Univ. u Novom Sadu
Zb. Rad. Prirod.–Mat. Fak.
Ser. Mat. 18,2, 127–136 (1988)

REVIEW OF RESEARCH
FACULTY OF SCIENCE
MATHEMATICS SERIES

COMPUTATIONAL GEOMETRY ON A MESH-CONNECTED COMPUTER

Ivan Stojmenović

Institute of Mathematics, Faculty of Science,
University of Novi Sad, dr Ilije Djuričića 4,
21000 Novi Sad, Yugoslavia

ABSTRACT

This paper describes optimal algorithms for solving some fundamental geometric problems on a mesh connected parallel computer: minimum enclosing circle, detecting circular separability of two point sets, recognition of digital disks, finding the smallest and the largest separating circle of two sets of points, detecting linear separability of two sets, inclusion of points in a convex polygon, maximum gap problem and the diameter of a binary image. The problem of finding the diameter of a black/white picture is solved in general, for k-dimensional binary image stored one pixel per processor in a k-dimensional mesh connected computer, and for any monotone metric.

AMS Mathematical Subject Classification (1980):
Primary 68U05, Secondary 68Q10.
Key words and phrases: Parallel computation, computational geometry, mesh connected computers, design and analysis of algorithms, linear programming, digital geometry, circles, convex polygons, diameter, digital disks, separability, inclusion.

1. INTRODUCTION

This paper is concerned with efficient parallel algorithms for determining several fundamental geometric properties of figures on a model of parallel computation which has extensively been used in practice: mesh-connected computer (MCC). Any figure is represented by the cartesian coordinates of $O(n)$ planar points, distributed O(1) points per processor on a MCC. A MCC of size n is a set of n synchronized processing elements (PEs) arranged in a $\sqrt{n}$ x$\sqrt{n}$ grid (see Fig. 1). Each PE is connected via bidirectional unit-time communication links to its four neighbors, if they exist. Each processor has a fixed number of registers and can perform standard arithmetic and comparisons in constant time. It can also send the contents of a register to a neighbor and receive a value from a neighbor in a designated register in O(1) time. Each PE in the leftmost column has an input/output port. Thus, one can 'load' a set of $O(n)$ data in $O(\sqrt{n})$ time such that each processor obtains constant amount of data. Each PE contains a unique identification register (ID), the contents of which correspond to that PE's snake-like index (Fig. 1 presents the snake-like indices for a MCC with 16 processors). We use standard MCC data movement operations: rotating data within a row (column), sorting, compression of data, Random Access Read (RAR) and Random Access Write (RAW) (cf. [8,9,12]). For all of the algorithms discussed in this paper, we will assume that geometric objects are given by $O(n)$ planar points, distributed O(1) points per processor on a MCC of size n. Each planar point p is represented by its cartesian coordinates. The communication time between two processors on a MCC is $O(\sqrt{n})$. Hence, no parallel MCC algorithm for solving considered problems can be constructed in faster than $O(\sqrt{n})$ time. In this light, all the algorithms presented here are optimal.

2. LINEAR PROGRAMMING AND RELATED GEOMETRIC PROBLEMS

The two-variable linear programming problem is given as:

minimize $ax+by$
subject to $a[i]x+b[i]y+c[i]\leq 0$, $i=1,2,\ldots,n$.

Meggido [7] and Dyer [5] have independently discovered a sequential linear time solution to the problem. Moreover, they

modified their technique and solved the corresponding three-dimensional problem.

For both two-dimensional and three-dimensional linear programming problems, using one processor per a linear constraint, their technique can be transformed to the parallel one in a straightforward way. We omit details and state only that $O(\sqrt{n})$ optimal MCC solutions for the two-variable and three-variable linear programming problems can be derived from their technique.

