An $O(\sqrt{n})$ Algorithm for the ECDF Searching Problem for Arbitrary Dimensions on a Mesh-of-Processors

Frank Dehne[1], Ivan Stojmenovic[2]

SCS-TR-121

Oct. 1987

1) School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6
Research supported by NSERC grant No. A9173
2) Institute of Mathematics, University of Novi Sad, 2100 Novi Sad, Yugoslavia

Abstract

[1] presented an optimal $O(\sqrt{n})$ time parallel algorithm for solving the ECDF searching problem for a set of n points in two- and three-dimensional space on a Mesh-of-Processors of size n. However, it remained an open problem whether such an optimal solution exists for the d-dimensional ECDF searching problem for $d \geq 4$.

In this paper we solve this problem by presenting an optimal $O(\sqrt{n})$ time parallel solution to the d-dimensional ECDF searching problem for arbitrary dimension $d = O(1)$ on a Mesh-of-Processors of size n.

The algorithm has several interesting implications. Among others the following problems can now be solved on a Mesh-of-Processors in (asymptotically optimal) time $O(\sqrt{n})$ for arbitrary dimension $d=O(1)$: the d-dimensional maximal element determination problem, the d-dimensional hypercube containment counting problem, and the d-dimensional hypercube intersection counting problem. The latter two problems can be mapped to the 2d-dimensional ECDF searching problem but require an efficient solution to this problem for at least $d \geq 4$.

keywords: ECDF searching, Mesh-of-Processors, Parallel Computational Geometry

1. Introduction

Given a set $S=\{p_1,...,p_n\}$ of n points in d-dimensional space; $d=O(1)$. A point p_i dominates a point p_j ($p_i > p_j$) if and only if $p_i[k] > p_j[k]$ for all $k \in \{1,...,d\}$, where $p[k]$ denotes the k^{th} coordinate of a point p. The d-dimensional ECDF searching problem consists of computing for each $p \in S$ the number $D(p,S)$ of points of S dominated by p (For more details on this problem consult e.g. [5], [6]).

An efficient solution to the ECDF searching problem has several interesting applications ([2], [5], [6]). One of these is the well known transformation of the rectangle containment counting problem to the ECDF searching problem ([2], [6]). The rectangle containment counting problem consist of counting for each rectangle R of a set of iso-oriented rectangles the number of rectangles R which are contained in R. If we map each rectangle $R=[x_1,x_2]x[y_1,y_2]$ into the four-dimensional point $R'= (-x_1, x_2, -y_1,y_2)$ then a rectangle R_1 contains a rectangle R_2 if and only if $R_2' <= R_1'$, hence, the problem is easily transformed into a four-dimensional ECDF searching problem.

In [1] an optimal $O(\sqrt{n})$ time parallel algorithm was introduced for solving the two- and three-dimensional ECDF searching problem on a Mesh-of-Processors of size n, i.e. a set of n processing elements (PEs) arranged on a $\sqrt{n} \times \sqrt{n}$ grid where each PE is connected to its direct neighbors by bidirectional communication links. (For a more detailed description of the Mesh-of-Processors architecture and basic algorithm design techniques on these machines consult e.g. [4], [8].)

However, the existence of an optimal $O(\sqrt{n})$ time solution to the d-dimensional ECDF searching problem for $d \geq 4$ remained an open problem.

In this paper we will solve this problem by introducing an optimal $O(\sqrt{n})$ time solution to the d-dimensional ECDF searching problem for arbitrary dimension $d=O(1)$.

2. Description and Analysis of the Proposed Algorithm

In order to obtain a convenient description of the algorithm we introduce the following definitions :

(1) Let p,q be two points in d-space and $1 \leq k \leq d$, then $q <_k p$ if and only if $q[1]<p[1], \ldots, q[k]<p[k]$.

(2) Let p be a point in d-space, S_1 be a subset of S, and $1 \leq k \leq d$, then $M^k(p,S_1)$ denotes the number of those $q \in S_1$ such that $q <_k p$.

