AD-A250 894

N PAGE

Form Approved
OMB No. 0704-0188

Public report... gathering an... collection of... Davis Highway...

...hour per response, including the time for reviewing instructions, searching existing data sources, ...ction of information. Send comments regarding this burden estimate or any other aspect of this ...ngton Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson ...work and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

| 1. AGENC | May 1992 | 3. REPORT TYPE AND DATES COVERED final technical 10/88-2/92 |
|---|---|---|

**4. TITLE AND SUBTITLE**

Design and Implementation of Parallel Algorithms

**5. FUNDING NUMBERS**

C: N00014-88-K-0166

**6. AUTHOR(S)**

Jeffrey D. Ullman

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Computer Science Department
Stanford University
Stanford, CA 94305

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Office of Naval Research
Code 1133
800 N. Quincy St., BCT No. 1
Arlington, VA 22217-5000

**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**

**12a. DISTRIBUTION / AVAILABILITY STATEMENT**

Approved for public release: distribution unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*

See attached report.

DTIC
ELECTE
MAY 26 1992
S D

| 14. SUBJECT TERMS | 15. NUMBER OF PAGES 17 |
|---|---|
| | 16. PRICE CODE |

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UL | UL | UL | UL |

Principal Investigator Name: Jeffrey D. Ullman
PI Institution: Stanford University
PI Phone Number: (415) 723-9745
PI Email Address: ullman@cs.stanford.edu
Contract Title: Design and Implementation of Parallel Algorithms
Contract Number: N00014-88-K-0166
Reporting Period: 10/88-2/92

## 1. NUMERICAL PRODUCTIVITY MEASURES

Refereed papers, submitted, not published: 17
Refereed papers published: 12
Unrefereed reports and articles: 40
Books or parts submitted, not published: 0
Books or parts published: 2
Patents filed, not granted: 1
Patents granted: 0
Invited presentations: 12
Contributed presentations: 27
Honors received: 9
Prizes or awards received: 3
Promotions obtained: 1
Graduate students supported: 8
Postdocs supported: 6
Minorities supported: 0

92-13759

Principal Investigator Name: Jeffrey D. Ullman
PI Institution: Stanford University
PI Phone Number: (415) 723-9745
PI Email Address: ullman@cs.stanford.edu
Contract Title: Design and Implementation of Parallel Algorithms
Contract Number: N00014–88–K–0166
Reporting Period: 10/88–2/92

## 2. DETAILED SUMMARY OF TECHNICAL RESULTS

### Sorting and Related Operations

Greg Plaxton [1989b] won the "best paper" award at SPAA for his recent work on three related problems, each highly fundamental:

1. Load balancing: given a distribution of "tasks" to processors, move the task tokens so each processor has an equal number of tasks.
2. Selection: Given $n$ items and $k$ between 1 and $n$, find the $k$th item in sorted order.
3. Sorting: given $n$ items distributed equally among $p$ processors, sort the items.

Other papers on the subject are Mayr and Plaxton [1989] and Plaxton [1989a, c].

Greg's techniques for sorting make use of new, more efficient algorithms for performing (1) and (2) on the hypercube or similar networks. The end result is an algorithm like quicksort, but with many points of division in a stage (rather than a single division point), that outperforms other known algorithms such as cubesort or bitonic sort when $n$ is larger than $p$ (the number of processors) by a small, nonconstant factor.

The model used is the standard hypercube model, where nodes can send along one edge only at a given time. Previously, Plaxton reported an algorithm using a more powerful model, where messages can pass along all edges, although a node cannot read or write (just transmit) more than one message at a time. In that model, one can even improve upon bitonic sort for the case $n = p$; $O(\log^2 n / \log\log n)$ is achievable.

Recent work of Plaxton and Bob Cypher (U. Washington/IBM) gives an even better asymptotic result for the standard hypercube model; they can sort in $n$ elements on $n$ processors in $O(\log n(\log\log n)^3)$ time. In truth, $(\log\log n)^3$ only becomes better than $\log n$ when $n$ exceeds $2^{1000}$, but the method involves a new operation that one can reasonably hope will have a more efficient implementation, thus yielding an $O(\log n(\log\log n)^2)$ algorithm or better (then the crossover point would be about $n = 1000$, rather than $n = 2^{1000}$). The new operation involves looking at a $d$-dimensional hypercube as a two-dimensional grid, say with the row number equal to the first $d/2$ address bits and the column number the last $d/2$ address bits. The rows must be sorted in the order of the elements that appear in the first column of each row. Since the best known algorithm for this operation takes $O(\log n(\log\log n)^2)$ time, and this is the critical part of the Cypher/Plaxton algorithm, there is real hope that, eventually, we shall learn how to sort on the hypercube almost as fast as on (practically unbuildable) expander networks (where $O(\log n)$ sorting is theoretically achievable).