The solution of several geometric problems can be derived ones the solution to the two or three-variable linear programming problem is given. Because of $O(\sqrt{n})$ MCC transformations from the corresponding problems, a number of geometric problems can be solved in optimal time on a MCC. We list the following ones:

- Minimum enclosing circle of a set of n planar point can be transformed to the three-variable linear programming problem, as shown in [7],

- Detecting circular separability of two point sets A and B (detecting whether there is any circle so that each point from A is in its interior or on its boundary while each point from B is not in its interior) is transformed to linear separability in R^3 [10] which is further transformed to the two-variable linear programming [7],

- Recognition of digital discs (detecting whether there is a circle so that each point from A is in its interior or on its boundary while each point from B lies outside of the circle) can be transformed to two-variable linear programming [10],

- Finding the smallest separating circle of two point sets can be transformed to convex quadratic minimization problem in R^3 with linear constraints (as shown in [10]), which can be again solved by Megiddo's technique [7].

- The problem of finding the largest separating circles can be solved in $O(n \log n)$ sequential time [10]. The corresponding MCC solution can be obtained by using techniques developed in [3] for solving three-dimensional convex hull and related problems and runs in optimal time (but with O(log n) space per processor).

3. LINEAR SEPARABILITY OF SETS

In this section we consider the following problem:

Given two convex polygons $S1$ and $S2$, each composed of $O(n)$ points distributed one point per processor on a MCC of size n, determine whether they intersect (i.e. whether they are linearly separable).

Assume that there are r extreme points of $S1$ and t extreme points of $S2$. For $S1$, choose the extreme points numbered 1, $r/4$, $r/2$ and $3r/4$. For $S2$, choose the extreme points numbered 1, $t/4$, $t/2$ and $3t/4$. Points 1, $r/4$, $r/2$, $3r/4$ divide the plane into eight regions as shown in Fig. 2 (some of regions may intersect). Assuming the special edges $(1, r/4)$, $(r/4, r/2)$, $(r/2, 3r/4)$ and $(3r/4, 1)$ do not intersect $S2$ and the points 1, $r/4$, $r/2$ and $3r/4$ do not belong to $S2$ we determine the region the point $t/2$ belongs to. In each case there is a special edge so that points lying between corresponding special extreme points can be discarded (i.e. these points cannot belong to the other figure and the edges of the convex polygons determined by the points do not intersect the other figure). Similarly, a fourth of points from $S2$ can be eliminated. By repeating this process and compressing the remaining data, in $O(\log n)$ iterations a solution will be known. The complexity of the algorithm can be expressed by the recurrence $T(n)=T(3n/4)+O(\sqrt{n})$. Hence, the algorithm runs in $O(\sqrt{n})$ time.

4. INCLUSION OF POINTS IN A CONVEX POLYGON

Some algorithms use a test for consistency (inclusion) of $O(n)$ points from a set P in a convex polygon S with $O(n)$ vertices. We give an optimal MCC algorithm for solving the problem.

Let M be an internal point of S (the median point of a three element subset of vertices will do). By broadcasting, PEs containing vertices of S and P can learn the cartesian coordinates of M and can calculate their slope. We sort slopes of points from P, S and $P \cup S$ by their x-coordinate in nondecreasing order. Let rank $r(Y,X)$ be the position of slope Y in the sorted list X. Each PE containing a point A from P can determine two neighboring slopes $A'M$ and $A''M$ (A', A'' in S) for the slope AM by the relations $r(A'M,S)=r(AM, P \cup S)$-

$r(AM,P)$ and $r(A''M,S)=r(A'M,S)+1$. Then A is outside S if and only if line segments AM and $A'A''$ intersect.

5. MAXIMUM GAP PROBLEM

The problem of finding the largest empty circle with the center inside the convex hull of planar set S reduces in one dimension to the following problem:

Given n real numbers $x(1),\dots,x(n)$ find the length of the largest gap between two consecutive numbers in sorted order. Although sorting requires O(n log n) worst case time, a linear time sequential solution to the problem is given by Gonzalez [6]. Thus, he has shown that the largest gap can be found efficiently without the need to sort the numbers. We give analog MCC solution of the problem in optimal time.

