ON THE PACKET COMPLEXITY OF DISTRIBUTED SELECTION

by Alberto Negro*, Nicola Santoro**
and Jorge Urrutia***

SCS-TR-127

November 1987

School of Computer Science
Carleton University
Ottawa, Ontario
CANADA K1S 5B6

*Dipartimento di Informatica, Universita di Salerno, Salerno, Italy
**School of Computer Science, Carleton University, Ottawa, Canada
***Department of Computer Science, University of Ottawa, Ottawa, Canada

This research has been supported in part by the Natural Sciences and Engineering Research Council of Canada under Grants A0977 and A2415

ON THE PACKET COMPLEXITY OF DISTRIBUTED SELECTION

Alberto Negro[1], Nicola Santoro[2], Jorge Urrutia[3]

(1) Dipartimento di Informatica, Universitá di Salerno, Salerno, Italy
(2) School of Computer Science, Carleton University, Ottawa, Canada
(3) Department of Computer Science, University of Ottawa, Ottawa, Canada

1. INTRODUCTION

In this paper, we consider distributed environments where the *size of a message is fixed.* That is, a message is a *packet* which can contain at most c bits of *data*, where $c \geq 1$ is a constant; therefore, the communication of integer I will require the transmission of $\lceil \log(I)/c \rceil$ packets. The constant c, called *packet size*, is generally dependent on the type of network but not necessarily on the application (e.g., the problem values). This situation occurs in practice in packet-switching networks (e.g., ARPA) which are a natural application environment for distributed algorithms and are directly modeled by point-to-point networks. Notice that the *packet complexity* includes both *bit complexity* and (long or short) *message complexity* as special cases.

A general solution algorithm using packets is presented for the (well known) distributed selection problem, and its *packet complexity* is analyzed. The proposed solution is obtained by reducing the selection problem to an *unbounded search problem.* It is shown to require at most $O(d \log f_K \lceil \log N / c \rceil)$ packets, where d is the number of processors, N is the size of the set among which the selection takes place, and f_K is the sought element; this represents a significant improvement over the existing $O(d^2 \log(N/d) \lceil \log f_K / c \rceil)$ bounds. For specific classes of network topologies, bounds are obtained which either match the one of existing solutions or improve them for a large range of values of the packet size c; it is interesting to note that this range includes the situation where the packet size is dependent on the application size but not on the application values.

The actual bound on the packet complexity of the proposed selection algorithm is shown to depend on the location of the node which coordinates (i.e., synchronizes) the algorithm. This leads to the problem of determining, for a given network, a site (*s-center*) which minimizes the required amount of communication activities. A solution to this problem is presented for acyclic networks.

2. MESSAGES, BITS AND PACKETS

With the increasing interest in the design and analysis of *distributed algorithms*, it has increased also the difference in assumptions made by researchers when developing a solution or measuring its *communication complexity*. Traditionally, the main complexity focus has been on the amount of communication activities performed by an algorithm or required by a class of algorithms solving a given problem, where a *communication activity* is defined as the transmission of a *message*. Although the intuitive notion of *message* is well understood, the formal specification of what a message *is* is not equally clear nor commonly agreed upon, as a quick glance through the many papers dealing with distributed algorithms will immediately reveal. In fact, depending on the situation, a message is allowed (or restricted) to contain anything from a constant number of *bits* to an (encoding of) the entire network topology to an entire set of *values* (from an application-dependent universe). It is thus not surprising that terms such as *bit complexity*, *short message complexity*, or *long message complexity* need to be employed to distinguish the assumptions made. Amid these conflicting views and considerations, there seems to be a general agreement that *messages should not be unbounded*: this to avoid pathological cases (e.g., the encoding of an entire file in a single message).

In this paper, we consider environments where the *size of a message is fixed*. That is, a message is a *packet* which can contain at most c bits of *data*, where $c \geq 1$ is a constant; therefore, the communication of integer I will require the transmission of $\lceil \log(I)/c \rceil$ packets. The constant c, called *packet size*, is generally dependent on the type of network but not necessarily on the application (e.g., the problem values). This situation occurs in practice in packet-switching networks (e.g., ARPA) which are a natural application environment for distributed algorithms and are directly modeled by point-to-point networks. Notice that the *packet complexity* includes both *bit complexity* and (long or short) *message complexity* as special cases.

