Parallel Algorithms for All Maximal Equally-Spaced Collinear Sets and All Maximal Regular Coplanar Lattices *

Laurence Boxer † Russ Miller ‡

Abstract

The *All Maximal Equally-Spaced Collinear Subset (AMESCS) Problem* is defined as follows. Given a set P of n points in a Euclidean space E^d, find all maximal equally-spaced collinear subsets of P. An optimal $\Theta(n^2)$ time sequential solution to the problem is given in [K&R91].

A related problem is the *All Maximal Regularly-Spaced Subsets (AMRSS) Problem*, defined as follows. Given a set P of n points in E^d, find all maximal regularly-spaced coplanar subsets of P. An optimal $O(n^3)$ time sequential solution to the problem is given in [K&R91].

In this paper, we consider parallel solutions to the AMESCS and AMRSS Problems. Optimal sequential running times of $O(n^2)$ and $O(n^3)$, respectively, make parallel solutions of these problems desirable for large n. While the optimal sequential algorithms of [K&R91] are dominated by repetitions of a procedure that requires $\Theta(n)$ time per repetition, and appears inherently sequential, the algorithms we give are quite different. Our parallel algorithms solve both of these problems to within a logarithmic factor of optimality on the Arbitrary CRCW PRAM, and in optimal time on the mesh-connected computer.

1 Introduction

Let P be a subset of a Euclidean space E^d. We say $P' \subset P$ is *collinear* if $|P'| > 2$ and there is a line in E^d that contains all members of P'. A *maximal collinear subset (MCS)* of P is a collinear subset of P not properly contained in any other collinear subset of P. Several papers, including [D&H72, Riss89, B&S90] make use of the Hough transform to solve the MCS problem.

However, we are sometimes interested in the *spatial regularity* of collinear points, *e.g.*, we may require collinear points to be equally spaced. Kahng and Robins [K&R91] state:

> ...regularity is in many cases the distinguishing characteristic of 'interesting' regions in an image. A motivating (military) application ...is the examination of infrared ground

* *Pattern Recognition Letters* 14 (1993), 17-22

† Department of Computer Science, State University of New York - Buffalo, Buffalo, New York 14260, USA. Permanent address: Department of Computer and Information Sciences, Niagara University, Niagara University, NY 14109, USA. Research partially supported by a grant from the Niagara University Research Council.

‡ Department of Computer Science, State University of New York - Buffalo, Buffalo, New York 14260, USA. Research partially supported by NSF grant IRI-9108288.

surveillance bitmaps to find equally-spaced collinear 'hotspots' (rows of surface landmines, fenceposts in a region perimeter, etc.)

The sequential algorithm to solve the *All Maximal Equally-Spaced Collinear Subsets (AMESCS) Problem* given in [K&R91] runs in optimal $\Theta(n^2)$ time, where $n = |P|$. However, the algorithm does not seem to be amenable to an efficient implementation on a (massively) parallel machine, due to the fact that it consists of $\Theta(n)$ repetitions of a $\Theta(n)$ time routine that appears inherently sequential. In this paper, we give an efficient PRAM solution and an optimal mesh-connected solution for the AMESCS Problem.

The *All Maximal Regularly Spaced Subsets (AMRSS) Problem* is a generalization of the AMESCS Problem in the sense that it is concerned with finding all maximal patterns with 2-dimensional equally-spaced regularity. We give the appropriate definitions in Section 4.1. The sequential algorithm to solve the AMRSS Problem given in [K&R91] runs in optimal $\Theta(n^3)$ time, where $n = |P|$. As is the case of the AMESCS algorithm of [K&R91], the AMRSS algorithm does not seem to be amenable to an efficient implementation on a (massively) parallel machine, as it includes $\Theta(n^2)$ applications of a $\Theta(n)$ time routine similar to the one mentioned above. In this paper, we give an efficient PRAM solution and an optimal mesh-connected solution for the AMRSS Problem.

2 Preliminaries

In this section, we define the models of computation, as well as the assumptions about the input, used in this paper. Note that the terms *processor* and *processing element (PE)* will be used interchangeably throughout the paper.

