46 PARALLEL ALGORITHMS IN GEOMETRY

Michael T. Goodrich and Nodari Sitchinava

INTRODUCTION

The goal of parallel algorithm design is to develop parallel computational methods that run very fast with as few processors as possible, and there is an extensive literature of such algorithms for computational geometry problems. There are several different parallel computing models, and in order to maintain a focus in this chapter, we will describe results in the Parallel Random Access Machine (PRAM) model, which is a synchronous parallel machine model in which processors share a common memory address space (and all inter-processor communication takes place through this shared memory). Although it does not capture all aspects of parallel computing, it does model the essential properties of parallelism. Moreover, it is a widely accepted model of parallel computation, and all other reasonable models of parallel computation can easily simulate a PRAM.

Interestingly, parallel algorithms can have a direct impact on efficient sequential algorithms, using a technique called *parametric search*. This technique involves the use of a parallel algorithm to direct searches in a parameterized geometric space so as to find a critical location (e.g., where an important parameter changes sign or achieves a maximum or minimum value).

The PRAM model is subdivided into submodels based on how one wishes to handle concurrent memory access to the same location. The Exclusive-Read, Exclusive-Write (EREW) variant does not allow for concurrent access. The Concurrent-Read, Exclusive-Write (CREW) variant permits concurrent memory reads, but memory writes must be exclusive. Finally, the Concurrent-Read, Concurrent-Write (CRCW) variant allows for both concurrent memory reading and writing, with concurrent writes being resolved by some simple rule, such as having an arbitrary member of a collection of conflicting writes succeed. One can also define randomized versions of each of these models (e.g., an rCRCW PRAM), where in addition to the usual arithmetic and comparison operations, each processor can generate a random number from 1 to n in one step.

Early work in parallel computational geometry, in the way we define it here, began with the work of Chow [Cho80], who designed several parallel algorithms with polylogarithmic running times using a linear number of processors. Subsequent to this work, several researchers initiated a systematic study of work-efficient parallel algorithms for geometric problems, including Aggarwal *et al.* [ACG$^+$88], Akl [Akl82, Akl84, Akl85], Amato and Preparata [AP92, AP95], Atallah and Goodrich [AG86, Goo87], and Reif and Sen [RS92, Sen89].

In Section 46.1 we give a brief discussion of general techniques for parallel geometric algorithm design. We then partition the research in parallel computational geometry into problems dealing with convexity (Section 46.2), arrangements and decompositions (Section 46.3), proximity (Section 46.4), geometric searching (Section 46.5), and visibility, envelopes, and geometric optimization (Section 46.6).

46.1 SOME PARALLEL TECHNIQUES

The design of efficient parallel algorithms for computational geometry problems often depends upon the use of powerful general parallel techniques (e.g., see [AL93, Já92, KR90, Rei93]). We review some of these techniques below.

PARALLEL DIVIDE-AND-CONQUER

Possibly the most general technique is parallel divide-and-conquer. In applying this technique one divides a problem into two or more subproblems, solves the subproblems recursively in parallel, and then merges the subproblem solutions to solve the entire problem. As an example application of this technique, consider the problem of constructing the upper convex hull of a set S of n points in the plane presorted by x-coordinates. Divide the list S into $\lceil\sqrt{n}\rceil$ contiguous sublists of size $\lfloor\sqrt{n}\rfloor$ each and recursively construct the upper convex hull of the points in each list. Assign a processor to each pair of sublists and compute the common upper tangent line for the two upper convex hulls for these two lists, which can be done in $O(\log n)$ time using a well-known "binary search" computation [Ede87, O'R94, PS85]. By maximum computations on the left and right common tangents, respectively, for each subproblem S_i, one can determine which vertices on the upper convex hull of S_i belong to the upper convex hull of S. Compressing all the vertices identified to be on the upper convex hull of S constructs an array representation of this hull, completing the construction.

The running time of this method is characterized by the recurrence relation $T(n) \leq T(\sqrt{n}) + O(\log n)$, which implies that $T(n)$ is $O(\log n)$. It is important to note that the coefficient for the $T(\sqrt{n})$ term is 1 even though we had $\lceil\sqrt{n}\rceil$ subproblems, for all these subproblems were processed simultaneously in parallel. The number of processors needed for this computation can be characterized by the recurrence relation $P(n) = \lceil\sqrt{n}\rceil P(\sqrt{n}) + O(1)$, which implies that $P(n) = O(n)$. Thus, the *work* needed for this computation is $O(n \log n)$, which is not quite optimal. Still, this method can be adapted to achieve work-optimal algorithms [BSV96, Che95, GG97].

BUILD-AND-SEARCH

Another important technique in parallel computational geometry is the build-and-search technique. It is a paradigm that often yields efficient parallel adaptations of sequential algorithms designed using the powerful plane sweeping technique. In the build-and-search technique, the solution to a problem is partitioned into a *build* phase, where one constructs in parallel a data structure built from the geometric data present in the problem, and a *search* phase, where one searches this data structure in parallel to solve the problem at hand. An example of an application of this technique is for the trapezoidal decomposition problem: given a collection of nonintersecting line segments in the plane, determine the first segments intersected by vertical rays emanating from each segment endpoint (cf. Figure 44.0.1). The existing efficient parallel algorithm for this problem is based upon first building in parallel a data structure on the input set of segments that allows for such vertical ray-shooting queries to be answered in $O(\log n)$ time by a single processor, and then querying this structure for each segment endpoint in parallel. This results in a parallel algorithm with an efficient $O(n \log n)$ work bound and fast $O(\log n)$ query time.

