Colored Intersection Searching via Sparse Rectangular Matrix Multiplication

Haim Kaplan
School of Computer Science
Tel Aviv University, Tel Aviv
69978, Israel
haimk@tau.ac.il

Micha Sharir[*]
School of Computer Science
Tel Aviv University, Tel Aviv
69978, Israel, and
Courant Institute of
Mathematical Sciences
New York University, New
York, NY 10012, USA
michas@tau.ac.il

Elad Verbin
School of Computer Science
Tel Aviv University, Tel Aviv
69978, Israel
eladv@tau.ac.il

ABSTRACT

In a Batched Colored Intersection Searching Problem (CI), one is given a set of n geometric objects (of a certain class). Each object is colored by one of c colors, and the goal is to report all pairs of colors (c_1, c_2) such that there are two objects, one colored c_1 and one colored c_2, that intersect each other. We also consider the bipartite version of the problem, where we are interested in intersections between objects of one class with objects of another class (e.g., points and halfspaces).

In a Sparse Rectangular Matrix Multiplication Problem ($SRMM$), one is given an $n_1 \times n_2$ matrix A and an $n_2 \times n_3$ matrix B, each containing at most m non-zero entries, and the goal is to compute their product AB.

In this paper we present a technique for solving CI problems over a wide range of classes of geometric objects. The basic idea is first to use some decomposition method, such as geometric cuttings, to represent the intersection graph of the objects as a union of bi-cliques. Then, in each of these bi-cliques, contract all vertices of the same color. Finally, use an algorithm for sparse matrix multiplication (adapted from Yuster and Zwick [20]) to compute the union of the bi-cliques. We apply the technique to segments in $\mathbb{R}^1$, to segments in $\mathbb{R}^2$, to points and halfplanes in $\mathbb{R}^2$, and, more generally, to points and halfspaces in $\mathbb{R}^d$, for any fixed d. However, the technique extends to colored intersection searching in any class (or pair of classes) of geometric objects of constant descriptive complexity.

In particular, using our technique we obtain an algorithm that reports all the pairs of intersecting colors for n points and n halfplanes in $\mathbb{R}^2$, that are colored by c colors, in $O(n^{4/3}c^{0.46})$ time when $n \geq c^{1.44}$, and in $O(n^{1.04}c^{0.9} + c^2)$ time when $n \leq c^{1.44}$.

The algorithms that we give for CI use the algorithm for $SRMM$ as a black box, which means that any improved algorithm for $SRMM$ immediately leads to an improved algorithm for all colored intersection problems that our method applies to. We also show that the complexity of computing all intersecting colors in a set of segments on the real line is *identical*, up to a polylogarithmic multiplicative factor, to the complexity of $SRMM$ with the appropriate parameters.

[*]Supported by NSF Grants CCR-00-98246 and CCF-05-14079, by a grant from the U.S.-Israel Binational Science Foundation, by Grant 155/05 from the Israel Science Fund, and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University.

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems - *geometrical problems and computations*; I.3.5 [Computational Geometry and Object Modeling]: Hierarchy and geometric transformations;

General Terms: Algorithms, Theory

Keywords: Matrix multiplication, Colored intersection searching, Generalized intersection searching, Range searching

1. INTRODUCTION

Colored intersection searching. In a *Batched Colored Intersection Searching Problem* (CI)[1], one is given a set of n geometric objects of a class C (say, the class of all segments on the real line). Each of the objects is colored by one of c colors, and the goal is to report all pairs of colors (c_1, c_2) such that there are two objects, one colored c_1 and one colored c_2, that intersect each other. Often, we wish to consider the bipartite version of the problem, where we are given two sets of objects, from two different classes (e.g., points and halfplanes in $\mathbb{R}^2$), and the goal is to report all pairs (c_1, c_2) of colors, so that an object with color c_1 of the first set intersects an object with color c_2 of the second set.

Colored Intersection Searching Problems arise naturally when we are interested in detecting intersections between

[1]In the literature, Batched Colored Intersection Searching is also called Single-Shot Generalized Intersection Searching; see [12].

complex objects. Assume that we can decompose each object into simple pieces, so that two objects x and y intersect if and only if some piece of x intersects some piece of y. Then we choose a distinct color for each original object, assign this color to all pieces of that object, and solve a colored intersection searching problem associated with the resulting colored simple objects. Each pair of intersecting colors correspond to a pair of original objects that intersect. Some especially useful instances of this setup are the problems of detecting all intersecting pairs in a set of c polygonal regions in the plane with a total of n edges, or in a set of polyhedral objects in 3-space or in higher dimensions.

As another example, consider the following *terrain guarding* problem (see, e.g., [1]). Given a polyhedral terrain T in $\mathbb{R}^3$, and a parameter $h > 0$, determine whether it is possible to place two "watchtowers" (vertical segments of height h) at two respective vertices of T, so that together they (more precisely, their top endpoints) see the entire terrain. This can be reduced to colored intersection searching as follows. For each vertex u, compute the *invisibility region* $I(u)$ of the watchtower erected at u, that is, the portion of T that is not visible from that watchtower. Decompose each invisibility region $I(u)$ into triangles, color all triangles of $I(u)$ by the same distinct color associated with u, and compute all pairs of "intersecting" colors. Any pair of non-intersecting colors yield a pair of watchtowers that together see the entire terrain.

Gupta, Janardan and Smid (see [12]) have systematically studied many colored intersection searching problems, mostly in the preprocessing-and-query setting, where one wants to build a data structure for future queries. See their survey [12] and the references therein. Bozanis et al. [3] give some output-sensitive algorithms for the problems that we address. They solve the CI problem for iso-oriented boxes in $\mathbb{R}^d$ in time $O(n(1+\sqrt{k})\log^{2d} n)$, and the CI problem for line segments in $\mathbb{R}^2$ in time $O(n^{3/2+\varepsilon}(1+k^{1/4-\varepsilon/2}))$, where k is the number of intersecting pairs of colors, and $\varepsilon > 0$ is an arbitrarily small constant. In the "dense" case, where $k = \Theta(c^2)$ and output-sensitivity does not matter, the bounds in [3] are such that the sum of the exponents of n and c is 2, while in our bounds it is below 2. (As follows from the discussion in Section 3.3, any explicit algorithm for the case of iso-oriented boxes for which the sum of the exponents is below 2 would give a new algorithm for sparse matrix multiplication; the smaller is the sum, the faster is the algorithm.)