2.1 PRAM Models of Computation

A parallel random access machine (PRAM) is a computer with multiple processors that share memory. An *Arbitrary CRCW PRAM* permits multiple processors to read simultaneously from the same memory location. Further, if multiple processors simultaneously attempt to write to the same memory location, one of the processors of the Arbitrary CRCW PRAM will (arbitrarily) succeed. Another model we mention briefly is the *CREW PRAM*, which permits multiple processors to read simultaneously from the same memory location, but which does not permit simultaneous writes to the same memory location.

2.2 Mesh Computer

A *2-dimensional mesh-connected computer (mesh)* is a computer with multiple processors organized as a square lattice of processors. Each generic processor shares a bidirectional communication link with each adjacent processor in its row and in its column. A *mesh of size N* has N processors arranged as an $N^{1/2} \times N^{1/2}$ lattice.

A mesh of size N has communication diameter $\Theta(N^{1/2})$, meaning that the maximum number of communication links separating any pair of PEs in the mesh is $\Theta(N^{1/2})$. Therefore, if a problem on a mesh of size N requires the possibility of two processors that are $\Theta(N^{1/2})$ communication links apart to communicate, and the running time of an algorithm to solve the problem is $O(N^{1/2})$, then the algorithm is optimal.

The processors are labeled so that $PE(i,j)$ is the PE in row i, column j, where $0 \leq i \leq N^{1/2}-1$, $0 \leq j \leq N^{1/2}-1$.

2.3 Input Assumptions

We assume input to the problem consists of a set P of n points, $P = \{x_0, x_1, \ldots, x_{n-1}\} \subset E^d$, each described as a d-tuple of real numbers:

$$x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d}).$$

Since there are $O(n^2)$ maximal equally-spaced collinear subsets of the input set [K&R91], we will assume our parallel computers have n^2 processors for the AMESCS Problem. Since there are $O(n^3)$ maximal regularly-spaced coplanar subsets of the input set [K&R91], we will assume our parallel computers have n^3 processors for the AMRSS Problem.

When we are discussing the mesh computer, we will assume the n input points are distributed one per PE in the first n PEs of the top row of the mesh.

We say x_i *precedes* x_j if $x_{i,1} < x_{j,1}$, or if there is an integer $k \in \{2, 3, \ldots, d\}$ such that $x_{i,p} = x_{j,p}$ for $p < k$ and $x_{i,k} < x_{j,k}$.

3 Parallel AMESCS Algorithm

In this section, we give efficient PRAM and optimal mesh solutions to the AMESCS Problem. We use the term "PRAM" to mean an "Arbitrary CRCW PRAM." Our algorithm is an implementation of the "naive" algorithm of [K&R91]. The algorithm is as follows.

1. In parallel, each of $\Theta(n^2)$ PEs determines a unique ordered pair of distinct data points in P such that the first member of the pair precedes the second. This may be done by the PRAM in $\Theta(1)$ time.

   The mesh accomplishes this as follows:

   - In parallel, $PE(0,j), 0 \leq j \leq n-1$, broadcasts its data point down its column to $PE(i,j), 0 \leq i \leq j$. This takes $\Theta(n)$ time.
   - In parallel, $PE(i,i), 0 \leq i \leq n-1$, broadcasts its data point across its row to $PE(i,j), i \leq j \leq n-1$. Now $PE(i,j)$ has the set of points $\{x_i, x_j\}$, for $0 \leq i < j \leq n-1$. This takes $\Theta(n)$ time.

- In $\Theta(1)$ additional time, every PE with a pair can determine the precedence order to form the desired ordered pair.

In order to simplify exposition, we assume for the remainder of the algorithm that the ordered pair formed from the set $\{x_i, x_j\}$, $i < j$, is (x_i, x_j).

2. In parallel, every PE with an ordered pair (x_i, x_j) of distinct points determines a third point $z_{(i,j)}$ such that $(x_i, x_j, z_{(i,j)})$ is an equally spaced collinear triple with the second member of the ordered pair preceding $z_{(i,j)}$. This is done in $\Theta(1)$ time.