46.2 CONVEXITY

Results on the problem of constructing the convex hull of n points in $\mathbb{R}^d$ are summarized in Table 46.2.1, for various fixed values of d, and, in the case of $d = 2$, under assumptions about whether the input is presorted. We restrict our attention to parallel algorithms with efficient work bounds, where we use the term ***work*** of an algorithm here to refer to the product of its running time and the number of processors used by the algorithm. A parallel algorithm has an ***optimal*** work bound if the work used asymptotically matches the sequential lower bound for the problem. In the table, h denotes the size of the hull, and c is some fixed constant. Also, throughout this chapter we use $\bar{O}(f(n))$ to denote an asymptotic bound that holds with high probability.

TABLE 46.2.1 Parallel convex hull algorithms.

PROBLEM	MODEL	TIME	WORK	REF
2D presorted	rand-CRCW	$\bar{O}(\log^* n)$	$\bar{O}(n)$	[GG91]
2D presorted	CRCW	$O(\log\log n)$	$O(n)$	[BSV96]
2D presorted	EREW	$O(\log n)$	$O(n)$	[Che95]
2D polygon	EREW	$O(\log n)$	$O(n)$	[Che95]
2D	rand-CRCW	$\bar{O}(\log n)$	$\bar{O}(n \log h)$	[GG91]
2D	rand-CRCW	$O(\log h \log\log n)$	$\bar{O}(n \log h)$	[GS97]
2D	EREW	$O(\log n)$	$O(n \log n)$	[MS88]
2D	EREW	$O(\log^2 n)$	$O(n \log h)$	[GG91]
3D	rand-CRCW	$\bar{O}(\log n)$	$\bar{O}(n \log n)$	[RS92]
3D	rand-CRCW	$O(\log h \log^2 \log n)$	$\bar{O}(n \log h)$	[GS03]
3D	CREW	$O(\log n)$	$O(n^{1+1/c})$	[AP93]
3D	EREW	$O(\log^2 n)$	$O(n \log n)$	[AGR94]
3D	EREW	$O(\log^3 n)$	$O(n \log h)$	[AGR94]
Fixed $d \geq 4$	rand-EREW	$\bar{O}(\log^2 n)$	$\bar{O}(n^{\lfloor d/2 \rfloor})$	[AGR94]
Even $d \geq 4$	EREW	$O(\log^2 n)$	$O(n^{\lfloor d/2 \rfloor})$	[AGR94]
Odd $d > 4$	EREW	$O(\log^2 n)$	$O(n^{\lfloor d/2 \rfloor} \log^c n)$	[AGR94]

We discuss a few of these algorithms to illustrate their flavor.

2-DIMENSIONAL CONVEX HULLS

The two-dimensional convex hull algorithm of Miller and Stout [MS88] is based upon a parallel divide-and-conquer scheme where one presorts the input and then divides it into many subproblems ($O(n^{1/4})$ in their case), solves each subproblem independently in parallel, and then merges all the subproblem solutions together in $O(\log n)$ parallel time. Of course, the difficult step is the merge of all the subproblems, with the principal difficulty being the computation of common tangents between hulls. The total running time is characterized by the recurrence

$$T(n) \le T(n^{1/4}) + O(\log n),$$

which solves to $T(n) = O(\log n)$.

3-DIMENSIONAL CONVEX HULLS

All of the 3D convex hull algorithms listed in Table 46.2.1 are also based upon this many-way, divide-and-conquer paradigm, except that there is no notion of presorting in three dimensions, so the subdivision step also becomes nontrivial. Reif and Sen [RS92] use a random sample to perform the division, and the methods of Amato, Goodrich, and Ramos [AGR94] derandomize this approach. Amato and Preparata [AP93] use parallel separating planes, an approach extended to higher dimension in [AGR94].

LINEAR PROGRAMMING

A problem strongly related to convex hull construction, which has also been addressed in a parallel setting, is d-dimensional linear programming, for fixed dimensions d (see Chapter 49). Of course, one could solve this problem by transforming it to its dual problem, constructing a convex hull in this dual space, and then evaluating each vertex in the simplex that is dual to this convex hull. This would be quite inefficient, however, for $d \ge 4$. The best parallel bounds for this problem are listed in Table 46.2.2. See Section 49.6 for a detailed discussion.

TABLE 46.2.2 Fixed d-dimensional parallel linear programming.

MODEL	TIME	WORK	REF
Rand-CRCW	$\bar{O}(1)$	$\bar{O}(n)$	[AM94]
CRCW	$O((\log\log n)^{d-1})$	$O(n)$	[GR97]
EREW	$O(\log n(\log\log n)^{d-1})$	$O(n)$	[Goo96]
EREW (d=2)	$O(\log n(\log\log n)^{*})$	$O(n)$	[CX02]

OPEN PROBLEMS

There are a number of interesting open problems regarding convexity:

1. Can d-dimensional linear programming be solved (deterministically) in $O(\log n)$ time using $O(n)$ work in the CREW PRAM model?
2. Is there an efficient output-sensitive parallel convex hull algorithm for $d \ge 4$?
3. Is there a work-optimal $O(\log^2 n)$-time CREW PRAM convex hull algorithm for odd dimensions greater than 4?

46.3 ARRANGEMENTS AND DECOMPOSITIONS

Another important class of geometric problems that has been addressed in the parallel setting are arrangement and decomposition problems, which deal with ways of partitioning space. We review the best parallel bounds for such problems in Table 46.3.1.

