A Parallel Algorithm for Approximate Regularity [*]

Laurence Boxer [†] Russ Miller [‡]

Abstract

Spatial regularity amidst a seemingly chaotic image is often meaningful. Many papers in computational geometry are concerned with detecting some type of regularity via exact solutions to problems in geometric pattern recognition. However, real-world applications often have data that is approximate, and may rely on calculations that are approximate. Thus, it is useful to develop solutions that have an error tolerance.

A solution has recently been presented [7] to the problem of finding all maximal subsets of an input set in the Euclidean plane R^2 that are approximately equally-spaced and approximately collinear. This is a problem that arises in computer vision, military applications, and other areas. The algorithm of [7] is different in several important respects from the optimal algorithm given in [3] for the exact version of the problem. The algorithm of [7] seems inherently sequential and runs in $O(n^{5/2})$ time, where n is the size of the input set. In this paper, we give parallel solutions to this problem.

Key words and phrases: parallel algorithm, computational geometry, equally-spaced collinear sequence, parallel random access machine (PRAM), mesh

1 Introduction

The literature of computational geometry contains many papers concerned with exact solutions to problems involving geometric data. However, real-world problems generally have data with measurement errors; further, computations required for solutions may have truncation and/or roundoff errors. It is therefore desirable to develop efficient solutions that can accommodate such errors. Allowing for an error tolerance may have a profound effect on the nature of the solution, as is seen in the problem that is the subject of this paper.

[*] *Information Processing Letters* 80 (2001), 311-316

[†] Niagara University and State University of New York - Buffalo. Department of Computer and Information Sciences, Niagara University, Niagara University, NY 14109, USA. Paper was written during this author's sabbatical visit to the Center for Computational Research, State University of New York - Buffalo. 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 ACIR-9721373.

We seek to identify all maximal ε-*regular* subsets of a finite input set $S \subset R^2$. Roughly, a set is regular if it is collinear and its members are equally-spaced; it is ε-regular if it is approximately regular, within an error tolerance $\varepsilon > 0$. Precise definitions are presented in section 2.

An optimal sequential solution to the problem of finding all regular subsets of S was given in [3]. This algorithm runs in $O(n^2)$ time. An efficient parallel solution, using a significantly different algorithm, was given in [1] (see also [5]).

The ε-regular version of the problem requires a different approach. For example, it was shown in [7] that the worst-case volume of output for this problem is $\Omega(n^2 \log n)$ and $O(n^{5/2})$, while for the regular (exact) version of the problem, the worst-case volume of output is $\Theta(n^2)$. Roughly, this is due to the fact that a pair of points appears in at most one maximal regular subset of S, but may occur in many ε-regular maximal subsets (see Figure 4 of [7] for an illustration). The $O(n^{5/2})$ running time of the algorithm presented in [7] is linear in the volume of a set of intermediate results, the so-called "p-initiated sequences." The upper bound on the volume of p-initiated sequences is an upper bound (possibly not tight) on the output of the algorithm.

In this paper, we give a parallel solution to the problem discussed above. Our solution runs on the CREW PRAM in $O(n \log n)$ time using n^2 processors and $O(n^{5/2})$ memory. This yields a cost of $O(n^3 \log n)$, which seems inefficient compared to the result of [7]. Nevertheless, our result is of interest, as the algorithm of [7] appears inherently sequential, yet its slow (even if optimal) running time seems to beg for a parallel solution. We also show that our algorithm runs on a mesh of size $\Theta(n^{5/2})$ in $O(n^{5/4} \log n)$ time, which is within a factor of $\log n$ of being optimal for this architecture.

2 Preliminaries

In this section, we define the model of computation, as well as the assumptions about the input, used in this paper. We define terminology, state our problem, and discuss linear programming as a tool for solving our problem.

2.1 CREW PRAM Model of Computation

A parallel random access machine (PRAM) is a computer with multiple processors that share memory. The concurrent read, exclusive write (CREW) PRAM, permits multiple processors to read simultaneously from the same memory location, but 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 processors 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.

2.3 Terminology

A set of points P in a Euclidean space R^d is *collinear* if there is a line $L \subset R^d$ such that $P \subset L$. A finite set $P = \{p_i\}_{i=1}^k$ is *equally spaced* if the elements of P are indexed such that the vector difference $p_{i+1} - p_i$ has constant length for $1 \leq i < k$. We say P is *regular* if $|P| > 2$, P is collinear, and P is equally-spaced.

