AN OPTIMAL PARALLEL SOLUTION TO THE ECDF SEARCHING PROBLEM FOR HIGHER DIMENSIONS ON A MESH-OF-PROCESSORS

FRANK DEHNE[1], IVAN STOJMENOVIC [2]
[1]School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6
[2] Institute of Mathematics, University of Novi Sad, 2100 Novi Sad, Yugoslavia

Abstract

[D86] presented an optimal $O(n^{1/2})$ time parallel algorithm for solving the ECDF searching problem on a mesh-of-processors for a set of n points in two- and three-dimensional space. 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 introducing an optimal $O(n^{1/2})$ time parallel solution to the d-dimensional ECDF searching problem for arbitrary dimension $d = O(1)$. The algorithm has several interesting implications. Among others the following problems can now be solved on a mesh-of-processors in (asymptotically optimal) time $O(n^{1/2})$ 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 $d \geq 4$.

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 $(p_i > p_j)$, iff $p_i[k] > p_j[k]$ 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. [OL81], [PS85]). An efficient solution to the ECDF searching problem has several interesting applications; [EO82], [OL81], [PS85]. One of these is e.g. the well known transformation of the rectangle containment counting problem to the ECDF searching problem; [EO82], [PS85]. 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 iff $R_2' <= R_1'$, hence the problem is easily transformed into a four-dimensional ECDF searching problem. In [D86] an optimal $O(n^{1/2})$ parallel algorithm was introduced for solving the two- and three-dimensional ECDF searching problem on a mesh-connected parallel computer (For a description of mesh-connected parallel computers and basic algorithm design techniques on these machines consult [MS84], [UL84]). However, the existance of an optimal $O(n^{1/2})$ 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 $O(n^{1/2})$ time solution to the d-dimensional ECDF searching problem for arbitrary dimension $d = O(1)$.

2. Description and Analysis of Proposed Algorithm

In order to obtain a conveniant description of the algorithm we introduce the following definitions: Let p,q be two points in d-space $(1 \leq k \leq d)$ and S_1, S_2 be two subsets of S, then (a) $q <^k p$ iff $q[1]<p[1], ..., q[k]<p[k]$, (b) $M^k(p,S_1)$ denotes the number of those $q \in S_1$ such that $q <^k p$, and (c) *k-dimensional dominance merge*, denoted by $MERGE^k(S_2, S_1)$, consists of computing the value $M^k(p,S_1)$ for all $p \in S_2$.
Initially, each processing element (PE) 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.

2.1. Global Structure of Algorithm

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. 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. [TK77].)

(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 (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 $MERGE^k(S_2, S_1)$, $1 \leq k \leq d$, as required for step (III,a).

F. Dehne and I. Stojmenovic, "An optimal parallel solution to the ECDF searching problem for higher dimensions on a mesh-of-processors," in Proc. *Allerton Conference on Communication, Control and Computing*, Monticello, Ill., 1987, pp. 660-661.

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 is decremented 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 all $p \varepsilon S_2$, where p is the point stored in the respective PE.

$k \geq 2$: (I) Divide: Partition S_1 into two subsets S_{11} and S_{12} and S_2 into two subsets S_{21} and S_{22} by comparing the k-th coordinate of the points with the median d-th coordinate of $S_2 \cup S_1$.

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

(b) Update: Each PE updates his register M as follows:

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

$k=1$: Sort $S_2 \cup S_1$ with respect to the first coordinate in snake like ordering [TK77]. For each $p \varepsilon S_2$, $M^k(p,S_1)$ is the number of $q \varepsilon S_1$ with lower rank.

2.3. Time Complexity of 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) $T_{ECDF}(n)=T_{ECDF}(n/2)+m^{d-1}(n)+O(n^{1/2})$ (2) $m^k(n)=m^k(n/2)+m^{k-1}(n)+O(n^{1/2})$, $m^1(n)=O(n^{1/2})$.

Since $k \leq d = O(1)$, it follows from (2) that $m^k(n) = O(n^{1/2})$. Hence, $m^{d-1}(n) = O(n^{1/2})$ and, thus, it follows from (1) that $T_{ECDF}(n) = O(n^{1/2})$, 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 $O(n^{1/2})$ which is asymptotically optimal.

3. Conclusion

The algorithm has several interesting implications. Among others the following problems can now be solved on a mesh-of-processors in (asymptotically optimal) time $O(n^{1/2})$:

- 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 iff $R_2'' \leq R_1'$ or, equivalently, $R_1'' \leq R_2'$; [EO82], [PS85]. Hence, with S' and S" denoting the set of all R' and R", respectively, for $R \varepsilon S$ the rectangle intersection counting problem is equivalent to 4-dimensional dominance merge, i.e. $MERGE^4(S', S'')$.

References

[D86] 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

[EO82] H.Edelsbrunner, M.H.Overmars, "On the Equivalence of Some Rectangle Problems", Information Processing Letters, 14, 3, 1982, pp.124-127

[H85] S.Hambrusch, private communication

[MS84] R.Miller, Q.F.Stout, "Computational Geometry on a Mesh-Connected Computer", Proc. IEEE 1984 Int. Conf. on Parallel Processing, pp.66-73

[OL81] M.H.Overmars, J.v.Leeuwen, "Maintenance of Configurations in the Plane", Report RUU-CS-81-3, Dept. of Computer Science, Univ. of Utrecht, Febr. 1981

[PS85] F.P.Preparata, M.I.Shamos, "Computational Geometry, An Introduction", Springer-Verlag, New York, 1985

[TK77] C.D.Thompson, H.T.Kung, "Sorting on a Mesh-Connected Parallel Computer", C.ACM, Vol.20, No.4, April 1977

[UL84] J.D.Ullman, "Computational Aspects of VLSI", Principles of Computer Science Series, Computer Science Press, 1984