An Optimal Parallel L_1 Metric Voronoi Diagram Algorithm

Young C. Wee [*]
Samsung Institute for Advanced Technology
Seoul, Korea

Seth Chaiken
Department of Computer Science
State University of New York at Albany
Albany, New York 12222

May 14, 1990

Abstract

An $O(\log n)$ time $O(n)$ processor parallel algorithm on a CREW-PRAM for constructing the planar Delaunay diagram (thus the Voronoi diagram) for a set of points under the L_1 metric is described. The algorithm uses the geographic nearest neighbor approach and efficient parallel range query techniques. This algorithm is both practical and asymptotically optimal.

The Voronoi diagram (VD) and its generalizations and varieties are among the most intensively studied objects in computational geometry. Although optimal sequential algorithms to compute wide classes of these diagrams are known (see for example [SH75, Le80, CD85, Fo86, ES86, Ed87, Kl89] and references therein), the goal of a provably optimal parallel algorithm to compute even a single class of VD, as far as we know, remained elusive. For the Euclidean distance VD, on the CREW-PRAM, the best $O(n)$ processor algorithms take $O(\log^2 n)$ time [ACGDY88, CG88] and the best [1] $O(\log n)$ time algorithm requires $O(n^3)$ processors [PM88]. We present here an $O(\log n)$ time $O(n)$ processor algorithm for the CREW-PRAM that computes the planar L_1 (i. e. rectilinear or Manhattan distance) nearest site VD for a set S of n points (which are called *sites*). It is optimal in any model for which sorting has the above complexity [SH75].

[*]Supported in part by NSF Grants IRI 8703430 and CCR 8805782.

[1]We recently heard that Chew and/or Fortune improved the number of processors to $O(n \log n)$.

The first main idea is to begin by computing an $O(n)$ size subset of site pairs (which we call NN_8; a pair of sites is called an *edge*) determined from nearest neighbors under a fixed number of different "angle restrictions" [WCR89] of the relevant distance function which in this case is L_1. This so-called *geographic nearest neighbor approach* was introduced by Yao [Ya82]. The space around each site is partitioned into k angular regions (k depends on the problem, the metric and the dimension) and for each site the nearest neighbor, if any, within each region is found. The $O(n)$ size subset is input for further computation that solves the problem at hand. This approach is useful for several problems such as minimum spanning tree [Ya82, GS83], relative neighborhood graph [Su83, Ka88] and shortest path motion planning [Cl87].

The second main idea is to compute NN_8 and to identify each edge in the dual of VD (the Delaunay diagram DD) either in NN_8 or from an NN_8 edge by using efficient parallel range query techniques in $O(\log n)$ time. The range tree techniques for the dominance counting problem and dominance counting queries given in [GBT84] and their $O(\log n)$ time parallelization given in [CG88] (which uses $O(n)$ processors for preprocessing and a single processor for one query) use a scheme that is easily adapted to our purpose as in [WW88, WCR89]. This scheme can be adapted with no loss of efficiency whenever the query answer for a given range can be computed in constant time from the one or two precomputed subranges and their answers. See [BS79] for further details. The range query techniques work because the distance function $d(p,.)$ restricted to each of these angular regions is *affine*. Thus, the main result of this paper is an $O(\log n)$ time $O(n)$ processor algorithm for constructing DD from NN_8 under the L_1 metric. For simplicity, we restrict our description to the L_1 VD; however the design of the algorithm can be adapted to any metric whose unit circle is a convex quadrilateral. Further, we make the assumptions that no more than 2 of the sites lie on a single line of slope ± 1 and that no 4 sites are co-circular. Without these assumptions, the DD or the VD may be larger than $O(n)$; a version of our algorithm may still be built to solve a variation of the problem whose output size is $O(n)$.

For our purpose, the boundaries of each angular region will be vertical, horizontal or oblique with slope ± 1. Let $\alpha(p)$ denote the convex cone between the two rays in directions r_1, r_2 from p that form an angle $\alpha = [r_1, r_2]$ where points on each vertical or horizontal boundary and point p are included and other points on each oblique boundary are excluded. A site $q \in S$ is said to be the *α-nearest neighbor of p among S* if and only if q is one of the sites nearest to p

from $\alpha(p) \cap (S - \{p\})$, chosen arbitrarilly in the case of ties. We denote the α-nearest neighbor of p in S by $N_\alpha(p)$. The *geographic nearest neighbor problem* is for given α to find $N_\alpha(p)$ for all $p \in S$. Let $\alpha_k[i] = [2i\pi/k, 2(i+1)\pi/k]$ for $k \geq 1$ and $\Delta_k = \{\alpha_k[i] \mid 0 \leq i \leq k-1\}$. We define NN_k to be the graph (V, E) where

$$V = S \text{ and } E = \bigcup_{\alpha \in \Delta_k} \{(p,q) \mid p \in S, q = N_\alpha(p)\}$$

Thus NN_k is the union of the solutions to k geographic nearest neighbor problems. In the following, $k = 8$ and NN_8 is used as an undirected graph.

