The design and analysis of algorithms is one of the fundamental areas in computer science. This also involves the development of suitable methods for structuring the data to be manipulated by these algorithms. In this way, algorithms and data structures form a unit and the right choice of algorithms and data structures is a crucial step in the solution of many problems. For this reason, the design, analysis and implementation of data structures form a classical field of computer science both in research and teaching.

The development of the research in this area has been influenced by new application fields such as CAD, geographic information systems, molecular biology and genetics. Not only new methods and paradigms, such as randomization or competitive anlysis of algorithms, have been developed, but there is also some shift of interest away from theory, e.g., the classical analysis of asymptotic behavior of algorithms, to more practical issues, such as implementation problems and the usefulness of algorithms in practical applications. One can observe that more and more researches in computer science also want to make their results available to the computer science and programming community in form of programs or software packages. This trend is also reflected in important international conferences.

The workshop brought together researchers working on different areas of the field of efficient data structures and algorithms from all over the world. Many new interesting results and solutions for theoretical and practical problems were presented. The organizers want to thank all participants and especially the team of Schloß Dagstuhl for helping to make the workshop a success.

Faster Deterministic Sorting and Searching in Linear Space

Arne Andersson

Department of Computer Science, Lund University

Abstract

We present a significant improvement on linear space deterministic sorting and searching. On a unit-cost RAM with word size w, an ordered set of nw-bit keys (viewed as binary strings or integers) can be maintained in

$$O\left(\min\left(\sqrt{\log n}, \frac{\log n}{\log w} + \log\log n, \log w \log\log n\right)\right)$$

time per operation, including insert, delete, member search and neighbor search. The cost for searching is worst-case while the cost for updates is amortized. For range queries, there is an aditional cost of reporting the found keys. As an application, n keys can be sorted in linear space at a worst-case cost of $O(n\sqrt{n})$.

The best previous method for deterministic sorting and searching in linear space has been the fusion trees which supports queries in $O(\log n/\log\log n)$ amortized time and sorting in $O(n\log n/\log\log n)$ worst-case time.

We also make two minor observations on adapting our data structure to the input distribution and on the complexity of perfect hashing.

Weighted Matching in Nonbipartite Graphs by the Hungarian Method

Norbert Blum

Universität Bonn

Abstract

A weighted matching algorithm for nonbipartite graphs which mimics the Hungarian method for the assignment problem is presented. The algorithm does not use linear programming and can be implemented such that its runtime is not worse than the runtimes of the best previous algorithms. The goal of the approach is to simplify the used techniques and to make weighted matching in nonbipartite graphs more understandably.

Priority Queues on Parallel Machines

Gerth Stolting Brodal

Abstract

Time and work optimal priority queues for the CREW PRAM are presented. Our priority queues support FINDMIN in constant time with one processor, and MAKEQUEUE, INSERT, MELD, FINDMIN, EXTRACTMIN, DELETE and DECREASEKEY in constant time with $O(\log n)$ processors. A priority queue can be built in time $O(\log n)$ with $O(n/\log n)$ processors. In parallel k elements can be inserted into a priority queue in time $O(\log k)$ with $O((\log n + k)/\log k)$ processors. With a slowdown of $O(\log\log n)$ in time the priority queues adopt to the EREW PRAM without increasing the work. A pipelined version of our priority queues can be implemented on a procesor array of sizes $O(\log n)$, supporting the operations MAKEQUEUE, INSERT, MELD, FINDMIN, EXTRACTMIN, DELETE and DECREASEKEY in constant time.

A Data Structure for the Closest Point Problem

Andrej Brodnik

Abstract

We adress the problem of a data structure representing points on an MxM grid ($M = 2^m$, where m is size of a word) that permits to answer the question of finding the closest point to a query point under the $L\infty$ or L_1 norm in constant time. Our data structure takes essentially minimum space. These results are extended to d dimensions under $L\infty$.

This work was done mostly while being a student at University of Waterloo (joint work with Ian Munro).