Sparse rectangular matrix multiplication. In a Sparse Rectangular Matrix Multiplication Problem (SRMM), one is given an $n_1 \times n_2$ matrix A and an $n_2 \times n_3$ matrix B, where A contains at most m_1 non-zero entries and B contains at most m_2 non-zero entries. The goal is to compute the product AB. Non-sparse rectangular matrix multiplication has been studied in depth, see Section 2. A simple efficient algorithm for sparse *square* matrix multiplication has recently been given by Yuster and Zwick [20] (this algorithm uses an algorithm for multiplying dense matrices as a black box). We adapt the algorithm of [20] to the case of rectangular sparse matrices (see Section 2.1).

A common characteristic of many problems that can be solved using fast matrix multiplication techniques is that they require some aggregation (or consolidation) of information, involving interactions between pairs of objects, where the same pair may arise multiple times but we need to record it only once. This idea, in fact, has been one of the primary insights that has led to our results. A typical example, which will also play an eminent role in our analysis, is the problem of computing the *union of bi-cliques*. Specifically, we wish to compute $\bigcup_i A_i \times B_i$ efficiently, faster than the trivial approach, whose running time is about $\sum_i |A_i| \cdot |B_i|$. As we will shortly see, this problem *is* in fact sparse (rectangular) matrix multiplication in the Boolean ring, in a different guise (see Section 2.1).

Our results. Let S be a set of colored objects. Let $G(S)$ denote the *intersection graph* of S, whose vertices are the elements of S, and $(s_1, s_2) \in E(G(S))$ if s_1 and s_2 intersect each other. Our approach for solving CI on S consists of the following three steps:

1. Use some geometric decomposition method (e.g., cuttings) to obtain a representation of $G(S)$ as a (not necessarily disjoint) union of bi-cliques.

2. In each bi-clique, contract each group of vertices that correspond to objects of the same color, to a single vertex. The solution to the CI problem is the union of the contracted bi-cliques.

3. Transform the bi-clique-union problem into a sparse matrix multiplication problem (Section 2.1) and solve it using an efficient algorithm adapted from Yuster and Zwick [20].

We demonstrate this approach on several concrete examples, obtaining the following results (the exponents in the bounds are functions of the exponents ω, α, used in the complexity bounds of algorithms for rectangular matrix multiplication – see Section 2).

(i) For the CI problem on the line, where we have n segments colored by c different colors, we obtain an algorithm with running time

$$O\left(\begin{cases} nc^{0.69} & \text{if } n \geq c^{1.69} \\ n^{0.7}c^{1.21} + c^2 & \text{if } n \leq c^{1.69} \end{cases}\right).$$

This result is described in Section 3. A similar result holds for iso-oriented boxes in $\mathbb{R}^d$.

(ii) For the CI problem in the plane, where we have n points and n halfplanes colored by c colors, and we want to report all pairs of colors (c_1, c_2) such that there is a point colored c_1 in a halfplane colored c_2, we obtain an algorithm with running time

$$O\left(\begin{cases} n^{4/3}c^{0.46} & \text{if } n \geq c^{1.44} \\ n^{1.04}c^{0.9} + c^2 & \text{if } n \leq c^{1.44} \end{cases}\right).$$

This result is described in Section 4.

(iii) The preceding result can easily be extended to points and halfspaces in any fixed dimension d, involving n points and n halfspaces, where the running time is

$$O\left(\begin{cases} n^{\frac{2d}{d+1}} c^{\frac{1.376}{d+1}} & \text{if } n \geq c^{\frac{1+2.376d}{2d}}, \\ n^{\frac{1.066d}{0.533d+1}} c^{\frac{1.844}{0.533d+1}} + c^2 & \text{if } n \leq c^{\frac{1+2.376d}{2d}} \end{cases}\right).$$

This result is also described in Section 4.

These are just a few concrete examples. The technique can be applied to any CI problem involving geometric objects of any simple shape (namely, objects having constant descriptive complexity) in any fixed dimension.

As mentioned, we can apply these results to the problem of reporting all intersecting pairs in a set of c polygonal regions in the plane, with a total of n edges. The running time is asymptotically the same as that for the colored segment intersection problem (ii) given above. In a similar fashion, we can report all intersecting pairs in a set of polyhedral regions in any $\mathbb{R}^d$ (for d fixed), as well as detecting other kinds of interaction between pairs of complex objects. Some of these generalizations will be presented in the full version of the paper.

Many results use fast matrix multiplication techniques (over the real field, the Boolean ring, or polynomial rings) as a subroutine to derive efficient algorithms for various problems in graph theory (see, e.g., [15, 21]), learning theory [18], computational linguistics [16], numerical algorithms [14], and many more. Much less common are results where the reduction to matrix multiplication can also be reversed. That is, in such a case the complexity of the problem is shown to be equivalent (or nearly so) to the complexity of matrix multiplication (one case where this does occur is [16]). Here we provide such a result. We show, in Section 3.2, that the complexity of CI for segments on the line is equivalent, up to a polylogarithmic factor, to the complexity of $SRMM$ over the Boolean ring.