GLOSSARY

Arrangement: The partition of space determined by the intersections of a collection of geometric objects, such as lines, line segments, or (in higher dimensions) hyperplanes. In this chapter, algorithms for constructing arrangements produce the *incidence graph*, which stores all adjacency information between the various primitive topological entities determined by the partition, such as intersection points, edges, faces, etc. See Section 28.3.1.

Red-blue arrangement: An arrangement defined by two sets of objects A and B such that the objects in A (resp. B) are nonintersecting.

Axis-parallel: All segments/lines are parallel to one of the coordinate axes.

Monotone polygon: A polygon, which is intersected by any line parallel to some fixed direction at most twice. See Section 30.1.

Polygon triangulation: A decomposition of the interior of a polygon into triangles by adding non-crossing diagonals between vertices. See Section 30.2.

Trapezoidal decomposition: A decomposition of the plane into trapezoids (and possibly triangles) by adding appropriate vertical line segments incident to vertices. See Section 38.3.

Star-shaped polygon: A (simple) polygon that is completely visible from a single point. A polygon with nonempty kernel. See Section 30.1.

$1/r$-cutting: Given n hyperplanes in $\mathbb{R}^d$ and a parameter $1 \leq r \leq n$, a $1/r$-cutting is a partition of $\mathbb{R}^d$ into (relatively open) simplicies such that each simplex intersects at most n/r hyperplanes. See Sections 40.2 and 44.1.

We sketch one randomized algorithm in Table 46.3.1 to illustrate how randomization and parallel computation can be mixed. Let S be a set of segments in the plane with k intersecting pairs. The goal is to construct $\mathcal{A}(S)$, the arrangement induced by S. First, an estimate $\hat{k}$ for k is obtained from a random sample. Then a random subset $R \subset S$ of a size r dependent on $\hat{k}$ is selected. $\mathcal{A}(R)$ is constructed using a suboptimal parallel algorithm, and processed (in parallel) for point location. Next the segments intersecting each cell of $\mathcal{A}(R)$ are found using a parallel point-location algorithm, together with some ad hoc techniques. Visibility information among the segments meeting each cell is computed using another suboptimal parallel algorithm. Finally, the resulting cells are merged in parallel. Because various key parameters in the suboptimal algorithms are kept small by the sampling, optimal expected work is achieved.

All of the algorithms for computing segment arrangements are ***output-sensitive***, in that their work bounds depend upon both the input size and the output size. In these cases we must slightly extend our computational model to allow for

TABLE 46.3.1 Parallel arrangement and decomposition algorithms.

PROBLEM	MODEL	TIME	WORK	REF
d-dim hyperplane arr	EREW	$O(\log n)$	$O(n^d)$	[AGR94]
2D seg arr	rand-CRCW	$\bar{O}(\log n)$	$\bar{O}(n \log n + k)$	[CCT92a, CCT92b]
2D axis-par seg arr	CREW	$O(\log n)$	$O(n \log n + k)$	[Goo91a]
2D red-blue seg arr	CREW	$O(\log n)$	$O(n \log n + k)$	[GSG92, GSG93, Rüb92]
2D seg arr	EREW	$O(\log^2 n)$	$O(n \log n + k)$	[AGR95]
Monotone polygon triangulation	EREW	$O(\log n)$	$O(n)$	[Che95]
Polygon triangulation	CRCW	$O(\log n)$	$O(n)$	[Goo95]
Polygon triangulation	CREW	$O(\log n)$	$O(n \log n)$	[Goo89, Yap88]
2D nonint seg trap decomp	CREW	$O(\log n)$	$O(n \log n)$	[ACG89]
2D quadtree decomp	EREW	$O(\log n)$	$O(n \log n + k)$	[BET99]

the machine to request additional processors if necessary. In all these algorithms, this request may originate only from a single "master" processor, however, so this modification is not that different from our assumption that the number of processors assigned to a problem can be a function of the input size. Of course, to solve a problem on a real parallel computer, one would simulate one of these efficient parallel algorithms to achieve an optimal speed-up over what would be possible using a sequential method.

A class of related problems deals with methods for detecting intersections. Testing whether a collection of objects contains at least one intersecting pair is frequently easier than finding all such intersections. Table 46.3.2 reviews such results in the parallel domain.

TABLE 46.3.2 Parallel intersection detection algorithms.

PROBLEM	MODEL	TIME	WORK	REF
2 convex polygons	CREW	$O(1)$	$O(n^{1/c})$	[DK89a]
2 star-shaped polygons	CREW	$O(\log n)$	$O(n)$	[GM91]
2 convex polyhedra	CREW	$O(\log n)$	$O(n)$	[DK89a]

Given a collection of n hyperplanes in $\mathbb{R}^d$, another important decomposition problem is the construction of a $(1/r)$-cutting. For this problem there exists an EREW algorithm running in $O(\log n \log r)$ time using $O(nr^{d-1})$ work [Goo93].

OPEN PROBLEMS

1. Is there a work-optimal $O(\log n)$-time polygon triangulation algorithm that does not use concurrent writes?

2. Can a line segment arrangement be constructed in $O(\log n)$ time using $O(n \log n + k)$ work in the CREW PRAM model?