Some Algorithmic Problems in Computer Networking

Swante Carlsson

Lulea University and CTD

Abstract

We present some problems from the Internet protocol community where the use of efficient advanced data structuring and algorithmic techniques will be benificial. These problems are:

1) Advanced reservation: How can we handle bandwidth reservations on the Internet and combine this with a predicted service.

2) Fair queueing: How could we handle the problem of sharing the sending capacity of a congsted router with using minimal time overhead

3) Connection ID selection: Each package of a data flow at the Internet must contain an identification tag. We are studying the problem of selecting the smallest possible such tag for a large number of flows.

4) Connection ID compression: A variant of the previous problem is discussed. We would like to give a shorter (in the number of bits) tags to frequent senders than to other flows.

5) Environmental IT: By scheduling our task in an efficient way we could save batteries on mobile computers. This also has other applications.

Compact Suffix Trees

David R. Clark

Abstract

We present a new representation for Suffix Trees that occupies little more storage than a suffix array but retains the efficiency and functionality of the suffix tree. The representation is based on the compact traversable tree encodings developed by Guy Jacobson and some simple observations on the PAT tree representation for suffix trees of Gonnet et al. The total storage used by the new representation is $(3 + \log n + \log\log\log n \pm k)n$ bits where k is used for a time-space tradeoff but is typically in $[-s, s]$. We then show a

min-max optimal page partitioning of the structure that allows very efficient string searching on secondary storage and is well suited to CD-ROM.

This is joint work with Ian Munro

From Parallel Time Complexity To Parallel Time: On the Design of Efficient Scalable Parallel Programs.

Frank Dehne

School of Computer Sciece, Carleton University, Ottawa

Abstract

It is well known that theoretical time complexity results for the PRAM as well as other fine grained parallel machine models are not always reflected in the running times observed in actual implementations. We study the design of efficient scalable parallel programs for coarse grained multiprocessors. We present techniques which result in parallel algorithms that are good on paper as well as in real implementations. Many of our methods also have the advantage of being portable across a variety of different platforms without loss in speedup observed.

Universal hashing and k-wise independent random variables via integer arithmetic without primes

Martin Dietzfelbinger

Abstract

Let $u, m \geq 1$ be arbitrary integers and let $r \geq um$ be any multiple of m. The basis result of this work is that the multiset $\mathcal{H} = \{h_{a,b} \mid 0 \leq a, b < um\}$ of functions from $U = \{0, \ldots, u-1\}$ to $M = \{0, \ldots, m-1\}$, where

$$h_{a,b}(x) = ((ax + b) \bmod r) \text{ div } (r/m), \text{ for } x \in U,$$

is a $(c, 2)$-universal class of hash functions from U to M in the sense of Carter and Wegman (1979), with $c = 5/4$. More precisely, we show that

$$\left|\text{Prob}\,(h(x_1) = i_1 \wedge h(x_2) = i_2) - 1/m^2\right| \leq (u/2r)^2 \leq 1/4m^2,$$

for distinct $x_1, x_2 \in M$ and arbitrary $i_1, i_2 \in M$. Among the many known constructions of $(c, 2)$-universal classes there was none that would get by with pure integer arithmetic without assuming that a prime number of the order of $|U|$ or at least $|M|$ was given.

Varying this result, we obtain

(1) two-independent (or almost two-independent) sequences of random variables;

(2) universal hash classes of higher degree ("(c, k)-universal" classes) and k-wise independent random variables, for $k \geq 2$;

(3) algorithms for static and dynamic perfect hashing with an optimal number of random bits,

all using pure integer arithmetic without the need for providing prime numbers (arbitrary or even random) of a certain size. Our results may be helpful in practical applications of hashing or two-independent sampling as well as in theoretical applications of two-independent sequences. It should be noted that the focus here is not on minimizing the size of the probability space used, as in much of the recent work on "almost k-independent random variables" but on the realization of such variables or hash classes using the most natural and most widely available operations, viz., integer arithmetic. Incidentally, our construction provides $(1, 2)$-universal hash classes with the smallest known circuit complexity and, using a result by Mansour, Nisan, Tiwari (1993), yields the best known time-space tradeoff for multiplication of integers in binary representation.

