Computational Geometry on Optical Multi-Trees (OMULT) Computer System

Rabiul Islam, Nahid Afroz, Subir Bandyopadhyay* and Bhabani P Sinha†

Abstract

This paper presents simple and efficient algorithms for fundamental computational geometry problems on OMULT (Optical Multi-Trees) system [5] that uses both electronic and optical links among processors. We show that the algorithms for convex hull and the smallest enclosing box can be computed on this network in $O(\log n)$ time, compared to $O(\sqrt{n})$ algorithms on the OTIS-Mesh[6] for each of these problems.

1 Introduction

In order to achieve better execution performance of computer systems through parallelization, there have been considerable efforts in designing interconnection networks for parallel computers over the last few decades. One of the recent architectures for this purpose is the Optical Interconnect System [3], [4], [7] in which processors are partitioned into groups so that processors within each group are interconnected by electronic links and processors in different groups are interconnected by optical links. The Optical Transpose Interconnect System (OTIS) proposed by Marsden et al [4] is an example of such a hybrid architecture and various fundamental algorithms have been conveniently implemented on the OTIS [6].

Recently, Sinha and Bandyopadhyay [5] have introduced another opto-electronic computer system, called the Optical Multi-Trees (OMULT). Basic broadcast operations like single data broadcast, row/column group-broadcast, complete group-broadcast on the OMULT topology can all be done in $O(\log n)$ time [5]. Fundamental algorithms such as the summation/average/maximum/minimum of n^3 elements, prefix computation of n^2 elements, multiplication of two $n \times n$ matrices can all be done in $O(\log n)$ time, and n^2 elements can be sorted in $O(\log^2 n)$ time on the OMULT system [5]. On the OTIS-Mesh, various fundamental problems in computational geometry were efficiently mapped by Wang and Sahni [6]. For example, finding the

i) convex hull (CH),
ii) the smallest enclosing box (SEB),
iii) the empirical cumulative distribution function (ECDF) and
iv) all-nearest neighbor(ANN) problem

were solved on an OTIS-Mesh in $O(\sqrt{n})$ time for n data inputs. In this paper, we present the algorithms for the CH and the SEB algorithms on the OMULT system that can be solved only in $O(\log n)$ time for n data inputs. The ECDF and the ANN problems could not be presented here due to lack of space.

The paper is organized in the following way. In section 2, we briefly present the basic topological features of the OMULT system. In section 3, we describe our proposed algorithms for solving the above mentioned problems in computational geometry using the OMULT system. Finally, we conclude in section 4.

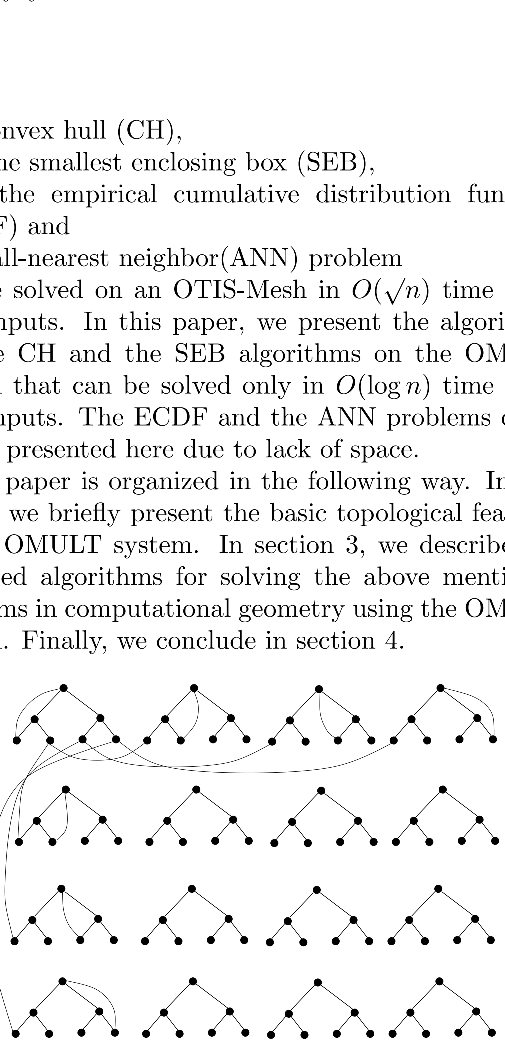

Figure 1: Optical Multi-Trees (OMULT) System

2 OMULT Topology

Figure 1 shows the schematic diagram of the Optical Multi-Trees (OMULT) architecture[1] [5] which uses n^2 complete binary trees T_{ij}'s ($1 \leqslant i, j \leqslant n$) each having n leaf nodes and $n-1$ internal nodes, as the basic building blocks. These n^2 trees are organized in the form of an $n \times n$ array. Nodes (processors) in an individual tree are all connected by electronic links and those in different trees are connected by bi-directional optical links in both horizontal and vertical directions according to rules given below. To describe the architecture,