## Parallel Transitive Closure

Mihalis Yannakakis (Bell Labs) and Jeff Ullman (Ullman and Yannakakis [1989] have developed a family of parallel algorithms for solving transitive-closure-type problems. These represent the first parallel algorithms for these fundamental problems that use less work (time $\times$ processors) than the serial time to do matrix multiplication, yet take sublinear time, in fact time $n^\epsilon$ for any $\epsilon > 0$.

Given a directed graph and a collection of "source" nodes, we may ask what nodes are reachable from each source? The case where all nodes are sources is standard transitive closure, and the case of a single source is standard "reachability." Problems with more than one but fewer than all nodes as sources come up as subproblems in our solutions to the standard problems.

If we have an $n$-node graph, we can take its TC in $O(\log^2 n)$ time with somewhat fewer than $n^3$ processors, which is about optimal work, since a serial algorithm takes $n^3$ time.† However, this approach uses much too much work in the cases where

1. There is a single source and/or
2. The graph is sparse, with $e$ arcs, $e << n^2$.

The new approach uses less work in either of these cases. The algorithms are "Las Vegas," in the sense that they give the correct answer at least half the time; when they give no answer, they "give up" in the time stated. In the latter case, one restarts the algorithm, so the expected time to success is, to within a factor of 2, the same as stated below.

a) Single-source (reachability): For allowed time $= t$, work $= e\sqrt{n} + en/t + n^3/t^4$. Example: $t = \sqrt{n}$; work $= e\sqrt{n}$ (i.e., $e$ processors).

b) For all-pairs TC: For allowed time $= t$, work $= en + n^3/t^2$. Example: $t = \sqrt{n}$; work $= en$ (optimal work).

## Primitives for Data Sharing

It is an ongoing research question to determine the power of various primitive operations for sharing data among many processors in a reliable manner. The goal is to understand what primitive(s) should be implemented in hardware, and what can then be built from them in software. (Uniprocessor analogy: choose to build test-and-set in hardware, then implement P and V in software.)

Serge Plotkin [1989] has shown that a primitive called a "sticky bit" is universal, in the sense that it can implement any wait-free object (data type with operations whose execution cannot be delayed indefinitely by the slowness or failure of a process accessing it). Specifically, the paper proves "universality of consensus" in the sense that given an algorithm to achieve consensus, it shows how to construct an atomic wait-free implementation of any sequential object using bounded memory. Thus, there is no wait-free implementation of 2-value Read-Modify-Write (RMW) from safe bits and there is no wait-free implementation of 3-value RMW from 2-value RMW. The 3-value RMW is universal in

† Really $n^3$ stands for "whatever time you believe matrix multiplication takes."



Availability Codes
Dist | Avail and/or Special
A-1

the sense that any RMW can be implemented from a 3-value RMW in a wait-free fashion.

### Parallel Matching and Flow Algorithms

Andrew Goldberg and Serge Plotkin, along with Pravin Vaidya (Bell Labs) [1988] have constructed the first sublinear-time deterministic parallel algorithm for bipartite matching. More precisely, the algorithm runs in $O(n^{2/3}\log^3 n)$ time and uses $O(n^3)$ processors. These techniques also yield the first sublinear time $[O(\sqrt{n}\log^5 n)]$ deterministic parallel algorithm for constructing a depth-first search tree in a graph.

Goldberg, Plotkin, and Eva Tardos (MIT) have shown [1989] how to use interior-point methods for linear programming, developed in the context of sequential computation, to obtain parallel algorithms for bipartite matching and related problems. These techniques yield an $O^*(\sqrt{n}\log C)$-time† algorithm for computing min-cost flow in networks with unit capacities and integer costs, where the maximum cost is bounded by $C$ and $n$ is the number of nodes in the network. This is the fastest currently known parallel deterministic algorithm for this problem. It implies a weighted bipartite matching algorithm with the same running time.

We have also made some progress on parallelizing interior-point methods for linear programming. For example, we can solve bipartite matching with weights polynomial in the number of nodes in approximately $\sqrt{e}$ time for a graph of $e$ edges.

Also, a parallel algorithm for the blocking flow problem that uses time approximately equal to the number of nodes and processors equal to the number of edges has been discovered (Goldberg, Tardos, and Tarjan [1989], Goldberg and Tarjan [1989]).

### Message-Passing vs. Shared Memory

Message-passing and shared memory models of computation are apparently distinct ideas. However, Bar-Noy and Dolev [1989] shows that the models are related in that each can implement a "synchronizer" building block efficiently. A synchronizer is a way to obtain all the data needed by a given procedure, no matter where that data may be located, in such a way that all procedures using the synchronizer methodology appear to have executed atomically, in some serial order.

