A Parallel Algorithm for Partitioning a Point Set to Minimize the Maximum of Diameters

Muhammad H. Alsuwaiyel

Department of Information and Computer Science

King Fahd University of Petroleum & Minerals

Dhahran 31261, Saudi Arabia

e-mail:suwaiyel@ccse.kfupm.edu.sa

Abstract. Given a set S of n points in the plane, we consider the problem of partitioning S into two subsets such that the maximum of their diameters is minimized. We present a parallel algorithm to solve this problem that runs in time $O(\log n)$ using the CREW PRAM with $O(n^2)$ processors.

1 Introduction

Let S be a set of n points in the plane. The problems of partitioning S into k partitions, or covering S with k convex objects, e.g. disks, are intractable. When the number of partitions or convex objects is restricted to two, a number of algorithms have been developed in order to meet a given criterion. For the case when the problem is partitioning S into two partitions S_1 and S_2 and the criterion is to minimize the maximum of the two diameters, Avis [3] gave a sequential algorithm to solve this problem in $O(n^2 \log^2 n)$ time and $O(n^2)$ space. Later, Asano et al [2] improved the bound to $O(n \log n)$ time and $O(n)$ space. Another $O(n \log n)$ time algorithm that is simpler was given by Monma and Suri [4]. Other parallel algorithm for this problem were not found in the literature.

In this paper, we present a special purpose parallel algorithm to solve this problem that runs in time $O(\log n)$ using the CREW PRAM model of computation with $O(n^2)$ processors.

2 The structure of an optimal bipartition

For simplicity, we will assume that each point has only one farthest neighbor. Modifying the algorithm to the more general case where a point may have more than one farthest neighbor is easy. If p and q are two points in S, we will denote by $d(p,q)$ the Euclidean distance between p and q. For $A \subseteq S$, we define $Diam(A)$ to be the diameter of A, that is, the largest distance realized by two points in A. A bipartition $\{A,B\}$ will be called *optimal* if the maximum of the two diameters of A and B is minimum among all bipartitions of S. For each point p in S, let $f(p)$ denote the farthest neighbor of p. In general, we define $f^0(p) = p$, and $f^j(p) = f(f^{j-1}(p))$.

Definition 1 Let $C \subset S$ and $C' \subseteq S - C$ such that for all $p \in C$ and all $q \in C'$, $f(p) \in C'$ and $f(q) \in C$. Then C and C' will be called *clusters*, and the pair (C, C') a *cluster pair.*

The following lemma provides the basis for the algorithm to be developed.

Lemma 1 Let $\{A,B\}$ be a bipartition of the set of points in S, and $p \in A$. If $f(p) \in A$, then $\max\{Diam(A - \{p\}), Diam(B \cup \{p\})\} \leq \max\{Diam(A), Diam(B)\}$.

proof. Let $D = \max\{Diam(A), Diam(B)\}$, and suppose that both p and $f(p)$ are in A. If for all $x \in S$, $d(p,x) < D$, then moving p to B will not increase the maximum of diameters. On the other hand, if for some $x \in S$, $d(p,x) = D$, then since $D = d(p,x) \leq d(p, f(p))$, moving p to B will not increase the maximum of diameters. It follows that in both cases

$$\max\{Diam(\{A - \{p\}), Diam(B \cup \{p\})\} \leq D.$$

Corollary 1 There is an optimal bipartition $\{A,B\}$ such that for all cluster pairs (C,C'), either $C \subseteq A$ and $C' \subseteq B$ or vise-versa.

The function f defined above has the property that if $q = f(p)$, $w = f(q)$ and $p \neq w$, then $d(p,q) < d(q,w)$. In general, for any $j \geq 1$ and any point $p \in S$, if $f^{j-1}(p) \neq f^{j+1}(p)$, then

$$d(f^{j-1}(p), f^j(p)) < d(f^j(p), f^{j+1}(p)). \tag{1}$$

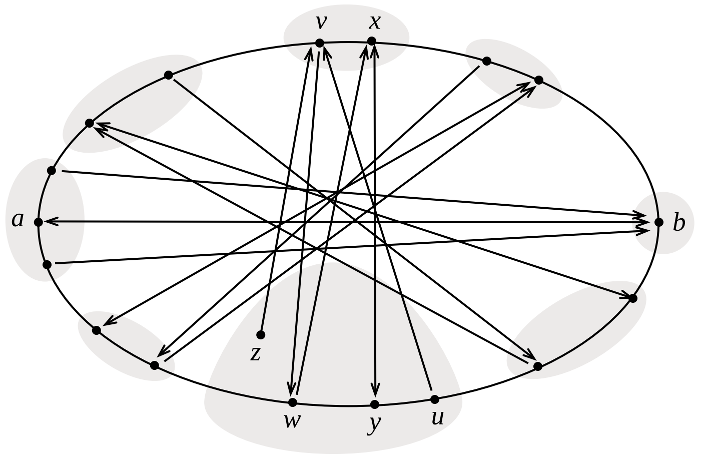

Figure 1: Partitioning the points into clusters.

