Time-Optimal Algorithm for Computing the Diameter of a Point Set on a Completely Overlapping Network

Prapaporn Techa-angkoon and Saowaluk Rattanaudomsawat

***Abstract*- Given a finite set *P* of *n* points in *d*-dimensional Euclidean space, the diameter is defined as the maximum Euclidean distance between any two points in the set *P*. In this paper, we illustrate a time-optimal algorithm to compute the diameter of a point set on a theoretical network called a *completely overlapping network (CON)*. This network model has an applicable potential in real-life applications because it is an extension of LANs that are widely used at present.**

***Index Terms*- computational geometry, diameter, overlapping network, time-optimal algorithm.**

I. Introduction

Parallel computation has been around for decades. Several parallel applications and architectures are available for use. However, parallel architectures such as hypercube and mesh are generally expensive [18] and hence their use is limited to only those who can afford them. Some attempts have been made to find an alternative to these expensive parallel machines. One alternative is called a cluster of workstations and personal computers. A typical cluster of workstations is essentially a group of numerous workstations and personal computers connected through a single communication line. Each computer can send a message, bit by bit, when the communication line is free. If the communication line is currently occupied, the computer must wait before it is allowed to send its message.

One significant problem with this model of communication via a single communication line is the line can only serve one computer at any time. To lessen this problem, during the recent decade or so, a group of computer scientists in the United States has developed an experimental network called *an overlapping network* [2]-[3], [10]. They have worked on the concept of using multiple Ethernet lines in some certain configurations. These configurations are in the general classification of overlapping connectivity networks. Overlapping connectivity networks have the characteristic that regions of connectivity are provided and the regions overlap so as to provide parallelism. The overlapping connectivity scheme is suitable for processors having local memory and can be applied to both fine-grained and coarse-grained processors.

Recently, Kantabutra *et al.* [15]-[17] extended the network model of Wilkinson one step further to provide complete overlapping of communication. This theoretical network model called *a completely overlapping network (CON)* is more general than but similar to the experimental overlapping network. Since Wilkinson's multiple bus network model yields a good result [2], it is worth studying properties of CON and investigating its potential. Therefore, in this paper we will refer to this network as CON and demonstrate the use of CON and its usefulness by solving problem of computing the diameter of a point set.

Computing the diameter of a point set is one of the classical problems in computational geometry [1], [6]-[8], [12]. Given a finite set P of n points in d-dimensional Euclidean space, the diameter of P is defined as the maximum Euclidean distance between any two points of P. This problem is actually known as the *diameter problem* or the *farthest pair problem*. There are several interesting applications of computing the diameter such as image databases, visualization, clustering, and data mining [5]. Additionally, the diameter is quite useful as it provides a reliable estimate of the point-set extent and it can be used in computing a tight fitting bounding box for the point set.

Computing the diameter of a point set has a long history. On a sequential machine, computing the diameter of n points in d dimensions requires $\Omega(n \log n)$ operations [8] while a trivial $O(n^2)$ upper bound is provided by the brute force algorithm that compares the distance between all pairs of points. In the dimensions 2 and 3, this problem can be solved optimally in $O(n \log n)$ [5] but in the higher dimensions it becomes non-trivial.

We now discuss a little bit about parallel computation of the diameter of a point set. Computing the diameter of a point set is a basic problem. Several parallel algorithms for computing the diameter of a point set exist in computer science literature depending on the kinds of network architectures they use. Most existing parallel algorithms are described in the context of networks such as mesh [9], [11], [16], hypercube [11], tree [14], etc. The rest are mostly in the environment of cluster of

Manuscript received December 30, 2008.

P. Techa-angkoon is with The Theory of Computation Group, Computer Science Department, Faculty of Science, Chiang Mai University, Chiang Mai, 50200, Thailand. Phone: 6653943409, Fax: 6653943433 (e-mail: prapapon@chiangmai.ac.th).

S. Rattanaudomsawat is with The Theory of Computation Group, Computer Science Department, Faculty of Science, Chiang Mai University, Chiang Mai, 50200, Thailand (e-mail: saowaluk.r@gmail.com).

workstations with on communication bus, i.e., LANS [13]. Nowadays, we are not aware of any algorithm for computing the diameter of a point set on the architecture like CON.

