

Short Communication

An optimal parallel algorithm for solving the maximal elements problem in the plane

Ivan STOJMENOVIĆ

Institute of Mathematics, University of Novi Sad, 21000 Novi Sad, Yugoslavia

Masahiro MIYAKAWA

Electrotechnical Laboratory, 1-1-4 Umezono, Tsukuba, Ibaraki 305, Japan

Received June 1987

Revised November 1987

Abstract. We describe an $O(\log(n))$ time with $O(n)$ processors optimal algorithm for finding the maximal elements of a set. The model of parallel computation we consider is the CREW-PRAM, i.e. it is the synchronous shared memory model where concurrent reads are allowed but no two processors can simultaneously attempt to write in the same memory location (even if they are trying to write the same thing).

Keywords. Parallel computation, maximal elements problem in the plane, optimal parallel algorithm, CREW-PRAM model.

Since they involve asking basic questions about sets of points, polygons, etc., geometric problems arise in many applications [7] and often it is critical to have algorithms whose speeds are close to real-time [1]. Using the CREW-PRAM model of parallel computation we are interested in achieving the optimal algorithm for solving the maximal elements problem. Dehne [4] gave a parallel solution to the problem for a different model of parallel computation and by using another idea. In [3] it is claimed that the 2-dimensional maximal elements problem can easily be solved in $O(\log(n))$ time and $O(n)$ space using the sorted network [2] and parallel prefix [5]. However, no details are given. We present another optimal solution to the same problem.

Given a point p , $p.x$ and $p.y$ denote the x - and y -coordinate of p , respectively.

Let $S = \{s(1), \dots, s(n)\}$ be a set of n points in the Euclidean plane. A point q dominates a point p , written $p \leq q$, iff $p.x \leq q.x$ and $p.y \leq q.y$. A point $s \in S$ is called *maximal* if there is no other $s' \in S$ with $s \leq s'$.

We are interested in computing the maximal elements of S , called *m-contour* $m(S)$. Our algorithm is optimal with respect to both the time and number of processors, since the problem has an $\Omega(n * \log(n))$ time sequential lower bound [7] and an obvious $\Omega(\log(n))$ time lower bound for the parallel machine considered in this paper.

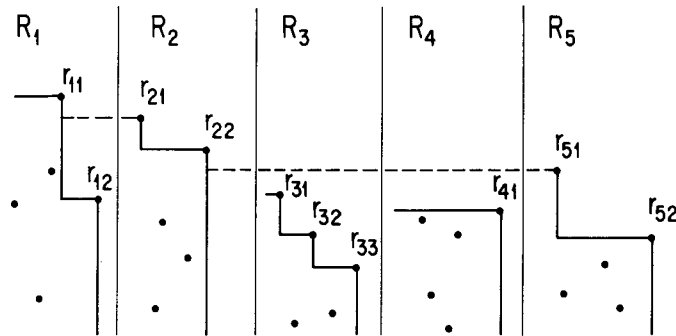


Fig. 1.

The following algorithm will compute m -contour $m(S)$. The main idea is to divide the problem into \sqrt{n} subproblems of size \sqrt{n} each, solve the subproblem recursively in parallel, and combine the solutions of the subproblems quickly (that is, in $O(\log(n))$ time) and with a linear number of processors. It is well known that parallel sorting (initial step in the algorithm) can be done in $O(\log(n))$ steps with $O(n)$ processors [2,6]

Algorithm.

Step 1. Sort the n points by x -coordinate and partition S into sets $R(1), R(2), \dots, R(\sqrt{n})$, each of size \sqrt{n} , divided by vertical cut-lines, such that $R(i)$ is left of $R(j)$ if $i < j$.

Step 2. Recursively solve the problem for each $R(i)$, $i \leq \sqrt{n}$ in parallel. After these parallel recursive call returns we will have $m(R(i))$ for each $R(i)$. Let $r(i, 1), \dots, r(i, u(i))$ ($u(i) \leq \sqrt{n}$) be the sequence of points in $m(R(i))$ (see Fig. 1). In parallel for each $i \in \{1, 2, \dots, \sqrt{n}\}$ use $\sqrt{n} - i$ processors to find those points of $m(R(i))$ which belong to $m(S)$ by doing the following.

Step 2.1. For given $j \in \{i+1, \dots, \sqrt{n}\}$ find the maximal index $t = t(i, j)$ such that $r(i, t).y \geq r(j, 1).y$. If there is no such point, then let $t(i, j) := 0$. This can be done by binary search technique in $\log(n)$ steps in parallel.

Step 2.2. In $\log(n)$ time by the $\sqrt{n} - i$ processors assigned to $m(R(i))$ one can find $t(i) := t(i, s) := \min(t(i, i+1), \dots, t(i, \sqrt{n}))$. If $t(i) = 0$, then no point from $R(i)$ is an element of $m(S)$. Otherwise the next element of $r(i, t(i))$ in $m(S)$ is $r(s, 1)$.

The proof that the algorithm works correctly in optimal $O(\log(n))$ time and with $O(n)$ processors is straightforward.

Acknowledgment

We are indebted to Professors Tetsuo Asano and Hiroshi Imai for providing useful comments.

References

- [1] A. Aggarwal, B. Chazalle, L. Guibas, C. Ó'Dúnlaing and C. Yap, Parallel computational geometry, *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science*, Portland (1985) 468–477.
- [2] M. Ajtai, J. Komlos and E. Szemerédi, An $O(n \cdot \log(n))$ sorting network, *Combinatorica* 3 (1) (1983) 1–19.
- [3] M.J. Atallah and M.T. Goodrich, Efficient plane sweeping in parallel, *Proc. ACM Symposium on Computational Geometry* (1986) 216–225.

- [4] F. Dehne, $O(\sqrt{n})$ algorithm for the maximal elements and ECDF searching problem on a mesh-connected parallel computer, *Inform. Process. Lett.* **22** (1986) 303–306.
- [5] L. Kruskal, L. Rudolph and M. Snir, The power of parallel prefix, *Proc. 1985 International Conference on Parallel Processing* (1985) 180–185.
- [6] T. Leighton, Tight bound on the complexity of parallel sorting, *IEEE Trans. Comput.* **34** (4) (1985) 344–354.
- [7] F.P. Preparata and M.I. Shamos, *Computational Geometry, An Introduction* (Springer, New York, 1985).