*School of Computer Science, University of Windsor, Ontario, Canada, subir@uwindsor.ca

†Indian Statistical Institute, Calcutta, India, bhabani@isical.ac.in

[1]All optical interconnections are not shown for clarity

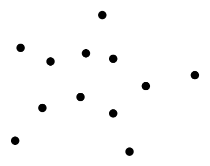
Figure 2: Set of points

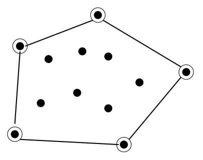
Figure 3: Convex hull for the points shown in Figure 2

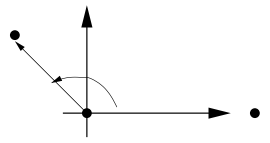
Figure 4: Polar angle

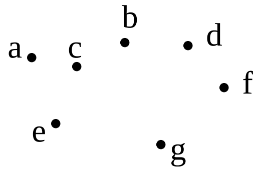

Figure 5: Set of points

we label the nodes in each tree $T_{ij}, (1 \leqslant i, j \leqslant n)$, by distinct integers from 1 to $2n-1$ in reverse level order, i.e., the leaf nodes in each tree are numbered from 1 to n, in order from left to right, and the internal nodes are also numbered from left to right in successive lower levels (the root node being at the lowest level - level 0). Thus, the root node in each tree is given the node number $2n-1$, and the node k in a tree T_{ij} will be referred to by the processor node $P(i,j,k), (1 \leqslant i, j \leqslant n)$, $(1 \leqslant k \leqslant 2n-1)$. The total number of nodes in the system is $N = n^2(2n-1) = 2n^3 - n^2$. The optical links are given by the following rules:
1. Bi-directional horizontal interconnection between processors $P(i,j,k)$ and $P(i,k,j), 1 \leqslant i,j,k \leqslant n$, if $j \neq k$.
2. Bi-directional vertical interconnection between processors $P(i,j,k)$ and $P(k,j,i), 1 \leq i,j,k \leq n$, if $i \neq k$.
3. Bi-directional interconnection between processor $P(i,j,k)$ and $P(i,j,2n-1), 1 \leq i,j,k \leq n$, if $i = k$ and/or $j = k$.

3 Mapping of Algorithms in Computational Geometry

3.1 Convex Hull

To find the convex hull [8] for a given set of points S on a plane ($|S| = n$), we need to identify the extreme points. We assume that no three points in S are collinear. Figure 2 shows an example for the set of points S and Figure 3 shows the corresponding convex hull of S. Corresponding to a point $p_i \epsilon S$, let $p_{i0}, p_{i1}, ..., p_{i,n-2}$ be the points in $S - p_i$, (i.e., $p_{ik} \neq p_i$ for $0 \leq k \leq n-2$), sorted by the polar angle made by the vector $\overrightarrow{p_i p_{ik}}$, $0 \leq k \leq n-2$. Then, by the results shown in [6], p_i is an extreme point of S if the counterclockwise angle between some pair of consecutive vectors $\overrightarrow{p_i p_{ik}}$ and $\overrightarrow{p_i p_{i,|k+1|_{n-1}}}$ is more than π. For example, for $S = a, b, c$, if the counter-clockwise angle (polar angle) between the vectors $\overrightarrow{ab}$ and $\overrightarrow{ac}$, as shown in Figure 5, is greater than π, then point a is an extreme point.

Figure 5 shows a set of points $S = \{a, b, c, d, e, f, g\}$, for which an extreme point is illustrated in Figure 6. The counterclockwise angle between the vectors $\overrightarrow{ea}$ and $\overrightarrow{eg}$ is more than π, so e is an extreme point, whereas c is not an extreme point shown in Figure 7 because the counterclockwise angle between no two consecutive vectors (sorted by their polar angles) originating at the point c is more than π.