(Appeared in the proceedings of STACS 96, Springer LNCS 1046.)

A Tight Layout of the Butterfly Network

Shimon Even

Technion

Abstract

We establish upper and lower bounds on the layout area of the butterfly network, which differ only in low-order terms. Specifically, the N-input, N-output butterfly network can be laid out in area $(1+o(1))N^2$, while no layout of the network can have area smaller than $(1-o(1))N^2$. These results improve both the known upper bound on the area of butterfly network layouts.

Joint work with Avior, Calamoneri, Litman, and Rosenberg.

Binary Search Trees: How Low Can You Go ?

Rolf Fagerberg

Odense University, Denmark

Abstract

We prove that any algorithm maintaining binary search trees during insertions and deletions while guaranteeing a height bounded by $\lceil \log(n+1) + 1/f(n) \rceil$ for some function f in $O(n)$, can be forced to do amortized $\Omega(f(n))$ restructuring work per update. We improve the existing upper bounds by showing that height $\lceil \log(n+1) + \log^3(f(n))/f(n) \rceil$ is maintainable in amortized $O(f(n))$ restructuring work per update, thus almost matching our lower bound. We also improve the existing upper bounds for worst case algorithms, and give a lower bound for the semi-dynamic case.

New Heuristics for the Protein Multiple Alignment Problem

Gaston H. Gonnet

ETH Informatik Zürich

Abstract

Pairwise alignment of proteins has been well understood for two and a half decades. Pairwise alignments are mostly based on dynamic programming. For multiple alignments, 3 or more sequences, the situation is much worse. The problem is very hard, exact solutions require exponential time and heuristics are few and generally unsatisfactory. On the other hand, to the expert "eye" the multiple alignment problem does not appear very difficult and it is often the case that humans can produce better alignments than computers.

We have used until now (in our automatic server at ETH) an algorithm based on a correct phylogenetic tree which does prohabilistic dynamic programming at each internal mode of the tree. When this is done, in order from the leaves towards the roof, the last alignment produces a desired multiple alignment. But this procedure depends on a correct phylogenetic tree, which is not always available, and when it has minor errors, it can potentially distort the final result.

We present a scaring function, which is originally intended to rate how good the multiple alignments are. This scaring function can be computed independently of a phylogenetic tree. This function is theoretically sound and proven efficient and accurate in practice. The main point of the presentation is that once that we have and trust such a scaring function, the optimization of this function hads to new and quit simple algorithms and heuristics to solve the M.A. problem.

This represents work in progress.

Adaptive Dynamic Dictionaries

Torben Hagerup

MPI für Informatik Saarbrücken

Abstract

We develop new schemes for universal hashing that extend existing schemes by allowing the universe from which keys are drawn to be an infinite set of all natural numbers. Previous schemes work only with a finite universe that must be specified in advance. As a result, previously known dynamic dictionary algorithms are all restricted, in that the keys to be inserted must be taken from an a priori known bounded universe. Our results lead to dynamic dictionary data structures for unbounded universes that match the performance bounds of their counterparts for bounded universes. Our schemes are fully adaptive in that the cost of hashing an arbitrary key is a function of the size of that key alone, and not of a hypothetical limit ("universe size") or even of the largest key seen before it. This may be significant when keys are allowed to be long strings, but long strings occur infrequently.

Joint work with Martin Dietzfelbinger and Yossi Matias.

