International Journal of Computational Geometry & Applications, Vol. 2, No. 3 (1992) 341–343

ERRATUM

RANDOMIZED PARALLEL ALGORITHMS FOR TRAPEZOIDAL DIAGRAMS

[INT. J. COMPUT. GEOMETRY APPL., Vol. 2, No. 2 (1992) 117–133]
K. L. CLARKSON, R. COLE and R. E. TARJAN

The following are the references and appendix that should appear after Ref. 19 on page 133 of the paper.

of hyperplanes in d dimensions", *Proc. Second Symposium on Parallel Algorithms and Architectures*, pages 290–297, 1990.

21. K. Hoffman, K. Mehlhorn, P. Rosenstiehl, and R. Tarjan, "Sorting jordan sequences in linear time using level-linked search trees", *Information and Control*, pages 170–184, 1986.
22. K. Mulmuley, "A fast planar point location algorithm: part I", *Proc. 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 580–589, 1988.
23. S. Rajasekaran and J.H. Reif, "Optimal and sublogarithmic time randomized parallel sorting algorithms", *SIAM Journal on Computing*, 18:594–607, 1988.
24. R. Seidel, "A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons", *Computational Geometry: Theory and Applications*, 1:51 – 64, 1991.

Appendix H

This appendix gives an algorithm for computing the trapezoidal diagram of a set S of n line segments in $O(n^2)$ expected time. The algorithm uses randomized divide-and-conquer: take random $R \subset S$ of size $r = n/\log n$, and compute $\mathcal{T}(R)$ using Goodrich's algorithm.[15] (Alternatively, use an algorithm analogous to the

suboptimal one given in Ref. 20.) Insert the remaining segments of S into $\mathcal{T}(R)$, and use subsampling of the segments meeting each $T \in \mathcal{T}(R)$ to yield subproblems all of size $O(\sqrt{\log n})$ with high probability. Now use an optimal serial algorithm for such subproblems, and Goodrich's algorithm for the remainder.

The insertion step, in more detail: note that the complete adjacency representation of $\mathcal{T}(R)$ divides the segments of R into $O(r^2)$ total pieces, with $O(r^2)$ on any segment of R (actually $O(r\alpha(r))$ trapezoids meet any given segment, but the larger bound will suffice). We use list ranking[2] to obtain, for each segment in R, a sorted array of the pieces induced by $\mathcal{T}(R)$, with pointers to the trapezoids meeting each piece. Now for each segment $a \in S$, use binary search on the array for every segment $b \in R$, to find the trapezoids containing the intersection point of a and b.

This does not give all the trapezoids meeting a; the remaining trapezoids are obtained as follows: consider the *convex diagram* of R, the subdivision induced using visibility edges from segment endpoints only, not intersection points. The trapezoids within each region of the convex diagram can be ordered top to bottom; such orderings can be obtained by list ranking, applied to adjacencies between trapezoids with common edges that are visibility segments from intersection points. The trapezoids in a convex cell that meet a segment $a \in S$ are an interval in that list, and the highest and lowest trapezoids contain either intersection points of a with R, or endpoints of a. Binary searches suffice to locate the endpoints of a in the list, if either are contained in trapezoids in the cell. Finally, the trapezoids meeting a, other than the highest and lowest, can be obtained after sufficient processors are allocated to do this. (Note that the number of trapezoids met is available.) We have, for every $a \in S$, the set of trapezoids of $\mathcal{T}(R)$ that it meets. By integer sorting we obtain a collection of subproblems as discussed in §3.1. This completes the first phase of processing.

Now for each $T \in \mathcal{T}(R)$, take a random subset of the segments that meet it, of size $K|n_T| \log |n_T|/\sqrt{\log n}$, where n_T is the number of segments meeting T, and K is an appropriate constant. Again apply Goodrich's algorithm to each such subset, and use the same insertion technique, to obtain a collection of subproblems. Now after appropriate processor allocations, apply an optimal serial algorithm to those subproblems with no more than $\sqrt{\log n}$ segments, and apply Goodrich's algorithm to the remainder.

Analysis. It is easy to verify that the algorithm consists of a constant number of stages each requiring $O(\log n)$ time in the worst case.

It is also easy to verify that $O(n^2)$ expected work is required, using Lemma 1. For example, computing the trapezoidal diagrams of the random subsets in the second phase of processing requires expected work proportional to

$$\sum_{T \in \mathcal{T}(R)} [O(|n_T|) \log |n_T|/\sqrt{\log n}]^2 \log |n_T|;$$

since $|n_T| = O(\log^2 n)$ for all T with probability $1 - 1/n$, the above is no more than

$$\sum_{T \in \mathcal{T}(R)} O(|n_T|^2)(\log\log n)^2 / \log n,$$

which is $O(n^2(\log\log n)^2 / \log n)$.