46.4 PROXIMITY

An important property of Euclidean space is that it is a metric space, and distance plays an important role in many computational geometry applications. For example, computing a closest pair of points can be used in collision detection, as can the more general problem of computing the nearest neighbor of each point in a set S, a problem we will call the ***all-nearest neighbors (ANN)*** problem. Perhaps the most fundamental problem in this domain is the subdivision of space into regions where each region $V(s)$ is defined by a *site* s in a set S of geometric objects such that each point in $V(s)$ is closer to s than to any other object in S. This subdivision is the ***Voronoi diagram*** (Chapter 27); its graph-theoretic dual, which is also an important geometric structure, is the ***Delaunay triangulation*** (Section 29.1). For a set of points S in $\mathbb{R}^d$, there is a simple "lifting" transformation that takes each point $(x_1, x_2, \ldots, x_d) \in S$ to the point $(x_1, x_2, \ldots, x_d, x_1^2 + x_2^2 + \ldots + x_d^2)$, forming a set of points S' in $\mathbb{R}^{d+1}$ (Section 27.1). Each simplex on the convex hull of S' with a negative $(d{+}1)$-st component in its normal vector projects back to a simplex of the Delaunay triangulation in $\mathbb{R}^d$. Thus, any $(d{+}1)$-dimensional convex hull algorithm immediately implies a d-dimensional Voronoi diagram (VD) algorithm. Table 46.4.1 summarizes the bounds of efficient parallel algorithms for constructing Voronoi diagrams in this way, as well as methods that are designed particularly for Voronoi diagram construction or other specific proximity problems. (In the table, the underlying objects are points unless stated otherwise.)

GLOSSARY

Convex position: A set of points are in convex position if they are all on the boundary of their convex hull.

Voronoi diagram for line segments: A Voronoi diagram that is defined by a set of nonintersecting line segments, with distance from a point p to a segment s being defined as the distance from p to a closest point on s. See Section 27.3.

TABLE 46.4.1 Parallel proximity algorithms.

PROBLEM	MODEL	TIME	WORK	REF
2D ANN in convex pos	EREW	$O(\log n)$	$O(n)$	[CG92]
2D ANN	EREW	$O(\log n)$	$O(n \log n)$	[CG92]
d-dim ANN	CREW	$O(\log n)$	$O(n \log n)$	[Cal93]
2D VD in L_1 metric	CREW	$O(\log n)$	$O(n \log n)$	[WC90]
2D VD for points & segments	rand-CREW	$\bar{O}(\log n)$	$\bar{O}(n \log n)$	[Ram97]
2D VD for points & segments	CRCW	$O(\log n \log\log n)$	$O(n \log n)$	[Ram97]
2D VD	CREW	$O(\log n \log\log n)$	$O(n \log^2 n)$	[CGÓ96]
2D VD	EREW	$O(\log^2 n)$	$O(n \log n)$	[AGR94]
2D VD for segments	EREW	$O(\log^2 n)$	$O(n \log n)$	[Ram97]
3D VD	EREW	$O(\log^2 n)$	$O(n^2)$	[AGR94]

OPEN PROBLEMS

1. Can a 2D Voronoi diagram be constructed deterministically in $O(\log n)$ time using $O(n \log n)$ work in either of the PRAM models?
2. Is there an efficient output-sensitive parallel algorithm for constructing 3D Voronoi diagrams?

46.5 GEOMETRIC SEARCHING

Given a subdivision of space by a collection S of geometric objects, such as line segments, the point location problem is to build a data structure for this set that can quickly answer *vertical ray-shooting queries*, where one is given a point p and asked to report the first object in S hit by a vertical ray from p. We summarize efficient parallel algorithms for planar point location in Table 46.5.1. The time and work bounds listed, as well as the computational model, are for building the data structure to achieve an $O(\log n)$ query time. We do not list the space bounds for any of these methods in the table since, in every case, they are equal to the preprocessing work bounds.

GLOSSARY

Arbitrary planar subdivision: A subdivision of the plane (not necessarily connected), defined by a set of line segments that intersect only at their endpoints.

Monotone subdivision: A connected subdivision of the plane in which each face is intersected by a vertical line in a single segment.

Triangulated subdivision: A connected subdivision of the plane into triangles whose corners are vertices of the subdivision (see Chapter 29).

Shortest path in a polygon: The shortest path between two points that does not go outside of the polygon (see Section 30.4).

Ray-shooting query: A query whose answer is the first object hit by a ray oriented in a specified direction from a specified point.

TABLE 46.5.1 Parallel geometric searching algorithms.

QUERY PROBLEM	MODEL	TIME	WORK	REF
Point loc in arb subdivision	CREW	$O(\log n)$	$O(n \log n)$	[ACG89]
Point loc in monotone subdivision	EREW	$O(\log n)$	$O(n)$	[TV91]
Point loc in triangulated subdivision	CREW	$O(\log n)$	$O(n)$	[CZ90]
Point loc in d-dim hyp arr	EREW	$O(\log n)$	$O(n^d)$	[AGR94]
Shortest path in triangulated polygon	CREW	$O(\log n)$	$O(n)$	[GSG92]
Ray shooting in triangulated polygon	CREW	$O(\log n)$	$O(n)$	[HS95]
Line & convex polyhedra intersection	CREW	$O(\log n)$	$O(n)$	[DK89b, CZ90]

OPEN PROBLEMS

1. Is there an efficient data structure that allows n simultaneous point locations to be performed in $O(\log n)$ time using $O(n)$ processors in the EREW PRAM model?
2. Is there an efficient data structure for 3-dimensional point location in convex subdivisions that can be constructed in $O(n \log n)$ work and at most $O(\log^2 n)$ time and which allows for a query time that is at most $O(\log^2 n)$?