We say $P = \{p_i = (x_i, y_i)\}_{i=1}^k \subset R^2$ is ε*-regular* if for all i there are points $p'_i = (x'_i, y'_i) \in R^2$ such that $|x_i - x'_i| \leq \varepsilon$, $|y_i - y'_i| \leq \varepsilon$, and the set $P' = \{p'_i\}_{i=1}^k$ is regular. A *maximal* ε-regular sequence is an ε-regular sequence that is not properly contained as a contiguous subsequence in any other ε-regular sequence.

It was noted in [7] that these definitions may result in "redundant" sequence detection, in the sense that, *e.g.*, if (a, b, c, d, e) is an ε-regular sequence, then so is (a, c, e). However, [7] goes on to point out that

- these definitions, permitting such "redundant" sequences to be considered maximal, do not raise the asymptotic cost of the algorithm; and

- this approach is consistent with the application of landmine detection, in which false positives are preferable to false negatives.

2.4 Input Assumptions

Input to our problem consists of a set S of $n \geq 3$ points in R^2, and a number $\varepsilon > 0$ that serves as an error tolerance. We will use the following.

Assumption 2.1 *The minimal (Euclidean) distance between distinct members of S is greater than 8ε.*

It was shown in [7] that this assumption, together with the following Lemma, enables us to conduct binary searches for members of S such that each search either fails or returns a unique point.

Lemma 2.2 [7] *Let (x_1, y_2) and (x_2, y_2) be the first two points of an ε-regular triple. Then the third point (x_3, y_3) of the triple satisfies*

$$2x_2 - x_1 - 4\varepsilon \leq x_3 \leq 2x_2 - x_1 + 4\varepsilon,$$

$$2y_2 - y_1 - 4\varepsilon \leq y_3 \leq 2y_2 - y_1 + 4\varepsilon. \blacksquare$$

2.5 Our problem

The problem we wish to solve may now be stated as follows: Given a set $S \subset R^2$ consisting of n points and an error tolerance $\varepsilon > 0$ satisfying Assumption 2.1, find *all* maximal ε-regular subsets of S.

It was noted in [7] that Assumption 2.1 precludes the possibility of exponential-sized output. For example, consider

$$S = \bigcup_{i=1}^{n/2} \{(i, \varepsilon/2), (i, -\varepsilon/2)\}$$

for n even. This is a set for which Assumption 2.1 is not valid. Then there are $2^{n/2}$ distinct maximal ε-regular subsets of S of the form $\{a_i\}_{i=1}^{n/2}$, with $a_i \in \{(i, \varepsilon/2), (i, -\varepsilon/2)\}$, each ε-approximated by the regular set $\{(i, 0)\}_{i=1}^{n/2}$.

2.6 Linear Programming

Suppose that we have an ε-regular sequence $(p_1, p_2, \ldots, p_k)$ and an ε-regular triple containing the last pair, (p_{k-1}, p_k, p_{k+1}). It does not automatically follow that $(p_1, p_2, \ldots, p_k, p_{k+1})$ is ε-regular. That is, finding a point $p_{k+1} \in S$ such that (p_{k-1}, p_k, p_{k+1}) is an ε-regular triple, is a necessary but insufficient condition to extending $(p_1, p_2, \ldots, p_k)$ as an ε-regular sequence.

For a sequence $(p_1, p_2, \ldots, p_m)$ to be ε-regular, there must be a regular sequence $(p'_1, p'_2, \ldots, p'_m)$ such that for each index i, p_i and p'_i are separated in each coordinate by at most ε. Thus, there are unknowns d_x, d_y, p'_{1_x}, p'_{1_y} such that for all $i \in \{1, \ldots, m\}$,

$$p'_{1_x} \ + \ (i-1)d_x \ - \ \varepsilon \leq p_{i_x} \leq p'_{1_x} \ + \ (i-1)d_x \ + \ \varepsilon,$$

$$p'_{1_y} \ + \ (i-1)d_y \ - \ \varepsilon \leq p_{i_y} \leq p'_{1_y} \ + \ (i-1)d_y \ + \ \varepsilon,$$

where the x and y subscripts refer, respectively, to the x and y coordinates. As noted in [7], it may be determined whether or not these conditions are all satisfied by solving a linear programming (LP) problem in the four unknowns d_x, d_y, p'_{1_x}, p'_{1_y}.

