ZBORNIK RADOVA
Prirodno-matematičkog fakulteta
Univerziteta u Novom Sadu
Serija za matematiku, 17,1(1987)

REVIEW OF RESEARCH
Faculty of Science
University of Novi Sad
Mathematics Series, 17,1(1987)

FINDING THE DIAMETER OF A POINT SET ON MESH-CONNECTED COMPUTERS

Ivan Stojemnović and Ljubomir Jerinić

University of Novi Sad, Faculty of Science,
Institute of Mathematics, Dr. I. Djuričića 4,
21000 Novi Sad, Yugoslavia

ABSTRACT

This paper presents an efficient algorithm for finding the diameter of a point set on mesh-connected parallel computers. The running time of the algorithm is O(n**(1/2)* log(n)).

1. INTRODUCTION

A mesh-connected parallel computer of size n is a set of n synchronized processing elements (PEs) arranged in an n**(1/2) x n**(1/2) grid. Each PE is connected via bi-directional unit-time communication links to its four neighbours, if they exist (see Fig.1).

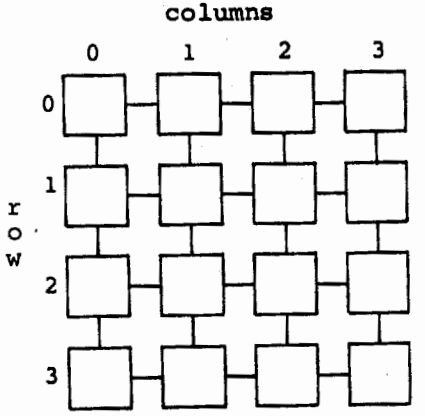
Each processor has a fixed number of registers and can perform standard arithmetic and comparisons in a constant time. It can also send the contents of a register to a neigh-

---

AMS Mathematics Subject Classification (1980): Primary 68U05
Secondary 68Q10.
Key words: Parallel computation, mesh-connected computer, computational geometry, diameter of a point set.

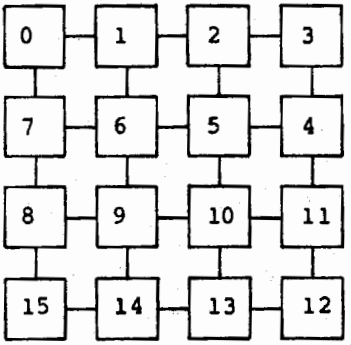
bour and receive a value from a neighbour in a designated register in O(1) time units. Each PE in the leftmost column has an I/O port. Thus, we can 'load' S in O(n**(1/2)) time units such that each processor contains exactly one arbitrary point of S.

Each PE contains a unique identification register (ID), the contents of which correspond to that PE's snage-like index (see Fig.2).



MCC of size 16

Fig.1



snake-like order of an MCC of size 16

Fig.2

For the algorithm presented in this paper, we shall assume that there initially exists n or fewer planar points, distributed one point per PE on a MCC (mesh-connected computer) of size n. Each planar point p is represented by its Cartesian coordinates p.x and p.y. Any figure is represented by the Cartesian coordinates of O(n) planar points, distributed O(1) points per processor on a MCC. For some of the problems that allow more than one figure as input, each point will also have an associated label to indicate which figure it is a member of.

To simplify the exposition of our algorithm we shall assume that n=4**k for some integer k and all points have distinct x- and y-coordinates.

We shall use the standard MCC data movement operations: rotating data within a row (column) sorting, compression of data, Random Access Read (RAR) and Random Access Write (RAW). We shortly describe them (see [2,3,4] for details).

1) Rotating data within a row (column)

For each row (column), every PE can transmit a fixed number of pieces of information to all other PEs in its row (column) in O(n**(1/2)) time.

2) Sorting

Thompson and Kung [7] have shown that n elements, distributed one element per PE on an MCC of size n, can be sorted in O(n**1/2)) time. In particular, the elements can be sorted into a snake-like ordering is illustrated in Fig.2.

3) Compression

Suppose that in an MCC of size n, m of the PEs each contain a fixed amount of information that needs to be exchanged in an efficient manner. In O(n**1/2)) time, the O(m) relevant pieces of information from these m PEs can be placed into a subsquare of size m where communication time will now be O(m**(1/2)).

Two other common data movement operations for the MCC are the random access read (RAR) and random access write (RAW). These operations involve two sets of PEs, the SOURCES and the DESTINATIONS. Source PEs send a fixed number of records, each consisting of a key and one or more data parts (a record may also be null). Destination PEs receive a fixed number of records sent by the source PEs. We allow the possibility that a PE is both a source and a destination.

