocessing takes $O(n)$ amortized time, and point deletions $O(\log n \cdot \log\log n)$ amortized time. Applications of the new dynamic convex hull data structure are improved deterministic algorithms for the k-level problem and for the

red–blue segment intersection problem where all red and all blue segments are connected.
Joint work with Riko Jacob, BRICS.

Range Searching Over Tree Cross Products

Adam L. Buchsbaum, AT&T Labs–Research

We formalize the tree cross-product problem. Given are two trees, T_1 and T_2, and a ground set $E \subseteq V(T_1) \times V(T_2)$. The problem is to perform online queries of the form (x, y), where $(x, y) \in V(T_1) \times V(T_2)$. Each query returns information about the supporting edges, if any, of (x, y), i.e., elements $(x', y') \in E$ such that $x' \in \mathrm{desc}(x) \wedge y' \in \mathrm{desc}(y)$. This problem arises in a graph visualization application, in text indexing with one error, and in software system analysis. We give data structures that solve the problem with a variety of tradeoffs, yielding improvements in the first two applications and new results in the third.
Joint work with Michael T. Goodrich (Johns Hopkins University) and Jeffery R. Westbrook (20th Century Television).

Parallelizing The Data Cube

Frank Dehne, School of Computer Science, Carleton University

This paper presents a general methodology for the *efficient parallelization of existing data cube construction algorithms.* We describe two different partitioning strategies, one for top-down and one for bottom-up cube algorithms. Both partitioning strategies assign subcubes to individual processors in such a way that the loads assigned to the processors are balanced. Our methods reduce inter-processor communication overhead by partitioning the load in advance instead of computing each individual group-by in parallel as is done in previous parallel approaches. In fact, after the initial load distribution phase, each processor can compute its assigned subcube without any communication with the other processors. Our methods enable code reuse by permitting the use of existing sequential (external memory) data cube algorithms for the subcube computations on each processor. This supports the transfer of optimized sequential data cube code to the parallel setting. Note that, such code reuse is not possible for existing parallel solutions which parallelize each individual group-by computation.

The bottom-up partitioning strategy balances the number of single attribute external memory sorts made by each processor. The top-down strategy partitions a weighted tree in which weights reflect algorithm specific cost measures like estimated group-by sizes. These are the cost measures used by the original proponents of bottom-up and top-down data cube algorithms. Both partitioning approaches can be implemented on any *shared disk* type parallel machine composed of p processors connected via an interconnection fabric and with access to a shared parallel disk array. Experimental results presented show that our partitioning strategies generate a close to optimal load balance between processors.
This is joint work with S. Hambrusch and A. Rau-Chaplin

Adaptive Set Computations

Erik D. Demaine, Dept. of Computer Science, University of Waterloo

Motivated by boolean queries in text database systems, we consider the problems of finding the intersection, union, or difference of a collection of sorted sets. While the worst-case complexity of these problems (in the comparison model) is straightforward, we consider a notion of complexity that depends on the particular instance. We develop the idea of a proof that a given set is indeed the correct answer. We present adaptive algorithms that make no a priori assumptions about the problem instance, and show that their running times are within a constant factor of optimal with respect to a natural measure of the difficulty of an instance. In the process, we develop a framework for designing and evaluating adaptive algorithms in the comparison model.
This is joint work with Alejandro Lopez-Ortiz and J. Ian Munro, and it appears in the proceedings of SODA 2000 (11th Annual ACM-SIAM Symposium on Discrete Algorithms).

(Simple) Minimal Perfect Hashing in Less Space

Martin Dietzfelbinger, Ilmenau Technical University

A minimal perfect hash function for a set $S \subseteq U$, $|S| = n$, where U is the "universe" of all keys, is a one-to-one onto function from S to $\{0, \ldots, n-1\}$. A family of minimal perfect hash functions for U and n is a set $\mathcal{H}$ of functions

from U to $\{0,\ldots,n-1\}$ with the property that for every $S \subseteq U$, $|S| = n$, there is some $h \in \mathcal{H}$ that is one-to-one on S.
R. Pagh [WADS'99] described a very elegant construction of such families. A function from the class can be evaluated by a few arithmetic operations and one table lookup; more precisely, it is given by the formula