46.6 VISIBILITY, ENVELOPES, AND OPTIMIZATION

We summarize efficient parallel methods for various visibility and lower envelope problems for a simple polygon with n vertices in Table 46.6.1. In the table, m denotes the number of edges in a visibility graph. For definitions see Chapter 33.

TABLE 46.6.1 Parallel visibility algorithms for a simple polygon.

PROBLEM	MODEL	TIME	WORK	REF
Kernel	EREW	$O(\log n)$	$O(n)$	[Che95]
Vis from a point	EREW	$O(\log n)$	$O(n)$	[ACW91]
Vis from an edge	CRCW	$O(\log n)$	$O(n)$	[Her95]
Vis from an edge	CREW	$O(\log n)$	$O(n \log n)$	[GSG92, GSG93]
Vis graph	CREW	$O(\log n)$	$O(n \log^2 n + m)$	[GSG92, GSG93]

We sketch the algorithm for computing the point visibility polygon [ACW91], which is notable for two reasons: first, it is employed as a subprogram in many other algorithms; and second, it requires much more intricate processing and analysis than the relatively simple optimal sequential algorithm (Section 25.3). The parallel algorithm is recursive, partitioning the boundary into $n^{1/4}$ subchains, and computing *visibility chains* from the source point of visibility x. Each of these chains is star-shaped with respect to x, i.e., effectively "monotone" (see Section 30.1). This monotonicity property is, however, insufficient to intersect the visibility chains quickly enough in the merge step to obtain optimal bounds. Rather, the fact that the chains are subchains of the boundary of a simple polygon must be exploited to achieve logarithmic-time computation of the intersection of two chains. This then leads to the optimal bounds quoted in Table 46.6.1.

The bounds of efficient parallel methods for visibility problems on general sets of segments and curves in the plane are summarized in Table 46.6.2.

GLOSSARY

Lower envelope: The function $F(x)$ defined as the pointwise minimum of a collection of functions $\{f_1, f_2, \ldots, f_n\}$: $F(x) = \min_i f_i(x)$ (see Section 21.2).

k-intersecting curves: A set of curves every two of which intersect at most k times (where they cross).

$\lambda_s(n)$: The maximum length of a Davenport-Schinzel sequence [SA95, AS00] of order s on n symbols. If s is a constant, $\lambda_s(n)$ is $o(n \log^* n)$. See Section 40.4.

TABLE 46.6.2 General parallel visibility and enveloping algorithms.

PROBLEM	MODEL	TIME	WORK	REF
Lower env for segments	EREW	$O(\log n)$	$O(n \log n)$	[CW02]
Lower env for k-int curves	rand-CRCW	$\bar{O}(\log n \log^* n)$	$\bar{O}(\lambda_k(n) \log n)$	[Goo91b]
Lower env for k-int curves	EREW	$O(\log^{1+\epsilon} n)$	$O(\lambda_{k+1}(n) \log n \log^* n)$	[CW02]

Finally, we summarize some efficient parallel algorithms for solving several geometric optimization problems in Table 46.6.3.

GLOSSARY

Largest-area empty rectangle: For a collection S of n points in the plane, the largest-area rectangle that does not contain any point of S in its interior.

All-farthest neighbors problem in a simple polygon: Determine for each vertex p of a simple polygon the vertex q such that the shortest path from p is longest.

Closest visible-pair between polygons: A closest pair of mutually-visible vertices between two nonintersecting simple polygons in the plane.

Minimum circular-arc cover: For a collection of n arcs of a given circle C, a minimum-cardinality subset that covers C.

Optimal-area inscribed/circumscribed triangle: For a convex polygon P, the largest-area triangle inscribed in P, or, respectively, the smallest-area triangle circumscribing P.

Min-link path in a polygon: A piecewise-linear path of fewest "links" inside a simple polygon between two given points p and q; see Sections 23.4 and 24.3.

TABLE 46.6.3 Parallel geometric optimization algorithms.

PROBLEM	MODEL	TIME	WORK	REF
Largest-area empty rectangle	CREW	$O(\log^2 n)$	$O(n \log^3 n)$	[AKPS90]
All-farthest neighbors in polygon	CREW	$O(\log^2 n)$	$O(n \log^2 n)$	[Guh92]
Closest visible-pair btw polygons	CREW	$O(\log n)$	$O(n \log n)$	[HCL92]
Min circular-arc cover	EREW	$O(\log n)$	$O(n \log n)$	[AC89]
Opt-area inscr/circum triangle	CRCW	$O(\log \log n)$	$O(n)$	[CM92]
Opt-area inscr/circum triangle	CREW	$O(\log n)$	$O(n)$	[CM92]
Min-link path in a polygon	CREW	$O(\log n \log \log n)$	$O(n \log n \log \log n)$	[CGM+95]

OPEN PROBLEMS

1. Can the visibility graph of a set of n nonintersecting line segments be constructed using $O(n \log n + m)$ work in time at most $O(\log^2 n)$ in the CREW model, where m is the size of the graph?
2. Can the visibility graph of a triangulated polygon be computed in $O(\log n)$ time using $O(n + m)$ work in the CREW model?

46.7 SOURCES AND RELATED MATERIAL

FURTHER READING

Our presentation has been results-oriented and has not provided much problem intuition or algorithmic techniques. There are several excellent surveys available in the literature [Ata92, AC94, AC00, AG93, RS93, RS00] that are more techniques-oriented. Another good location for related material is the book by Akl and Lyons [AL93].