For example, in Figure 1,

$$d(u,v) < d(v,w) < d(w,x) < d(x,y) = d(y,x).$$

In this figure, $f(p) = q$ is represented by an arrow directed from p to q. $f(p) = q$ and $f(q) = p$ is represented by a line segment with arrows at both ends. For clarity, only one point not on the convex hull is shown.

Thus, f induces a directed graph $G = (S, E)$ with the property that the only cycles in G are of length 2, and they are pairs of points (p, q) such that $f(p) = q$ and $f(q) = p$. Since, by assumption, f is a function (each point has exactly one farthest neighbor), it follows that if G contains more than one cycle, then it is disconnected. By Inequality 1 and the definition of a cluster pair, each cluster pair (C, C') contains exactly two points $p \in C_1$ and $q \in C_2$ such that $f(p) = q$ and $f(q) = p$. It follows that each component of the directed graph corresponds to exactly one cluster pair. Figure 2 shows the component of the directed graph G corresponding to the cluster pair $(\{v, x\}, \{z, w, y, u\})$ shown in Figure 1. Note that all points in the interior of the convex hull are leaves in their respective components.

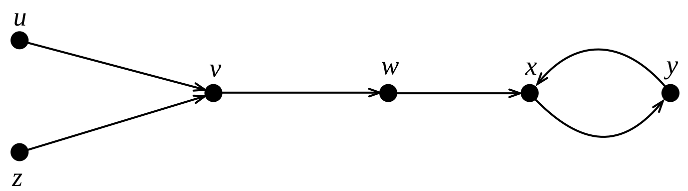

Figure 2: One component of the graph.

For any component H of the graph G corresponding to a cluster pair (C, C'), let r_1 and r_2 denote the two points constituting the cycle in H. Define the function $g(p) = f^{2n}(p)$. Since there is only one cycle in H whose vertices, namely r_1 and r_2, are reachable from all other points in H, $g(p)$ is either r_1 or r_2 . As $g(p)$ is within even edge distance from p, $g(p)$ is in the same cluster as p. Consequently, the function g induces two shallow directed trees rooted at r_1 and r_2. Each tree consists of a root r and a number of children that point to it, namely those points within even distance from r. Thus, each directed tree represents exactly one cluster, and each cluster is represented by a directed tree. As an example, the component shown in Figure 2 is transformed by the function g into the two directed trees shown in Figure 3. It follows that H is transformed by g into two directed trees corresponding to C and C'. It should be noted that both the directed graph defined by f and the directed trees defined by g are represented by the algorithm using two arrays of size n.

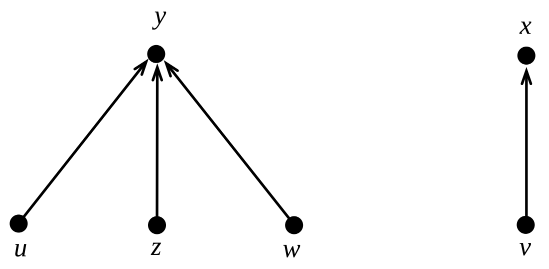

Figure 3: Two trees corresponding to two clusters in a cluster pair.

Let P be the polygon formed by the points on $CH(S)$, the convex hull of S. It is well-known that for all $p \in S$, $f(p)$ is a vertex of P. Let k be the number of cluster pairs, and C_0 any cluster. For $1 \leq j \leq 2k - 1$, let C_j be such that $C_j \cap CH(S)$ follows $C_{j-1} \cap CH(S)$ in a clockwise traversal of the boundary of P. It is fairly easy to show that for $0 \leq j \leq k - 1$, (C_j, C_{j+k}) is a cluster pair. By Corollary 1, we may assume that C_j belongs to one partition, and C_{j+k} belongs to the other. Clearly, each partition must consist of exactly k *adjacent* clusters, that is, their intersection with $CH(S)$ consists of one polygonal chain. As a result, there are only k possibilities for bipartitioning S with the property that the two clusters in a cluster pair are assigned to different partitions. By exhaustively computing the k maximums of diameters, the partitioning that results in the minimum of these maximums is selected.