$$h(x) = (f(x) + d_{g(x)}) \bmod n,$$

where $d_0, \ldots, d_{m-1}$ are $m = (2+\varepsilon)n$ numbers in $\{0,\ldots,n-1\}$ and f and g are functions chosen from some universal class of hash functions mapping U into $\{0,\ldots,n-1\}$ and $\{0,\ldots,m-1\}$, resp.
The construction is motivated by the displacement method introduced by Tarjan and Yao in 1979.
(It is known that the asymptotically optimal description size of minimal perfect hash functions is $\Theta(n + \log\log|U|)$ bits. No construction that yields functions with this description size and as simple as those described by Pagh is known, though.)
We modify Pagh's approach so as to obtain minimal perfect hash functions of the same structure whose description essentially consists of $m = (1+\varepsilon)n$ numbers. Partly, the improvement in the constant (down to about $m = 1.44n$) is due to the use of stronger universal hash classes and a modified analysis, but the main contribution of this work is a modified algorithm for finding the coefficients $d_0, \ldots, d_{m-1}$, and a new analysis.
Joint work with Torben Hagerup.

The Complexity of Rebalancing a Binary Search Tree

Rolf Fagerberg, Aarhus University Dept. of Computer Science BRICS

For any function f, we give a rebalancing scheme for binary search trees which uses amortized $O(f(n))$ work per update while maintaining a height bounded by $\lceil \log(n+1) + 1/f(n) \rceil$. This improves on previous algorithms for maintaining binary search trees of very small height, and matches an existing lower bound. The main implication is the exact characterization of the amortized cost of rebalancing binary search trees, seen as a function of the height bound maintained. We also show that in the semi-dynamic case, a height of $\lceil \log(n+1) \rceil$ can be maintained with amortized $O(\log n)$ work per

insertion. This implies new results for *TreeSort*, and proves that it is optimal among all comparison based sorting algorithms for online sorting.

An Optimal Data Structure for the Static Predecessor Problem

Faith E. Fich, University of Toronto

A new way of representing a set S of n elements from the universe $\{0, \ldots, N-1\}$ is presented. It supports predecessor queries, i.e. searching for the largest element of S that is smaller than a given input $x \in U$, in worst case time

$$O\left(\min\left\{\frac{\log\log N}{\log\log\log N}, \sqrt{\frac{\log n}{\log\log n}}\right\}\right).$$

This result improves previous upper bounds and matches matches our lower bound for the problem.
This work is joint with Paul Beame.

Balanced k-Colorings

Rudolf Fleischer, Department of Computer Science, University of Waterloo

While discrepancy theory is normally only studied in the context of 2-colorings, we explore the problem of k-coloring, for $k \geq 2$, a set of vertices to minimize imbalance among a family of nonempty subsets of vertices. The *imbalance* is the maximum, over all subsets in the family, of the largest difference between the size of any two color classes in that subset.
The *discrepancy* is the minimum possible imbalance. We show that the discrepancy is always at most $4d-3$, where d (the "dimension") is the maximum number of subsets containing a common vertex. For 2-colorings, the bound on the discrepancy is at most $\max\{2d-3, d\}$.
In a geometric setting, where vertices are points and subsets are lines, we show that the 2-color discrepancy is at most $2d-3$ where d is the number of distinct line directions. Finally, we prove that several restricted versions of computing discrepancy are NP-complete.
Joint work with Therese C. Biedl, Eowyn Čenek, Timothy M. Chan, Erik D. Demaine, Martin L. Demaine and Ming-Wei Wang.

A smooth start and finish of quicksort

Martin Fürer, Pennsylvania State University

An efficient version of quicksort is presented where the work to finish with insertion sort is done earlier, resulting in an improved splitting procedure. For a random input sequence, the expected number of comparisons is $\log n! + o(n \log n)$. The resulting algorithm is quite simple, and its speed seems to compare well with every existing variant of quicksort. If there are only $m < n$ distinct keys, then there is no need to exchange keys equal to the pivot, and the running time is still only $O(n \log m)$ rather than $O(n \log n)$ or $O(n^2)$.