In the following, we will consider the packet complexity of a well known problem, the distributed selection problem, for which solutions based on messages have been proposed. The seemingly innocuous shift in viewpoint (i.e., the use of packets instead of bits or messages) is actually instrumental for the development of a new simple and efficient solution to the problem, as well as for sheding some light on the impact of the constant c on complexity.

3. THE DISTRIBUTED SELECTION PROBLEM

The classical problem of selecting the K-th smallest element of a set F drawn from a totally ordered set has been extensively studied in serial and parallel environments. In a distributed context, it has different formulations and complexity measures.

A *communication network* of size d is a set $S=\{S_1, \ldots, S_d\}$ of *sites*, where each site has a

local non-shared memory as well as processing and communication capabilities. Given a totally ordered set $F=\{f_1,...,f_N\}$, a *distribution* of F on S is a d-tuple $X=<X_1,...,X_d>$ where $X_i \subseteq F$ is the subset stored at site S_i, $X_i \cap X_j = \emptyset$ for $i \neq j$, and $\cup_i X_i = F$. Without loss of generality, let $f_i<f_{i+1}$ for $1 \leq i < N$. Order-statistics queries about F can be originated at any site and will activate a query resolution process at that site. Since only a subset of F is available at each site, the resolution of a query will in general require the cooperation of several (possibly all) sites according to some predetermined algorithm. Since local processing time is usually negligible when compared with transmission and queueing delays, the goal is to design resolution algorithms which minimize the amount of communication activity rather than the amount of processing activity.

The *distributed selection problem* is the general problem of resolving a query for locating f_K, the K-th smallest element of F. Any efficient solution to this problem can be employed as a building block for a distributed sorting algorithm [10]. The complexity of this problem (i.e., the number of communication activities required to resolve an order-statistics query) depends on many parameters, including the number of sites d, the size $|F|=N$ of the set, the rank K of the element being sought, and the topology and type of the network.

In *point-to-point* networks, associated with S is a set $L \subseteq SxS$ of direct communication lines between sites; if $(S_i,S_j) \in L$, S_i and S_j are said to be *neighbours*. Sites communicate by sending messages; a message can only be sent to and received from a neighbour. The couple G=(S,L) can be thought of as an undirected graph; hence, graph-theoretical notation can be employed in the design and analysis of distributed algorithms in the point-to-point model.

Several solution algorithms for the distributed selection problem have been presented in the literature, some tailored for specific network topologies (e.g., ring [3], complete binary tree [3], star and complete graph [6, 9,12], mesh [3,7], torus [7]), others executable in any network (e.g., [13,15]). In all these solutions, it is assumed that one or more set elements can be transmitted within one *message*, and the communication complexity of the proposed solutions reflect this assumption.

In the following, a general solution algorithm is presented for the distributed selection problem using packets, and its *packet complexity* is analyzed.

4. A DISTRIBUTED SELECTION ALGORITHM

4.1 Selection as Unbounded Search

In the proposed solution stategy for the distributed selection problem, the K-th smallest element is determined by *guessing* its value (following an approach developed in [5]). The query resolution mechanism consists in making an initial guess, x_1, and determining whether x_1 is greater than,

smaller than, or equal to f_K; depending on the outcome, either the algorithm terminates or another guess x_2 is made and its relationship with f_k is determined. In general, depending on the outcome of guesses $x_1,x_2,...,x_i$ a new guess x_{i+1} is made; this process is repeated until the value of the sought element is unambigously established. The relationship between f_K (which is unknown) and a guess x_i is easily established by determining the relationship between K and the rank of x_i (i.e., the number of elements smaller or equal to x_i) in the set. Thus, the entire process consists of a sequence of two simple steps: (1) make a guess, and (2) determine its rank.

It is not difficult to see that this stategy reduces the distributed selection problem to the well known problem of *searching in an unbounded table*. The number of iterations as well as which value will be chosen next depend on the searching strategy adopted. Using the known stategy and results for the unbounded search problem [1], it follows:

Property 4.1 The proposed algorithm determines the K-th smallest element using at most $g(f_K)$ iterations, where $g(x)=\log x+ \log\log x+ \log\log\log x+ ....$

4.2 Packet complexity

The proposed algorithm is a sequence of iterations; in each iteration a value is determined and the relationship between its rank and K is determined. The number of iterations as well as which value will be chosen in an iteration depend on the searching strategy adopted; once a strategy is selected, to determine the packet complexity of the entire algorithm it suffices to establish the packet complexity of each iteration. To this end and similarly to [13,15], we assume that a spanning-tree of the graph is available; this tree will be used for all communications.