Exact Solutions for the Rectilinear Steiner Tree Problem

Michael Kaufmann

Universität Tübingen

Abstract

The Rectilinear Steiner Tree Problem asks for a shortest tree connecting given points in the plane with rectilinear distance. The best theoretically analysed algorithms for this problem are based on dynamic programming and have a running time of $O(n^2 \cdot 2.62^n)$ (Ganley/Cohoon) resp. $n^{o(sqrtn)}$ (Smith). The best implementations perform pooly even on small problem instances, they can solve random problems of size 27 resp. 35 (Salowe/Warme) within a day. In this talk we present an improvement of the theoretical worst case time

bound to $O(n \cdot 2.38^n)$, for random problem instances we achieve a running time of less than 2^n.

This is joint work sith Ulli Fößmeier.

Pagination Reconsidered

Rolf Klein

FernUniversität Hagen

Abstract

Findind an admissible pagination of a document requires to position text blocks and floating objects like figures, tables etc., on pages such that the following conditions are met. The order of text blocks and figures must be preserved, no figure must appeare before the text block in which it is cited (for the first time), and the contents of a page must be within certain limits.

In addition, a good pagination should place figures close to their citations, in order to make the document easy to read. M. Plass has shown that finding an optimal pagination is an NP-complete problem. However, he was using as objective function the sum of the squares of the distances, counted in pages, between the figures and their citations. We feel that this badness function puts too much weight on single figures being far away. If, instead, a linear badness function is used, dynamic programming can be applied, and an optimal pagination can be found in time $O(mn)$, where n denotes the number of figures, and m the number of text blocks. But adding up the citation-figure distances is not satisfying in practice as it tends to produce loosely filled pages. Therefore, we suggest to count the total number of page turns that are required for reading the document and looking up the figures, with a variable weight on either constituent part. Experiments show that the resulting layouts are of high quality.

This is joint work with Anne Brüggemann-Klein and Stefan Wohlfeil.

Routing and Sorting on Reconfigurable Meshes with Bounded Bus Length

Manfred Kunde

Technical University of Munich
Technical University of Ilmenau

Abstract

Two fundamental problems on nets of processors are discussed in context: sorting and packet routing. We start with so-called $h-h$ problems on grids where each processor contains at most h packets, initially and finally. We present a method where in the first phase the data are locally ordered to some order criteria and then distributed uniformly over the whole grid. Using this method we obtain the deterministic sorting and routing procedures which asymptotically match the simple bisection bound on grids of arbitrary dimensions. The algorithm was also successfully implemented on reconfigurable meshes where packets can be transported over arbitrarily long distances in one step. To overcome this unrealistic model assumption we introduce reconfigurable meshes with bus length bounded by a constant. It turns out that $h-h$ problems, $h \geq 1$, can be solved within $hn + o(n)$ steps on r-dimensional reconfigurable meshes with side length n and with bus length at most $4r$, $r \geq 1$.

New Contact Measures for the Protein Docking Problem

Hans-Peter Lenhof

MPI für Informatik

Abstract

Docking reactions play an important role in a large number of biochemical processes. Although the mechanisms of docking reactions are not well understood, two complementarity principles seem to be important for the recognition and binding of docking partners. The first principle is the “shape

complementarity principle". The shapes of the molecules that build a docking complex are (locally geometrically) complementary, that is, there is a large fit between the surfaces of the docking partners. The second complementary principle is the "chemical principle". It states that there is a strong chemical complementarity (with respect to hydrogen bonds, electrostatic interactions, hydrophobicity and so on) between sites of docking partners. We introduced new "complementarity measures" for the geometric and the chemical complementarity that count and evaluate "van der Waals contacts" between the atoms the (two) molecules. Real world experiments with these new measures produced docking results of excellent quality.

Experimental Methods for Algorithm and Data Structure Analysis

Catherine Cole Mc Geoch

Abstract

There is a growing body of experience to suggest that computational experiments can provide deep new insights for algorithms and data structure analysis. But very little is known about proper experimental methods and techniques for the problems that typically arise in algorithm analysis. This talk focuses on two areas of experimental methodology that can have significant impact on the quality of insight gained. The first area concerns techniques for deciding which quantity to measure; the second concerns techniques for finding asymptotic bounds in experimental data sets.