Theorem 2.3 [4] *An LP problem in a fixed number of unknowns can be solved in time linear in the number of constraints.* ■

Another tool for determining whether a sequence is ε-regular is to solve a problem related to linear programming, that of describing an intersection of half-planes. We have the following.

Theorem 2.4 [6] *Given the description of n or fewer half-planes distributed one per processor on a mesh of size n, their intersection can be computed in $\Theta(n^{1/2})$ time.* ■

3 Our algorithm

We prove the following.

Theorem 3.1 *Let S be a set of n points in R^2 satisfying Assumption 2.1 for a given $\varepsilon > 0$. Then all maximal ε-regular subsets of S may be identified using the following resources:*

- $O(n \log n)$ *time on a CREW PRAM of* n^2 *processors and* $\Theta(n^{5/2})$ *memory.*
- $O(n^{5/4} \log n)$ *time on a mesh of* $\Theta(n^{5/2})$ *processors.*

Proof: We remark that in the algorithm that follows, the casual reader may find it strange that determining whether a sequence is ε-regular is faster on the mesh than on the PRAM. This is explained by the fact that we deploy parallelism differently for these models of computation. Our PRAM implementation is medium-grained, while our mesh implementation is fine-grained. On the PRAM, one processor will solve a linear programming problem of non-constant size, while on the mesh, $\Theta(m)$ processors will describe the intersection of m half-planes. Thus, the time required by the PRAM for this portion of the algorithm is the time for a sequential algorithm; but the time required by the mesh for this portion of the algorithm is the time for a fine-grained parallel algorithm.

For the mesh, we use the snake-like order of processors [6, 5]. Also for the mesh, we assume S is initially distributed one point per processor in an $n^{1/2} \times n^{1/2}$ block of processors. Our algorithm follows.

1. Let every processor have the value of ε. Since we assume a CR PRAM, every PRAM processor can read this value in $\Theta(1)$ time. For the mesh, broadcast the value of ε to all processors. This takes $\Theta(n^{5/4})$ time [6, 5].

2. Sort the members of S as in [7]. That is, we conceptually partition the plane into a grid in which each square has edges of length 8ε. For each $s \in S$, we compute the coordinates of the lower left corner of the grid square containing s and store these grid coordinates as part of a record that also contains the coordinates of s. Sort the members of S with respect to the lexicographic order of the grid coordinates in $\Theta(\log n)$ time for the PRAM [2]. For the mesh, sort S in its $n^{1/2} \times n^{1/2}$ block of processors in $\Theta(n^{1/2})$ time [6, 5].

3. For the mesh, we may think of the processors as being partitioned into $\Theta(n^{3/2})$ distinct blocks of $n^{1/2} \times n^{1/2}$ processors apiece. Use parallel row and column rotations so that each such block gets a copy (now sorted) of S. This takes $\Theta(n^{5/4})$ time.

4. We can map the processors to ordered pairs of members of the input set S so that each processor is associated with at most $\Theta(1)$ such pairs. For the PRAM, this takes $\Theta(1)$ time,

since we assume the CR property. For the mesh, we can create these pairs by using random access write operations so that the processor in row i and column j gets the pair (s_i, s_j), $1 \le i \le n$, $1 \le j \le n$, in $O(n^{5/4})$ time [6, 5].

5. Associate with each pair created above, a record containing the pair and the *count* of the sequence that the pair begins. Initialize the *count* as 2. This takes $\Theta(1)$ time.

6. Determine the successor pair (s_j, s_k) of each pair (s_i, s_j), if the successor pair exists, where the pair (s_j, s_k) is the *successor* of (s_i, s_j) if the triple (s_i, s_j, s_k) is ε-regular. This is done as follows.

   - Compute the grid coordinates of the desired point, using the inequalities required of the third point in Lemma 2.2. This takes $\Theta(1)$ time.
   - Conduct a search for the unique (if it exists) $s_k \in S$ that has these grid coordinates. The processors of the PRAM can use parallel binary searches to perform this step in $O(\log n)$ time. The mesh can perform the search by a $\Theta(n^{1/2})$ time random access read operation within its local $n^{1/2} \times n^{1/2}$ block that contains a copy of S [6, 5].
   - Solve the appropriate LP or half-plane intersection problem in $\Theta(1)$ time to determine whether or not the triple is ε-regular.
   - If the triple is ε-regular, update the *count* associated with the triple's initial pair to 3. This takes $\Theta(1)$ time.