In following sections, the definition and rules of operations of a completely overlapping network is firstly given. Secondly, the algorithm for computing the diameter of a point set on a completely overlapping network is presented. Also, we show a rigorous proof that our diameter computing algorithm is optimal. Lastly, we conclude our paper and propose a transformation method of our fine-grained model into a course-grain model.

II. Completely overlapping network

In this section, we give a review of a completely overlapping network as shown in [15], [17]. A completely overlapping network is composed of several overlapped communication lines that connect among several nodes (or processors) to provide parallelism. There are vertical and horizontal communication lines as shown in Fig. 1.

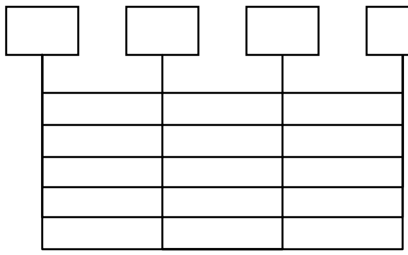

Fig. 1. Four-node completely overlapping network.

The number of vertical lines is equal to the number of nodes n and the number of horizontal lines is equal to $\frac{n(n-1)}{2}$. Additionally, one straight line segment equates one step horizontally and vertically. For instance, Fig. 2 shows a communication of 5 steps between the leftmost node and the rightmost node.

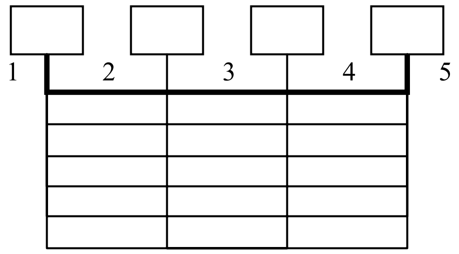

Fig. 2. Five-step communication between nodes.

Like any other networks, there are rules of operations which our proposed algorithm runs follow them. The rules are as follows.

- Horizontal and vertical line segments cannot be shared. That is, any line segment can be used only one at a time.
- Each line segment is bidirectional.
- Each node has a constant memory size.
- A same message can be concurrently sent from one source node to several destination nodes as long as there is no collision of messages.
- If there exists contention for a communication line segment, some kind of priority can be applied.

In order to enable readers to understand our communication method, a numbering of both nodes and communication lines is necessary. Our numbering scheme is illustrated in Fig. 3. This figure shows a four-node completely overlapping network with node and line identification numbers, *myID* and *lineID*, respectively. It is easy to generalize this numbering scheme for a *n*-node completely overlapping network. Hereafter, we will regularly refer to this numbering scheme when explaining our algorithm.

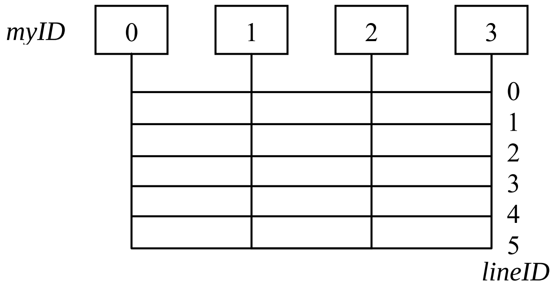

Fig. 3. Numbering scheme for a four-node completely overlapping network.

III. Time- Optimal Algorithm for computing the diameter of a point set on a Completely overlapping network

In this section, we will describe a time-optimal algorithm that solves the diameter problem of a point set on a completely overlapping network. The definition of this problem and our algorithm are given respectively. Then we present a rigorous proof that our diameter computing algorithm is optimal.

Definition 1 (Diameter Problem) Given a finite set P of n distinct points p_i in d dimensions where $n \geq 2$, $0 \leq i \leq n\text{-}1$. Let $d(p_i,p_j)$ be a Euclidean distance between two points $p_i(x_1,x_2,\ldots,x_d)$ and $p_j(y_1,y_2,\ldots,y_d)$ when $i \neq j$, and

$$ed(p_i,p_j) = \sqrt{\sum_{m=1}^{d} (x_m - y_m)^2}$$

The diameter of the set P is the maximum Euclidean distance between any two points in the set P, or that is, any pair of points of the set P that is farthest away from each other.