This paper shows that every algorithm using synchronizers for interprocessor communication works correctly and efficiently in both message-passing and shared-memory models. The interesting consequence comes from the fact that there are algorithms known for one of these models that are better than any known algorithm for the same problem in the other model. If these algorithms can be couched in terms of synchronizers, as many can, then the algorithms can be carried from one model to the other, yielding best-known algorithms for certain problems in both models. Examples are given in the paper.

### Converting Randomized Algorithms to Deterministic Ones

The effective use of randomization in parallel algorithms is far more common than in serial algorithms. In many cases, after discovery of a randomized parallel algorithm, there is later discovered a deterministic algorithm that has the same worst-case performance

† $O^*$ means "on the order of, neglecting factors of $\log n$ as well as constant factors."

as the original had expected performance. Seffi Naor [1989] has developed a technique that, for a large class of problems, called "lattice approximation problems," automatically converts randomized algorithms to deterministic ones.

### Adaptive Routing

In a really large network, say one million nodes, it is infeasible to store complete routing information at each node. Thus, schemes have been developed that store partial information at each node, so that the total space used to store routing tables is small and the path taken by any packet will not be longer than some constant times the minimum possible path. There is a known time-space tradeoff, in the sense that the larger you make the tables, the closer to optimal routes can be.

In Awerbuch, Bar-Noy, Linial, and Peleg [1989a, b], schemes are proposed that not only give close-to-optimal time-space tradeoff, but also offer the following, which previously had not been achievable simultaneously.

1. *Balanced memory.* The sizes of the tables are the same at each node.
2. *Name independence.* One can move a node, only changing information about the name of that node. (Contrast with the internet: if I physically took my machine, known as nimbin.stanford.edu, and moved it to Berkeley, I could never receive mail unless I renamed it nimbin.berkeley.edu.)
3. *Arbitrary edge costs*: "Shortest" paths can be defined by weighted path length, rather than just number of hops.
4. *Efficient preprocessing*: There is an efficient algorithm to construct the routing tables from the network.

### Radio Networks

A *radio network* is one in which a node can receive a message only if exactly one neighbor is talking to it. When many nodes have something to say, it is nontrivial to get the nodes organized quickly, so each will have an exclusive time slot for the "ear" of a node to which it wants to talk.

Alon, Bar-Noy, Linial, and Peleg [1989] shows that the minimum number of rounds needed for one of $n$ nodes to broadcast the same message to all nodes is $\Omega(\log^2 n)$. The result holds even for networks of radius 2, where "ordinary" broadcast takes only constant time, independent of $n$.

### Fast Communication Model

It is reasonable to suppose that we are headed for an era where we can transmit bits faster than a processor can read them. It thus becomes advantageous to design algorithms that allow routing of messages with minimal delay. Essentially, each node must read a few bits from the front of the message, delay the message only by the length of time it takes to read those bits, and then know where to pass the rest of the message without delaying it.

This model of networks was studied in Bar-Noy, Naor, and Naor [1989a]. The principal results are examples of problems where this constraint does not slow down the algorithm as a whole, and other problems where there is provably an $O(\log n)$ factor slowdown.

### Computation on a Grid

Bar-Noy and Peleg [1989] studies computation on a "mesh of busses," a two- (or more) dimensional network, in which any processor can broadcast to its row or column in one step. Previous algorithms for the model had assumed that a square mesh was the most efficient shape. However, this paper shows that for census functions (associative and commutative operations like addition) applied to a value at each node, a somewhat rectangular shape is optimal. For example, 256 nodes should be arranged as $8 \times 32$ rather than $16 \times 16$.

### Computation in Fault-Tolerant Networks

Bar-Noy, Dolev, Koller, and Peleg [1989] describes some positive results on solving coordination problems in unreliable distributed systems. It is possible to solve many variants of the basic critical-section problem. In comparison, the consensus (Byzantine agreement) problem is known to be unsolvable in this model.

### Circuit-Value Problems and Network Stability Problems

Mayr and Subramanian [1989] develops a method for non-trivially limiting fanout in a circuit, and studies two problems on limited-fanout circuits: circuit value and stability. The circuit-value problem (given a combinational circuit and input values, what is the output?) for these classes of circuits each defines a class of problems whose parallelizability is unknown, in the sense that each lies between NC and P. Some common problems are shown to be in this class, including corresponding stability problems (in a network with cycles, is there a consistent assignment of truth values to the outputs of all gates?).

In the *stable matching problem*, we seek to match people in pairs, subject to some preference information, in "divorce-proof" ways. The paper Subramanian [1989] shows that this problem may be viewed as a problem about fanout-limited circuits with feedback. Consequences include putting this problem into the class CC (one of the classes defined in Mayr and Subramanian [1989]), thus giving evidence that this problem has intermediate difficulty of parallelization.