7. For each ε-regular set detected so far, we do recursive doubling to extend the ε-regular set until it is no longer possible to do so. On the mesh, as we do so, we also compactify the data so that each initial pair "controls" enough processors to store its sequence so that at most $\Theta(1)$ sequence points per processor are in any processor of the mesh. This is done as follows. In each iteration of the loop, for every ordered ε-regular set $(s_{i_1}, s_{i_2}, \ldots, s_{i_{k-1}}, s_{i_k})$ not yet marked as stopped or maximal (with respect to those that start (s_{i_1}, s_{i_2})), do the following.

   - Find the ordered ε-regular set begun by the successor pair (if it exists) $(s_{i_k}, s_{i_{k+1}})$ of the last pair $(s_{i_{k-1}}, s_{i_k})$ of the sequence. On the PRAM, this can be done in $\Theta(1)$ time. On the mesh, this requires a copy of the ε-regular set begun by the successor pair (if it exists) to be sent to the block of processors currently storing the sequence with initial pair (s_{i_1}, s_{i_2}). This can be done by a random access read operation in $\Theta(n^{5/4})$ time. If no successor exists, mark the sequence as maximal (among those that start with the first pair of the sequence).

- For the mesh, use a parallel prefix step to redistribute the data so that the union of the two ε-regular sequences discussed in the previous step is stored in the mesh at most $\Theta(1)$ sequence entries per processor in contiguous processors. This takes $\Theta(n^{5/4})$ time [6, 5].
- Suppose the ε-regular set found above is $(s_{i_k}, s_{i_{k+1}}, \ldots, s_{i_m})$. Solve an LP problem (PRAM) or half-plane intersection problem (mesh) to determine if

$$(s_{i_1}, s_{i_2}, \ldots, s_{i_{k-1}}, s_{i_k}, s_{i_{k+1}}, s_{i_{k+2}}, \ldots, s_{i_m})$$

  is an ε-regular set. If so, update the *count* of the ε-regular sequence starting with (s_{i_1}, s_{i_2}). This takes $\Theta(m)$ time on the PRAM (by Theorem 2.3) and $\Theta(m^{1/2})$ time on the mesh (by Theorem 2.4), since snake-like order is used.
- If the concatenated set formed above is not ε-regular, mark the sequence $(s_{i_1}, s_{i_2}, \ldots, s_{i_k})$ as *stopped* in $\Theta(1)$ time.
- The sequence starting with a given initial pair is no longer processed by this loop when the sequence is marked either as *stopped* or as maximal among those that start with the first pair of the sequence. Thus, the parallel computer can test whether or not to exit this loop by a semigroup (AND) operation in $\Theta(\log n)$ time for the PRAM, and in $O(n^{5/4})$ time for the mesh.

The recursive doubling steps build up a sequence of *count* $O(n)$ in a geometric progression of (sizes of) LP or half-plane intersection steps. Therefore, they require $O(n)$ time on the PRAM, $O(n^{5/4})$ time on the mesh [6, 5]. Since the loop has $O(\log n)$ iterations, it requires altogether $O(n \log n)$ time for the PRAM and $O(n^{5/4} \log n)$ time for the mesh.

8. For each sequence $(s_{i_1}, s_{i_2}, \ldots, s_{i_{k-1}}, s_{i_k})$ marked as *stopped* when we attempted to extend the sequence with $(s_{i_{k+1}}, s_{i_{k+2}}, \ldots, s_{i_m})$, we must determine how much of $(s_{i_{k+1}}, s_{i_{k+2}}, \ldots, s_{i_m})$ can be concatenated to $(s_{i_1}, s_{i_2}, \ldots, s_{i_{k-1}}, s_{i_k})$ while preserving ε-regularity. This is done by

applying a binary search technique to $(s_{i_{k+1}}, s_{i_{k+2}}, \ldots, s_{i_m})$, during each stage of which we solve an LP problem (for the PRAM) or a half-plane intersection problem (for the mesh) to determine if the corresponding concatenated sequence is ε-regular. Since the binary search technique requires $O(\log n)$ iterations, it follows from Theorem 2.3 that this takes $O(n \log n)$ time for the PRAM, and from Theorem 2.4 that this takes $O(n^{5/4} \log n)$ time on the mesh. In an additional $\Theta(1)$ time, mark the resulting sequence as maximal (among those starting (s_{i_1}, s_{i_2})) and update its *count*.

9. The previous steps have constructed, for each pair (s_i, s_j), the maximal ε-regular set among those that start with (s_i, s_j). In the terminology of [7], these are *maximal s_i-initiated sequences.* We may determine which of these are maximal ε-regular sets as follows.