Our Diameter Computing Algorithm is specifically designed to suit the completely overlapping network. The following are our assumptions and the description of some variables in the algorithm.

- Each processor (or node) has a point stored in it initially.

- There is a total of $n \geq 2$ processors in the completely overlapping network where n is the number of points to be computed the diameter.
- One point per one processor and *myID* is its own point identification number.
- All processors are fine-grained.
- A variable S is a finite set of any pair of points that represents the diameter of the set P.
- A variable *max* is the diameter of the set P.

There are two communication subroutines in the algorithm: ***send***(*data*, *destination process*, *communication line number*) and ***receive***(*data*, *source process*). One of the arguments in subroutine *send*() indicates the communication lines to use. (There is no such argument in *receive*().) These subroutines require identification numbers for both lines and nodes. These identification numbers, *myID* and *lineID*, were described in the previous section. Also note that ***pack***(*item1*, *item2*,…, *itemN*) is a subroutine that packs all stated items together as one larger item and the subroutine ***unpack***(*packedItems*, *item1*, *item2*,…, *itemN*) does just the opposite.

The description of our algorithm called *Diameter Computing Algorithm* for any processor in CON is given as follows. Some of pseudocode conventions are borrowed from [4].

Diameter Computing Algorithm for Processor P_i

```
1.  $S = \varnothing$, $max = -\infty$
2.  for $i$ = 0 to $n$-1 and $i \neq myID$
3.      send(myPoint, $P_i$, myID)
4.  for $i$ = 0 to $n$-2
5.      receive(point, $P_{ANY}$)
6.      dist = ed(myPoint, point)
7.      if (dist > max)
8.          $S = \varnothing$
9.          max = dist
10.         $S$ = {(myPoint, point)}
11.     else if (dist = max)
12.         $S = S \cup$ {(myPoint, point)}
13. twoItems = pack(max, S)
14. for $i$ = 0 to $n$-1 and $i \neq myID$
15.     send(twoItems, $P_i$, myID)
16. for $i$ = 0 to $n$-2
17.     receive(packedItems, $P_{ANY}$)
18.     unpack(packedItems, max_dist, SS)
19.     if (max_dist > max)
20.         $S = \varnothing$
21.         max = max_dist
22.         S = SS
23.     else if (max_dist = max)
24.         $S = S \cup SS$
```

Like any communication scheme, it is essential that there be no collision of messages on any of these communication lines (or any line segment) at any point in the algorithm. Kantabutra *et al.* [15], [17] showed that this communication scheme produces no collision.

Theorem 1 (Algorithm's Correctness). *The Diameter Computing Algorithm is correct.*

Proof. Let P_i be an arbitrary processor i, $0 \leq i \leq n$-1, in a completely overlapping network. In the algorithm, two variables, S and *max*, are initially set to $\varnothing$ and $-\infty$, respectively (line 1). Then, each processor P_i sends out its own point *myPoint* to the other processors P_j where $j \neq i$ (lines 2-3). Upon receiving these points, each processor P_i computes the Euclidean distance between its own point *myPoint* and the just-received point *point*, and then keeps this distance in *dist*. Each processor P_i compares *dist* and *max* that if *dist* is greater than *max*, it replaces *max* with the new maximum distance *dist* and also replaces any pair of points in the set S with the new pair of points, *myPoint* and *point*, that is just computed. If *dist* is equal to *max*, each processor P_i collects that pair of points which also has the maximum distance in the set S (lines 4-12). Presently, each processor P_i knows that which point is the farthest away from its own point and also their corresponding distance. However, it still does not know the exact pair(s) of points which is the farthest away from each other. Therefore, it sends the set S of pairs of points and the maximum distance *max* between its own point and the other points as a variable *twoItems* to the other processors P_j when $j \neq i$ (lines 13-15). Upon receiving the set S and the maximum distance *max* as *SS* and *max_dist*, respectively, each processor P_i compares *max_dist* to *max*, if max_dist is greater than *max* then the processor P_i replaces *max* with the new maximum distance *max_dist* and also replaces any pair of points in the set S with any pair of points in the set *SS*. If *max_dist* is equal to *max*, the processor P_i includes the set *SS* in the set S (lines 16-24). Eventually, each processor P_i knows the diameter or the maximum distance of the set P, and any pair of points that gives the diameter. Thus, the Time-Optimal Diameter Computing Algorithm on a completely overlapping network is correct. ☐

In addition, the algorithm should be also efficient in terms of time complexity. The following theorem shows that our algorithm has a running time of O(n).