4) Random access read (RAR)

In an RAR, we require that no two records sent by source PEs have the same key. Each destination PE specifies the keys of the records it wishes to receive, or it specifies a null key, in which case it receives nothing. Several destination PEs can request the same key. A destination PE may specify a key for which there is no source record, in which case it receives a null message. The RAR is accomplished through a fixed number of sort steps and rotations in O(n**(1/2)) time.

5) Random access write (RAW)

In an RAW the destination PEs do not specify the keys they want to receive. They merely indicate their willingness to receive. At the end of the RAW, the number of records received by a destination PE is between zero and a fixed number requested. Each key is received by exactly one destination PE. If two or more source PEs send records with the same key, then a destination PE will receive the minimum such records. (In other circumstances, one could replace the minimum with any other commutative, associative, binary operation.) The RAW is accomplished through a fixed number of sort steps in O(n**(1/2)) time.

Sorting [7] and the component labelling algorithm [4] are basic algorithms which provide an efficient solution to some geometric problems given in [2,3].

O(n**(1/2)) is a lower bound of each algorithm on MCC since it cannot be faster than the time it takes to combine information starting at opposite corners of the mesh.

The diameter of a set S of n points in the plane is the distance between two points form S that are farthest apart. We shall give an O(n**(1/2)*log(n)) time MCC algorithm for finding the diameter of S. A sequential one described in [5] works in optimal O(n*log(n)) time.

2. COMPRESSION OF DATA AND BINARY SEARCH

Let a function f be defined on a set S containing O(n) elements (points, line segments,...). The function f is unimodular on S if there is an order of elements from S such that there are two elements A and B and f is nonincreasing from A to B and nondecreasing from B to A. For a given function f the elements with the minimum (or maximum) value of f can be determined by a binary search in log(n) steps by comparing the value of f for the middle element and two neighbouring elements, discarding half of elements from consideration and compressing of the remaining data. The running time of the algorithm is expressed by T(n)=T(n/2)+c*n**(1/2) whose solution is O(n**(1/2)).

Let us consider the problem of determining the elements with the minimum (or maximum) values for O(n) different unimodular functions in parallel. Binary search in this case differs from the one described above in that multiple binary searches are occurring simultaneously. In each step for each function, half of the elements can be discarded but there is no straightforward way how to avoid bottleneck during compressing the remaining data. If the algorithm continued its work without compressing the data then the communication time would not be cut in half during each iteration of the algorithm and because of log(n) steps the total running time would be O(n**(1/2)*log(n)). With a correct compression technique, the algorithm will be completed in O(n**(1/2)) time.

So, in each particular case we have either to find a way of compressing the data to avoid bottleneck or use an additional log(n) factor in the running time of the given algorithm.

In order to determine the diameter for a point set S, each PE containing an edge e of the H(S)(the convex hull of S, i.e. the smallest convex figure containing S), one needs to know one or two additional extreme points of the H(S). Namely, N, the last extreme point(s) of S encountered as a line paral-

lel to e, starting colinear to e, passes through the H(S). These points are extreme values of some unimodular functions defined on extreme points H(S) and can be found by multiple binary searches simultaneously for each edge e.

3. DIAMETER ON A MESH

The diameter of a set is equal to the diameter of its convex hull [1] and the diameter of a convex figure is the greatest distance between parallel lines of support [8]. A pair of points that does admit parallel supporting lines will be called antipodal. All antipodal points of an extreme point P(i) form an interval [P(i)',P(i)"] of consecutive extreme points on convex hull CH(S) [5](we call it the P(i)-interval).

First we shall find the convex hull of S by algorithm described in [2] in O(n**(1/2)) time. The number of extreme points of the convex hull is m<=n. Let P(1),...,P(m) be the points of the convex hull of S ordered in counterclockwise order. We assign a PE to each convex hull edge P(i-1)P(i) (1<=i<=m, P(0)=P(m)). In the parallel we find for each edge the extreme point which is farthest from it. There are one or two such points. If there is one point P(j) then the point is the right endpoint of the P(i-1)-interval and the left endpoint of the P(i)-interval (i.e. j=(i-1)"=i'). If there are two such points P(j-1) and P(j)(in this case edges P(i-1)P(i) and P(j-1)P(j) are parallel) then the point P(j-1) is the right endpoint of P(i-1)-interval and the point P(j) is the left endpoint of the P(i)-interval (any time two polygon edges are parallel, only the diagonals of the trapezoid they determine need be considered).