Data Structures for Strings

Roberto Grossi, Universita degli Studi di Pisa

The proliferation of online text, such as on the World Wide Web and in databases, motivates the need for space-efficient index methods that support fast search.
Consider a text T of n binary symbols to index. Given any query pattern P of m binary symbols, the goal is to search for P in T quickly, with T being fully scanned only once, namely, when the index is created. All indexing schemes devised in the last thirty years support searching in $\Theta(m)$ worst-case time and require $\Theta(n)$ memory words (or $\Theta(n \log n)$ bits), which is significantly larger than the text itself.
In this talk we describe how to break through both searching time and index space under the same model of computation as the one adopted in previous work. Based upon new compressed representations of suffix arrays and suffix trees, we show how to construct an index structure that occupies only $O(n)$ bits and compares favorably with inverted lists in space. We can search any binary pattern P, stored in $O(m/\log n)$ words, in only $o(m)$ time. Specifically, we achieve optimal $O(m/\log n)$ search time for sufficiently large $m = \Omega(\log^{1+\epsilon} n)$, for any fixed $0 < \epsilon < 1$, and therefore provide a generalization of perfect hash and classical dictionary searching to full text searching. We can list all the occ pattern occurrences in optimal $O(\mathrm{occ})$ time when $m = \Omega(\mathrm{polylog}(n))$ (or $\mathrm{occ} = \Omega(n^\epsilon)$); otherwise, listing takes $O(\mathrm{occ} \log^\epsilon n)$ time.

SSSP as a jigsaw puzzle

Torben Hagerup, Universität Frankfurt

The key to a fast computation of shortest paths using Dijkstra's algorithm is an efficient priority queue (PQ). Recent developments have exploited the fact that Dijkstra's algorithm does not require a completely general PQ, but can work with a so-called monotonic PQ. We show how a visual representation of general and monotonic PQs, sorting algorithms, and reductions between PQs and sorting algorithms can help in designing and understanding new algorithms for computing shortest paths. Specifically, we give a simpler, more correct and unified treatment of four of the theoretically fastest currently known realizations of Dijkstra's algorithm.
Joint work with Rajeev Raman, King's College, London.

Similarity search in multimedia databases

Gisli R. Hjaltason, University of Maryland

Similarity search is a very important operation in multimedia databases and other database applications involving complex objects. Such search involves identifying objects that are similar to a target object, and is usually classified as being a range query or a nearest neighbor query. In this setting, the similarity between two objects is usually represented by a distance metric.
There are two approaches for handling similarity search. The first directly indexes the objects based on distances, while the second is based on mapping to a low-dimensional geometric space (making use of multidimensional indexes for the search).
We present a general framework for incremental nearest neighbor search, where by incremental we mean that neighbors are reported one-by-one in order of similarity, with as little effort as possible expended to produce each new result object.
We show how this framework can be applied to similarity search to obtain specific algorithms for both similarity search approaches. In the case of distance-based indexing, numerous data structures have been proposed.
We have applied our framework to many of them, including vp-tree, M-tree, sa-tree, and LAESA, in each case fully exploiting their special nature. In the case of the mapping-based approach, we obtained an algorithm that observes a filter and refine query processing strategy.

Exponentially Decreasing Rebalancing in Balanced Search Trees

Lars Jacobsen, University of Southern Denmark, Odense

We consider tree-like data structures and the following conditions: First, updates and rebalancing operations are local transformations, involving only a constant number of nodes. Second, rebalancing can be proven to be amortized constant using a potential function which only assigns potential to the nodes, and only constant potential to each node. Third, there exists a function, satisfying similar locality constraints, that assigns a layer to each node. If these conditions are fulfilled, then rebalancing is exponentially decreasing, meaning that the number of rebalancing operations carried out on any given layer decreases exponentially in the distance from the leaves. Our framework also seems to provide reasonable constants.
This is joint work with Kim S. Larsen.