Theorem 2 (Time Complexity). *The Diameter Computing Algorithm on a completely overlapping network has a running time of O(n) when n is the number of the distinct points in the set P.*

Proof. In any parallel algorithm, running time is divided into communication time T^{comm} and computation time T^{comp}. For simplicity, assume that one step in communication is equal to one step in computation. There are four phases in our algorithm.

<u>Phase 1 (Communication):</u> Each processor sends its own point *myPoint* to the other processors (lines 2-3). Since this sending is done in parallel, the time of the longest communication path dominates the whole communication.

$$T_1^{comm} = 3n - 1$$

<u>Phase 2 (Computation):</u> Each processor computes the Euclidean distances between its own point *myPoint* and the other points *point* that receives from the other processors. By comparing, the maximum distance is eventually kept in *max*, and any pair of points corresponding to the maximum distance is also kept in the set *S*. (lines 6-12).

$$T_1^{comp} = n - 1$$

Phase 3 (Communication): Each processor sends a variable *twoItems* consisting of the set *S* and the maximum distance *max* between its own point and the other points to the other processors (lines 14-15) for finding other pairs of points that have the distance greater than or equal to *max*. For simplicity, assuming that time of sending one item and two items are the same, we therefore have

$$T_2^{comm} = 3n - 1$$

Phase 4 (Computation): Each processor checks for whether the distance that receives from the other processors *max_dist* is greater than or equal to the maximum distance *max* that it keeps (lines 19-24). There are in the worst case *n*-1 times to check. Therefore, we have

$$T_2^{comp} = n - 1$$

Hence, the total time complexity T_{total} is

$$T_{total} = T_1^{comm} + T_2^{comm} + T_1^{comp} + T_2^{comp} = O(n). \square$$

Theorem 3 (Optimality). *The Diameter Computing Algorithm is time-optimal on CON.*

Proof. In order to solve *any* diameter computing problem on CON, the parallel diameter computing algorithm must *at least* communicate between the two farthest nodes 0 and *n*-1. Let us call the shortest distance between the two farthest nodes a *diameter*. In CON, the diameter is *n* + 1 steps. This establishes the lower bound Ω(*n*) for the diameter computing problem on CON. Theorem 2 states that the Diameter Computing Algorithm has a time complexity of O(*n*). Hence, theorem 3 holds. □

IV. CONCLUSION

In this paper, we proposed a time-optimal algorithm called *Diameter Computing Algorithm* for calculating the diameter of a finite set of *n* points on a completely overlapping network. Our algorithm not only gives the diameter of the point set but also outputs the set of pair(s) of points that produce(s) the diameter. This algorithm optimally works in O(*n*) time. We can also take satisfaction in knowing that there is no faster algorithm for solving the diameter computing problem on CON. Additionally, our algorithm has a speedup of O(log *n*) over the fastest sequential algorithm of this problem.

A theoretical network CON is an extension of Wilkinson's model. One may ask about practicality of our theoretical CON network. We believe that CON can be implemented cost-effectively since it is similar to Wilkinson's experimental networks that are known to be cheaper than most parallel machines.

Throughout this paper, we only discussed the case in which all processors are fine-grained. However, like the multiple bus network with overlapping connectivity model, the concept of our theoretical network can also be applied to coarse-grained processors with larger memory. This is the case particularly worth attention because it can be applied to existing, widely-used local area networks. Fig. 4 shows an example of embedding a 16-node CON into a 8-node CON.