### Some Algorithmic Ideas

Chrobak and Naor [1989] shows how to use local transformations on a graph, which may be performed in parallel, for a fast solution to the problem of finding a large independent set. Chrobak, Naor, and Novick [1989] shows how to use the technique of finding a spanning tree where each node is of limited degree.

### Network Decomposition

Awerbuch, Goldberg, Plotkin, and Luby [1990] develop a technique for designing distributed algorithms: they show how to partition a network into small-diameter clusters such that the graph of clusters has low chromatic number. For example, they apply the technique to find maximal independent sets. That is, they assume a graph is distributed over several nodes, and they try to find a set of nodes, no two of which are connected by an edge, such that all other nodes are adjacent to some node in the set. This classical problem is very important for parallel computation in general; it lets us find maximal sets

of operations that can be performed in parallel without conflict.

They show how to find maximal independent sets in a distributed network of $n$ nodes in $n^\epsilon$ time for any $\epsilon > 0$. The same technique is used to find near-optimal colorings of graphs and to find breadth-first orders for graphs. All these algorithms are faster than previously known distributed algorithms for their problems.

### Interior-point Methods

Goldberg, Plotkin, Shmoys, and Tardos [1989] have a technique for applying interior-point methods (like Karmarkar's algorithm) for linear programming, in parallel. They produced sublinear parallel algorithms for bipartite matching, and certain cases of the assignment and flow problems. Again, these algorithms are the fastest known parallel algorithms for their problems.

### Max Flow

Goldberg [1990] gives a parallelization of the "maximum distance discharge" algorithm for max flow, which he shows experimentally to be efficient (serially) compared to other serial flow algorithms. The parallel implementation of this algorithm uses work (processor-time product) within a log factor of the serial time, and thus is the best known parallel flow algorithm.

### Secure Transmission

Dolev, Dwork, Waarts, and Yung [1990] addressed the question of secure (secret) transmission in faulty networks. The assumption is that an adversary can listen in on some limited number of lines of the network, and that the adversary can also disrupt transmission, that is, introduce false signals, into some limited number of lines. The goal is to guarantee correct transmission of messages, without the adversary learning anything.

The authors show lower bounds on the connectivity of the network involved, and give provably secure algorithms for networks of adequate connectivity. Their algorithms, for the first time, do not rely on any complexity-theoretic assumptions, like $\mathcal{P} \neq \mathcal{NP}$. Further, their algorithm runs in time that is linear in the network size.

### Eventual Byzantine Agreement

Halpern, and Moses, and Waarts [1990] investigated eventual Byzantine agreement (*EBA*) in networks of faulty processors (the ability of good processors to agree on a value in the presence of failures). By applying a new construction, twice, they can turn any EBA protocol into an optimal one (which runs as fast as any other on all possible data).

### Asynchronous PRAM

Phil Gibbons has continued his work on a PRAM model (the A-PRAM) that accounts for the cost of synchronizing processors and the cost of accessing nonlocal memory. In Gibbons [1990b], the relative power of several methods for performing pairwise synchronization of processors was established, and algorithms for minimizing synchronization in the A-PRAM were developed. Lower bounds on the performance of the A-PRAM, and simulations of

conventional PRAM's were also found.

Gibbons [1990a] also has developed and shown correct, a cache-management algorithm for simulating A-PRAM's. His algorithm involves prefetching and overlapping of instructions with the service of cache misses, and represents a novel compromise between the programmer's and the hardware's responsibility for achieving fast parallel algorithms.

### Small-Bias Probability Spaces

Naor and Naor [1990] offers a technique for limiting, and in some cases, eliminating, randomness in parallel algorithms. The idea centers around a construction for selecting a small (polynomial-size) set of assignments to n random variables, such that for any subset of these variables, the number of assignments making an odd number of the variables in this subset equal to 1 is approximately 50%.

### Fast Routing in Fast Networks

Azar, Naor, and Rom [1990] consider networks in which the transmission speed is too great to allow much computation at the intermediate nodes. In such a network, the route selection must primarily be the function of the source node, with only minimal work at each intermediate node. One approach is to fix the route completely, in advance. For this strategy, they give a polynomial algorithm for balancing the average use of network links. Another approach is to "flood" the network, effectively broadcasting each message. They show how to limit the degree of flooding, while still guaranteeing message delivery.

### Comparison Merging

Azar [1990a] has given lower bounds and matching algorithms for the general problem of merging $n/m$ ordered lists of $m$ elements each, using $p$ processors. For example, sorting is the case $m = n$.