Each iteration will comprise of the originator node broadcasting a control message: "start" to initiate the query resolution algorithm,"<" to indicate that the previous guess was too low, ">" to indicate that the previous guess was too high, and "stop" to indicate that the previous guess was the sought element. Upon reception of a control message other than "stop", a new value is chosen according to the searching strategy and its rank is collected at the originator by an "echo" process starting from the leaves.

Note that the purpose of this ranking process is to determine the relationship between K and the rank r(x) of the guess x ; thus, if it is determined at a node (on the basis of the partial ranks received) that r(x)>K, the ranking itself can stop and a "rank high" message can be sent instead. In other words, during an iteration, every node sends in the direction of the originator either a counter (which will never exceed the value K) or a control message ("rank high"). Combining the cost of this

operation with the cost for the broadcast started by the originator, we obtain the following:

Property 4.2 The K-th smallest element can be determined in a network G with d sites using at most

$$(d-1)\ g(f_K)\ (1+\lceil \log K / c \rceil)$$

packets.

Let us draw the reader's attention to some points of interest. First note that in the proposed algorithm, set values are *never* transmitted. Secondly observe that the established bound applies to any network, regardless of its topology. Finally, consider the effect of the "ceiling" operation: the bound, as a function of c, decreases at the increasing of the packet size until c = log K; from that point on, the function becomes a constant. The value c=log K is of interest in itself since it expresses a packet size which is dependent on the problem *size* but not on the problem *values*.

Let us now compare this bound with the existing results. In the literature, only two general selection algorithms have been proposed [13,15], both focusing on the *expected* rather than *worst case* complexity. The *worst case* complexity of the algorithm described in [15] is O(d N) messages, each carrying a constant number of values, for a total of

$$W = O(d\ N \lceil \log f_K / c \rceil) \qquad (1)$$

packets, while its *expected case* performance in terms of packets is

$$E = O(d \log N \lceil \log f_K / c \rceil). \qquad (2)$$

Thus, the proposed algorithm represents an improvement on the worst case performance of this solution for all packet sizes in the range $1 \le c \le \log f_K$; for even larger packet sizes, the proposed solution is still more efficient provided that $\log f_K \le N$. The improvement is particulary significant for $1 \le c \le \log K$. Moreover, for this range of the packet size ($1 \le c \le \log K$), the *worst case* complexit of the proposed algorithm coincides with the *expected* performance of [15]. The other general algorithm, described in [13], has a worst case performance of

$$O(d^2 \log(N/d) \lceil \log f_K / c \rceil) \qquad (3)$$

packets. Thus, also in this case, the proposed algorithm yields a significative improvement for all packet sizes in the range $1 \le c \le \log f_K$; for even larger packet sizes, the proposed solution is still more efficient provided that $\log f_K \le d \log (N/d)$.

5. SPECIAL CLASSES OF NETWORKS

For special classes of networks, algorithms have been presented which by exploiting topological features of networks yield better bounds than in the general case. In the rest of this

section, the classes of networks studied in the literature are considered. The obtained results will be contrasted with the existing bounds. As in [3], it will be assumed that the file is equi-distributed; that is, N/d elements are stored at each site.

5.1 Networks of diameter 1

Consider the class of networks having a spanning-star subgraph (e.g., complete networks). In these graphs, the best existing bound is [14]:

$$C \leq 9.64\ d\ \log(K/d)\ \lceil \log f_K / c \rceil + 1.o.t.. \qquad (4)$$

The use of the proposed solution yields the following bound:

Property 5.1 Let G be a star graph. Then the K-th smallest element can be determined at the center using at most

$$(d-1)\ \lceil \log (\mathrm{Min}\{K,N/d\})\ /\ c \rceil\ g(f_K) + d - 1$$

packets.