3. In parallel, every PE with a pair (x_i, x_j) searches through the members of P to determine whether $z_{(i,j)} \in P$.

   This is done on the PRAM by first sorting $\{x_i\}$ with respect to the precedence order discussed above, and then allowing every processor to conduct a binary search for its query point. Sorting on the PRAM can be performed in $O(\log n)$ time [Cole88], as can a binary search.

   On the mesh, a $\Theta(n)$ time random access read operation [M&S89] will suffice.

4. In parallel, if $z_{(i,j)} = x_k \in P$, $PE(i,j)$ creates an edge of a graph G whose vertices represent the pairs of indices (i,j) and (j,k). This requires $\Theta(1)$ time. It is noted in [K&R91] that vertices representing the pairs (i,j) and (p,q) belong to the same component of G if and only if (x_i, x_j) and (x_p, x_q) are ordered pairs of consecutive members of the same maximal equally-spaced collinear subset of P.

5. Label the components of G. Notice that every component of G corresponds to a maximal equally-spaced collinear subset of P. Component labeling requires $O(\log n)$ time on the PRAM [Sh&V82], and $\Theta(n)$ time on the mesh [M&S89].

Therefore, on the Arbitrary CRCW PRAM, the algorithm given above requires $\Theta(\log n)$ time. Since we are using n^2 PEs, the total work is $\Theta(n^2 \log n)$, which is within a factor of $\Theta(\log n)$ of the optimal sequential bound.

On the mesh, the algorithm requires $\Theta(n)$ time, which is optimal for the mesh since the communication diameter of the mesh is $\Theta(n)$.

4 Parallel AMRSS Algorithm

In this section, we give efficient PRAM and optimal mesh solutions to the AMRSS Problem. In Section 4.1, we state appropriate terminology. In Section 4.2, we give our algorithm.

4.1 Terminology

We first present some terminology from [K&R91]. Let A, B, and C be vectors in E^d, with A and B linearly independent. A *lattice* $L(A, B, C)$ is the set

$$L(A, B, C) = \{V | V = jA + kB + C, \text{ for all integers } j, k\}.$$

The lattice point V has *lattice coordinates* (j, k). A finite coplanar set L' is a *sublattice* of $L(A, B, C)$ if $L' \subset L$. Four points of a lattice $L(A, B, C)$ define a *cell* of the lattice if their lattice coordinates are of the form

$$\{(j, k), (j + 1, k), (j, k + 1), (j + 1, k + 1)\};$$

we then say the cell has *cell coordinates* (j, k). Thus, four points define a cell if and only if they are the vertices of a parallelogram.

Two cells p and q of a given lattice are *neighbors* if their respective cell coordinates (j_1, k_1) and (j_2, k_2) satisfy

$$|j_1 - j_2| + |k_1 - k_2| = 1.$$

The neighbor relation over the cells of a sublattice induces a graph structure. Given a fixed lattice L, the *cell graph* of a sublattice $L' \subset L$ is defined by $G(L') = (V, E)$, where

$$V = \{u | u \text{ is a cell in } L'\},$$

and

$$E = \{\overline{uv} | u \text{ and } v \text{ are neighboring cells in } L'\}.$$

A set of points $L' \subset E^2$ is *regularly-spaced* with respect to some fixed lattice L if

1. L' is a sublattice of L;
2. every point of L' belongs to some cell of L'; and
3. the cell graph of L' is connected.

A regularly-spaced subset is *maximal* if it is not a proper subset of any other regularly-spaced subset. Thus, the AMRSS Problem is defined as follows. Given a finite pointset $P \subset E^d$, find all its maximal regularly-spaced coplanar subsets.

4.2 Parallel AMRSS Algorithm for E^2

In this section, we assume our parallel computers have n^3 PEs. As above, we use "PRAM" in this section to mean "Arbitrary CRCW PRAM". We note that a mesh of n^3 PEs requires $\Theta(n^{3/2})$ time to perform operations involving global communication. A parallel algorithm for the AMRSS Problem in E^2 follows.