### Single-Cell Communication

The problem of computing using a local-area network (or a bus for that matter) can be represented abstractly by a PRAM with a single memory cell. That is, there are $n$ processors, each able to communicate only by reading and writing this one cell. It is normal to assume some arbitration rule for writing in all PRAM models, and in this case, arbitration is essential. Azar [1990b] shows some lower bounds on what this model can achieve. The arbitration rule is "priority," that is, the lowest-numbered processor that wishes to write is allowed to write. For instance, Azar shows that if each of $n$ processors is given a bit, it takes at least $\Omega(m)$ time to determine if at least $m$ of the bits are 1, provided $m$ grows slower than $\sqrt{n}$.

### Coupled Execute/Control Processor Architecture

Andy Freeman has designed and simulated a new processor architecture for use in multiprocessors. The basic problem is designing processors that can cope with long memory latency. His approach is to couple two processors by two-way queues; neither processor has the capability associated with conventional processors. One executes only "straight-line

code," i.e., computations, and the second executes only tests and branches. Through simulation he has shown the superiority of this architecture on applications that are similar to "vectorizable" computations, including those on which vector architectures perform well and some that are (slightly) too sequential for vector processors. No reports on this work have yet been produced.

### Timestamping in Networks

Orli Waarts and Cynthia Dwork of IBM have looked at the question of assigning timestamps in an asynchronous network of processors (Dwork and Waarts [1991]). The model is that of shared memory, where the processors communicate through a memory accessible to all; the memory, like the processors, may be distributed. The problem is to issue tickets in a first-come-first served order when there are unlimited numbers of requests for service that may arrive asynchronously. Their solution uses time proportional to the number of processors involved, while previous solutions were superlinear.

A key problem to solve is avoiding a bottleneck at one memory location. Obvious approaches have the processors queue up for access to a counter that assigns timestamps. The solution of Dwork and Waarts uses instead a bounded timestamp, which is in the form of a vector of values, one for each processor. Each processor controls one component of the vector. In particular, we must be able to recycle timestamps so we do not confuse a new value with an old, identical value. We avoid this problem by allowing each processor to designate which of its values comes first; the others follow in a fixed, cyclic order.

### Linearizable Counting

A related problem, called "linearizable counting," assigns consecutive integers to processors (timestamping assigns tokens, which might be integers, to processes) in a first-come-first served order. The algorithm of Herlihy, Shavit, and Waarts [1991] performs this assignment in time proportional to the number of processors, and does so in a way that does not produce a bottleneck at a shared memory location.

### Processor Assignment

Yossi Azar and Joseph Naor, with R. Rom of IBM, looked at the problem of on-line assignment of tasks to processors (Azar, Naor, and Rom [1991]). That is, a sequence of tasks enter the system asynchronously, and each can be performed by any of a subset of the $n$ processors. We must assign tasks to processors as they enter, and the object is to balance the load on the processors. They offer an algorithm that can be no worse than a factor $\log_2 n$, compared with a "clairvoyant" algorithm that can predict future demand for the various processors. They also show this ratio is best possible. Further, they consider the use of a randomized algorithm for the same assignment problem, and show that a ratio of $\log_e n$ is sufficient and best-possible.

### Derandomization of Algorithms

Many of the best known parallel algorithms are probabilistic, in the sense that they perform well with high probability, but there is no guarantee they will finish in any particular

amount of time. In many cases, it is possible to replace a probabilistic, parallel algorithm by a deterministic parallel algorithm by discovering a small set of bit strings that can represent possible sequences of "coin flips" (the steps that introduce the randomization). It is necessary that these small sets have certain properties of pseudo-randomness, in the sense that on any input there is at least one among them on which the probabilistic algorithm will terminate relatively quickly.

Azar, Motwani, and Naor [1991] address the problem of approximating arbitrary joint distributions of random variables. They construct, for any $\epsilon > 0$, a set of strings that is of size polynomial in both the number of variables and $1/\epsilon$, whose joint distribution approximates that of the given distribution, in the sense that none of the coefficients of the Fourier transform of the two distributions differ by more than $\epsilon$. This result can be used in two ways. One, it is possible to run the original random algorithm using each of the strings of "coin flips" in the set, in paralllel, terminating as soon as the first among them terminates. Second, it allows one to replace a large number of coin flips by a smaller number (the logarithm of the size of the set) that is "almost as random" as the original.

### Bipartite Matching

Andy Goldberg and Serge Plotkin looked at the open question of sublinear (NC) algorithms for parallel bipartite matching (Fisher, Goldberg, and Plotkin [1991]). This problem, in addition to being one of the most important of the classical combinatorial problems, has special significance for parallel computation because a number of other important problems, such as depth-first search, are known to have NC algorithms if bipartite matching does. They show is that there is an NC algorithm for a strong form of approximate matching, where, with different versions of the algorithm, they can guarantee a matching that comes within $1 - \epsilon$ of the maximum match for any $\epsilon > 0$.