Let $l_t(p)$ denote the line through point p with slope t. Given an edge $e = (a, b)$, the two points $l_1(a) \cap l_{-1}(b)$ and $l_1(b) \cap l_{-1}(a)$ will be called the *corner points* of e. For our algorithm, we need to compute NN_8, to find for certain points p the site nearest to p on closed vertical, horizontal or oblique rays from p, and to answer $N_\alpha(r)$ queries for each corner point r of $e = (a, b) \in NN_8$ for the right angle α whose (oblique) sides contain a and b. The latter query angle is denoted by $\alpha(r, e)$. These $\alpha(r, e)$-nearest neighbor queries are implemented by two $N_{\alpha'}(r)$ queries where the width of each α' is $\pi/4$. For all this computation, we use 8 range trees. Each search answer has 3 fields: site nearest to p in the interior of $\alpha(p)$ and the sites nearest to p in each of the two closed rays that bound $\alpha(p)$.

Let us call any square whose sides have slope ± 1 a *diamond.* Note that the shape of the L_1 unit circle is a diamond. A diamond is called *empty* when its interior contains no sites. An empty diamond *touches* a site means that the site is on the boundary of the diamond. Two sites a and b are *adjacent on* a diamond if and only if a and b lie respectively on a pair of adjacent sides. Here, one site might coincide with the common corner of the adjacent sides or each site is contained in exactly one of the two adjacent sides. In the latter case, (a, b) does not have slope ± 1 and we say that this edge is *generally positioned.* Two VD regions have a common boundary point if and only if there is an empty diamond that touches the two sites corresponding to the regions.

The following theorem holds under the assumption that $n > 2$ and no 3 sites lie on a common line of slope $+1$ or -1. The correctness of the algorithm DD under the already given assumptions follows from this theorem and the preceeding discussion.

We remark that if the sites have integer coordinates then half-integer precision suffices to eliminate any round-off error during the computation.

Theorem 0.1 $(a, b) \in \mathrm{DD}$ *if and only if*

(1) $(a, b) \in \mathrm{NN}_8$ *and* a, b *are adjacent on an empty diamond*

or (2) there is a site c*, there is an empty diamond* D *that touches* a*,* b *and* c*, and for at least one edge* $e = \{x, y\} \subset \{a, b, c\}$*,* $e \in \mathrm{NN}_8$ *and* x, y *are adjacent on* D.

Proof sketch. "If" is immediate. "Only if" follows from a case analysis of how a and b lie on the boundary of an empty diamond D'.

Case 1. If a and b are adjacent on D' and (a, b) is in general position, then either $a \in \alpha(b)$ or $b \in \alpha'(a)$ for appropriate $\alpha \neq \alpha'$. Say $a \in \alpha(b)$. Either $N_\alpha(b) = a$ so $(b, a) \in \mathrm{NN}_8$ or $N_\alpha(b) = c$ is another site on the same side of $D = D'$ as a because c won a tie settlement against a.

Case 2. If a and b are not adjacent on D', then either there is a third site that is adjacent to at least one of a and b on D' or there exists a different empty diamond D obtained by "sliding" D' while maintaining contact with a and b until a corner of D touches a or b, or a (closed) side of D adjacent to a and b touches a third site. Thus, for some empty diamond, either the other cases apply or there is a third site c' such that (a, c') or (b, c') has ends adjacent on D and is in general position. In the latter case, the argument of case 1. is applied.