This can be done in log(m) steps using the binary search technique (the distance from an edge is an unimodular function) and O(m**(1/2)*log(m)) time, because the compression of data remains as a problem in the process.

At the end of the process each extreme point P(i) has its P(i)-interval calculated by knowing its endpoints.

There are exactly m antipodal pairs (cf. [5]) that remain to be examined. Each antipodal pair P(i)P(j) is contained in both the P(i)-interval and P(j)-interval. Thus intervals contain 2m members in all. However, one interval can contain O(m) members and thus we cannot find the greatest distance among antipodal pairs by storing one interval (one extreme point) per a PE because bottleneck can appear. We are going to devise a procedure for storing all antipodal pairs without bottleneck. We shall use a mesh of size O(m)(i.e. O(n)).

We shall send each antipodal pair P(i)P(j) into PE with ordinal number i+j in snage-like ordering of PEs. We shall show that each PE can receive at most two antipodal pairs.

Consider the natural order of antipodal pairs: P(1)--interval, P(2)-interval,..., P(m)-interval (we keep in each interval the order from the left endpoint to the right endpoint as they are constructed). In fact, each antipodal pair appears twice in the order. All the time, the sum of the indexes i+j of the antipodal pairs increase by 1 or 2 (the latter appear twice per each pair of parallel edges; so at most m times). The only exception is when P(1) appears as a member of an interval of antipodal pairs. But, it happens exactly once. Therefore a sum k can appear at most twice.

If [P(u),P(v)] is P(i)-interval then we first send P(i) to PEs i+u and i+v (in snake-like ordering) in parallel for each i (1<=i<=m). This can be done in O(n**(1/2)) time by performing an RAR (the key of a PE is the same as the ordinal number in the snake-like ordering and each PE with extreme point P(i) broadcast two records with keys i+u and i+v both with data i,u,v). Now all PEs between PEs i+u and i+v can receive the index i in O(n**(1/2)) in parallel for each i by the following: if i+u and i+v are in the same row then it is obvious; otherwise first they inform PEs in the rows containing i+u and i+v having the ordinal number between i+u and i+v (and the PEs in the leftmost column of the rows) about i. If there are some rows between two containing i+u and i+v then first PEs in the leftmost column of these rows are informed

about i and afterward broadcast the information to the remaining PEs in these rows.

Now each PE containing one or more antipodal pairs computes the distances in O(1) time. By an RAW one PE can know the greatest of these distances in additional O(n**(1/2)) time.

4. CONCLUSION

Describing an optimal time MCC-algorithm for finding the diameter of a point set remains an open problem. Our method could be improved only by solving the compression problem during a simultaneous binary search. However it seems that implementing such a technique in order to avoid bottleneck (recall a PE has a fixed number of registers) is a rather difficult task. For instance, a point P from H(S) can be encountered as point N for O(n) different edges of H(S). Such points P and their neighbour points must be stored in O(n) compressed squares (or a square in each step of the iteration has a size O(n)).

REFERENCES

[1] Hocking,J.G., Young,G.S., *Topology, Addison Wesley, Reading, MA, 1961.*

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

[3] Miller,R., Stout,Q.F., *Mesh Computer Algorithms for Computational Geometry, Technical Report 86-18, Dept.of Comp.Sci., State Univ. of New York at Buffalo, July 1986, pp.50.*

[4] Nassimi,D., Sahni,S., *Finding Connected Components and Connected Ones on a Mesh-Connected Parallel Computer, SIAM J., Computing, 9, 1980, 744-757.*

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

[6] Stout,Q.F., *Broadcasting in Mesh-Connected Computers, Proc. 1982, Conf. on Info. Sciences and Systems, Princeton Univ., pp. 85-90.*

[7] Thompson,C.D., Kung,H.T., *Sorting on a Mesh-Connected Parallel Computer, Comm. ACM 20(4)(1977) 263-271.*

[8] Yaglom,I.M., Boltyanskii,V.G., *Convex Figures, Holt, Rinehart and Winston, New York, 1961.*

REZIME

NALAŽENJE NAJVEĆEG RASTOJANJA IZMEDJU TAČAKA NA MREŽNO POVEZANOM PARALELNOM RAČUNARU

U radu je predstavljen efikasan algoritam za nalaženje najvećeg rastojanja izmedju tačaka za mrežno povezane paralelne računare. Vreme izvršavanja algoritma je O(n**(1/2)* log(n)).

Received by the editors December 15, 1986.