Earlier work on using interior point methods in parallel algorithms has appeared in a journal (Goldberg, Plotkin, Shmoys, and Tardos [1991]).

### Network Flow Problems

A survey of parallel algorithms for network flow was published (Goldberg, Tardos, and Tarjan [1990]). Also, earlier work on reducing the number of processors needed to solve network flow problems in parallel appeared in a journal (Goldberg [1991]).

Principal Investigator Name: Jeffrey D. Ullman
PI Institution: Stanford University
PI Phone Number: (415) 723-9745
PI Email Address: ullman@cs.stanford.edu
Contract Title: Design and Implementation of Parallel Algorithms
Contract Number: N00014-88-K-0166
Reporting Period: 10/88-2/92

## 3. REFERENCES

Alon, N., Y. Azar, and Y. Ravid [1990]. "Universal sequences for complete graphs," *SIAM J. Discrete Math* **27**.

Alon, N., A. Bar-Noy, N. Linial, and D. Peleg [1989]. "On the complexity of radio broadcast," *Proc. Twenty-First Annual ACM Symposium on the Theory of Computing*, pp. 274–285 (May, 1989).

Alon, N., J. Bruck, J. Naor, M. Naor, and R. Roth [1991]. "Constructing asymptotically good low-rate error-correcting codes through pseudo-random graphs," to appear in *IEEE Trans. Inf. Th.*

Attiya, H., A. Bar-Noy, D. Dolev, D. Peleg, and R. Reischuk [1990]. "Renaming in an asynchronous environment," *J. ACM* **37**:3, pp. 524–548.

Awerbuch, B., A. Bar-Noy, N. Linial, and D. Peleg [1989a]. "Compact distributed data structures for adaptive routing," *Proc. Twenty-First Annual ACM Symposium on the Theory of Computing*, pp. 479–489 (May, 1989).

Awerbuch, B., A. Bar-Noy, N. Linial, and D. Peleg [1989b]. "Improved routing strategies with succinct tables," to appear in *J. Algorithms.*

Awerbuch, B., A. V. Goldberg, M. Luby, and S. A. Plotkin [1989]. "Network decomposition and locality in distributed computation," *Proc. Thirtieth Annual IEEE Symposium on Foundations of Computer Science*, pp. 364–369.

Awerbuch, B., O. Goldreich, D. Peleg, and R. Vainish [1990]. "A trade-off between information and communication in broadcast protocols," *J. ACM* **37**:2, pp. 238–256.

Azar, Y. [1990]. "Lower bounds for threshold and symmetric functions in parallel computation," submitted to *SIAM J. Computing.*

Azar, Y. [1991a]. "Parallel merging of many ordered lists," to appear in *Theor. CS.*

Azar, Y. [1991b]. "Parallel computation of threshold and symmetric functions," to appear in *SICOMP.*

Azar, Y., R. Motwani, and J. Naor [1991]. "Approximating distributions using small sample spaces," submitted to *Combinatorica.*

Azar, Y., J. Naor, and R. Rom [1990]. "Routing strategies for fast .etworks," IBM, Almaden Research Center.

Azar, Y., J. Naor, and R. Rom [1991]. "The competitiveness of on-line assignments," submitted to 1992 *SODA*.

Bar-Noy, A. and D. Dolev [1989]. "Shared-memory vs. message-passing in an asynchronous distributed environment," *Proc. 8th ACM Symp. on Principles of Distributed Computing*, Aug., 1989.

Bar-Noy, A., D. Dolev, D. Koller, and D. Peleg [1989]. "Fault-tolerant critical section management in asynchronous networks," *Third Intl. Workshop on Distributed Algorithms*, 1989.

Bar-Noy, A., R. Motwani, and J. Naor [1989]. "A linear time approach to the set maxima problem," Dept. of CS, Satanford.

Bar-Noy, A. and J. Naor [1990]. "Sorting, feedback sets, and Hamilton paths in tournaments," *Siam J. Disc. Math* **3**:1, pp. 7-20.

Bar-Noy, A., J. Naor, and M. Naor [1990]. "One-bit algorithms," *Distributed Computing*, pp. 3-8, 1990.

Bar-Noy, A. and D. Peleg [1989]. "Square meshes are not always optimal," *Proc. ACM Symp. on Parallel Algorithms and Architectures*, pp. 138–147, June, 1989.

Chrobak, M. and J. Naor [1989]. "An efficient parallel algorithm for computing a large independent set in planar graphs," *Proc. ACM Symp. on Parallel Algorithms and Architectures*, pp. 379–387, June, 1989.