    - Each (s_i, s_j) finds the record of its opposite-directed pair, (s_j, s_i). Suppose the successor of (s_j, s_i) is (s_i, s_u). Then (s_i, s_j) is the successor of (s_u, s_i). Since the pairs have been ordered, this takes $O(\log n)$ time for the PRAM. For the mesh, this operation can be done via a random access read in $\Theta(n^{5/4})$ time.
    - If the successor (s_i, s_u) of (s_j, s_i) exists and the maximal ε-regular set among those that start with (s_u, s_i) has *count* greater than the *count* of the maximal ε-regular set among those that start with (s_i, s_j), then the latter is not maximal; otherwise, it is maximal if and only if the *count* of the sequence is at least 3. This may be determined from the above in $\Theta(1)$ time.

10. Now we have $\Theta(n^2)$ sequences, some of which are marked as maximal ε-regular sets, the rest of which are marked as not maximal ε-regular sets. We may use a parallel prefix operation to initialize an array X of pointers so that X_i points to the i^{th} maximal ε-regular set. This takes $\Theta(\log n)$ time for the PRAM, $O(n^{5/4})$ time for the mesh.

11. The same set may be indexed more than once by members of X (in reverse order). If desired, we may remove duplicate sets from a listing of the sets indexed by X as follows.

    - For each $X_i \in X$, create a quadruple (a, b, c, d) as follows. Suppose the sequence indexed by X_i is $(s_{i_1}, s_{i_2}, \ldots, s_{i_{k-1}}, s_{i_k})$. If, with respect to lexicographic order, $(s_{i_1}, s_{i_2}) <$

$(s_{i_k}, s_{i_{k-1}})$, let

$$(a,b,c,d) \;=\; (s_{i_1}, s_{i_2}, s_{i_k}, s_{i_{k-1}});$$

otherwise, let

$$(a,b,c,d) \;=\; (s_{i_k}, s_{i_{k-1}}, s_{i_1}, s_{i_2}).$$

This takes $\Theta(1)$ time on the PRAM. On the mesh, since each sequence is stored in a submesh of size n, it takes $O(n^{1/2})$ time.

- Sort X by the quadruples constructed above. This may be done in $\Theta(\log n)$ time on the PRAM, $O(n^{5/4})$ time on the mesh.
- Perform a parallel prefix on X that eliminates the sequence indexed by X_i if its quadruple coincides with the quadruple of the sequence indexed by X_{i-1}. This takes $\Theta(\log n)$ time for the PRAM, $O(n^{5/4})$ time for the mesh.

Thus, our algorithm takes $O(n \log n)$ time on a CREW PRAM of $\Theta(n^2)$ processors and $\Theta(n^{5/2})$ memory, and $O(n^{5/4} \log n)$ time on a mesh of $\Theta(n^{5/2})$ processors. ■

4 Further Remarks

In this paper, we have given a parallel algorithm to find all maximal ε-regular subsets of a finite planar set. Our algorithm is significantly different than that of [7], which seems inherently sequential. The algorithm of [7] runs in $O(n^{5/2})$ time. Our algorithm runs in $O(n \log n)$ time on a medium-grained CREW PRAM of $\Theta(n^2)$ processors and $\Theta(n^{5/2})$ memory, and in $O(n^{5/4} \log n)$ time on a fine-grained mesh of $\Theta(n^{5/2})$ processors. Note the latter is within a factor of $\log n$ of being optimal for the mesh.

5 Acknowledgment

We thank the anonymous referees for several corrections and suggestions.

References

[1] Boxer, L. and R. Miller, Parallel algorithms for all maximal equally-spaced collinear sets and all maximal regular coplanar lattices, *Pattern Recognition Letters* 14 (1993), 14-20.

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

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

[4] Megiddo, N., Linear programming in linear time when the dimension is fixed, *J. Assoc. Comput. Mach.* 31 (1984), 114-127.

[5] Miller, R. and L. Boxer, *Algorithms Sequential and Parallel*, Prentice Hall, 2000.

[6] Miller, R. and Q.F. Stout, *Parallel Algorithms for Regular Architectures: Meshes and Pyramids*, The MIT Press, Cambridge, MA, 1996.

[7] Robins, G., B.L. Robinson, and B.S. Sethi, On detecting spatial regularity in noisy images, *Information Processing Letters* 69 (1999), 189-195.