Finally, while we have focused on the classical PRAM model, recently, several extensions have been proposed that consider the effects of the hierarchical memory design of modern multicore processors on the runtime [BG04, CGK$^+$07, BCG$^+$08, AGNS08, CRSB13]. In these models, the goal is to organize data and schedule computations in such a way as to minimize the number of accesses to the slow shared memory by utilizing fast private and/or shared caches. This is still a relatively new research direction with many problems remaining open. However, some results have already been obtained in these models for 2D axis-parallel line segment and rectangle intersection reporting [ASZ10, ASZ11], 1D and 2D range reporting [ASZ10, SZ12], 2D lower envelope and convex hull computation [ASZ10] and 3D maxima reporting [ASZ10].

RELATED CHAPTERS

Chapter 26: Convex hull computations
Chapter 27: Voronoi diagrams and Delaunay triangulations
Chapter 28: Arrangements
Chapter 30: Polygons
Chapter 38: Point location
Chapter 42: Geometric intersection
Chapter 44: Randomization and derandomization
Chapter 49: Linear programming

REFERENCES

[AC89] M.J. Atallah and D.Z. Chen. An optimal parallel algorithm for the minimum circle-cover problem. *Information Processing Letters*, 34:159–165, 1989.

[AC94] M.J. Atallah and D. Z. Chen. Parallel computational geometry. In A.Y. Zomaya, editor, *Parallel Computations: Paradigms and Applications.* World Scientific, Singapore, 1994.

[AC00] M. J. Atallah and D. Z. Chen. Deterministic parallel computational geometry. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 155–200. Elsevier, Amsterdam, 2000.

[ACG$^+$88] A. Aggarwal, B. Chazelle, L.J. Guibas, C. Ó'Dúnlaing, and C. Yap. Parallel computational geometry. *Algorithmica*, 3:293–327, 1988.

[ACG89] M.J. Atallah, R. Cole, and M.T. Goodrich. Cascading divide-and-conquer: A technique for designing parallel algorithms. *SIAM J. Comput.*, 18:499–532, 1989.

[ACW91] M.J. Atallah, D.Z. Chen, and H. Wagener. Optimal parallel algorithm for visibility of a simple polygon from a point. *J. ACM*, 38:516–553, 1991.

[AG86] M.J. Atallah and M.T. Goodrich. Efficient parallel solutions to some geometric problems. *J. Parallel Distrib. Comput.*, 3:492–507, 1986.

[AG93] M.J. Atallah and M.T. Goodrich. Deterministic parallel computational geometry. In J.H. Reif, editor, *Synthesis of Parallel Algorithms*, pages 497–536. Morgan Kaufmann, San Mateo, 1993.

[AGNS08] L. Arge, M.T. Goodrich, M. Nelson, and N. Sitchinava. Fundamental parallel algorithms for private-cache chip multiprocessors. In *Proc. 20th ACM Sympos. Parallel. Algorithms Architect.*, pages 197–206, 2008.

[AGR94] N.M. Amato, M.T. Goodrich, and E.A. Ramos. Parallel algorithms for higher-dimensional convex hulls. In *Proc. 35th IEEE Sympos. Found. Comp. Sci.*, pages 683–694, 1994.

[AGR95] N.M. Amato, M.T. Goodrich, and E.A. Ramos. Computing faces in segment and simplex arrangements. In *Proc. 27th ACM Sympos. Theory Comput.*, pages 672–682, 1995.

[Akl82] S.G. Akl. A constant-time parallel algorithm for computing convex hulls. *BIT*, 22:130–134, 1982.

[Akl84] S.G. Akl. Optimal parallel algorithms for computing convex hulls and for sorting. *Computing*, 33:1–11, 1984.

[Akl85] S.G. Akl. Optimal parallel algorithms for selection, sorting and computing convex hulls. In G. T. Toussaint, editor, *Computational Geometry*, volume 2 of *Machine Intelligence and Pattern Recognition*, pages 1–22. North-Holland, Amsterdam, 1985.

[AKPS90] A. Aggarwal, D. Kravets, J.K. Park, and S. Sen. Parallel searching in generalized monge arrays with applications. In *Proc. 2nd ACM Sympos. Parallel Algorithms Architect.*, pages 259–268, 1990.

[AL93] S.G. Akl and K. A. Lyons. *Parallel Computational Geometry.* Prentice Hall, Englewood Cliffs, 1993.

[AM94] N. Alon and N. Megiddo. Parallel linear programming in fixed dimension almost surely in constant time. *J. ACM*, 41:422–434, 1994.

[AP92] N.M. Amato and F.P. Preparata. The parallel 3D convex hull problem revisited. *Internat. J. Comput. Geom. Appl.*, 2:163–173, 1992.

[AP93] N.M. Amato and F. P. Preparata. An NC^1 parallel 3D convex hull algorithm. In *Proc. 9th Sympos. Comput. Geom.*, pages 289–297, ACM Press, 1993.

[AP95] N.M. Amato and F.P. Preparata. A time-optimal parallel algorithm for three-dimensional convex hulls. *Algorithmica*, 14:169–182, 1995.

[AS00] P.K. Agarwal and M. Sharir. Davenport-Schinzel sequences and their geometric applications. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 1–47. Elsevier, Amsterdam, 2000.

[ASZ10] D. Ajwani, N. Sitchinava, and N. Zeh. Geometric algorithms for private-cache chip multiprocessors. In *Proc. 18th Europ. Sympos. Alg.*, part II, volume 6347 of *Lecture Notes Comp. Sci.*, pages 75–86. Springer, Berlin, 2010.