1. In parallel, each of $\Theta(n^3)$ PEs determines a unique ordered triple (x_i, x_j, x_k) of data points such that $i < j < k$. This may be done in $\Theta(1)$ time by the PRAM. The mesh can achieve this step by 3 $\Theta(n^{3/2})$ time random access read operations.

2. In $\Theta(1)$ time, each PE with a triple (x_i, x_j, x_k) determines whether x_i, x_j, and x_k are collinear, and, if not, determines the three points $\{z_{(i,j,k),m}\}_{m=1}^{3}$ such that, for each value of m, the set $\{x_i, x_j, x_k, z_{(i,j,k),m}\}$ defines a lattice cell.

3. Each PE with a non-collinear triple searches P to determine which, if any, of the points $\{z_{(i,j,k),m}\}_{m=1}^{3}$ belong to P.

   The PRAM does this as follows.

   - Sort P by precedence order in $\Theta(\log n)$ time.
   - In parallel, each PE with a non-collinear triple conducts 3 binary searches on its query points in $\Theta(\log n)$ time.

   The mesh does this by 3 $\Theta(n^{3/2})$ time random access read operations.

4. Each PE with a triple now knows that it has between 0 and 3 lattice cells. In $\Theta(1)$ time, represent each such cell by a $5-$tuple consisting of its leading (in precedence order) vertex, the lesser slope of its pairs of parallel segments, the length of that pair of segments, the greater slope of its pairs of parallel segments, and the length of that pair of segments.

5. Determine for each of the $O(n^3)$ cells u, the cells u_1, u_2, u_3, u_4 that are neighbors of u. This is done in $\Theta(1)$ time.

6. In parallel, every PE with a cell determines for each of its neighbor cells whether the neighbor cell belongs to the array U.

   This is done on the PRAM as follows.

   - Sort the cells with respect to the members of the $5-$tuples created above, in decreasing order of priority. The sorted cells are now in an array U. This requires $\Theta(n)$ time on the PRAM.
   - Every processor with a cell conducts a binary search through U for each of its query cells. This requires $\Theta(\log n)$ time.

   On the mesh, 4 $\Theta(n^{3/2})$ time random access read operations will suffice.

   Each time we find that a cell u has a neighbor cell $v \in U$ with $u < v$ in the order of U, create an edge $\overline{uv}$ of a graph G. The creation of edges requires $\Theta(1)$ time. As above, components of G correspond to maximal regularly-spaced subsets of P [K&R91].

7. Label the components of G. Notice that every component of G corresponds to a maximal regularly-spaced subset of P. Component labeling requires $O(\log n)$ time on the PRAM and $\Theta(n^{3/2})$ time on the mesh.

Therefore, on the Arbitrary CRCW PRAM, the algorithm given above requires $\Theta(\log n)$ time. Since we are using n^3 PEs, the total work is $\Theta(n^3 \log n)$, which is within a factor of $\Theta(\log n)$ of the optimal sequential bound.

On the mesh, the algorithm requires $\Theta(n^{3/2})$ time, which is optimal for the mesh since the communication diameter of the mesh is $\Theta(n^{3/2})$.

4.3 Extension to Higher Dimensional Spaces

As discussed in [K&R91], our AMRSS algorithm can be extended to higher dimensional Euclidean spaces, using the same time and processor resources, as follows. Project the input onto a plane and solve the AMRSS problem in 2 dimensions for the projected pointset. This must be done with some care, due to the following problems.

- Although congruent neighboring cells in E^d will project into the plane as congruent neighboring cells, there may be congruent cells that are not neighbors in E^d whose planar projections are neighbors; however, for both the PRAM and the mesh, checking for such spurious neighbors requires only $\Theta(1)$ additional time for any fixed dimension d.

- Three non-collinear points in E^d, $d > 2$, may have collinear projections in the plane determined by a fixed pair of coordinates. However, if $S = \{x_i, x_j, x_k\}$ is a non-collinear subset of E^d, then there is a pair of coordinate axes that determines a plane $\mathcal{P}$ such that the projection of S into $\mathcal{P}$ is not collinear. Thus, the collection of lattice cells with vertices in P can be constructed from the projections of P into all of the planes determined by pairs of coordinate axes. This requires several steps of the algorithm to be performed for each of the $\binom{d}{2}$ pairs of coordinate axes. Thus for any fixed d, our analysis of the algorithm's running time is not changed.