Proof: From Property 4.2 and from the fact that each site is directly connected to the center. []

For small packet sizes (i.e., $1 \leq c \leq \log(\mathrm{Min}\{K,N/d\})$), this represents an improvement in the multiplicative constant of (4). Due to the different position of the "ceiling" operator in the two bounds, the complexity of the proposed algorithm stays unchanged to $O(d\ g(f_K))$ while the other bound decreases as the packets became increasingly larger than $\log(\mathrm{Min}\{K,N/d\})$. This can be reinterpreted as saying that, *if the packet size is independent of the application values* (i.e., the f_i's) then the solution based on the *guessing* approach yields an improvement not only in the *bit* complexity but also in the actual *message* complexity.

5.2 Ring Networks

The best available bound for selection in a ring network is [3]:

$$R \leq O(d\ (\log d)^2 \log(N/\log d) \lceil \log f_K / c \rceil) \qquad (5)$$

Using the proposed method, from Property 4.2 it follows

Property 5.2 Let G be a line graph, and let each site have N/d elements. Then the K-th smallest element can be determined at one end node of the line using at most

$$O(d \lceil \log K / c \rceil\ g(f_K))$$

packets.

which improves the bound (5) for c≤log K; for larger packet sizes, it still improves whenever (log d log(N/log d) ≥ log f_K.

5.3 Complete Binary Trees

The best available bound for selection in a complete binary tree is [3]:

$$B \leq O(d \log(N/d) \lceil \log f_K / c \rceil) \qquad (6)$$

Using the method proposed here, the same number of messages (in order of magnitude) would be exchanged for 1≤c≤ log (N/d) as indicated by the following

Property 5.3 Let G be a complete binary tree, and let each site have N/d elements. Then the K-th smallest element can be determined at the root node using at most

$$O(d \lceil \log (\text{Min}\{K,N/d\}) / c \rceil \ g(f_K))$$

packets.

6. S-CENTERS OF A TREE

The method used for an arbitrary network is to apply the proposed technique on a spanning-tree of the network, and having a node (the "root" of the tree) coordinating the algorithm execution. This node could be the site where the query originated; it could however be a different site. Once a root is given, there is a fixed cost associated with the algorithm, and the actual number of packets sent in the worst case depends on the location of the root. This can bee seen by refining the result of Property 4.2 as follows.

Given a tree T and a node S_j, let T[j] denote T when S_j is chosen as the root; and let $T_i[x]$ and $|T_i[x]|$ denote the subtree of T[j] rooted at S_i and the number of nodes in that subtree respectively.

Property 6.1 Given a tree T and a node S_j, the K-th smallest element can be determined at S_j using at most

$$g(f_K) (d-1) (1 + \Sigma_{i \neq j} \lceil \log (|T_i[j]| N / d) / c \rceil)$$

packets.

Thus, it is whorthwhile to select as a root the site S_j which minimizes the sum $\Sigma_{i \neq j} \lceil \log |T_i[j]| \rceil / c \rceil$ or, equivalently, the product $f(j)=\Pi_{i \neq j} |T_i[j]|$.

Given a tree T, a node S_j is said to be a *s-center* of T if $f(T,j) \leq f(T,i)$ for all S_i in T. It is not difficult to see that the s-centers do not coincide neither with the centers nor with the medians of a tree (except in very special cases); they do however share with centers and medians similiar properties as indicated by the following:

Property 6.2 In any tree T, either there is a unique s-center or there are two s-centers and these s-centers are neighbours.

The proof follows from monotonicity of the cost function: the function decreases moving from a leaf to an s-center where it is minimized; using this property, an s-center of a tree can be easily determined using $O(d\lceil \log d / c \rceil)$ packets with simple modifications to the distributed center-finding algorithm [4]. Given a graph G, a site S_j in G is said to be a s-center of G if S_j is an s-center of a spanning tree T^j of G and, for all S_i which are s-centers of a spanning tree T^i of G, $f(T,j) \leq f(T^i,i)$. For special classes of graphs, an s-center is easily determined (e.g., graphs of diameter 1, meshes); however, to determine the s-center of an arbitrary graph G is rather complex and no solution algorithm (distibuted or serial) is known besides the brute force approach.

Acknowledgement

This research has been supported in part by the Natural Sciences and Engineering Research Council of Canada under Grants A0977 and A2415.