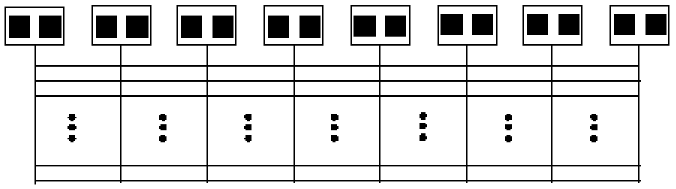

Fig. 4. 16-node CON embedded in a 8-node CON

ACKNOWLEDGMENT

The authors would like to thank Assistant Professor Dr. Sanpawat Kantabutra and the reviewers for their suggestions and significant comments on this paper.

REFERENCES

[1] B. Chazelle, H. Edelsbrunner, L. J. Guibas, and M. Sharir, "Diameter, Width, Closest Line Pair, and Parametric Searching," *Proceeding of the Eighth Annual Symposium on Computational Geometry*, Berlin, Germany, ACM Press, June 1992, pp. 120-129.
[2] B. Wilkinson, "On Crossbar Switch and Multiple Bus Interconnection Networks with Overlapping Connectivity," *IEEE Trans. Computers*, vol. 41, 1992, pp. 738-746.
[3] B. Wilkinson and J.M. Farmer. "Reflective Interconnection Networks", Computer and Electrical Engineering, vol.20, pp. 289-308, 1994.
[4] Cormen T.H., Leiserson C.E., and Rivest R.L., *Introduction to Algorithms*. New York: McGraw-Hill Book Company, 1992.
[5] D. V. Finocchiaro and M. Pellegrini, "On Computing the Diameter of a Point Set in High Dimension Euclidean Space," *Theoretical Computer Science*, vol. 287 (2), pp. 501-514, September 2000.
[6] E. A. Ramos, "An Optimal Deterministic Algorithm for Computing the Diameter of a Three-Dimensional Point Set," *Discrete and Computational Geometry*, vol. 26 (2), Springer New York, pp. 233-244, May 2001.
[7] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer Verlag, October 1990, 3rd edition.
[8] G. Maladian and J.-D. Boissonnat, "Computing the Diameter of a Point Set," *DGCI: Lecture Notes in Computer Science*, vol. 2301, Springer Berlin/Heidelberg, pp. 197-208, 2002.
[9] I. Stojmenović, L. Jerinić, "Finding the diameter of a point set on mesh-connected computers", NSJOM, Vol.17 No.1, pp.251-259, 1987.
[10] K. Hoganson, B. Wilkinson, and W.H. Carsile, "Applications of Rhombic Multiprocessors", Proceeding of the International Conference on Parallel and Distributed Processing Techniques and Applications, 1997.
[11] L. Boxer, R. Miller, "Dynamic Computational Geometry on Meshes and Hypercubes", Hypercube Concurrent Computers and Applications, Proceedings of the third conference on Hypercube concurrent computers and applications , Vol. 2, pp.1212 – 1219,1989.
[12] M.I. Shamos, "Geometry complexity," *Proceedings 7th Annual ACM Symposium on the Theory of Computing*, 1975, pp.224-233.
[13] M. Pal and G. P. Bhattacharjee, "Optimal sequential and parallel algorithms for computing the diameter and the center of an interval graph", *Intern. J. Computer Maths.*, vol. 59, pp. 1-13, 1995.

[14] N. Deo and A. Abdalla, "Computing a Diameter-Constrained Minimum Spanning Tree in Parallel**",** *Lecture Notes in Computer Science*, vol. 1767, Springer-Verlag Berlin Heidelberg, pp. 17-31, 2000.
[15] P. Techa-angkoon, "The Closest Pair of Points Finding Algorithm on a Completely overlapping Network," Proceeding of JCSSE, November 2005.
[16] R. Miller, Q. F. Stout, "Geometric Algorithms for Digitized Pictures on a Mesh-Connected Computer", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.7, pp. 216-228, 1985.
[17] S. Kantabutra, W. Jindaluang, and P. Techa-angkoon, "It's Elementary, My Dear Watson: Time-optimal Sorting Algorithms on a Completely Overlapping Network," *Lecture Notes in Computer Science*, vol. 3758, Springer Berlin/Heidelberg, pp. 252-262, 2005.
[18] T. E. Anderson, D. E. Culler, D. Patterson, "A case for NOW (Networks of Workstations)", IEEE Micro. . 15, No. 1 (Feb.) 54-64, 1995.