(3) Let S_1, S_2 be two subsets of S, and $1 \leq k \leq d$, then k-dimensional dominance merge, denoted by $MERGE^k(S_2,S_1)$, consists of computing the value $M^k(p,S_1)$ for all $p \in S_2$.

2.1. Global Structure of the Algorithm

Initially, each processing element of the mesh contains the d coordinates of one point of S. Each PE is assumed to have a register D which will contain the value D(p,S), where p is the point stored in the respective PE, after the algorithm has terminated.

The global structure of the proposed algorithm is a divide-and-conquer mechanism which solves the problem as follows :

(I) Divide

Partition S into two subsets S_1 and S_2 by comparing the d^{th} coordinate of the points with their median d^{th} coordinate (points in S_2 have larger d^{th} coordinate). S_1 and S_2 are stored in one half of the Mesh-of-Processors, each.

(This step is easily obtained by sorting S with resp. to the d-coordinate; see e.g. [7]. The mesh is split into two submeshes of equal size by either a vertical or a horizontal line to minimize the diameter of the submeshes.)

(II) Recur

Solve the d-dimensional ECDF searching problem for S_1 and S_2, respectively, on each half of the Mesh-of-Processors in parallel.

(III) Merge

(a) Solve the (d-1)-dimensional dominance merge problem $MERGE_{d-1}(S_2, S_1)$.

(b) Update:

Each PE updates his register D as follows:

$$D(p,S) := \begin{cases} D(p,S_1) & \text{for } p \in S_1 \\ D(p,S_2) + M_{d-1}(p,S_1) & \text{for } p \in S_2 \end{cases}$$

The following section shows how to solve the k-dimensional dominance merge problem $MERGE^k(S_2, S_1)$, $1 \le k \le d$, as required for step (III,a).

2.2. K-Dimensional Dominance Merge $MERGE^k(S_2, S_1)$

The structure of the k-dimensional dominance merge algorithm is again a divide-and-conquer mechanism. In each iteration, k decreases by one, i.e. the merge step for k-dimensional dominance merge involves the solution of a (k-1)-dimensional dominance merge problem. This process is iterated until k=1.

Each PE is assumed to have a register M which will finally contain the value $M^k(p,S_1)$ for $p \in S_2$, where p is the point stored in the respective PE.

$k \ge 2$:

(I) Divide

Partition S_1 into two subsets S_{11} and S_{12} and, simultaneously, S_2 into two subsets S_{21} and S_{22} by comparing the k^{th} coordinate of the points with the median k^{th} coordinate of $S_1 \cup S_2$ (points in S_{12} and S_{22} have larger k^{th} coordinate). Store $S_{21} \cup S_{11}$ and $S_{22} \cup S_{12}$ on one half of the current submesh, each. (Again, split the current submesh into two submeshes of equal size using either a vertical or a horizontal split line to minimize the diameter of the submeshes.)

(II) Recur

Solve the k-dimensional dominance merge problems $MERGE^k(S_{21}, S_{11})$ and $MERGE^k(S_{22}, S_{12})$, respectively, on each half of the Mesh-of-Processors in parallel.

(III) Merge

(a) Solve the (k-1)-dimensional dominance merge problem $MERGE^{k-1}(S_{22}, S_{11})$.

(b) Update:

Each PE updates his register M as follows:

$$M^k(p,S_1) := \begin{cases} M^k(p,S_{11}) & \text{for } p \in S_{21} \\ M^k(p,S_{12}) + M^{k-1}(p,S_{11}) & \text{for } p \in S_{22} \end{cases}$$

k=1 :

Sort $S_1 \cup S_2$ with respect to the first coordinate in snake like ordering ([7]). For each $p \in S_2$, $M^k(p,S_1)$ is the number of $q \in S_1$ with lower rank.

2.3. Time Complexity of the Proposed Algorithm

Let $T_{ECDF}(n)$ and $m^k(n)$ denote the time complexity for solving the d-dimensional ECDF searching problem (d=O(1)) for a set of n points and the k-dimensional dominance merge problem $MERGE^k(S_2,S_1)$ for $|S_2 \cup S_1|=n$, respectively, as described above.