The second part is joint work with Paul Cohen.

The talk concludes with a discussion of the DIMACS Challenge, which covers priority queues, dictionaries and multi-dimensional point sets.

Checking Geometric Programs or Verification of Geometric Structures

Kurt Mehlhorn

MPI für Informatik, Saarbrücken

Abstract

A program checker verifies that a particular program execution is correct. We give simple and efficient program checkers for some basic geometric tasks. We report about our experiences with program checking in the context of the LEDA system. We discuss program checking for data structures that have to rely on user-provided functions.

This is joint work with Stefan Näher, Michael Seel, Raimund Seidel, Thomas Schilz, Stefan Schirra, and Christian Uhrig

Instruction Set Dependent Lower Bounds for the Static Dictionary Problem

Peter Bro Miltersen

Abstract

A **classical** RAM is a RAM with direct and indirect adressing, conditional jump, addition, subtraction and multiplication.

A **practical** RAM is a RAM with direct and indirect adressing, conditional jump, addition, bituise Bookan oprutions and shifts.

The static dictionary problem is the task of storing a set $S \subseteq \{1, \ldots, m\}$ usgin s registers each containing $O(\log m)$ bits so that membership queues can be answered in the time t.

We show (for $|S| = n, \log m << n << m$):

For classical RAMs

$t = O(1) \Longrightarrow s = \epsilon m$ is sufficient and necessary.

$s = O(n) \Longrightarrow t = \Theta(\log n)$ is sufficient and necessary.

For practical RAMs
$t = O(1) \Longrightarrow s = m^\epsilon$ is sufficient and necessary.
$s = O(n) \Longrightarrow s = \Omega(\log\log n)$ is necessary and $t = O(\log n)$ is sufficient.

A Constsnt Time Priority Queue ... on the Rambo Model

Ian Munro

University of Waterloo

Abstract

The stratified binary tree, introduced by Peter van Emde-Boas in 1975, is now a classic data structure and an important building block. It permits the operations insert, delete and find nearest value above or below a query value over the universe $[0..M-1]$. Runtime for each opration is $O(\log\log M)$. The space used is easily limited to about M bits or if the number of elements present is small, $O(N)$ words of $\log M$ bits (Melhorn, Näher, and Alt). Melhorn and Näher have shown the $\Theta(\log\log N)$ runtime to be optimal on a pointer machine. In this talk we discuss the same problem on the Random Access Machine with Byte Overlap (RAMBO) of Fredman and Saks. Under this model an individual bit may be shared by a large number of words. We exploit this memory model to adapt the van Emde-Boas stucture into one which supports all its operations in constant time.

This is joint work with Andrej Brodnik

Algorithms for the Set-Union-Intersection Problem

Enrico Nardelli

University of L'Aquile, Italy

Abstract

We consider an algorithmic problem related to the incremental constraint maintenance in a context of concurrent constraint logic programming (CC). We consider CC languages defined on symbolic, non-structured, finite domains, where concurrent agents working on different sets of constraints define different partitions into equivalence classes over the universe of feasible values. An equivalence class of a paritition contains the values which are equivalent as far as the satisfaction of constraints is concerned. The problem to know at a certain time which are the equivalence classes with respect to the agents can be modelled by means of a variant of the well-known disjoint set union problem. We introduce data structures for this variant which lead to an optimal worst-case cost dealing with the new defined operations, paying a cost which is proportional to the size of the intersection.

This is joint work with Carlo Gaibisso and Gundo Prielli.

Chess Theory for Man and for Machines

Jürg Nievergelt

ETH Zürich

Abstract

Half a century after pioneering papers by Shannon and Turing on "Programming a computer to play chess", machines have reached a playing strength exceeded by only the top human players. In recent decades, computer's chess performance, as measured on the scale of ELO ratings, exhibits a linear relation with search depth. This empirical relation appears to have been confirmed, and extended by one new data point in the february 1996 match between IBM's Deep Blue (10^8 positions/sec) and world champion Garry Kasparov. We explain examples and compare typical samples of chess knowledge used by human players, and others used by machines.