A more general problem is the following "counting version" of CI. In this counting version, each object s has a real weight $w(s)$, and we want to compute for each pair of colors (c_1, c_2) the sum $\sum_{(s_1,s_2)} w(s_1)w(s_2)$ over all pairs of objects (s_1, s_2) such that s_1 is of color c_1, s_2 is of color c_2, and s_1 and s_2 intersect each other. Our technique extends to this counting version of CI with similar running times. However, here we need to represent the intersection graph as a union of *disjoint* bi-cliques, and then we can use matrix multiplication over the reals rather than over the Boolean ring. Furthermore, we can show that here too, this weight-counting problem for segments on the line is equivalent, up to a polylogarithmic factor, to the complexity of $SRMM$ over the reals.

In a CI problem our goal is to report all *pairs* of colors c_1, c_2 such that there are objects s_1, s_2 of these respective colors that intersect. Our method can actually be generalized to report all triples of colors c_1, c_2, c_3, or, more generally, all k-tuples of colors $c_1, \ldots, c_k$, such that there is a set of k objects of these respective colors that have a common point. In such a case we need to decompose the k-uniform hypergraph of k-wise intersections of the given objects into a union of k-cliques, and then use a variant of matrix multiplication that computes the union of these k-cliques. Further details of this extension will be given in the full version of the paper.

To obtain the best asymptotic results, we use algorithms for matrix multiplication which are asymptotically efficient but are far from being practical. However, even using Strassen's algorithm for matrix multiplication [19], or any other non-trivial algorithm, our results give improvements over the naive $O(n^2)$ algorithm for CI. Implementation of our method using simple and efficient algorithms for matrix multiplication should be practical.

2. PRELIMINARIES

We use the notation $\tilde{O}(\cdot)$ to hide poly-logarithmic multiplicative factors. That is, we write $f(n) = \tilde{O}(g(n))$ if there exists c such that $f(n) \le g(n) \log^c n$.

Notation for CI problems. Let $CI_{Segments,\mathbb{R}^1}(c, n, m)$ denote the problem where one is given a set S_1 consisting of n segments on the line and a set S_2 consisting of m segments on the line, and the goal is to report for each ordered pair of colors (c_1, c_2) whether there is a segment $s_1 \in S_1$ of color c_1 and a segment $s_2 \in S_2$ of color c_2 such that s_1 and s_2 intersect each other. Let $CI_{Segments,\mathbb{R}^2}(c, n, m)$ denote the analogous problem where S_1 (resp., S_2) consists of n (resp., m) segments in $\mathbb{R}^2$. A special case of $CI_{Segments,\mathbb{R}^1}(c, n, m)$ and $CI_{Segments,\mathbb{R}^2}(c, n, m)$ is when $S_1 = S_2$. We denote these special cases by $CI_{Segments,\mathbb{R}^1}(c, n)$ and $CI_{Segments,\mathbb{R}^2}(c, n)$, respectively.

Let $CI_{Points,Halfplanes,\mathbb{R}^2}(c, n, m)$ denote the analogous colored intersection searching problem where S_1 consists of n points in $\mathbb{R}^2$ and S_2 consists of m halfplanes in $\mathbb{R}^2$. Finally, $CI_{Points,Halfspaces,\mathbb{R}^d}(c, n, m)$ denotes the analogous problem involving points and halfspaces in $\mathbb{R}^d$.

We somewhat abuse notation so that the same notation (e.g., $CI_{Segments,\mathbb{R}^1}(c, n)$) refers both to the problem and to (an upper bound on) its running time.

Notation for $SRMM$ problems. We use $RMM(n_1, n_2, n_3)$ to denote (the complexity of) the problem of multiplying an $n_1 \times n_2$ matrix by an $n_2 \times n_3$ matrix.

We denote by $SRMM(n_1, n_2, n_3, m_1, m_2)$ the problem of computing AB when A is an $n_1 \times n_2$ matrix with at most m_1 non-zero entries, and B is an $n_2 \times n_3$ matrix with at most m_2 non-zero entries. Again, we abuse notation so that RMM, $SRMM$ refer both to the problems and to (upper bounds on) their respective complexities. We also use the following notations:

$$\begin{aligned} &SRMM(n_1, *, n_3, m_1, m_2) = \\ &SRMM(n_1, max(m_1, m_2), n_3, m_1, m_2) \end{aligned}$$

(this means that we practically impose no restriction on n_2), and

$$SRMM(n_1, *, n_3, m) = SRMM(n_1, *, n_3, m, m) \ .$$

The ring containing the elements of A and B is not specified in this notation. We usually use the 2-element *Boolean ring*, and sometimes the real field. In most cases, the specific choice of the ring is immaterial, because the complexity of all known algorithms for matrix multiplication does not seem to be different for different rings (for example, no better algorithms are known for the Boolean ring than for the reals). We will sometimes denote the problems as RMM_{Bool} or $SRMM_{Bool}$, when we want to stress the fact that we are using the Boolean ring.

Throughout, we will assume that the input matrices are given to us in sparse representation, that is as a collection of triplets $(row, column, value)$ which lists all positions where the matrix is non-zero (in the Boolean ring, the first two components suffice). Thus, the size of the representation of an input matrix is linear in the number $m_1 + m_2$ of non-zero entries of the matrices.

Bounds for Matrix Multiplication.

Let ω be the matrix multiplication exponent, i.e. ω is the smallest number such that $RMM(n, n, n) = O(n^\omega)$. Let

$\alpha = \max\{0 \le r \le 1 \mid RMM(n, n^r, n) = O(n^2)\}$, and let $\beta = \frac{\omega-2}{1-\alpha}$. It is well known that:

THEOREM 2.1 (COPPERSMITH AND WINOGRAD [9]). *$\omega < 2.376$.*

THEOREM 2.2 (COPPERSMITH [8]). *$\alpha > 0.294$.*

THEOREM 2.3 (COPPERSMITH [8], HUANG AND PAN [13]).

$$RMM(n_1, n_2, n_1) = O\left(\begin{cases} n_1^{\omega-1} n_2 & \text{if } n_2 \ge n_1 \\ n_1^{2-\alpha\beta} n_2^{\beta} + n_1^2 & \text{if } n_2 \le n_1 \end{cases}\right).$$

Setting $\omega = 2.376$ and $\alpha = 0.294$ gives $\beta \approx 0.533$.