[ASZ11] D. Ajwani, N. Sitchinava, and N. Zeh. I/O-optimal distribution sweeping on private-cache chip multiprocessors. In *Proc. 25th IEEE Internat. Parallel Distrib. Proc. Sympos.*, ‘pages 1114–1123, 2011.

[Ata92] M.J. Atallah. Parallel techniques for computational geometry. *Proc. IEEE*, 80:1435–1448, 1992.

[BCG$^+$08] G.E. Blelloch, R.A. Chowdhury, P.B. Gibbons, V. Ramachandran, S. Chen, and M. Kozuch. Provably good multicore cache performance for divide-and-conquer algorithms. In *Proc. 19th ACM-SIAM Sympos. Discrete Algorithms*, pages 501–510, 2008.

[BET99] M. Bern, D. Eppstein, and S.-H. Teng. Parallel construction of quadtrees and quality triangulations. *Internat. J. Comput. Geom. Appl.*, 9:517–532, 1999.

[BG04] G.E. Blelloch and P.B. Gibbons. Effectively sharing a cache among threads. In *Proc. 16th ACM Sympos. Parallel. Algorithms Architect.*, pages 235–244, 2004.

[BSV96] O. Berkman, B. Schieber, and U. Vishkin. A fast parallel algorithm for finding the convex hull of a sorted point set. *Internat. J. Comput. Geom. Appl.*, 6:231–242, 1996.

[Cal93] P.B. Callahan. Optimal parallel all-nearest-neighbors using the well-separated pair decomposition. In *Proc. 34th IEEE Sympos. Found. Comp. Sci.*, pages 332–340, 1993.

[CCT92a] K.L. Clarkson, R. Cole, and R.E. Tarjan. Erratum: Randomized parallel algorithms for trapezoidal diagrams. *Internat. J. Comput. Geom. Appl.*, 2:341–343, 1992.

[CCT92b] K.L. Clarkson, R. Cole, and R.E. Tarjan. Randomized parallel algorithms for trapezoidal diagrams. *Internat. J. Comput. Geom. Appl.*, 2:117–133, 1992.

[CG92] R. Cole and M.T. Goodrich. Optimal parallel algorithms for polygon and point-set problems. *Algorithmica*, 7:3–23, 1992.

[CGK$^+$07] S. Chen, P.B. Gibbons, M. Kozuch, V. Liaskovitis, A. Ailamaki, G.E. Blelloch, B. Falsafi, L. Fix, N. Hardavellas, T.C. Mowry, and C. Wilkerson. Scheduling threads for constructive cache sharing on CMPs. In *Proc. 19th ACM Sympos. Parallel. Algorithms Architect.*, pages 105–115, 2007.

[CGM$^+$95] V. Chandru, S.K. Ghosh, A. Maheshwari, V.T. Rajan, and S. Saluja. NC-algorithms for minimum link path and related problems. *J. Algorithms*, 19:173–203, 1995.

[CGÓ96] R. Cole, M.T. Goodrich, and C. Ó'Dúnlaing. A nearly optimal deterministic parallel Voronoi diagram algorithm. *Algorithmica*, 16:569–617, 1996.

[Che95] D. Chen. Efficient geometric algorithms on the EREW PRAM. *IEEE Trans. Parallel Distrib Syst.*, 6:41–47, 1995.

[Cho80] A.L. Chow. *Parallel algorithms for geometric problems.* Ph.D. thesis, Dept. Comp. Sci., Univ. Illinois, Urbana, IL, 1980.

[CM92] S. Chandran and D.M. Mount. A parallel algorithm for enclosed and enclosing triangles. *Internat. J. Comput. Geom. Appl.*, 2:191–214, 1992.

[CRSB13] R.A. Chowdhury, V. Ramachandran, F. Silvestri, and B. Blakeley. Oblivious algorithms for multicores and networks of processors. *J. Parallel Distrib. Comput.*, 73:911–925, 2013.

[CW02] W. Chen and K. Wada. On computing the upper envelope of segments in parallel. *IEEE Trans. Parallel Distrib. Syst.*, 13:5–13, 2002.

[CX02] D.Z. Chen and J. Xu. Two-variable linear programming in parallel. *Computat Geom*, 21:155–165, 2002.

[CZ90] R. Cole and O. Zajicek. An optimal parallel algorithm for building a data structure for planar point location. *J. Parallel Distrib. Comput.*, 8:280–285, 1990.

[DK89a] N. Dadoun and D.G. Kirkpatrick. Cooperative subdivision search algorithms with applications. In *Proc. 27th Allerton Conf. Commun. Control Comput.*, pages 538–547, 1989.

[DK89b] N. Dadoun and D. G. Kirkpatrick. Parallel construction of subdivision hierarchies. *J. Comp. Syst. Sci.*, 39:153–165, 1989.

[Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Volume 10 of *EATCS Monogr. Theoret. Comp. Sci.*. Springer-Verlag, Heidelberg, 1987.

[GG91] M.R. Ghouse and M.T. Goodrich. In-place techniques for parallel convex hull algorithms. In *Proc. 3rd ACM Sympos. Parallel Algorithms Architect.*, pages 192–203, 1991.

[GG97] M.R. Ghouse and M.T. Goodrich. Fast randomized parallel methods for planar convex hull construction. *Comput. Geom.*, 7:219–235, 1997.

[GM91] S.K. Ghosh and A. Maheshwari. An optimal parallel algorithm for determining the intersection type of two star-shaped polygons. In *Proc. 3rd Canadian Conf. Comput. Geom.*, pages 2–6, 1991.