Chrobak, M., J. Naor, and M. Novick [1989]. "Using bounded degree spanning trees in the design of efficient algorithms on claw-free graphs," 1989 workshop on algorithms and data structures, Carleton Univ.

Cohen, E. and N. Megiddo [1989]. "Strongly polynomial time and NC algorithms for detecting cycles in dynamic graphs," STOC, 1989.

Dolev, D., C. Dwork, O. Waarts, and M. Yung [1990]. "Perfectly secure message transmission," to appear in *JACM*. Also in 1990 *FOCS*.

Dwork, C. and O. Waarts [1991]. "Simple and efficient bounded concurrent timestamp system," IBM Almaden Research Center.

Fischer, T., A. V. Goldberg, and S. A. Plotkin [1991]. "Approximating matchings in parallel," STAN–CS–91–1369, Dept. of CS, Stanford Univ, June, 1991.

Gafni, E. and J. Naor [1989]. "Separating the EREW and CREW PRAM models," *Theor. CS*.

Gafni, E., J. Naor, and P. Ragde [1989]. "On separating the EREW and CREW PRAM models," *Theoretical Computer Science* **68**:3, pp. 343–346.

Gharachorloo, K., D. Lenoski, J. Laudon, P. B. Gibbons, A. Gupta, and J. L. Hennessy [1990]. "Memory consistency and event ordering in scalable shared-memory processors," *Proc. Seventeenth Intl. Symp. on Computer Architecture*, May, 1990.

Gibbons, P. B. [1990a]. "Cache support for the asynchronous PRAM," *Proc. 19th Intl. Conf. on Parallel Processing*, Aug., 1990.

Gibbons, P. B. [1990b]. "Asynchronous PRAM algorithms," *A Synthesis of Parallel Algorithms* (J. Reif, ed.), Morgan-Kaufmann, San Mateo.

Goldberg, A. V. [1990]. "Processor-efficient implementation of network flow algorithms," STAN-CS-90-1301.

Goldberg, A. V., S. A. Plotkin, D. Shmoys, and E. Tardos [1989]. "Interior-point methods in parallel computation," *Proc. Thirtieth Annual IEEE Symposium on Foundations of Computer Science*, pp. 350–356.

Goldberg, A. V., S. A. Plotkin, D. Shmoys, and E. Tardos [1991]. "Interior point methods in parallel computation," *SIAM J. Computing* **16**:2, pp. 351–381.

Goldberg, A. V., S. A. Plotkin, and É. Tardos [1990]. "Combinatorial algorithms for the generalized circulation problem," to appear in *Math. of Oper. Res.*.

Goldberg, A. V., S. A. Plotkin, and P. M. Vaidya [1988]. "Sublinear-time parallel algorithms for matching and related problems," *Proc. Twenty-ninth Annual IEEE Symposium on Foundations of Computer Science*, pp. 174–185 (Oct., 1988).

Goldberg, A. V., E. Tardos, and R. E. Tarjan [1989]. "Network flow algorithms," STAN–CS–89–1252, 1989.

Goldberg, A. V., E. Tardos, and R. E. Tarjan [1990]. "Network flow algorithms," in *Flows, Paths, and VLSI Layout* (B. Korte, L. Lovasz, H. J. Promel, and A. Schrijver, eds.), Springer-Verlag, pp. 101–164.

Goldberg, A. V. and R. E. Tarjan [1989]. "A parallel algorithm for finding a blocking flow in an acyclic network," *Information Processing Letters* **31**:3, pp. 265–271.

Haibt-Norton, C., S. A. Plotkin, and É. Tardos [1990]. "Using separation algorithms in fixed dimension," *First Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1990.

Halpern, J. Y., Y. Moses, and O. Waarts [1990]. "A characterization of eventual Byzantine agreement," *Proc. 9th ACM Symp. on Principles of Distributed Computing*, Aug., 1990.

Herlihy, M., N. Shavit, and O. Waarts [1991]. "Low contention linearizable counting," 1991 *FOCS*.

Hochbaum, D. and J. Naor [1991]. "Simple and fast algorithms for linear and integer programs with two variables per inequality," submitted to *JACM*.

Khuller S. and J. Naor [1990]. "Flow in planar graphs with vertex capacities," Proc. Integer *Programming and Combinatorial Optimization*, pp. 367-383, Waterloo, Ont., Canada.

Klein, P., S. Khuller, and J. Naor [1991]. "The lattice structure of planar flow," submitted to *SIAM J. Discrete Math.*

Klein, P., S. A. Plotkin, C. Stein, and E. Tardos [1991]. "Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts," TR-961 OR/IE, Cornell Univ.