Based on the above property, we describe below our proposed algorithm for finding the convex hull for a set of points $S = \{p_1, p_2, ..., p_n\}$, no three of which is collinear. We assume that each processor $P(i,j,k)$ in the OMULT system has two registers $A(i,j,k)$ and $B(i,j,k)$, where we would use the A-register for any data movement operation. The coordinates of all n points are initially stored in the A-registers of the leaf nodes of the tree T_{11}.

3.1.1 Algorithm CH :

```
Step 1: /* move coordinates of $p_i$ and $p_j$ to the tree $T_{ij}$ */
Step 1.1 : /* using horizontal optical links, move data from $T_{11}$ to $T_{1j}$, $1 \leq j \leq n$ */
    $\forall j, 1 \leq j \leq n$, do in parallel
    $A(1,j,1) \leftarrow A(1,1,j)$;
Step 1.2 : /* broadcast data within each tree $T_{1j}$ */
    $\forall j,k, 1 \leq j,k \leq n$, do in parallel
    $A(1,j,k) \leftarrow A(1,j,1)$;
Step 1.3 : /* using vertical optical links, move data from $T_{1j}$ to $T_{ij}, 1 \leq i,j \leq n$*/
    $\forall i,j, 1 \leq i,j \leq n$, do in parallel
    $A(i,j,1) \leftarrow A(1,j,i)$;
Step 1.4 : /* broadcast data within each tree $T_{ij}$ */
    $\forall i,j,k, 1 \leq i,j,k \leq n$, do in parallel
    $A(i,j,k) \leftarrow A(i,j,1)$;
Step 1.5 : /* using horizontal optical links, move data across tree $T_{ij}$ , $\forall i, 1 \leq i,j \leq n$ */
```

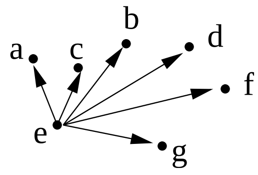

Figure 6: e is an extremum point

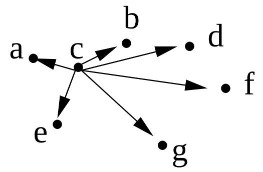

Figure 7: c is not an extremum point

```
    $\forall i, j, k, 1 \le i, j, k \le n$, do in parallel
    $A(i, k, j) \leftarrow A(i, j, k)$;
/* After step 1.5, all $p_i$ values, $1 \le i, j \le n$, are stored in the leaf nodes of each tree */
```

Remark : After step 1, the A-registers of the processors $P(i, j, i)$ and $P(i, j, j)$, $1 \le i, j \le n$, in the leaf nodes of the tree T_{ij} store the coordinate values of the points p_i and p_j , respectively.

Step 2 : $\forall i, j$, $1 \le i, j \le n$, compute the vector $\overrightarrow{p_i p_{ik}}$ in the tree T_{ij}, and store it in the register $A(i, j, 2n-1)$ of the respective root node. (Note that for $i = j$, a 0 value will be stored for the vector).

Step 3 : /* Broadcast the vector $\overrightarrow{p_i p_{ik}}$ to all trees $T_{ij}, 1 \le j \le n$ in the same row */

```
Step 3.1 : /* using electronic links */
    $\forall i, j, k, 1 \le i, j, k \le n$, do in parallel
    $A(i, j, k) \leftarrow A(i, j, 2n-1)$;
Step 3.2 : /* using horizontal optical links */
    $\forall i, j, k, 1 \le i, j, k \le n$, do in parallel
    $A(i, k, j) \leftarrow A(i, j, k)$;
    $B(i, k, j) \leftarrow A(i, k, j)$;
```

Remark : After step 3.2, both $A(i, j, k)$ and $B(i, j, k), 1 \le i, j, k \le n$, store the vector $\overrightarrow{p_i p_{ik}}$ originating at point p_i.

Step 4 : Sort the n vectors $\overrightarrow{p_i p_{ik}}$ (including the zero vector) stored in the leaf nodes of the trees $T_{ij}, 1 \le i, j \le n$ in the order of their polar angles by rank computation, in a manner similar to that described in the algorithm SORT in [5]. The A-register in each processor will still be used for data movements across different processors needed for this rank computation. Store the sorted list of vectors $\overrightarrow{p_i p_{ik}}$ for all $k, 1 \le k \le n$, in order from left to right in the leaf nodes of the tree $T_{ij}, 1 \le i \le n$.

Step 5 : Broadcast the sorted list of vectors in $T_{ij}, 1 \le i \le n$ to leaf nodes of all trees $T_{ij}, 1 \le i, j \le n$ in the same row.