We place each $x(i)$ in a PE. First we find minimal and maximal number MIN and MAX in the given set by performing a RAW. We create n-1 buckets by dividing the interval from MIN to MAX with n-2 equally spaced points. For each point (except MIN and MAX), in parallel, determine its bucket by $BUCKET:=1+\lfloor(n-1)(x(i)-MIN)/(MAX-MIN)\rfloor$.

We now put the minimal and maximal number of each bucket in a PE so that these data are placed in n-1 PEs in snake-like order. By performing RAWs in parallel, each bucket can find its minimal and maximal number. n-2 points have been placed in n-1 buckets, so by the pigeonhole principle some bucket must be empty. This means that the largest gap cannot occur between two points in the same bucket.

In parallel, we find the distance of the minimal and maximal point in each bucket to the corresponding elements in the nearest nonempty buckets. First we inform PE in leftmost column about the following data: if the corresponding row contain a nonempty bucket. Now each PE search for his neighbor first in his row, then through leftmost column find the nearest row containing a nonempty bucket and find the bucket and calculate the distance from it. In this way it can be done in optimal time in parallel. By a RAW a processing element can find maximal gap as the maximal distance among obtained distances.

Note that another optimal time solution for the maximum gap problem can be obtained using the optimal sorting procedure [12]

and finding the largest distance between two consecutive elements in sorted order, but such solution requires larger constants.

6. THE DIAMETER OF A k-DIMENSIONAL BINARY IMAGE

Given a black/white picture stored one pixel per processing element in an $n \times n$ mesh-connected computer, and a l_p metric, where for $1 \le p < \infty$, the l_p distance from (a,b) to (c,d) is $[|a-c|^p + |b-d|^p]^{1/p}$ (the l_∞ is max $\{|a-c|, |b-d|\}$) an optimal $O(n)$ algorithm for computing the diameter (the farthest pair of black pixels) is given in [4,9]). However, computing the diameter of a $n \times n \times n$ three-dimensional picture remained an open problem in [4,9]. Here we give a general solution, for the case of a k-dimensional digital picture, where one pixel per processor of a k-dimensional mesh computer is stored. The solution has the time complexity $O(kn)$ which is optimal for fixed k. Note that l_2 correspond to the Euclidean metrics, while l_1 and l_∞ are called "city block" and "chessboard" metrics, respectively.

A k-dimensional mesh-connected computer (k-D MCC) is a set of n^k synchronized procesing elements arranged in a $n \times n \times \ldots \times n$ grid of dimension k. Each processing element is connected via bidirectional unit time communication links to its $2k$ neighbors, if they exist. More precisely, a node m of a k-D MCC has coordinates $m = m_k m_{k-1} \ldots m_1$, $0 \le m_i < n$ for all i, $1 \le i \le k$, and two nodes m' and m'' are connected by a link if and only if there exists j, $1 \le j \le k$, such that $m'_j = m''_j + 1$ and $m'_i = m''_i$ for $i \ne j$. Each processor has a fixed number of registers and can perform standard arithmetic and comparison operations in constant time. It can also send the contents of a register to a neighbor and receive a value from a neighbor in a designated register in constant time. The only one data movement operations on k-D MCC we use is rotating data within a row (in the direction of an arbitrary dimension) in $O(n)$ time: for each row in parallel, every processing element can transmit a data to all other processing elements in its row. For $n=2$ this scheme describes a k-dimensional hypercube, a model which has appeared on the commercial market.

Given two points $x=(x_1,\ldots,x_k)$ and $y=(y_1,\ldots,y_k)$ in k-dimensional space, the l_p distance from x to y is defined as $[|x_1-y_1|^p + \ldots + |x_k-y_k|^p]^{1/p}$. In order to find the diameter of a k-dimensional binary image we solve the following problem: for each pixel (black or white) find the farthest black pixel from it. The solution goes in k steps, each time increasing the dimension of searched sub-grid by 1.