LP-Formulations for Orthogonal Graph Drawing

Michael Kaufmann, Universität Tübingen

We review some recent LP-based approaches for orthogonal graph drawing and motivate why they may perform poorly as stand-alone products and why they should be combined. Then we propose a very general LP-formulation for orthogonal drawings with straight lines. Several refinements that include bends, planarity, crossings, orthogonal representations and angles lead to the previous approaches which now can be expressed in the new formulations. The progress of this approach is the possibility to combine several optimization criteria. The open problem is the efficiency of the approach, which should probably be elaborated and improved.
This is joint work with Markus Eiglsperger (Tübingen).

Reducing the diameter of networks

Manfred Kunde, TU Ilmenau

In an ATM network an express graph is built on the top of the physical network in order to make the network faster. An express edge (a, b) is built by connecting a physical path from a to b in the original network to one edge

in the modified network. By establishing several express egdes an embedded express graph is generated. The aim is to construct an express graph with a very small diameter.
We show that for an r-dimensional mesh with sidelength n and with $N = n^r$ there is an express graph with diameter of $2r \log n = 2 \log N$ which asymptotically matches the bisection bound. Further we can show that an arbitrary graph with N nodes and containing k directed Hamiltonian cycles has an express graph with a diameter of $O(kN^{1/k})$. The result can be extended to a class of graphs having bigger subgraphs with k Hamiltonian cycles.

Network Simplex Revisited - Simplified Data Structures and Implementation Through Very Lazy Evaluation

Ulrich Lauther, Siemens ZT SE 4, Germany

The implementation of the network simplex algorithm for solving minimum cost flow problems can be greatly simplified when node potentials are not maintained but calculated on the fly when needed (lazy evaluation). Instead of threaded indices which are usually used to maintain subtrees, just one pointer per node (to the incoming tree edge) is sufficient and the size of subtrees needs not to be stored and updated. Thus out of usually 7 fields per node 4 can be saved. Nevertheless, node potentials can be calculated in $O(1)$ time per node using a simple recursive function. Further speed up can be achieved by "very lazy evaluation": As most potentials do not change during an iteration, we can suspend the recalculation and use wrong values to find violating edges until "shortly before the algorithm terminates".

A combinatorial approach to protein docking with flexible side chains

Hans-Peter Lenhof, MPI Informatik, Saarbrücken, Germany

Rigid body docking approaches are not sufficient to predict the structure of a protein complex from the unbound (native) structures of the two proteins. Accounting for side–chain flexibility is an important step towards fully flexible protein docking. This work describes an approach that allows conformational flexibility for the side–chains while keeping the protein backbone rigid. Starting from candidates created by a rigid docking algorithm, we demangle the side–chains of the docking site, thus creating reasonable approximations of the true complex structure. These structures are ranked with respect to the binding free energy. We present two new techniques for side–chain demangling. Both approaches are based on a discrete representation of the side–chain conformational space by the use of a rotamer library. This leads to a combinatorial optimization problem. For the solution of this problem we propose a fast heuristic approach and an exact, albeit slower method using branch–&–cut techniques. As a test set we use the unbound structures of three proteases and the corresponding protein inhibitors. For each of the examples the highest–ranking conformation produced was a good approximation of the true complex structure.

On Roberto's First Problem

Alejandro Lopez-Ortiz, University of New Brunswick at Fredericton

Given a text string N bits long, we present a linear space lower bound for the size of an index that reports the location, if any, of an occurrence patther in the text in time proportional to the length of the the occurrence of the pattern. This lower bound applies to inverted word indices as well.
This work solves an open question posed by Roberto Grossi during the Dagstuhl open session of 24 February 2000.
This is joint work with Erik Demaine, University of Waterloo.

Semi-External-Memory DFS

Ulrich Meyer, Max-Planck Institut für Informatik, Saarbrücken

Given a graph G with n nodes and m edges, computing a depth first search tree with the associated DFS numbers can be done sequentially in $\mathcal{O}(n+m)$ time. However, in external memory computing where the graph is too big to fit into the main memory of size M, DFS constitutes a major bottleneck. All previous external DFS algorithms for general graphs follow the basic idea of the sequential approach and incur at least $\Theta(n)$ random I/Os in order to access the adjacency lists of nodes that are visited for the first time. Sometimes it is easier to develop I/O efficient algorithms for the practically important *semi-external* case. That is, $n \leq M/c$, for some appropriate constant $c \geq 1$ and $m > M$. Thus, some node related information can be kept internally,