We note that in recent years there have been some developments in algorithms for matrix multiplication. Cohn et al. [7, 6] propose a new approach to matrix multiplication, which is based on group theory. They give a framework which is different (and perhaps deeper) than that of previous methods. They in a relatively simple way derive an exponent of 2.41 (which still falls short of the current $\omega = 2.376$), and claim to have unpublished results that re-derive the known matrix multiplication exponent 2.376, eliminating some of the complications of the original algorithm.

2.1 Sparse Rectangular Matrix Multiplication

We use the following lemma in the proof of Lemma 3.2:

LEMMA 2.4. *Any instance of $SRMM(n_1, n_2, n_1, m)$ can be reduced to an instance of $SRMM(2n_1, n_2, 2n_1, 2m)$, in which $B = A^T$. The reduction takes linear time.*

Proof. Given an instance (A_1, B_1) of $SRMM(n_1, n_2, n_1, m)$, where B_1 is not necessarily equal to A_1^T, define a new $2n_1 \times n_2$ matrix A which consists of A_1 placed on top of B_1^T, i.e. $A = \begin{pmatrix} A_1 \\ B_1^T \end{pmatrix}$. Then $AA^T = \begin{pmatrix} A_1A_1^T & A_1B_1 \\ B_1^TA_1^T & B_1^TB_1 \end{pmatrix}$ and the upper-right quadrant of AA^T is equal to A_1B_1. □

The following theorem upper bounds $SRMM(n, *, n, m_1, m_2)$ and $SRMM(n, *, n, m)$. The proof is an extension of the one in [20], and is given here for the sake of completeness. With some additional care and case-analysis, one can generalize these bounds to $SRMM(n_1, n_2, n_3, m_1, m_2)$.

THEOREM 2.5.

(i) $SRMM(n, *, n, m) =$

$$O\left(\begin{cases} mn^{\frac{\omega-1}{2}} & \text{if } m \ge n^{\frac{\omega+1}{2}}, \\ m^{\frac{2\beta}{\beta+1}} n^{\frac{2-\alpha\beta}{\beta+1}} & \text{if } n^{1+\frac{\alpha}{2}} \le m \le n^{\frac{\omega+1}{2}}, \\ n^2 & \text{if } m \le n^{1+\frac{\alpha}{2}} \end{cases}\right).$$

(ii) $SRMM(n, *, n, m_1, m_2) =$

$$O\left(\begin{cases} (m_1m_2)^{1/2} n^{\frac{\omega-1}{2}} + m_1 + m_2 & \text{if } m_1m_2 \ge n^{\omega+1}, \\ (m_1m_2)^{\frac{\beta}{\beta+1}} n^{\frac{2-\alpha\beta}{\beta+1}} + m_2 + m_2 & \text{if } n^{2+\alpha} \le m_1m_2 \le n^{\omega+1}, \\ n^2 + m_1 + m_2 & \text{if } m_1m_2 \le n^{2+\alpha} \end{cases}\right)$$

Proof. The first part is a corollary of the second, setting $m_1 = m_2 = m$, and noticing that the term $m_1 + m_2$ in each of these bounds is dominated by the other term of the bound when $m_1 = m_2$. Therefore, we consider only the second part.

Suppose that the columns of A and the corresponding rows of B are indexed by the elements of a set I. Partition I into two subsets I_H and I_L, referred to as the *heavy* indices and the *light* indices, respectively, as follows. Denote by $|v|$ the number of nonzero elements in a vector v, and choose a threshold parameter k that we will fix later. An index i is in I_H if $|A_{*i}| \cdot |B_{i*}| \ge k$, where A_{*i} denotes the i^{th} column of A, and B_{i*} denotes the i^{th} row of B; otherwise $i \in I_L$.

Partition the matrix A into two matrices, A^H and A^L, such that A^H (resp., A^L) consists of the columns of A which are indexed by I_H (resp., I_L) in their original order. Likewise, partition B into B^H and B^L, such that B^H (resp., B_L) consists of the heavy (resp., light) rows of B in their original order. Clearly, $AB = A^HB^H + A^LB^L$, and we therefore consider each of the two multiplications A^HB^H and A^LB^L separately.

We compute A^LB^L in the following brute-force manner. Put $x_i = |A_{*i}|$, $y_i = |B_{i*}|$, for $i \in I_L$. For each $i \in I_L$, we compute the product of the i^{th} column of A and the i^{th} row of B. This is an $n \times n$ matrix $C^{(i)}$ satisfying $C^{(i)}_{uv} = A_{ui}B_{iv}$. We fill in only the nonzero entries of $C^{(i)}$, whose number is $x_iy_i \le k$, in $O(x_iy_i)$ time. We bound the sum of these costs over all light columns, as follows. Choose another threshold parameter t, and write

$$\sum_{i \in I_L} x_iy_i = \sum_{i \in I_L,\, x_i \le t} x_iy_i + \sum_{i \in I_L,\, x_i > t} x_iy_i \ .$$

The first sum is at most $\sum_{i \in I_L} ty_i \le tm_2$. For the second sum, since $x_iy_i \le k$, we have $y_i \le k/t$ for each i in that sum, so the sum is at most $\sum_{i \in I_L} kx_i/t \le km_1/t$. (We have used the fact that A (resp., B) contains at most m_1 (resp., m_2) nonzero entries.) Optimizing for t, we conclude that A^LB^L can be computed in $O(\sqrt{km_1m_2})$ time.