Step 6 : Assuming that the sorted list of vectors in T_{ij} is $(0, p_i p_{i0}, p_i p_{i1}, ..., p_i p_{iq})$, where $q = n-2, p_{ik} \ne p_i$ for $0 \le k \le q$, compute the counterclockwise polar angle between vectors $\overrightarrow{p_i p_{ik}}$ and $\overrightarrow{p_i p_{i,|k+1|_{n-1}}}$ in the tree T_{ij}, where p_{ik} is actually the point p_j , and store it in the processor $P(i, j, 2n-1)$.

Step 7 : If the polar angle computed in any tree T_{ij}, (where $p_{ik} = p_j$) is more than π, then point p_i is an extreme point formed by the sides $\overrightarrow{p_i p_{ik}}$ and $\overrightarrow{p_i p_{i,|k+1|_{n-1}}}$, and this information is conveyed to the processor $P(i, 1, 1)$ in T_{i1} by setting an appropriate tag bit (tag = 1, if p_i is an extreme point, and 0 otherwise) along with the 3-tuple $(p_{ik}, p_i, p_{i,|k+1|_{n-1}})$. This is accomplished by first transferring the above information from $P(i, j, 2n-1)$ to $P(i, j, 1)$ in $\log n$ steps, then from $P(i, j, 1)$ to $P(i, 1, j)$ in one step by using a horizontal optical link, and then from $P(i, 1, j)$ to $P(i, 1, 1)$ in $2\log n$ steps. Next move the information regarding all such convex hull points to the leaf nodes of the processors $P(1, 1, k), 1 \le k \le n$, in the tree T_{11}, in one step using the vertical optical links.

3.1.2 Time Complexity :

Steps 1.2 and 1.4 need $2\log n$ data transfer steps each. Hence, step 1 needs $4\log n + 3$ time units. Step 2 needs $\log n$ time units. Also, step 3.1 requires $\log n$ time units. Step 3.2 needs 2 time units. Step 4 will require $3\log n + 2$ time units. Step 5 needs $2\log n + 3$ time units. Step 6 needs $\log n$ time units. Step 7 requires $3\log n + 3$ time units (assuming that setting the tag bit and related information about an extreme point requires one time unit). Hence, we have the following result.

Theorem 1 *Algorithm CH computes the convex hull of n points in $O(\log n)$ time.*

4 Smallest Enclosing Box

In the smallest enclosing box (SEB) problem, given a set S of coplanar points, we are to find a minimum area rectangle that encloses all points in S [6]. Freeman and Shapiro showed [1] that the SEB of S has one side that is collinear with an edge of the convex hull of S and that the remaining three sides of the SEB pass through at least one convex hull vertex each. We describe below the algorithm for solving the SEB problem on the OMULT system.

4.0.3 Algorithm SEB

```
Step 1 : /* compute the convex hull vertices and store the corresponding information in the leaf nodes of the tree $T_{11}$ */
  $\forall i, 1 \le i \le n$, do in parallel
  if (point $p_i$ is a convex hull vertex) then $P(1, 1, i) \leftarrow (p_{ik}, p_i, p_{i,|k+1|n-1})$
  else $P(1, 1, i) \leftarrow 0$;
```

Step 2 : Sort the convex hull vertices in the order of their polar angles using the sort algorithm described in [5].

Step 3 : Broadcast the information about the convex hull vertices from T_{11} to all trees $T_{i1}, 1 \leq i \leq n$.

Step 4 :

i) $\forall i, 1 \leq i \leq n$, compute the i^{th} hull edge $(p_i, p_{i,|k+1|_{n-1}})$ in T_{i1} and broadcast to the leaf nodes $1, 2, \ldots, n$ of the trees $T_{ij}, 1 \leq j \leq n$ in the same row.

ii) broadcast all the hull vertices to the leaf nodes of $T_{ij}, 1 \leq i, j \leq n$ (node $P(i, j, j)$ gets the relevant information regarding the vertex p_j , if p_j is a hull vertex).

Step 5 :

i) If p_j is a hull vertex, $\forall i, j, 1 \leq i, j \leq n$, using the leaf node $P(i, j, j)$, compute the height d_1 between the hull vertex p_j and the hull edge $(p_i, p_{i,|k+1|_{n-1}})$.

ii) Compute, using $P(i, j, j)$, the perpendicular bisector L of the hull edge $(p_i, p_{i,|k+1|_{n-1}})$.

iii) Calculate, using $P(i, j, j)$, the distance d_2 from the vertex p_j to this perpendicular bisector L.

iv) If p_i and p_j are on the same side of L, using processor $P(i, j, j)$, set left $\leftarrow d_2$ and right $\leftarrow 0$; otherwise using processor $P(i, j, j)$, set left $\leftarrow 0$ and right $\leftarrow d_2$.