but the number of edges is too big. However, the previous external DFS algorithms cannot take advantage of the semi-external setting and still need $\Theta(n)$ random I/Os.
We propose a new algorithm for semi-external unordered DFS. After an initial random permutation of the edges it uses one scan over the graph to build an internal tentative DFS tree. Then the algorithm keeps on scanning through the non-tree edges and modifying the internal tree until it eventually fulfills the DFS properties. No random I/Os are needed. The full potential of the approach is not yet completely determined, however on random graphs with n nodes and $m \geq n$ edges it requires $\mathcal{O}(m/B \cdot \log n)$ I/Os whp where B denotes the block size. Also on any other graph class we have tested including real-world data the performance is amazingly good. The worst (artificial) inputs we can construct either need expected $\mathcal{O}((n/\log n)^{1/4} \cdot m/B)$ I/Os for a simple modification rule or expected $\mathcal{O}((m/B) \cdot \log n)$ I/Os for a slightly extended modification rule.
Joint work with Jop Sibeyn, MPI für Informatik.

Caching in Networks

Friedhelm Meyer auf der Heide, Heinz Nixdorf Institute, Paderborn University, Germany

We present a general framework for the development of on-line algorithms for data management in networks with limited memory capacities. These algorithms dynamically create and delete copies of shared data objects that can be read and updated by the nodes in the network. Our algorithms aim to minimize the congestion, i.e., the maximum communication load over all network resources, so that that none of these resources can become a communication bottleneck.
We give several examples of networks for which our framework yields efficient algorithms, including meshes, fat-trees, hypercubic networks, and complete networks. For example, our framework yields an $O(d \cdot \log n)$-competitive caching algorithm for d-dimensional meshes of size n with respect to the congestion on the communication links, and an $O(1)$-competitive algorithm for complete networks with respect to the congestion at the memory modules due to remote accesses.
Previous work on data management in networks either neglects memory capacity constraints or minimizes only the total communication load, i.e., the

sum of the communication load over all resources, which may produce congestion as some of the links become bottlenecks.
(Joint work with Berthold Vöcking and Matthias Westermann)

On the Competitiveness of Linear Search

J. Ian Munro, Department of Computer Science, University of Waterloo

Offline techniques for linear search are considered; hence the competitiveness of selforganizing linear search, and in particular the move to front heuristic, is reopened. We consider a model under which each search must start at the beginning of the list and scan at least to the element requested. The process may scan further and then may reorder the portion of the list inspected. The cost of this task is taken to be the number of elements inspected. Indeed the technique actually used is easily implemented in linear time. This is sharp contrast with the model of Sleator and Tarjan who restrict reordering of the list to be performed by swapping adjacent elements at a cost of one per interchange.
Under the previously studied model, the move to front heuristic was shown to be 2-competitive with the best possible offline method. Under the model considered here, it is demonstrated that a simple offline heuristic, that involves scanning past the requested element by at most a factor of two then reordering the scanned elements in order of next request, gives an amortized cost within a constant factor of the entropy of the sequence of requests. That is, the amortized search cost is at most logarithmic, or indeed, comparable to that under an optimal binary search tree. This is in sharp contrast with the potentially linear amortized cost of the move to front heuristic. It follows that, under a very reasonable model, no online technique can have an amortized cost within a constant factor of that which one could obtain if given the request sequence in advance, i.e., there is no competitive linear search algorithm.