We compute A^HB^H using the fast matrix multiplication algorithms of Theorem 2.3. We claim that $|I_H| \le \sqrt{m_1m_2/k}$. Indeed, as above, put $x_i = |A_{*i}|$, $y_i = |B_{i*}|$, for $i \in I_H$. We have $x_iy_i \ge k$ for each $i \in I_H$, and $\sum_{i \in I_H} x_i \le m_1$, $\sum_{i \in I_H} y_i \le m_2$. Hence

$$m_1m_2 \ge \left(\sum_{i \in I_H} x_i\right) \cdot \left(\sum_{i \in I_H} y_i\right) \ge$$

$$k\left(\sum_{i \in I_H} x_i\right) \cdot \left(\sum_{i \in I_H} \frac{1}{x_i}\right) \ge k|I_H|^2,$$

as follows from the arithmetic-harmonic inequality (or simply from the Cauchy-Schwarz inequality). This implies the claim. Hence, the cost of computing A^HB^H is at most

$$RMM(n, |I_H|, n) \le RMM(n, \sqrt{m_1m_2/k}, n).$$

Thus the total cost of computing AB is at most

$$RMM(n, \sqrt{m_1m_2/k}, n) + O\left(\sqrt{m_1m_2k}\right) + O(m_1 + m_2), .$$

where the last term bounds the steps of processing the input, finding the heavy and light indices, and partitioning the matrices into their heavy and light parts. We choose the optimal k by case analysis, depending on the value of m_1m_2. Specifically, using Theorem 2.3, we have: If $m_1m_2 \ge n^{\omega+1}$ then we choose $k = n^{\omega-1}$, and obtain the first bound. If $m_1m_2 \le n^{\omega+1}$, we choose $k = (m_1m_2)^{\frac{\beta-1}{\beta+1}} n^{\frac{2(2-\alpha\beta)}{\beta+1}}$. This gives us a running time of

$$O((m_1m_2)^{\frac{\beta}{\beta+1}} n^{\frac{2-\alpha\beta}{\beta+1}} + n^2 + m_1 + m_2),$$

which means that we get either the second or the third bound of the theorem, depending on whether or not $m_1 m_2 \geq n^{2+\alpha}$.

We remark that the final term in the running time, $m_1 + m_2$, may dominate the running time, when one of $m_1/m_2, m_2/m_1$ is very large compared to n. □

Computing the union of cliques or bi-cliques.

As discussed in the introduction, matrix multiplication is related to problems in which we need to eliminate duplicates efficiently. The following two examples, on which all our geometric applications are based, are in fact matrix multiplication in disguise.

LEMMA 2.6. *(i) Let $\{K_i\}$ be a set of cliques over a vertex set V where $|V| = n$. Define the* size *of a clique to be the number of vertices it contains. Assume that the total size of the cliques is m. Then the problem of computing the union of the cliques is equivalent to $SRMM_{Bool}(n, *, n, m)$.*

(ii) Let $\{K_i\}$ be a set of bi-cliques (that is, complete bipartite graphs) whose left side is a subset of some vertex set V_1, and whose right side is a subset of another vertex set V_2, where $|V_1| = |V_2| = n$. Suppose that the sum of the sizes of the left (resp., right) sides of the bi-cliques is m_1 (resp., m_2). Then the problem of computing the union of the bi-cliques is equivalent to
$SRMM_{Bool}(n, *, n, m_1, m_2)$.

Proof. It suffices to consider (ii). Given the collection of bi-cliques, construct two Boolean matrices A, B, where A is an $n \times k$ matrix and B is a $k \times n$ matrix, where k is the number of bi-cliques, so that $A_{ij} = 1$ if the i^{th} element of V_1 belongs to the j^{th} bi-clique, and $B_{jl} = 1$ if the l^{th} element of V_2 belongs to the j^{th} bi-clique. It is now obvious that the union of the bi-cliques is represented by the 1-entries of AB. The converse transformation follows from the same argument in reverse. □

3. SEGMENTS ON THE LINE AND ISO-ORIENTED BOXES IN HIGHER DIMENSIONS

In this section we give two solutions to $CI_{Segments,\mathbb{R}^1}(c, n)$. In Section 3.1 we give a simple solution, which exhibits some similarity with the algorithm for solving $SRMM$ that is presented in Theorem 2.5. In Section 3.2 we show that $CI_{Segments,\mathbb{R}^1}(c, n)$ is in fact equivalent, up to poly-logarithmic factors, to $SRMM_{Bool}(c, *, c, n)$.

The latter result is of interest because it shows that we can use any algorithm for $SRMM_{Bool}$ as a black box to get an algorithm for $CI_{Segments,\mathbb{R}^1}(c, n)$, whereas in the simple solution we, in a sense, simulate the algorithm of Theorem 2.5 and so use it in an explicit, non-black-box manner.

3.1 An Explicit Solution via RMM

THEOREM 3.1. *$CI_{Segments,\mathbb{R}^1}(c, n)$ can be solved in time*

$$O\left(\begin{cases} nc^{\frac{\omega-1}{2}} & \textit{if } n \geq c^{\frac{\omega+1}{2}}, \\ n^{\frac{2\beta}{\beta+1}} c^{\frac{2-\alpha\beta}{\beta+1}} & \textit{if } c^{1+\frac{\alpha}{2}} \leq n \leq c^{\frac{\omega+1}{2}}, \\ c^2 & \textit{if } n \leq c^{1+\frac{\alpha}{2}} \end{cases}\right).$$

Note: The expression for the running time in the last Theorem is the same as the upper bound shown in Theorem 2.5 for $SRMM_{Bool}(c, *, c, n)$. This is because, as just noted, we essentially "simulate" the $SRMM$ algorithm of Theorem 2.5.

Proof. Assume without loss of generality that all segments of the same color are disjoint. We can also assume that the segments have nonempty interiors, and that their endpoints are all distinct, because otherwise it is easy to perturb the endpoints to achieve this property, so that all intersections remain and no new intersections are created. We can sort the endpoints, and assume that all endpoints are integers between 1 and $2n$, and that every integer between 1 and $2n$ is the endpoint of exactly one segment.