3 The partitioning algorithm

The partitioning algorithm can now be described as follows.

1. Compute $CH(S)$, the convex hull of S, and let $h = |CH(S)|$. This takes $O(\log n)$ time using $O(n)$ processors [1].

2. For each point $p \in S$, allocate $\lceil h/\log h \rceil$ processors. Since $f(p) \in CH(S)$, these processors will compute $f(p)$ in $O(\log h)$ time.

3. In this step we construct the function g using pointer jumping in which each processor associated with point p executes the assignment $f(p) = f(f(p))$ repeatedly until $f(f(f(p))) = f(p)$. The number of applications of the function f is $O(\log n)$, as the size of each component is $O(n)$ and the unique cycle it contains is of length 2. Hence, at the end of this step, we will have computed the $2k$ clusters as directed trees. We number them clockwise around the boundary of $CH(S)$ as $C_0, C_1, \ldots, C_{2k-1}$, where C_0 is chosen arbitrarily. Finally, we assign label j to every point in cluster C_j.

4. Label kn processors as $P_{j,i}$, $0 \le j \le k-1$, $1 \le i \le n$.

   For $j = 0, 1, \ldots, k-1$ do in parallel: Processors $P_{j,i}$, $1 \le i \le n$, partition the point set into S_j and S'_j, where S_j is the set of points whose label is in $\{j, j+1, \ldots, j+k-1\}$, and S'_j is the set of points whose label is in $\{j+k, j+k+1, \ldots, j-1\}$. Compute $\mathit{Diam}(S_j)$ and $\mathit{Diam}(S'_j)$. Computing the two diameters using n processors takes $O(\log n)$ time [1]. Thus, the total time required in this step is $O(\log n)$ using $kn = O(hn)$ processors, as h is an upper bound on the number of clusters.

5. Among the pairs $(D_0, D'_0), (D_1, D'_1), \ldots, (D_{k-1}, D'_{k-1})$, Return that pair (D_j, D'_j) in which $\max\{D_j, D'_j\}$ is minimum, and the corresponding partitions.

Figure 4 shows an example in which the set of points is partitioned into 8 clusters. In this example the following 8 diameters are computed in parallel:

$D_0 = \mathit{Diam}(C_0 \cup C_1 \cup C_2 \cup C_3) \quad D'_0 = \mathit{Diam}(C_4 \cup C_5 \cup C_6 \cup C_7)$

$D_1 = \mathit{Diam}(C_1 \cup C_2 \cup C_3 \cup C_4) \quad D'_1 = \mathit{Diam}(C_5 \cup C_6 \cup C_7 \cup C_0)$

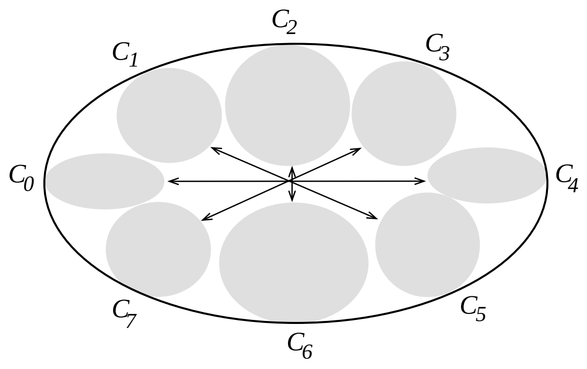

Figure 4: The point set partitioned into $2k$ clusters.

$$D_2 = Diam(C_2 \cup C_3 \cup C_4 \cup C_5) \quad D'_2 = Diam(C_6 \cup C_7 \cup C_0 \cup C_1)$$
$$D_3 = Diam(C_3 \cup C_4 \cup C_5 \cup C_6) \quad D'_3 = Diam(C_7 \cup C_0 \cup C_1 \cup C_2).$$

References

[1] S. G. Akl and K. A. Lyons, *Parallel Computational Geometry*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[2] T. Asano, B. K. Bhattacharya, J. M. Keil and F. F. Yao, "Clustering algorithms based on minimum and maximum spanning trees," *Proc. of the Fourth Annual Symposium on Computational Geometry*, 1988, 252-257.

[3] D. Avis, Diameter Partitioning, *Discrete and Computational Geometry*, 1, 1986, 265-276.

[4] C. Monma and S. Suri , "Partitioning points and graphs to minimize the maximum or the sum of diameters," *Proceedings of the Sixth International Conference on the Theory and Applications of Graphs* (Wiley, New York, 1989), 899-912.