How to Swap a Failing Edge of a Single Source Shortest Path Tree

Guido Proietti, Universita degli Studi di l'Aquila

Let $S(r) = (V, E_S)$ be a single source shortest paths tree (SPT) rooted in $r \in V$ in a weighted, 2-edge connected graph $G = (V, E)$. Whenever an edge $e = (u, v) \in E_S$ fails, a subtree S_v of $S(r)$ is detached from the root r, and one could be interested in reconnecting S_v to r by means of a single edge e' crossing the cut created by the removal of e. Such an edge e' is named a *swap edge* for edge e. Let $S_{e/e'}(r)$ be the tree obtained by swapping e with e', and let $F[S_{e/e'}(r)]$ be a functional defined on $S_{e/e'}(r)$ that expresses some feature of the tree, such as the average length of a path from the root r to all the nodes in S_v. A *best swap edge* f for e with respect to F is a swap edge such that $F[S_{e/f}(r)]$ is minimum among all possible swap edges for e.
In this paper we address the problem of finding a best swap edge for each edge e of $S(r)$, with respect to several functionals, presenting efficient algorithms that are much faster than recomputing, a true new SPT, say $S'_e(r)$. Moreover, an accurate comparison between $S_{e/f}(r)$ and the corresponding $S'_e(r)$, in terms of the tree features captured by the considered functionals, shows that these algorithms are also effective, in the sense that the tree $S_{e/f}(r)$ they create is functionally very similar to $S'_e(r)$.

A lower Bound for Range Search on the Grid

Theis Rauhe, The IT University of Copenhagen

The talk addresses the problem of orthogonal range search on the grid. The problem is to maintain a subset S of points in the $[n] \times [n]$ grid, under the dynamic operations: Insert(p): Insert p into S. Delete(p): Delete p from S. Empty(R): For rectangle R $\subset [n] \times [n]$, return yes iff the intersection of R and S is empty.
In the talk I present a proof of a lower bound of $\Omega(\log^{1/2} n)$ per operation of the above problem on the unit-cost RAM with logartihmic word size. The proof uses a non-deterministic reduction of a partial sum problem proved hard on a non-determinstic variant of the cell-prope model due to Husfeldt & Rauhe (ICALP'98).

Approximation algorithms for geometric shortest path problems

Jörg-Rüdiger Sack, School of Comp. Sci., Carleton U. Ottawa

We consider the classical geometric problem of determining a shortest path through a weighted domain. We present approximation algorithms that compute *ε-short* paths, i.e., paths whose costs are within a factor of $1+\varepsilon$ of the shortest path costs, for an arbitrary constant $\varepsilon > 0$, for the following geometric configurations:

SPPS Problem: We are given a polyhedron $\mathcal{P}$ consisting of n convex faces and each face has a positive non-zero real valued weight. The *shortest path on polyhedral surface problem* (SPPS) is to compute a path of least cost that remains on the surface of $\mathcal{P}$ between any two vertices, where the cost of the path is defined to be the weighted sum of Euclidean lengths of the sub-paths within each face. Our algorithm runs in $O(\frac{n}{\varepsilon}\log\frac{1}{\varepsilon}(\frac{1}{\sqrt{\varepsilon}}+\log n))$ time for $0 < \varepsilon < 1$. The run time improves to $O(n\log n)$ for $\varepsilon \geq 1$ and to $O(\frac{n}{\varepsilon}\log\frac{1}{\varepsilon}\log n)$ when all weights are equal.

WRP-3D Problem: We are given a subdivision of $\Re^3$ consisting of n convex regions. Each face has associated with it a positive non-zero real valued weight. The *shortest path problem in three dimensions* (SP3D) is to compute a path of least cost between any two vertices, where the cost of the path is defined to be the weighted sum of Euclidean lengths of the sub-paths within each region. We present the first polynomial time approximation scheme. Our algorithm runs in $O(\frac{n}{\varepsilon^3}\log\frac{1}{\varepsilon}(\frac{1}{\sqrt{\varepsilon}}+\log n))$ time. The run time improves to $O(\frac{n}{\varepsilon^3}\log\frac{1}{\varepsilon}\log n))$ when all weights are equal. This can used to solve the shortest path problem amidst obstacles in 3-dimensional Euclidean space (ESP-3D).
Joint work with A. Maheshwari and L. Aleksandrov.

Visualizing Spatial Data Structures

Hanan Samet, University of Maryland

A demonstration of the effects of adding a MOVE operation to the VASCO system of spatial data structure applets written in JAVA was presented. The VASCO system enables users on the worldwide web to experiment with over

30 different spatial data structures for spatial data comprising points, line segments, and rectangles. The system supports insertion and deletion of data as well as basic search operations such as nearest neighbor and range queries of arbitrary shape and permits users to visualize the mechanics of these operations in an animated manner. The spatial data structures include many variants of quadtrees, k-d trees, R-trees, range trees, and priority search trees. These data structures find use in many applications including geographic information systems (GIS), computer graphics, and image processing. The addition of the MOVE operation enables users to change the position and sizes of individual spatial objects (e.g., line segments and rectangles) thereby permitting them to visualize the effect of their positions on the different data structures. Some of the main issues that are addressed include how to display the changes and how to keep track of the changing information so that the effect of the modifications on the different data structures can be compared and understood better. The applets can be found at http://www.cs.umd.edu/ hjs/quadtrees/index.html
Joint work with Frantisek Brabec.