Step 6 :

i) $\forall i, j, 1 \leq i, j \leq n$, store in the root node $P(i, j, 2n-1)$, the values of the height, left, and right computed in step 4

ii) send the values stored in $P(i, j, 2n-1)$ to the node $P(i, 1, j)$ of the tree T_{i1} (via $P(i, j, 1)$).

Step 7 : Compute $\forall i, 1 \leq i \leq n$,

h_{max}, the maximum of all height values in processors $P(i, 1, j)$,

r_{max}, the maximum of all right values in processors $P(i, 1, j)$,

l_{min}, the minimum of all the left values in processors $P(i, 1, j)$,

Store the results in the leaf nodes of T_{i1} to find the farthest, rightmost and leftmost points, respectively from the point p_i.

Step 8 : $\forall i, 1 \leq i \leq n$ compute, in processor P(i, 1, 2n-1), the area $A_i = h_{max}(r_{max} - l_{min})$.

Step 9 : $\forall i, 1 \leq i \leq n$, move the value of the area A_i from $P(i, 1, 2n-1)$ to the leaf node $P(1, 1, i)$. (This is done by first moving A_i to $P(i, 1, 1)$ in $\log n$ steps and then to $P(1, 1, i)$ in one step).

Step 10 : Find the minimum of all area values in the leaf nodes of the tree T_{11} (relevant information regarding the bounding edges may also be transferred in step 8 along with each A_i value).

4.0.4 Time Complexity

Each of steps 1, 5 and 8 needs constant time, while each of the remaining steps needs $O(\log n)$ time. Hence, we have the following result.

Theorem 2 *Algorithm SEB computes the smallest enclosing box of a given set of n points in $O(\log n)$ time.*

5 Conclusion

A number of efficient algorithms for computational geometry problems - finding the convex hull, the smallest enclosing box, the empirical cumulative distribution function and the all-nearest neighbor have been developed on OMULT architecture. Due to lack of space only the first two algorithms have been described here. All the algorithms uses n data elements and time complexity is only $O(\log n)$. It should be noted, however, that the number of processors involved in OMULT is $O(n^3)$ as opposed to OTIS-mesh which requires $O(n^2)$ processors. If we compare the performances of these algorithms implemented on the OMULT architecture with those implemented on the OTIS-Mesh, it is clear that the OMULT architecture performs better.

6 Acknowledgement

This research was supported by a grant from the Natural Science and Engineering Research council of Canada.

References

[1] H. Freeman and R. Shapiro,"Determining the minimal-area encasing rectangle for an arbitrary closed curve", Communications of ACM, 18:409 413, 1975.

[2] J-W. Jang, M. Nigam, V.K. Prasanna, S. Sahni, "Constant time algorithms for computational geometry on the reconfigurable mesh", IEEE Trans. on Parallel and Distr. Sys., Vol: 8, Iss: 1 , pp: 1 -12, Jan. 1997.

[3] A. Krishnamoorthy, P. Marchand, F. Kiamilev and S. Esener. "Grain-size considerations for optoelectronic multistage interconnection networks", Applied Optics, Vol. 31, No. 26, pp. 5480-5507, September 1992.

[4] G. C. Marsden, P. J. Marchand, P. Harvey and S. C. Esener. " Optical transpose interconnection system architectures" Optical Letters, Vol. 18, No. 13, pp. 1083-1085, July 1993.

[5] B.P Sinha and S. Bandyopadhyay, "OMULT: An Optical Interconnection System for Parallel Computing", presented at the EUROPAR'04, Pisa, Italy.

[6] C-F Wang and S. Sahni, "Computational geometry on the OTIS-Mesh opto-electronic computer", Proceedings International Conference on Parallel Processing, pp: 501 -507,18- 21 Aug. 2002.

[7] F. Zane, P. Marchand, R. Paturi and S. Esener, "Scalable network architectures using the optical transpose interconnection system (OTIS)", Journal of Parallel and Distributed Computing, Vol. 60, No. 5, pp. 521-538, 2000.

[8] F. P. Preparata and M. I. Shamos, "Computational geometry an introduction", Springer-Verlag, 1985.