Let k be a parameter which we fix later. We divide the points into $2n/k$ consecutive *blocks*, $B_1, \ldots, B_{2n/k}$, so that each block consists of at most k points. We are interested in the $2n/k$ endpoints of the blocks, which we call *cutoff points*.

We distinguish between two types of intersections between pairs of the segments: (i) two segments that contain a common cutoff point; (ii) two segments that have an endpoint in the same block. It is easy to see that each pair of intersecting segments are at least of one of these types (they may be of both types). Therefore it suffices to detect all intersections of type (i) and all intersections of type (ii).

To detect intersections of type (ii), we process each block B_i, go over all pairs of segments such that both have an endpoint in B_i and check whether they intersect each other. There are at most k such segments, so we perform $O(k^2)$ work per block, and in total $2n/k \cdot O(k^2) = O(nk)$ work.

To detect intersections of type (i), we define a $c \times 2n/k$ Boolean matrix A whose rows correspond to the c colors, and whose columns correspond to the $2n/k$ cutoff points. We put $A_{ij} = 1$ if there is a segment of color i that contains the j^{th} cutoff point, and $A_{ij} = 0$ otherwise. The matrix A can be constructed in time $O(cn/k + n \log n)$. Computing AA^T gives a Boolean $c \times c$ matrix which has 1 in location (c_1, c_2) if and only if there is a segment s_1 of color c_1 and a segment s_2 of color c_2 such that s_1 and s_2 have an intersection of type (i). This takes time $O(RMM_{Bool}(c, n/k, c))$. Using the bounds in Theorem 2.3, and optimizing over k, one gets the required expression for the running time, as is easily checked. □

Remark. Note that in the proof of Theorem 3.1, in order to detect intersections of type (ii) we essentially ran the naive algorithm that goes over all intersections of segments. The step in the proof where we saved work relative to the naive $O(n^2)$ algorithm is in the detection of the intersections of type (i). In this step, the use of matrix multiplication enabled us to aggregate information about the intersecting *pairs of colors* instead of running over all intersecting *pairs of segments*. Note also that using cutoff points is, in a sense, analogous to distinguishing between light and heavy columns/rows in the proof of Theorem 2.5.

3.2 Equivalence to $SRMM$

Consider the problem $CI_{Segments,\mathbb{R}^1}(c, n)$. It is easily seen that by taking an input where all segments are points (i.e., closed segments of length 0) and using Lemmas 2.4 and 2.6, we get that a special case of $CI_{Segments,\mathbb{R}^1}(c, n)$ is equivalent to $SRMM_{Bool}(c, *, c, n)$. Thus:

LEMMA 3.2.

$$CI_{Segments,\mathbb{R}^1}(c, n) = \Omega(SRMM_{Bool}(c, *, c, n)) \ .$$

In what follows we show that this is tight up to a polylogarithmic multiplicative factor:

THEOREM 3.3.

$$CI_{Segments,\mathbb{R}^1}(c,n) = O(SRMM_{Bool}(c,*,c,n\log n)) \ .$$

This easily follows from using our technique together with the following Lemma:

LEMMA 3.4. *Let S be a set of n segments on the real line. Then their intersection graph $G(S)$ can be represented as the disjoint union of bi-cliques of total size $O(n\log n)$. These bi-cliques can be found in time $O(n\log n)$.*

Proof. Construct a segment tree over the given segments (see [10]). For each node ξ of the tree, construct a bi-clique, consisting of the segments stored at ξ on one side, and the segments whose left endpoints are stored at the subtree rooted at ξ, on the other side. It is easy to see that each edge of the bi-clique corresponds to a pair of intersecting segments. Furthermore, for each pair of intersecting segments, there is a unique bi-clique in which they correspond to an edge. Hence $G(S)$ is equal to the union of these bi-cliques. It then follows that the sum of the sizes of the bi-cliques is $O(n\log n)$, and that they can all be constructed in $O(n\log n)$ time. $\square$

Using a similar and slightly modified approach, we can extend this result to any fixed dimension $d > 1$:

LEMMA 3.5. *Let S be a set of n iso-oriented boxes in $\mathbb{R}^d$. Then their intersection graph $G(S)$ can be represented as the union of bi-cliques of total size $O(n\log^d n)$. These bi-cliques can be found in time $O(n\log^d n)$.*

It can be seen that Lemma 3.4 is tight: there exist sets S of segments for which $G(S)$ has no representation as a union of bi-cliques of smaller order of magnitude. Such a set is the set S which consists of all segments of the form $I_i = [i, i + n/2]$ for $1 \le i \le n$. Here, the (bipartite) intersection graph between the segments $I_1, \ldots, I_{n/2}$ and $I_{n/2+1}, \ldots, I_n$ is the bipartite graph H where I_i is connected to $I_{j+n/2}$ if and only if $j \le i$. Any cover of $G(S)$ by bi-cliques is also a cover of H. Erickson [11, Theorem 3.1] showed that covering H by bi-cliques of total size $o(n\log n)$ is impossible.

Lemma 3.4 together with Lemma 2.6 immediately give the proof of Theorem 3.3.

Note: It is possible to generalize the results of this section to show that $CI_{Segments,\mathbb{R}^1}(c,n,m)$ is equivalent, up to a poly-logarithmic factor, to $SRMM_{Bool}(c,*,c,n,m)$.

Generalizations. Lemma 3.5 implies that CI of iso-oriented boxes in any fixed dimension is equivalent, up to polylogarithmic factors, to $SRMM_{Bool}(c,*,c,n)$. This also holds for a large variety of other iso-oriented objects.

Remark: One of the motivations that have led to this study has been the problem of guarding a terrain by two watchtowers, as reviewed in the Introduction (also see [1]). The bottleneck step in the solution of this problem is a CI problem on the line, involving $O(n^3)$ segments with n colors (where n is the number of vertices of the terrain). Theorem 3.1 then gives a solution that runs in time $O(n^{3+(\omega-1)/2}) \approx O(n^{3.688})$. However, using a different, more global geometric approach, which exploits the special structure of these segments of invisibility, Agarwal et al. [1] obtain a slightly improved solution with a running time of roughly $n^{11/3}$. We also note that an indirect use of matrix multiplication for terrain guarding in two dimensions was given earlier in [2].