Fast Priority Queues for Cached Memory — a Case Study in Algorithm Engineering

Peter Sanders, Max-Planck Institut für Informatik, Saarbrücken

The cache hierarchy prevalent in todays high performance processors has to be taken into account in order to design algorithms which perform well in practice. We advocates the approach to adapt external memory algorithms to this purpose. We exemplify this approach and the practical issues involved by engineering a fast priority queue suited to external memory and cached memory which is based on multi-way merging. For any sequence of operations containing I insertions the data structure needs $2I/B \log_{M/B} + \mathcal{O}(1)$ I/Os and $I \log I + \mathcal{O}(1)$ key comparisons where M is the size of the fast memory and B the size of a memory block. This improves previous external memory algorithms by constant factors crucial for transferring it to cached memory. Running in the cache hierarchy of a workstation a careful implementation of the algorithm is up to 4.7 times faster than an optimized binary heap implementation.

Lower Bounds For Randomized Searching on m Rays

Sven Schuierer, Institut für Informatik, Uni Freiburg

In this talk we consider the problem of on-line searching on m rays. A point robot is assumed to stand at the origin of m concurrent rays one of which contains a goal G that the point robot has to reach. Neither the ray containing G nor the distance to G are known to the robot. The only way the robot can detect G is by reaching its location. We use the *competitive ratio* to measure the performance of a strategy. We present a simple proof of a tight lower bound for randomized strategies to search on m rays.

Visualizing the Analysis of Algorithms

Robert Sedgewick, Princeton University

Visual representations of mathematical functions have played an important role in understanding their properties since antiquity. In this talk, we examine some examples of such representations in the analysis of algorithms and data structures, and discuss ways in which the software and hardware in everyday use makes it easy to develop useful pictures of the functions that we study. Included are several examples of PostScript programs that produce detailed visual representations of the performance properties of fundamental algorithms.

Approximating the stretch factor of Euclidean graphs

Michiel Smid, University of Magdeburg, Germany

There are several results available in the literature dealing with efficient construction of t-spanners for a given set S of n points in R^d. t-spanners are Euclidean graphs in which distances between vertices in G are at most t times the Euclidean distances between them; in other words, distances in G are "stretched" by a factor of at most t. We consider the interesting dual problem: given a Euclidean graph G whose vertex set corresponds to the set S, compute the stretch factor of G, i.e., the maximum ratio between distances in G and the corresponding Euclidean distances. It can trivially be solved by solving the All-Pairs-Shortest-Path problem. However, if an approximation to the stretch factor is sufficient, then we show it can be efficiently computed by making only $O(n)$ approximate shortest path queries in the graph G. We

apply this surprising result to obtain efficient algorithms for approximating the stretch factor of Euclidean graphs such as paths, cycles, trees, planar graphs, and general graphs. The main idea behind the algorithm is to use Callahan and Kosaraju's well-separated pair decomposition.
(joint work with Giri Narasimhan, The University of Memphis)

Generalization of the Minkowski Algebra and Its Applications

Kokichi Sugihara, Dept. of Math. Eng. and Inform. Physics, University of Tokyo

The two-dimensional Minkowski sum of figures is reformulated and generalized in the way that the sum operation is always invertible. The main idea is to represent a figure in terms of its boundary in a parametric form such that the value of the parameter represents the tangent direction of the boundary curve. The Minkowski sum of two such curves is defined as the vector sum at each value of the parameter. Then the resulting algebraic system can be extended to a large system in the same way as the natural numbers with multiplication is extended to the rational numbers. The elements is this large system is called "hyper-figures"; they all have their inverses and hence the sum and its inverse can always be computed in a well-defined manner. Practical interpretation of a hyper figure is also given.