REFERENCES

[1] J.L. Bentley, A.C. Yao, "An almost optimal algorithm for unbounded searching", *Information Processing Letters 5*, 3, Aug. 1976, 82-87.

[2] F. Chin, H.F. Ting, "A near-optimal algorithm for finding the median distributively", *Proc. 5th IEEE Conf. on Distributed Computing Systems*, Denver, May 1985.

[3] G.N. Frederickson, "Tradeoffs for selection in distributed networks", *Proc. 2nd ACM Symp. on Principles of Distributed Computing*, Montreal, Aug. 1983, 154-160.

[4] E. Korach, D. Rotem, N. Santoro, "Distributed algorithms for finding centers and medians in networks", *ACM Trans. on Programming Languages and Systems 6*, 3, July 1984, 380-401.

[5] J. van Leeuwen, N. Santoro, J. Urrutia, S. Zaks, "Guessing games and distributed computations in synchronous networks", *Proc. 14th Int. Conf. on Automata, Languages and Programming*, Karlsruhe, July 1987, 347-356.

[6] J.M. Marberg, E. Gafni, "An optimal shout-echo algorithm for selection in distributed sets", *Proc. 23rd Allerton Conf. on Comm., Control and Computing*, Monticello, Oct. 1985.

[7] T.A. Matsushita, "Distributed algorithms for selection", Master Thesis, Department of Computer Science, University of Illinois, Urbana, July 1983.

[8] M. Rodeh, "Finding the median distributively", *J. Computer and System Science 24*, 2, April 1982, 162-167.

[9] D. Rotem, N. Santoro, J.B. Sidney, "Shout-echo selection in distributed files", *Networks 16*, 1986, 77-86.

[10] D. Rotem, N. Santoro, J.B. Sidney, "Distributed sorting", *IEEE Transactions on Computers C-34*, 4, April 1985, 372-376.

[11] N. Santoro, M. Scheutzow, J.B. Sidney, "On the complexity of distributed selection", *J. Parallel and Distributed Computing*, to appear.

[12] N. Santoro, J.B. Sidney, "Order statistics on distributed sets", *Proc. 20th Allerton Conf. on Communication, Control and Computing*, Monticello, Oct. 1982, 251-256.

[13] N. Santoro, J.B. Sidney, S.J. Sidney, "On the expected complexity of distributed selection", *Proc. 4th Symp. on Theoretical Aspects of Computer Science*, Passau, Feb. 1987, 456-467.

[14] N. Santoro, E. Suen, "Reduction techniques for selection in distributed files", *IEEE Transactions on Computers*, to appear.

[15] L. Shrira, N. Francez, M. Rodeh, "Distributed k-selection: from a sequential to a distributed algorithm", *Proc. 2nd ACM Symp. on Principles of Distributed Computing*, Montreal, Aug. 1983, 143-153.

Carleton University, School of Computer Science
Bibliography of Technical Reports
Publications List (1985 -->)

School of Computer Science
Carleton University
Ottawa, Ontario, Canada
KIS 5B6

SCS-TR-66 **On the Futility of Arbitrarily Increasing Memory Capabilities of Stochastic Learning Automata**
_____ B.J. Oommen, October 1984. Revised May 1985.

SCS-TR-67 **Heaps in Heaps**
_____ T. Strothotte, J.-R. Sack, November 1984. Revised April 1985.

SCS-TR-68 **Partial Orders and Comparison Problems**
out-of-print M.D. Atkinson, November 1984. See Congressus Numerantium 47 ('86), 77-88

SCS-TR-69 **On the Expected Communication Complexity of Distributed Selection**
_____ N. Santoro, J.B. Sidney, S.J. Sidney, February 1985.

SCS-TR-70 **Features of Fifth Generation Languages: A Panoramic View**
_____ Wilf R. LaLonde, John R. Pugh, March 1985.

SCS-TR-71 **Actra: The Design of an Industrial Fifth Generation Smalltalk System**
_____ David A. Thomas, Wilf R. LaLonde, April 1985.

SCS-TR-72 **Minmaxheaps, Orderstatisticstrees and their Application to the Coursemarks Problem**
_____ M.D. Atkinson, J.-R. Sack, N. Santoro, T. Strothotte, March 1985.