[Goo87] M.T. Goodrich. *Efficient parallel techniques for computational geometry.* Ph.D. thesis, Dept. Comp. Sci., Purdue Univ., West Lafayette, 1987.

[Goo89] M.T. Goodrich. Triangulating a polygon in parallel. *J. Algorithms*, 10:327–351, 1989.

[Goo91a] M.T. Goodrich. Intersecting line segments in parallel with an output-sensitive number of processors. *SIAM J. Comput.*, 20:737–755, 1991.

[Goo91b] M.T. Goodrich. Using approximation algorithms to design parallel algorithms that may ignore processor allocation. In *Proc. 32nd IEEE Sympos. Found. Comp. Sci.*, pages 711–722, 1991.

[Goo93] M.T. Goodrich. Geometric partitioning made easier, even in parallel. In *Proc. 9th Sympos. Comput. Geom.*, pages 73–82, ACM Press, 1993.

[Goo95] M.T. Goodrich. Planar separators and parallel polygon triangulation. *J. Comp. Syst. Sci.*, 51:374–389, 1995.

[Goo96] M.T. Goodrich. Fixed-dimensional parallel linear programming via relative epsilon-approximations. In *Proc. 7th ACM-SIAM Sympos. Discrete Algorithms*, pages 132–141, 1996.

[GR97] M.T. Goodrich and E.A. Ramos. Bounded-independence derandomization of geometric partitioning with applications to parallel fixed-dimensional linear programming. *Discrete Comput. Geom.*, 18:397–420, 1997.

[GS97] N. Gupta and S. Sen. Optimal, output-sensitive algorithms for constructing planar hulls in parallel. *Comput. Geom.*, 8:151–166, 1997.

[GS03] N. Gupta and S. Sen. Faster output-sensitive parallel algorithms for 3d convex hulls and vector maxima. *J. Parallel Distrib. Comput.*, 63:488–500, 2003.

[GSG92] M.T. Goodrich, S. Shauck, and S. Guha. Parallel methods for visibility and shortest path problems in simple polygons. *Algorithmica*, 8:461–486, 1992.

[GSG93] M.T. Goodrich, S. Shauck, and S. Guha. Addendum to "parallel methods for visibility and shortest path problems in simple polygons". *Algorithmica*, 9:515–516, 1993.

[Guh92] S. Guha. Parallel computation of internal and external farthest neighbours in simple polygons. *Internat. J. Comput. Geom. Appl.*, 2:175–190, 1992.

[HCL92] F.R. Hsu, R.C. Chang, and R.C.T. Lee. Parallel algorithms for computing the closest visible vertex pair between two polygons. *Internat J. Comput. Geom. Appl.*, 2:135–162, 1992.

[Her95] J. Hershberger. Optimal parallel algorithms for triangulated simple polygons. *Internat. J. Comput. Geom. Appl.*, 5:145–170, 1995.

[HS95] J. Hershberger and S. Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *J. Algorithms*, 18:403–431, 1995.

[Já92] J. JáJá. *An Introduction to Parallel Algorithms.* Addison-Wesley, Reading, 1992.

[KR90] R.M. Karp and V. Ramachandran. Parallel algorithms for shared memory machines. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 869–941. Elsevier/The MIT Press, Amsterdam, 1990.

[MS88] R. Miller and Q.F. Stout. Efficient parallel convex hull algorithms. *IEEE Trans. Comp.*, 37:1605–1618, 1988.

[O'R94] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, 1st edition, 1994.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, 1985.

[Ram97] E.A. Ramos. Construction of 1-d lower envelopes and applications. In *Proc. 13th Sympos. Comput. Geom.*, pages 57–66, ACM Press, 1997.

[Rei93] J.H. Reif. *Synthesis of Parallel Algorithms.* Morgan Kaufmann, San Mateo, 1993.

[RS92] J.H. Reif and S. Sen. Optimal parallel randomized algorithms for three-dimensional convex hulls and related problems. *SIAM J. Comput.*, 21:466–485, 1992.

[RS93] S. Rajasekaran and S. Sen. Random sampling techniques and parallel algorithms design. In J.H. Reif, editor, *Synthesis of Parallel Algorithms*, pages 411–452. Morgan Kaufmann, San Mateo, 1993.

[RS00] J.H. Reif and S. Sen. Parallel computational geometry: An approach using randomization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 765–828. Elsevier, Amsterdam, 2000.

[Rüb92] C. Rüb. Computing intersections and arrangements for red-blue curve segments in parallel. In *Proc 4th Canadian Conf. Comput. Geom.*, pages 115–120, 1992.

[SA95] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications.* Cambridge University Press, 1995.

[Sen89] S. Sen. *Random Sampling Techniques for Efficient Parallel Algorithms in Computational Geometry.* Ph.D. thesis, Dept. Comp. Sci., Duke Univ., 1989.

[SZ12] N. Sitchinava and N. Zeh. A parallel buffer tree. In *Proc. 24th ACM Sympos. Parallel. Algorithms Architect.*, pages 214–223, 2012.

[TV91] R. Tamassia and J.S. Vitter. Parallel transitive closure and point location in planar structures. *SIAM J. Comput.*, 20:708–725, 1991.

[WC90] Y.C. Wee and S. Chaiken. An optimal parallel L_1-metric Voronoi diagram algorithm. In *Proc. 2nd Canadian Conf. Comput. Geom.*, pages 60–65, 1990.

[Yap88] C.K. Yap. Parallel triangulation of a polygon in two calls to the trapezoidal map. *Algorithmica*, 3:279–288, 1988.