In other words, at a step $j(1 \le j \le k)$, each pixel finds the farthest black pixel in a j-dimensional sub-grid containing the pixel, where all pixels of a sub-grid have the same last $k-j$ coordinates.

Initially the desired *distance(x)* of a pixel x is equal to 0 if x is a black pixel and undefined otherwise. Then the following algorithm is running for $j=1,2,...,k$ in parallel:

Rotate each row in dimension j and update *distance(x)* as follows:

$$distance(x) := max(distance(y)+|x_j-y_j|^p),$$

where y ranges over searched row, and *max* denotes the function which chooses the maximum over obtained values. Note that at given step $x_i=y_i$ for $1 \le i \le k$ except that $x_j \neq y_j$ and that only the points y with defined *distance(y)* are considered. The desired result is $distance(x)^{1/p}$ (the modification of the algorithm for l_∞ metric is straightforward).

The proof of correctness of the algorithm is based on the following property of l_p distances: for a given point $x=(x_1,..., x_n)$, its farthest point among points having coordinates $(y_1,...,y_{j-1},c,x_{j+1},...,x_n)$ (i.e. $j-1$ -dimensional space with last $n-j+1$ coordinates fixed) is the same point which is farthest from point $(x_1,...,x_{j-1},c,x_{j+1},...,x_n)$ among points of the same $j-1$ -dimensional space. Thus by rotating through the row in dimension j furthest distance can be updated on the basis of x_j-c and obtained distance for point with j-coordinate c $(1 \le c \le n)$.

In order to find the diameter of given digitized picture, a similar procedure is performed: for each j, $j=1, 2, ..., k$, in parallel rotate each row in dimension j and update *diameter(x)* as follows:

$$diameter(x) := max(diameter(y)),$$

where y denotes any element in the searched row (initially *diameter(x)* = *distance(x)*). At the end of the procedure each processor will learn the desired diameter (i. e. *diameter(x)* = *diameter(y)* for all pixels).

Our algorithm runs actually for any monotone metric, where a metric d is said to be monotone it the following property is valid for all points (processing elements) P, Q and R: if Q and R are neighbors and the l_1 distance from P to R exceeds the l_1 distance from P to Q, then $d(P,R) \ge d(P,Q)$ (all l_p metrics are clearly monotone).

The problem studied in the paper can also be defined as follows: given a set of selected processors in a k-D MCC find the furthest distance between a pair of selected processors. In case of a hypercube it gives an $O(k)$ algorithm to solve the problem.