4.4 Extension to Higher Dimensional Lattices

We observe that the results of [K&R91] and the current paper may be generalized easily to lattices of higher dimensions. Let $A_0, A_1, \ldots, A_D$ be vectors in E^d such that $D \leq d$ and such that $A_1, \ldots, A_D$ are linearly independent. A *lattice* $L(\{A_i\}_{i=0}^D)$ *of dimension* D is the set

$$L(\{A_i\}_{i=0}^D) = \{V \in E^d | V = A_0 + \Sigma_{i=1}^D \, j_i A_i, \text{ for all integers } j_1, j_2, \ldots, j_D\}.$$

We may define terms such as *cell, neighbor,* and *regularly-spaced* by obvious generalizations of the analogous terms of Section 4.1. The *All Maximal Regularly-Spaced* $D-$*dimensional Lattice Subsets*

(AMRSDLS) Problem is as follows. Given a finite pointset $P \subset E^d$, find all its maximal regularly-spaced $D-$dimensional lattice subsets. We then obtain the following.

Theorem 4.1 *Let D and d be fixed positive integers with $D \leq d$. Let P be a set of n points in E^d. Then the All Maximal Regularly-Spaced $D-$dimensional Lattice Subsets Problem can be solved sequentially in optimal $O(n^{D+1})$ time; on an Arbitrary CRCW PRAM with n^{D+1} PEs in $\Theta(\log n)$ time; and on a mesh with n^{D+1} PEs in optimal $\Theta(n^{\frac{D+1}{2}})$ time.*

Proofs of the claims of Theorem 4.1 are omitted, as the sequential algorithm is a minor generalization of the AMRSS algorithm of [K&R91], and the parallel solutions are minor generalizations of the AMRSS solutions of Section 4.2.

5 Further Remarks

In this paper, we have presented parallel solutions to the AMESCS and the AMRSS Problems, and we have shown how our solutions to the latter generalize to the AMRSDLS Problem. Our algorithms differ significantly from the optimal sequential algorithms presented in [K&R91], which do not scale well to (massively) parallel machines. The optimality of our Arbitrary CRCW PRAM algorithms is open; however, the algorithms we present are within a logarithmic factor of optimal. Further, our algorithms are optimal for the mesh-connected computer.

Finally, it is known that a CREW PRAM can simulate an algorithm for an Arbitrary CRCW PRAM using the same number of PEs with a logarithmic delay. This raises the following questions:

1. Can a CREW PRAM with n^2 PEs solve the AMESCS Problem in $o(\log^2 n)$ time?
2. Can a CREW PRAM with n^3 PEs solve the AMRSS Problem in $o(\log^2 n)$ time?

References

[B&S90] Ben-Tzvi, D. and M.B. Sandler, A combinatorial Hough transform, *Pattern Recognition Letters* 11 (1990), 167-174.

[Cole88] R. Cole, Parallel merge sort, *SIAM J. Comput.* 17(4) (1988), 770-785.

[D&H72] Duda, R. and P. Hart, Use of the Hough transform to detect lines and curves in pictures, *Comm. ACM* 15(1) (1972), 11-15.

[K&R91] Kahng, A.B. and G. Robins, Optimal algorithms for extracting spatial regularity in images, *Pattern Recognition Letters* 12 (1991), 757-764.

[M&S89] Miller, R. and Q.F. Stout, Mesh computer algorithms for computational geometry, *IEEE Transactions on Computers* 38(3) (1989), 321-340.

[Riss89] Risse, T., Hough transform for line recognition: complexity of evidence accumulation and cluster detection, *Computer Vision* 46 (1989), 327-345.

[Sh&V82] Shiloach, Y. and U. Vishkin, An $O(\log n)$ parallel connectivity algorithm, *Journal of Algorithms* 3 (1982), 57-67.