Relaxed Blancing Made Simple

Th. Ottmann

Universität Freiburg

Abstract

Relaxed balancing means that in a dictionary stored as a balanced tree the necessary rebalancing after updates may be delayed. This is in contrast to strict balancing meaning that rebalancing is performed immediately after the update. Relaxed balancing is important for efficiency in highly dynamic applictions where updates can occur in bursts. The rebalancing tasks can be performed gradually after all urgent updates, allowing the concurrent use of the dictionary even though the underlying tree is not completely in balance. The contribution of our work is that we introduce a new scheme of relaxed balancing, which is obtained by a simple generalization of strict balancing. Our approach implies a simple proof of the fact that the number of the needed rebalancing operations (to put the tree in balance) for the relaxed balancing is the same as strict balancing.

This is Joint work with E. Saisalon-Saininen, Techn. Univer. of Helsinki, Finland .

Space Filling Curves and their Use in the Design of Geometric Data Structures

Thomas Roos

ETH Zürich

Abstract

We are giving a two-dimensional square grid of size $N \times N$, where $N = 2^n$ and $n \geq 0$. A space filling curve (SFC) is a numbering of the cells of this grid with numbers from $c + 1$ to $c + N^2$ for some $c \geq 0$. We call a SFC recursive (RSFC) if it can be recursively divided into 4 square RSFC's of equal size.

We proof several useful and interesting combinatorial properties of recursive and general SFC's. For an optimality criterion that is important in the design of geometric data structures, we propose a RSFC that is optimal in the worst-case and outperforms the previously known RSFCs.

This is a joint work with Tetsuo Asano (Osaka), Desh Ranjan (Las Cruces), Emo Wetzl (FU Berlin), and Peter Widmayer (ETH Zürich).

Ranking in Graphical Databases

Hanan Samet

University of Maryland, College Park Maryland, USA

Abstract

An algorithm for ranking spatial objects according to increasing distance from a query object is introduced and analyzed. The algorithm makes use of a hierarchical spatial data structure. The intended application area is a database environment, where the spatial data structure serves as an index. The algorithm is incremental in the sense that objects are reported one by one, so that a query processor can use the algorithm in a pipelined fashion for complex queries involving proximity. It is well suited for k nearest neighbor queries, and has the property that k needs not to be fixed in advance. In particular, having computed the k nearest neighbors, if we want to find the $k + 1$st nearest neighbor, we can start the search from where we last left it rather than from the start. Thus there is no need to apply an algorithm to compute $k+1$ neighbors. The algorithm makes use of a priority queue in which the objects as well as their container blockes are stored and is applicable to arbitrary hierarchical spatial data structures that make use of a tree access. It has been used as the basis of a browser for graphical objects in a relational database.

This is joint work with Gisli Hjaltason, University of Maryland.

Optimal Robot Localization in Trees

Sven Schuierer

Universität Freiburg

Abstract

The problem of localization, i.e. of a robot finding its position on a map, is an important task for autonomous mobile robots. It has applications in numerous areas of robotics ranging from aerial photography to autonomous vehicle exploration. In this talk we present a new strategy for a robot to find its position on a map where the map is represented as a geometric tree. Our strategy exploits to a high degree the self-similarities that may occur in the environment.

We use the framework of competitive analysis to analyse the performance of our strategy. In particular, we show that the distance travelled by the robot is at most $O(\sqrt{n})$ times longer than the shortest possible route to localize the robot, where n is the number of vertices of the tree. This is a significant improvement over the best known previous bound of $O(n^{2/3})$. Moreover, since there is a lower bound of $\Omega(\sqrt{n})$, our strategy is optimal up to a constant factor.