4. COLORED INTERSECTION OF POINTS AND HALFSPACES

In this section we consider the problem $CI_{Points,Halfplanes,\mathbb{R}^2}(c,n,m)$, and, more generally, $CI_{Points,Halfspaces,\mathbb{R}^d}(c,n,m)$, for any fixed d. In Section 4.1 we give some necessary background on cuttings. We then show how to use cuttings to decompose the intersection graph of points and halfspaces into the union of bi-cliques. We then use these results in Section 4.2 to obtain an efficient solution of $CI_{Points,Halfplanes,\mathbb{R}^2}(c,n,n)$. For simplicity of exposition, we only present a detailed solution of the problem in $\mathbb{R}^2$, and only for the balanced case $n = m$. We then state (without proof) the extension to higher dimensions. We also mention several applications of these results. More details, including the handling of the unbalanced case $n \ne m$, are given in the full version.

4.1 Cuttings and Decomposition of the Intersection Graph

Let S and T be two sets of geometric objects. As above, the *intersection graph* $G(S,T)$ of S and T is the bipartite graph with S as the set of its left vertices, T as the set of its right vertices, and an edge connecting $s \in S$ and $t \in T$ if s and t intersect each other. In this section, we consider the case where S is a collection of n points in $\mathbb{R}^d$ and T is a collection of n halfspaces (bounded by hyperplanes) in $\mathbb{R}^d$. We denote by H the set of hyperplanes bounding the halfspaces in T. As in the iso-oriented case, we want to represent $G(S,T)$ as a union of bi-cliques. As it turns out, a full representation of this kind is too expensive, and we can improve the solution if we use only the following representation, which decomposes the graph into a collection of bi-cliques, and leaves a sparse residual graph that is dealt with separately.

The representation is based on the *hierarchical cutting* technique of Chazelle [4] (see also Matoušek [17]). We recall the definition of an *efficient hierarchical* $(1/r)$*-cutting* for H, with respect to constant parameters C, ρ. It is a sequence of cuttings $\Xi_0, \Xi_1, \ldots, \Xi_k$, such that (i) Ξ_0 is the cutting consisting of the single "simplex" $\mathbb{R}^d$; (ii) each Ξ_i, for $1 \le i \le k$, is a $(1/\rho^i)$-cutting for H (each of its cells is crossed by at most $|H|/\rho^i$ hyperplanes of H), of $O(\rho^{id})$ simplices, which C-refines Ξ_{i-1}, meaning that each simplex of Ξ_i is contained in a single simplex of Ξ_{i-1}, and each simplex of Ξ_{i-1} contains at most C simplices of Ξ_i; and (iii) $\rho^{k-1} < r \le \rho^k$ (so $k = \Theta(\log r)$).

THEOREM 4.1 (CHAZELLE [4]; CF. MATOUŠEK [17]). *There exist constants C, ρ that depend on d, such that, for any sets S of n points in $\mathbb{R}^d$ and H of n halfplanes in $\mathbb{R}^d$, and for any parameter r, there exists an efficient hierarchical $(1/r)$-cutting for H, with respect to the parameters C and ρ, such that each cell of the i^{th} cutting contains at most n/ρ^{id} points of S. Such a cutting can be constructed deterministically in time $O(nr^{d-1})$.*

As a corollary, we obtain:

Theorem 4.2. *Let S, T, and $r \le n^{1/(d+1)}$ be as above. Then $G(S,T)$ can be represented as $\left(\bigcup_{i=1}^{cr^{2d}} S_i \times T_i\right) \cup G_0(S,T)$, for some absolute constant c, where the residual graph $G_0(S,T)$ has $O(n^2/r^2)$ edges, and $\sum_i |S_i|,\ \sum_i |T_i| = O(nr^{d-1}\log r)$. This decomposition can be constructed deterministically in time $O(nr^{d-1}\log r + n^2/r^2)$.*

Proof. Construct an efficient hierarchical $(1/r)$-cutting for the set H of hyperplanes bounding the halfspaces in T, with respect to the constants C, ρ provided in Theorem 4.1, and denote the resulting sequence of cuttings as $\Xi_0, \Xi_1, \ldots, \Xi_k$. For each Ξ_i, $i \ge 1$, and for each simplex τ of Ξ_i, construct a bi-clique $S_\tau \times T_\tau$, where S_τ is the set of points that lie in τ, and T_τ is the set of halfspaces that fully contain τ. For each i we have $\sum_\tau |S_\tau| = n$, and (recalling that ρ is a constant)

$$\sum_\tau |T_\tau| = O(\rho^{id} \cdot n/\rho^{i-1}) = O(\rho n \rho^{i(d-1)}) = O(n\rho^{i(d-1)}) \ .$$

Summing these bounds over i, we get an overall bound of $O(n\rho^{k(d-1)}) = O(nr^{d-1})$ for T-sizes of the bi-cliques.

We next continue to decompose the residual graph $G_0(S,T)$ using duality. We note that if $(s,t) \in G(S,T)$ is an intersection that does not appear in the union of the bi-cliques constructed so far, then there must exist a cell τ of the final cutting Ξ_k, such that s lies in τ and the hyperplane that bounds t crosses τ. For each such τ, we pass to the dual space, where the points of S that lie in τ become halfspaces (we denote by H^*_τ the set of their bounding hyperplanes), and the halfspaces whose bounding hyperplanes cross τ become points (we denote by T^*_τ the set of these points).