SCS-TR-73 **Designing Communities of Data Types**
Wilf R. LaLonde, May 1985.
Replaced by SCS-TR-108

SCS-TR-74 **Absorbing and Ergodic Discretized Two Action Learning Automata**
out-of-print B. John Oommen, May 1985. See IEEE Trans. on Systems, Man and Cybernetics, March/April 1986, pp. 282-293.

SCS-TR-75 **Optimal Parallel Merging Without Memory Conflicts**
_____ Selim Akl and Nicola Santoro, May 1985

SCS-TR-76 **List Organizing Strategies Using Stochastic Move-to-Front and Stochastic Move-to-Rear Operations**
_____ B. John Oommen, May 1985.

SCS-TR-77 **Linearizing the Directory Growth in Order Preserving Extendible Hashing**
_____ E.J. Otoo, July 1985.

SCS-TR-78 **Improving Semijoin Evaluation in Distributed Query Processing**
_____ E.J. Otoo, N. Santoro, D. Rotem, July 1985.

Carleton University, School of Computer Science
Bibliography of Technical Reports

SCS-TR-79 ——— **On the Problem of Translating an Elliptic Object Through a Workspace of Elliptic Obstacles**
B.J. Oommen, I. Reichstein, July 1985.

SCS-TR-80 ——— **Smalltalk - Discovering the System**
W. LaLonde, J. Pugh, D. Thomas, October 1985.

SCS-TR-81 ——— **A Learning Automation Solution to the Stochastic Minimum Spanning Circle Problem**
B.J. Oommen, October 1985.

SCS-TR-82 ——— **Separability of Sets of Polygons**
Frank Dehne, Jörg-R. Sack, October 1985.

SCS-TR-83 out-of-print **Extensions of Partial Orders of Bounded Width**
M.D. Atkinson and H.W. Chang, November 1985. See Congressus Numerantium, Vol. 52 (May 1986), pp. 21-35.

SCS-TR-84 ——— **Deterministic Learning Automata Solutions to the Object Partitioning Problem**
B. John Oommen, D.C.Y. Ma, November 1985

SCS-TR-85 out-of-print **Selecting Subsets of the Correct Density**
M.D. Atkinson, December 1985. To appear in Congressus Numerantium, Proceedings of the 1986 South-Eastern conference on Graph theory, combinatorics and Computing.

SCS-TR-86 ——— **Robot Navigation in Unknown Terrains Using Learned Visibility Graphs. Part I: The Disjoint Convex Obstacles Case**
B. J. Oommen, S.S. Iyengar, S.V.N. Rao, R.L. Kashyap, February 1986

SCS-TR-87 ——— **Breaking Symmetry in Synchronous Networks**
Greg N. Frederickson, Nicola Santoro, April 1986

SCS-TR-88 ——— **Data Structures and Data Types: An Object-Oriented Approach**
John R. Pugh, Wilf R. LaLonde and David A. Thomas, April 1986

SCS-TR-89 ——— **Ergodic Learning Automata Capable of Incorporating Apriori Information**
B. J. Oommen, May 1986

SCS-TR-90 ——— **Iterative Decomposition of Digital Systems and its Applications**
Vaclav Dvorak, May 1986.

SCS-TR-91 ——— **Actors in a Smalltalk Multiprocessor: A Case for Limited Parallelism**
Wilf R. LaLonde, Dave A. Thomas and John R. Pugh, May 1986

SCS-TR-92 ——— **ACTRA - A Multitasking/Multiprocessing Smalltalk**
David A. Thomas, Wilf R. LaLonde, and John R. Pugh, May 1986

SCS-TR-93 ——— **Why Exemplars are Better Than Classes**
Wilf R. LaLonde, May 1986

SCS-TR-94 ——— **An Exemplar Based Smalltalk**
Wilf R. LaLonde, Dave A. Thomas and John R. Pugh, May 1986

SCS-TR-95 ——— **Recognition of Noisy Subsequences Using Constrained Edit Distances**
B. John Oommen, June 1986

Carleton University, School of Computer Science
Bibliography of Technical Reports

SCS-TR-96 **Guessing Games and Distributed Computations in Synchronous Networks**
J. van Leeuwen, N. Santoro, J. Urrutia and S. Zaks, June 1986.