Using the same approach we can also show that the problem of searching for a target in a geometric tree where the robot is given a map of the tree and the location of the target but does not know its own position, can be solved by a strategy with a competitive ratio of $O(\sqrt{n})$, which is again optimal up to a constant factor.

This is joint work with Kathleen Romanik, DIMACS, Rutgers University

Maintaining the Minimum of a Set

Shiva Chaudhuri

MPI Saarbrücken

Abstract

The problem of maintaining a data structure on a universe of n ordered elements is studied. The data structure should support the operations *Insert*, *Delete* and *Findmin*. It is shown that if *Insert* and *Delete* are restricted to using at most t comparisons, then *Findmin* requires $n/2^{4t+3}$ comparisons.

The above bounds hold even for the expected cost of algorithms that use randomization.

This is joint work with Gerth Stolting Brodal and Joikumar Radhakrishnan.

Static Dictionaries on AC $^{\mho}$ RAMs: Query Time $\Theta(\sqrt{\log n/\log\log n})$ **is necessary and sufficient.**

Mikkel Thorup

Abstract

In this paper we consider solutions to the static dictionary problem on AC^0 RAMs, i.e. random access machines where the only restriction on the finite instruction set is that all computational instructions are in AC^0. Our main result is a tight upper and lower bound of $\Theta(\sqrt{\log n/\log\log n})$ on the time for answering membership queries in a set of size n when reasonable space is used for the data structure storing the set; the upper bound can be obtained using $O(n)$ space, and the lower bound holds even if we allow space $2^{polylog(n)}$.

Several variations of this bound are also obtained, including tight upper and lower bound on the space if the query time must be constant, general (but not tight) time-space tradeoffs, and bounds valid for non-AC^0 RAMs if the execution time of an instruction is measured as the minimal depth of a

polynomially sized unbounded fan-in circuit computing the operation. As an exemple of the latter, we show that any RAM instruction set which permits a linear space, constant query time solution to the static dictionary problem must have an instruction of depth $\Omega(\log w / \log\log w)$, where w is the word size of the machine (and log the size of the universe). This matches the depth of multiplication and integer division, used in the solution of Fredman, Komlos and Szemeredi achieving these bounds.

One of the non-dictionary related consequences of our techniques is a randomized $O(n(\log\log n)^2)$ AC^0 sorting algorithm using linear space. This is the first linear space AC^0 algorithm beating the information theoretic lower bound of $\Omega(n \log n)$.

This is joint work with Arne Andersson, Peter Bro Miltersen, and Soren Riis.

Data Structures for Counting Knight's Tours

Ingo Wegener

Universität Dortmund

Abstract

The exact number of knight's tours on the 8×8 chessboard is determined. It equals

$$33,439,123,484,294$$

The message is not the number but the method. Ordered binary decision diagrams (OBDD'S) known from circuit verification, hardware design and timing analysis are used in a divide-and-conquer approach to present cycle free path systems. Because they are minimized in size they automatically prevent us from searching in regions without solutions and from searching a region where an isomorphic region has been searched before. The devide-and-conquer approach implies, that the number of solutions for subproblems can be multiplied. Not all solutions have to be enumerated.

This is joint work with Martin Löbbing (Univ. Dortmund).

Distributed Search Trees

Peter Widmayer

ETH Zürich

Abstract

We discuss the problem of maintaining a set of keys under the dictionary operations in a distributed setting. Client and server processors communicate by sending messages. We describe a distributed generalization of random search trees and show experimetally that these trees perform well on average. They suffer, not surprisingly, from a bad worst case. We show that balancing the tree in the sense that it has logarithmic height and logarithmic cost per operation is impossible within a certain model. We prove an $\Omega(\sqrt{n})$ lower bound for the height of a distributed tree in the model, and we present a tree that satisfies an $O(\sqrt{n})$ upper bound. As a consequence data structures for efficient distributed searching have to be looked for outside this model.

This is joint work woth Brigitte Kröll, ETH Zürich.