Case 3. If a and b are adjacent on D' and (a, b) has slope $+1$ or -1, then at least one of a or b is a corner of D'. There is no other site on (a, b) and $n > 2$, so there exists an empty diamond that touches a, b and some third site. The third site paired with a or with b is an edge in general position. A reduction to case 1. or to case 2. now applies. □

```
DD
$DD =$ empty.
for each edge $e = (a,b) \in$ NN$_8$ (Note that all NN$_8$ edges are generally positioned.)
    Determine the two corner points $r_1$ and $r_2$ of $e$.
    for $r = r_1, r_2$
        $t = \alpha(r,e)$-nearest neighbor of $r$ among $S$.
        if $t$ is defined then
            if $\max(d(r,a), d(r,b)) \leq d(r,t)$ then
                $DD = DD \cup \{e, (a,t), (b,t)\}$.
        else (i.e., $\alpha(r,e)$ contains no sites)
            if a site $x \in S - \{a,b\}$ lies in a boundary ray of $\alpha(r,e)$ then
                $DD = DD \cup \{e, (a,x), (b,x)\}$.
            else $DD = DD \cup \{e\}$.
Sort [Co86] $DD$ to remove duplications.
endDD
```

References

[BS79] J.L. Bentley, Decomposable searching problems, *Information Processing Letters*, 1979, 244-251.

[ACGDY88] A. Aggarwal, B. Chazelle, L. Guibas, C. Ó'Dúnlaing and C. Yap, Parallel Computational Geometry, *Algorithmica*, 1988, 293-327.

[CD85] L. P. Chew and R. L. Drysdale, Voronoi diagrams Based on Convex Distance Functions, *Proc. of the 1st Annual Symp. on Computational Geometry*, 1985, 235-244.

[CG88] R. Cole, M. T. Goodrich, Optimal Parallel Algorithms for Polygon and Point-Set Problems, *Proc. of the Fourth Ann. Symp. on Computational Geometry*, 1988, 201-210.

[Cl87] K. Clarkson, Approximation Problems for Shortest Path Motion Planning, *Proc. 19th Ann. ACM Symp. Theory of Computing*, 1987, 56-65.

[Co86] R. Cole, Parallel Merge Sort, *27th Foundations of Computer Science*, 1986, 511-516.

[Ed87] H. Edelsbrunner, Algorithms in Combinatorial Geometry, Springer-Verlag Berlin Heidelberg, 1987.

[ES86] H. Edelsbrunner and R. Seidel, Voronoi diagrams and Arrangements, *Discrete Computational Geometry 1*, 1986, 25-44.

[Fo86] S. J. Fortune, A Sweepline Algorithm for Voronoi Diagrams, *Proc. of 2nd Annual Symp. on Computational Geometry*, 1986, 313-322.

[GBT84] H.N. Gabow, J.L. Bentley and R.E. Tarjan, Scaling and Related Techniques for Geometry Problems, *Proc. 16th Ann. ACM Symp. Theory of Computing*, 1984, 135-143.

[GS83] L. J. Guibas and J. Stolfi, On Computing all North-east Nearest Neighbors in the L_1 Metric, *Info. Process. Lett.* **17**, 1983, 219-223.

[Ka88] J. Katajainen, The Region Approach for Computing Relative Neighborhood Graphs in the L_p Metric, *Computing* **40**, 1988, 147-161.

[Kl89] R. Klein, *Concrete and Abstract Voronoi Diagrams*, Lecture Notes in Computer Science **400**, Springer-Verlag, 1989.

[Le80] D. T. Lee, Two-Dimensional Voronoi Diagrams in the L_p-Metric, *J. ACM* **27**(4), 1980, 604-618.

[PM88] W. Preilowski and W. Mumbeck, A Time-Optimal Parallel Algorithm for the Computing of Voronoi-Diagrams, *Lect. Notes in Computer Science, Springer Verlag*, 1988, 424-433.

[SH75] M. I. Shamos and D. Hoey, Closest point problems, *Proc. 16th Ann. Symp. on the Foundations of Computer Science, IEEE*, 1975, 151-162.

[Su83] K. J. Supowit, The Relative Neighborhood Graph, with an Application to Minimum Spanning Trees, *J. ACM* **30**(3), 1983, 428-448.

[WCR89] Y. C. Wee, S. Chaiken and S. S. Ravi, Some Applications of Geographic Nearest Neighbor Approach, *Proceedings of the 27th Annual Allerton Conference*, 1989.

[WW88] D. E. Willard and Y. C. Wee, Quasi-valid Range Querying and its Implications for Nearest Neighbor Problems, *Proceedings of the Fourth Annual Symp. on Computational Geometry*, 1988, 34-43.

[Ya82] A. C. Yao, On Constructing Minimum Spanning Trees in k-dimensional Spaces and Related Problems, *SIAM J. Comput.* **11**(4), 1982, 721-736.