Information Processing Letters 28 (1988) 67–70
North-Holland

24 June 1988

An $O(\sqrt{n})$ TIME ALGORITHM FOR THE ECDF SEARCHING PROBLEM FOR ARBITRARY DIMENSIONS ON A MESH-OF-PROCESSORS

Frank DEHNE*

School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6

Ivan STOJMENOVIC

Institute of Mathematics, University of Novi Sad, 2100 Novi Sad, Yugoslavia

Communicated by E.C.R. Hehner
Received 16 November 1987
Revised 22 February 1988

Dehne (1986) 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

A set $S = \{p_1, p_2, \ldots, p_n\}$ of n points in d-dimensional space is given. A point p_i *dominates* a point p_j (denoted $p_i > p_j$) if and only if $p_i[k] > p_j[k]$ for all $k \in \{1, 2, \ldots, d\}$ where $p[k]$ denotes the kth 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 about this problem, consult, e.g., [7,8].)

An efficient solution to the ECDF searching problem has several interesting applications [3,7,8]. One of these is the well-known transformation of the rectangle containment counting problem to the ECDF searching problem [3,8]. The rectangle containment counting problem consists of counting for each rectangle R of a set of iso-oriented rectangles the number of rectangles which are contained in R. If we map each rectangle $R = [x_1, x_2] \times [y_1, y_2]$ 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 $R_2' \Leftarrow R_1'$; hence, the problem is easily transformed into a four-dimensional ECDF searching problem.

In [2], 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 $\sqrt{n} \times \sqrt{n}$

* The research reported here was supported by the National Science and Engineering Research Council under Grant No. A9173.

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., [5,10].)

However, the algorithm in [2] did not solve, in $O(\sqrt{n})$ time, the d-dimensional ECDF searching problem for $d \geq 4$, and the existence of an optimal solution for $d \geq 4$ remained an open problem [4].

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 $d = O(1)$. Miller and Stout [6] have also considered this problem and, independently, suggested the same solution based on Bentley's multidimensional divide-and-conquer technique [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 let $1 \leq k \leq d$; then $q <_k p$ if and only if $q[2] < p[2]$, ..., $q[k] < p[k]$.

(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 let $1 \leq k \leq d$; then k-dimensional dominance merge, denoted $\text{MERGE}^k(S_2, S_1)$, 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 dth coordinate of the points with the median dth coordinate (points in S_2 have larger dth 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 respect to the dth coordinate (see, e.g., [9]). 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 $\text{MERGE}^{d-1}(S_2, S_1)$.
(b) ***Update***
Each PE updates its 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 subsection shows how to solve the k-dimensional dominance merge problem $\text{MERGE}^k(S_2, S_1)$, $1 \leq k \leq d$, as required for step III(a).

2.2. k-dimensional dominance merge $\text{MERGE}^k(S_2, S_1)$

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.

Case $k \geqslant 2$

(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 kth coordinate of the points with the median kth coordinate of $S_1 \cup S_2$ (points in S_{12} and S_{22} have larger kth coordinate). Store $S_{21} \cup S_{11}$ and $S_{22} \cup S_{12}$ in 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 denoted $\text{MERGE}^k(S_{21}, S_{11})$ and $\text{MERGE}^k(S_{22}, S_{12})$, respectively, on each half of the mesh-of-processors in parallel.

(III) ***Merge***

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

(b) ***Update***

Each PE updates its register M as follows:

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

Case $k = 1$

Sort $S_1 \cup S_2$ with respect to the first coordinate in snake-like ordering [9]. For each $p \in S_2$, the value $M^k(p, S_1)$ is equal to the number of $q \in S_1$ with lower rank.

2.3. Time complexity of the proposed algorithm

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

With these definitions, the following recurrence relations are easily observed:

$$(1)\quad T_{\text{ECDF}}(n) = T_{\text{ECDF}}\left(\tfrac{1}{2}n\right) + m_{d-1}(n) + O(\sqrt{n}),$$
$$T_{\text{ECDF}}(n) = O(1).$$

$$(2)\quad m_k(n) = m_k\left(\tfrac{1}{2}n\right) + m_{k-1}(n) + O(\sqrt{n}),$$
$$m_1(n) = O(\sqrt{n}).$$

From (2) it follows that

$$m_k(n) = O(\mu^k \sqrt{n}),$$

$$\mu = \frac{\sqrt{2}}{\sqrt{2}-1} + \varepsilon \approx 3.414213\ldots + \varepsilon \quad (\varepsilon > 0).$$

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

$$T_{\text{ECDF}}(n) = O(\mu^{d-1}\sqrt{n}).$$

For any $d = O(1)$, i.e., any fixed dimension d, this yields the following result.

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* $O(\sqrt{n})$ *time which is asymptotically optimal.*

3. Applications

The above algorithm has several interesting applications. Among other, the following problems, for arbitrary dimension $d = O(1)$, can now be solved on a mesh-of-processors of linear size in (asymptotically optimal) time $O(\sqrt{n})$:

- The d-dimensional maximal element determination problem: compute the set of points which are not dominated by any other point.
- The d-dimensional hypercube containment counting problem, i.e., the d-dimensional generalization of the rectangle containment counting problem described in Section 1 (mapping the

d-dimensional problem into a $2d$-dimensional ECDF searching problem is straightforward).

The d-dimensional hypercube intersection counting problem, i.e., the d-dimensional generalization of the rectangle intersection counting problem: given a set S of iso-oriented rectangles, determine for each rectangle R the number of rectangles that intersect R. Each rectangle $R = [x_1, x_2] \times [y_1, y_2]$ 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 $R_2'' \Leftarrow R_1'$ or, equivalently, $R_1'' \Leftarrow R_2'$ [3,8]. Hence, with S', S'' denoting the set of all R', respectively R'', the rectangle intersection counting problem is equivalent to four-dimensional dominance merge MERGE$^4(S', S'')$. Analogously, the d-dimensional hypercube intersection counting problem can be mapped into a $2d$-dimensional dominance merge problem.

References

[1] J.L. Bentley, Multidimensional divide-and-conquer, *Comm. ACM* **23** (4) (1980) 214–229.

[2] F. Dehne, $O(n^{1/2})$ algorithms for the maximal elements and ECDF searching problem on a mesh-connected parallel computer *Inform. Process. Lett.* **22** (6) (1986) 303–306.

[3] H. Edelsbrunner and M.H. Overmars, On the equivalence of some rectangle problems, *Inform. Process. Lett.* **14** (3) (1982) 124–127.

[4] S. Hambrusch, Private communication, September 1985.

[5] R. Miller and Q.F. Stout, Computational geometry on a mesh-connected computer, *Proc. IEEE Internat. Conf. on Parallel Processing* (1984) 66–73.

[6] R. Miller and Q.F. Stout, *Mesh Computer Algorithms for Computational Geometry*, Tech. Rept. 86-18 (revised), Dept. of Computer Science, State Univ. of New York, Buffalo, NY, 1986.

[7] M.H. Overmars and J. Van Leeuwen, Maintenance of configurations in the plane, *J. Comput. System Sci.* **23** (1981) 166–204.

[8] F.P. Preparata and M.I. Shamos, *Computational Geometry — An Introduction* (Springer, New York, NY, 1985).

[9] C.D. Thompson and H.T. Kung, Sorting on a mesh-connected parallel computer, *Comm. ACM* **20** (4) (1977) 263–271.

[10] J.D. Ullman, *Computational Aspects of VLSI* (Computer Science Press, Rockville, MD, 1984).