With these definitions the following recurrence relations are easily observed:

(1) $T_{ECDF}(n) = T_{ECDF}(\frac{n}{2}) + m^{d-1}(n) + O(\sqrt{n})$

(2) $m^k(n) = m^k(\frac{n}{2}) + m^{k-1}(n) + O(\sqrt{n})$

$m^1(n) = O(\sqrt{n})$.

Since $k \leq d=O(1)$, it follows from (2) that

$m^k(n) = O(\sqrt{n})$.

Hence, $m^{d-1}(n) = O(\sqrt{n})$ and, thus, it follows from (1) that

$T_{ECDF}(n) = O(\sqrt{n})$, too.

This yields the following

theorem:

The d-dimensional ECDF searching problem, d=O(1), for a set of n points can be solved on a Mesh-of-Processors of size n in time $O(\sqrt{n})$ which is asymptotically optimal.

3. Conclusion

In this paper we introduced an optimal $O(\sqrt{n})$ time parallel solution to the d-dimensional ECDF searching problem for arbitrary dimension d=O(1) on a Mesh-of-Processors of linear size which solves the problem that remained open in [1].

The algorithm has several interesting implications. Among others the following problems can now be solved on a Mesh-of-Processors of linear size in (asymptotically optimal) time $O(\sqrt{n})$:

- d-dimensional maximal element determination (d=O(1)):
  compute the set of points which are not dominated by any other point
- d-dimensional hypercube containment counting problem (d=O(1)): d-dimensional generalization of the rectangle containment counting problem described above (mapping to 2d-dimensional ECDF searching problem is straight forward)
- d-dimensional hypercube intersection counting problem (d=O(1)): d-dimensional generalization of the rectangle intersection counting problem; i.e. given a set S of iso-oriented rectangles determine for each rectangle R the number of rectangles that intersects R.
  Each rectangle $R=[x_1,x_2]x[y_1,y_2]$ is mapped into the four-dimensional points $R'= (-x_1,x_2,-y_1,y_2)$ and $R''= (-x_2,x_1,-y_2,y_1)$. Two rectangles R_1 and R_2 intersect if and only if $R_2'' <= R_1'$ or, equivalently, $R_1'' <= R_2'$ ([2], [6]). Hence, with S' and S" denoting the set of all R' and R", respectively, then the rectangle intersection counting problem is equivalent to 4-dimensional dominance merge, i.e. $MERGE^4(S', S'')$.
  Analogously, the d-dimensional hypercube intersection counting problem can be mapped into a 2d-dimensional dominance merge problem.

References

[1] F.Dehne, "O($n^{1/2}$) Algorithms for the Maximal Elements and ECDF Searching Problem on a Mesh-Connected Parallel Computer", *Information Processing Letters*, 22, 1986, pp.303-306

[2] H.Edelsbrunner, M.H.Overmars, "On the Equivalence of Some Rectangle Problems", *Information Processing Letters*, 14, 1982, pp.124-127

[3] S.Hambrusch, private communication

[4] R.Miller, Q.F.Stout, "Computational Geometry on a Mesh-Connected Computer", Proc. IEEE Int. Conf. on Parallel Processing,1984, pp.66-73

[5] M.H.Overmars, J.v.Leeuwen, "Maintenance of Configurations in the Plane", Report RUU-CS-81-3, Dept. of Computer Science, Univ. of Utrecht, 1981

[6] F.P.Preparata, M.I.Shamos, "Computational Geometry, An Introduction", Springer-Verlag, New York, 1985

[7] C.D.Thompson, H.T.Kung, "Sorting on a Mesh-Connected Parallel Computer", *Comm. of the ACM*, Vol.20, No.4, April 1977

[8] J.D.Ullman, "Computational Aspects of VLSI", Principles of Computer Science Series, Computer Science Press, 1984

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 out-of-print — **Partial Orders and Comparison Problems**
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 out-of-print — **Absorbing and Ergodic Discretized Two Action Learning Automata**
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.

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 l985.

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 Visiblity 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

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 - $\Theta(n \log n)$ to $\Theta(n)$ 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.

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 Trie Hashing**
E.J. Otoo, August 1987.

SCS-TR-121 **An O($\sqrt{n}$) 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.