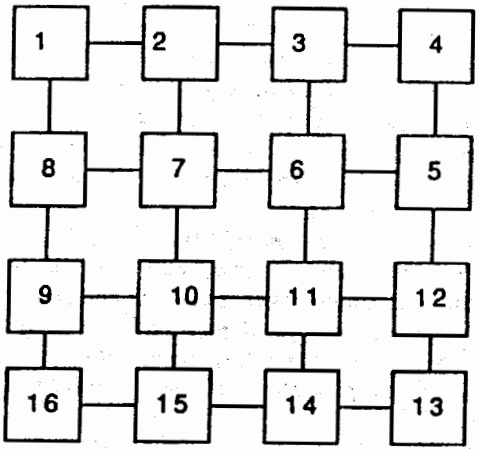
1
2
3
4
8
7
6
5
9
10
11
12
16
15
14
13

Fig. 1

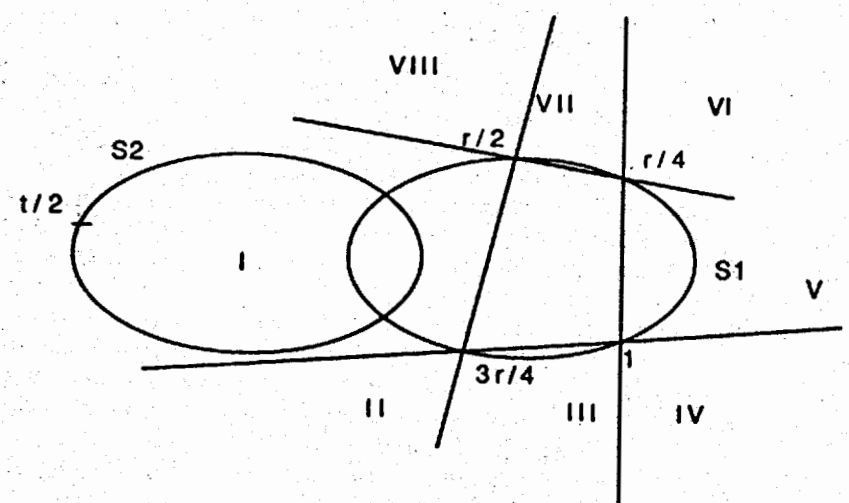
VIII
VII
VI
r/2
r/4
S2
t/2
I
S1
V
3r/4
1
II
III
IV

Fig. 2

REFERENCES

[1] Dehne, F., *Optimal algorithms for the maximal elements and ECDF searching problem on a mesh-connected parallel computer, Inform. Process. Lett., 22, 1986, 303-306.*

[2] Dehne,F., Stojmenović,I., *An optimal algorithm for the ECDF searching problem for arbitrary dimensions on a mesh-of-processors, Inform. Process. Lett., 28, 2, 1988, 67-70.*

[3] Dehne,F., Sack,J.-R., Stojmenović,I.,, *A note on determining the 3-dimensional convex hull of a set of points on a mesh of processors, Scandinavian Workshop on Algorithm Theory (SWAT), Lecture Notes in Computer Science, Springer-Verlag, 318, 1988, 154-161.*

[4] Dyer, C.R., Rosenfeld, A., *Parallel image processing by memory augmented cellular automata, IEEE Trans. PAMI-3, 1981, 29-41.*

[5] Dyer,M.E., *Linear time algorithms for two- and three- variable linear programs, SIAM J. Comput., 13,1, 1984, 31-45.*

[6] Gonzalez,T., *Algorithms on sets and related problems, TR Dept. Comp. Sci. Univ. of Oklahoma, 1975.*

[7] Megiddo,N., *Linear time algorithm for linear programming in R^3 and related problems, SIAM J. Comput., 12,4, 1983, 759-776.*

[8] Miller,R., Stout, Q.F., *Computational geometry on a mesh-connected computer, Proc. Int'l. Conf. on Parallel Processing, 1984, 66-73.*

[9] Miller,R., Stout,Q.F., *Geometric algorithms for digitized pictures on a mesh-connected computer, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. PAMI-7, 2, 1985, 216-228.*

[10] O'Rourke,J., Kosaraju,S.R., *Computing circular separability, Disc. Comp. Geom., 1, 1986, 105-113.*

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

[12] Thompson,C.D., Kung,H.T., *Sorting on a mesh-connected parallel Computer, Comm. ACM, 4, 20, 1977, 263-271.*

REZIME

KOMPJUTERSKA GEOMETRIJA NA MREŽNO POVEZANOM RAČUNARU

Rad opisuje optimalne algoritme za rešavanje nekih fundamentalnih geometrijskih problema na mrežno povezanom paralelnom računaru: nalaženje najmanjeg kruga koji obuhvata skup tačaka, ispitivanje kružne odvojivosti dva skupa tačaka, prepoznavanje digitalnog diska, nalaženje najmanjeg i najvećeg kruga koji razdvaja dva skupa tačaka, ispitivanje linearne odvojivosti dva skupa tačaka, ispitivanje pripadnosti tačaka datom konveksnom poligonu, odredjivanje najvećeg razmaka u skupu brojeva, i odredjivanje prečnika crno-bele slike. Rešenje poslednjeg problema je uopšteno, za slučaj više-dimenzionalne digitalne slike na više-dimenzionalnom mrežno povezanom računaru.

Received by the editors December 10, 1986.