SCS-TR-97 **Bit vs. Time Tradeoffs for Distributed Elections in Synchronous Rings**
M. Overmars and N. Santoro, June 1986.

SCS-TR-98 **Reduction Techniques for Distributed Selection**
N. Santoro and E. Suen, June 1986.

SCS-TR-99 **A Note on Lower Bounds for Min-Max Heaps**
A. Hasham and J.-R. Sack, June 1986.

SCS-TR-100 **Sums of Lexicographically Ordered Sets**
M.D. Atkinson, A. Negro, and N. Santoro, May 1987.

SCS-TR-102 **Computing on a Systolic Screen: Hulls, Contours, and Applications**
F. Dehne, J.-R. Sack and N. Santoro, October 1986.

SCS-TR-103 **Stochastic Automata Solutions to the Object Partitioning Problem**
B.J. Oommen and D.C.Y. Ma, November 1986.

SCS-TR-104 **Parallel Computational Geometry and Clustering Methods**
F. Dehne, December 1986.

SCS-TR-105 **On Adding *Constraint Accumulation* to Prolog**
Wilf R. LaLonde, January 1987.

SCS-TR-107 **On the Problem of Multiple Mobile Robots Cluttering a Workspace**
B. J. Oommen and I. Reichstein, January 1987.

SCS-TR-108 **Designing Families of Data Types Using Exemplars**
Wilf R. LaLonde, February 1987.

SCS-TR-109 **From Rings to Complete Graphs - Θ(n logn) to Θ(n) Distributed Leader Election**
Hagit Attiya, Nicola Santoro and Shmuel Zaks, March 1987.

SCS-TR-110 **A Transputer Based Adaptable Pipeline**
Anirban Basu, March 1987.

SCS-TR-111 **Impact of Prediction Accuracy on the Performance of a Pipeline Computer**
Anirban Basu, March 1987.

SCS-TR-112 **ε-Optimal Discretized Linear Reward-Penalty Learning Automata**
B.J. Oommen and J.P.R. Christensen, May 1987.

SCS-TR-113 **Angle Orders, Regular n-gon Orders and the Crossing Number of a Partial Order**
N. Santoro and J. Urrutia, June 1987.

SCS-TR-115 **Time is Not a Healer: Impossibility of Distributed Agreement in Synchronous Systems with Random Omissions**
N. Santoro, June 1987.

Carleton University, School of Computer Science
Bibliography of Technical Reports

SCS-TR-116 **A Practical Algorithm for Boolean Matrix Multiplication**
M.D. Atkinson and N. Santoro, June 1987.

SCS-TR-117 **Recognizing Polygons, or How to Spy**
James A. Dean, Andrzej Lingas and Jörg-R. Sack, August 1987.

SCS-TR-118 **Stochastic Rendezvous Network Performance - Fast, First-Order Approximations**
J.E. Neilson, C.M. Woodside, J.W. Miernik, D.C. Petriu, August 1987.

SCS-TR-120 **Searching on Alphanumeric Keys Using Local Balanced Tree Hashing**
E.J. Otoo, August 1987.

SCS-TR-121 **An O(√n) Algorithm for the ECDF Searching Problem for Arbitrary Dimensions on a Mesh-of-Processors**
Frank Dehne and Ivan Stojmenovic, October 1987.

SCS-TR-122 **An Optimal Algorithm for Computing the Voronoi Diagram on a Cone**
Frank Dehne and Rolf Klein, November 1987.

SCS-TR-123 **Solving Visibility and Separability Problems on a Mesh-of-Processors**
Frank Dehne, November 1987.

SCS-TR-124 **Deterministic Optimal and Expedient Move-to-Rear List Organizing Strategies**
B.J. Oommen, E.R. Hansen and J.I. Munro, October 1987.

SCS-TR-125 **Trajectory Planning of Robot Manipulators in Noisy Workspaces Using Stochastic Automata**
B.J. Oommen, S. Sitharam Iyengar and Nicte Andrade, October 1987.

SCS-TR-126 **Adaptive Structuring of Binary Search Trees Using Conditional Rotations**
R.P. Cheetham, B.J. Oommen and D.T.H. Ng, October 1987.

SCS-TR-127 **On the Packet Complexity of Distributed Selection**
A. Negro, N.Santoro and J. Urrutia, November 1987.