Leighton, T., F. Makedon, S. A. Plotkin, C. Stein, S. Tragoudas, and E. Tardos [1991]. "Fast approximation algorithms for multicommodity flow problems," *Proc. Twenty-third Annual ACM Symposium on the Theory of Computing*, pp. 101–111.

Mayr, E. W. and C. G. Plaxton [1989] "Pipelined parallel prefix computations, and sorting on a pipelined hypercube," STAN-CS-89-1261, May, 1989.

Mayr, E. W. and A. Subramanian [1989] "The complexity of circuit value and network stability," *Proc. 4th Annual Conference on Structure in Complexity Theory*, pp. 114-123, June 1989. Submitted to *J. CSS.* Also STAN-CS-89-1278.

Miller, G. and J. Naor [1989] "Flow in planar graphs with multiple sources and sinks," FOCS, 1989.

Naor, J. [1989] "The probabilistic method yields deterministic parallel algorithms," 1989 FOCS.

Naor, J. and M. Novick [1989] "An efficient reconstruction of a graph from its line graph in parallel," to appear in *J. Algorithms.*

Naor, J. and M. B. Novick [1990]. "An efficient reconstruction of a graph from its line graph in parallel," *J. Algorithms* **11**, pp. 132-143.

Naor S. and M. Naor [1990]. "Small bias probability spaces: efficient constructions and applications," *Proc. Twenty-Second Annual ACM Symposium on the Theory of Computing*, pp. 213–223.

Niblack, C. W., D. W. Capson, and P. B. Gibbons [1990]. "Generating skeletons and centerlines from the medial axis transform," *Proc. Tenth Intl. Conf. on Pattern Recognition*, June, 1990.

Peleg, D. and J. D. Ullman [1989] "An optimal synchronizer for the hypercube," *SIAM J. Computing* **18**:4 (Aug., 1989), pp. 740–747.

Plaxton, C. G. [1989a] "Load Balancing on the hypercube," *Proc. 4th Conf. on Hypercubes, Concurrent Computers, and Applications* [1989] March, 1989.

Plaxton, C. G. [1989b] "Load balancing, selection, and sorting on the hypercube," *Proc. ACM Symp. on Parallel Algorithms and Architectures* [1989] pp. 64–75, June, 1989.

Plaxton, C. G. [1989c] "On the network complexity of selection," 30th FOCS, Oct., 1989.

Plotkin, S. A. [1989] "Sticky bits and the universality of consensus," *Proc. 8th ACM Symp. on Principles of Distributed Computing*, pp. 159–174, Aug., 1989.

Plotkin, S. A., D. Shmoys, and E. Tardos [1991]. "Fast approximation algorithms for fractional packing and covering problems," 1991 *FOCS*.

Plotkin, S. A. and É. Tardos [1990]. "Improved dual network simplex," *First Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1990.

Subramanian, A. "A new approach to the stable matching problem," TR STAN-CS-89-1275, Stanford University, 1989.

Ullman, J. D. [1990]. "Parallel algorithms for logical inference," *A Synthesis of Parallel Algorithms* (J. Reif, ed.), Morgan-Kaufmann, San Mateo.

Ullman, J. D. and M. Yannakakis [1990]. "High-probability parallel transitive closure," *Symp. on Parallel Algorithms and Architectures*, Crete, June, 1990. Also, accepted for *SIAM J. Computing*.

Ullman, J. D. and M. Yannakakis [1991]. "High-probability parallel transitive closure algorithms," *SIAM J. Computing* **20**:1, pp. 100–125.

Principal Investigator Name: Jeffrey D. Ullman
PI Institution: Stanford University
PI Phone Number: (415) 723-9745
PI Email Address: ullman@cs.stanford.edu
Contract Title: Design and Implementation of Parallel Algorithms
Contract Number: N00014-88-K-0166
Reporting Period: 10/88-2/92

## 4. TRANSITIONS AND DOD INTERACTIONS

Ullman served on steering committee for the creation of SPAA conference (Parallel Algorithms and Architectures).

Surveys of relevant material were published as Goldberg, Tardos, and Tarjan [1990], Gibbons [1990b], and Ullman [1990].

Andy Freeman has discussed his architectural proposals with representatives of IDA.

Principal Investigator Name: Jeffrey D. Ullman
PI Institution: Stanford University
PI Phone Number: (415) 723-9745
PI Email Address: ullman@cs.stanford.edu
Contract Title: Design and Implementation of Parallel Algorithms
Contract Number: N00014-88-K-0166
Reporting Period: 10/88-2/92

## 5. SOFTWARE AND HARDWARE PROTOTYPES

Andy Freeman has an extensive simulator for microprocessors, and Andy Goldberg has been working on code for parallel flow on a connection machine, in connection with a competition for algorithms of this type.