To Seek Or Not To Seek: Random I/O Operations in Spatial Join Algorithms

Jan Vahrenhold, University of Münster, Germany

We study the effects of random and sequential I/O operations in spatial join algorithms. Furthermore, we develop a simple plane-sweep approach to the spatial join problem that unifies the data structure based and non-data structure based approaches and can be built from a few standard operations. We also report on the results of a comparative study of the new algorithm with several data structure based and non-data structure based spatial join algorithms. The experiments have been performed on several different architectures, and scaled up to data sets that are significantly larger than those used in previous work. We considered a number of factors, including the rel-

ative performance of CPU and disk, the quality of the spatial indices used, and the sizes of the input relations.
This is joint work with L. Arge, O. Procopiuc, T. Suel, S. Ramaswamy, and J. Vitter.

Multiresolution Approximation Techniques for Database Systems

Jeff Vitter, Duke University

Approximation has been an area of great interest and importance in database systems, a classic example being selectivity estimation. A more recent example involves approximate answers for OLAP (On-Line Analytical Processing) queries in data warehousing applications. In this talk we discuss some novel approximation techniques in these two areas. Our techniques are based on the wavelet technique of multiresolution decomposition and are fundamentally different from traditional histogram-based approaches. Experiments indicate that our methods offer substantial improvements in accuracy and efficiency over existing methods. We show how to construct the wavelet decomposition efficiently in terms of I/O by making use of sparse matrix techniques. Our online query processing algorithm is very fast and capable of refining answers as the user demands more accuracy. We also discuss current work involving the problems of dynamic update and of computing the wavelet decomposition of the dense partial sum array when the underlying raw data are sparse.

Dual Search Strategies on Planar Graphs

Dorothea Wagner, Universität Konstanz, Germany

We study search strategies like depth first search (DFS) and breadth first search (BFS) on a planar embedded graph and its dual graph. Let us consider the result of a search strategy to be just a numbering and an orientation of edges. Then a search σ on a planar embedded graph G has a dual search if there is a search σ^* in the dual graph G^* that visits the edges in the same ordering as the edges in G are visited by the search. We are interested in the following questions.
1. When does a dual search exist?

2. How can we formulate the dual search for search strategies like DFS and BFS?
The first question can be answered easily. To answer the second question, we consider natural versions of DFS and BFS on planar graphs that select in each search step the rigthmost resp. leftmost unvisited edge. It turns out that "right–first–DFS" and "left–first–BFS" are not dual in general. We characterize the class of graphs where these search strategies are dual and give a modifications of "left–first–BFS" and "right–first–DFS" that are dual to "right–first–DFS" and "left–first–BFS" respectively. Moreover, we study the "right–first-DFS" tree and the dual of its non–tree edges.
This is joint work with Ulrik Brandes and Vanessa Kääb.

On the analysis of evolutionary algorithms

Ingo Wegener, Universität Dortmund, Germany

The design of algorithms and data structures is influenced by the aim to analyze their behavior. Evolutionary algorithms and other randomized search heuristics like simulated annealing are designed as "robust" algorithms with a "quite good" behavior for many problems "typical in applications". They are useful in particular in situations where the target function is not given but can only be sampled (black box optimization). With three results it is argued why the analysis of evolutionary algorithms leads to interesting problems.
1.) Even the analysis of the simplest evolutionary algorithm, the (1+1) EA, on arbitrary linear functions defined on $0, 1^n$ is difficult. The expected run time for the mutation probability 1/n is proved to be equal to $\Theta(n \log n)$.
2.) A simple dynamic (1+1) EA with varying mutation probability may outperform all static (1+1) EAs with fixed mutation probability but it may also cause unneccessary exponential run times.
3.) Crossover leads to populations whose individuals depend on each other in a complicated way (quadratic dynamical system). Hence, evolutionary algorithms with crossover are hard to analyze. Nevertheless, it can be proved for some specific function that a steady-state GA with crossover takes polynomial time while "all" evolutionary algorithms without crossover take superpolynomial time.
(Result 1 together with S. Droste and T. Jansen, Results 2 and 3 together with T. Jansen.)