We apply Theorem 4.1 to H^*_τ and T^*_τ, to obtain an efficient hierarchical $(1/r)$-cutting for H^*_τ, with the above properties. Proceeding as above, we obtain a collection of bi-cliques, so that the sum of the sizes of their vertex sets that are contained in H^*_τ is $O(|H^*_\tau| r^{d-1})$, and the sum of the sizes of their vertex sets that are contained in T^*_τ is $O(|T^*_\tau|\log r)$. Summing these bounds over all cells τ of Ξ_k, and using the facts that $\sum_\tau |H^*_\tau| = n$ and $\sum_\tau |T^*_\tau| = O(nr^{d-1})$, we obtain that the total size of the vertex sets of all the bi-cliques constructed so far is $O(nr^{d-1}\log r)$.

We are left with $O(r^{2d})$ cells of all the dual cuttings, each intersected by at most n/r^{d+1} hyperplanes of the respective set H^*_τ, and containing at most n/r^{d+1} points of the respective set T^*_τ. Arguing as above, one can show that any containment between a point $s \in S$ and a halfspace $t \in T$ that is not represented in any of the bi-cliques constructed so far, must be such that there exists a cell σ of one of the dual cuttings, such that the hyperplane bounding the halfspace dual to s crosses σ, and the point dual to t lies in σ. Hence, the size of the residual graph is at most $O(r^{2d} \cdot (n/r^{d+1})^2) = O(n^2/r^2)$. The time bound for constructing this decomposition is an easy consequence of Theorem 4.1. This completes the proof. $\Box$

4.2 Solving $CI_{Points,Halfplanes,\mathbb{R}^2}(c,n,n)$

For simplicity of presentation, we restrict the foregoing analysis to the planar case $d = 2$ and to the balanced case $n = m$, and establish the following main result.

Theorem 4.3.

$$CI_{Points,Halfplanes,\mathbb{R}^2}(c,n,n) = \tilde{O}\left(\begin{cases} n^{4/3}c^{\frac{\omega-1}{3}} & \text{if } n \ge c^{\frac{1+2\omega}{4}}, \\ n^{\frac{4\beta}{2\beta+1}}c^{\frac{2-\alpha\beta}{2\beta+1}} & \text{if } c^{1+\frac{\alpha}{4}} \le n \le c^{\frac{1+2\omega}{4}}, \\ c^2 & \text{if } n \le c^{1+\frac{\alpha}{4}} \end{cases}\right) .$$

Proof. As above, let S and T be the given sets of n points and n halfplanes in $\mathbb{R}^2$, respectively. For an appropriate choice of r, that we will fix later, construct the decomposition $G(S,T) = \left(\bigcup_{i=1}^{cr^4} S_i \times T_i\right) \cup G_0(S,T)$, where $\sum_i |S_i|,\ \sum_i |T_i| = O(nr\log r)$, and $|E(G_0(S,T))| = O(n^2/r^2)$. Recall that this can be done in time $O(nr\log r + n^2/r^2)$. We then compute the union of bi-cliques $\bigcup_{i=1}^{cr^4} S_i \times T_i$, using Lemma 2.6, in time $O(SRMM_{Bool}(c,*,c,m))$, where $m = O(nr\log r) = \tilde{O}(nr)$. By Theorem 2.5(i), computing the union takes time

$$\tilde{O}\left(\begin{cases} nrc^{\frac{\omega-1}{2}} & \text{if } nr \ge c^{(\omega+1)/2}, \\ (nr)^{\frac{2\beta}{\beta+1}}c^{\frac{2-\alpha\beta}{\beta+1}} & \text{if } c^{1+\alpha/2} \le nr \le c^{(\omega+1)/2}, \\ c^2 & \text{if } nr \le c^{1+\alpha/2} \end{cases}\right) .$$

For the total running time, we add $O(n^2/r^2)$ to this bound, and optimize the choice of r as follows. For the first case we choose $r = n^{1/3}/c^{(\omega-1)/6}$, and obtain the first bound in the theorem. For the second and third case we choose $r = n^{1/(2\beta+1)}/c^{(2-\alpha\beta)/2(2\beta+1)}$, and obtain the second or the third bound, depending or whether or not $n \ge c^{1+\alpha/4}$. $\Box$

Remark. An alternative approach is to obtain a *complete* decomposition of $G(S,T)$ into the union of bi-cliques, by choosing $r = n^{1/(d+1)} = n^{1/3}$ in Theorem 4.2. However, this is too expensive: The overall size of the resulting bi-cliques is $O(n^{4/3}\log n)$, and then the first bound of Theorem 2.5 is only $O(n^{4/3}c^{(\omega-1)/2})$, which is weaker than our solution (the same happens for the second bound). This should be contrasted with the 1-dimensional case, where the solution that we obtained is based on a complete decomposition.

Extension to higher dimensions. Using the decomposition given by Theorem 4.2 for any fixed $d > 2$, we obtain

Theorem 4.4.

$$CI_{Points,Halfspaces,\mathbb{R}^d}(c,n,n) = \tilde{O}\left(\begin{cases} n^{2d/(d+1)}c^{\frac{\omega-1}{d+1}} & \text{if } n \ge c^{\frac{1+d\omega}{2d}}, \\ n^{\frac{2d\beta}{d\beta+1}}c^{\frac{2-\alpha\beta}{d\beta+1}} & \text{if } c^{1+\frac{\alpha}{2d}} \le n \le c^{\frac{1+d\omega}{2d}}, \\ c^2 & \text{if } n \le c^{1+\frac{\alpha}{2d}} \end{cases}\right) .$$

Generalizations. The cases of points and halfplanes in $\mathbb{R}^2$ and of points and halfspaces in $\mathbb{R}^d$ are only two simple applications of the technique. We can apply it to any other kind of batched range searching with colored objects. For example, we can consider the CI problems involving segments in the plane, triangles and points in the plane, triangles in $\mathbb{R}^3$, $(d-1)$-simplices in $\mathbb{R}^d$, a variety of ray-shooting problems in $\mathbb{R}^2$ and $\mathbb{R}^3$, and so on. The running time bounds for the first two problems (segments in the plane, and points and triangles in the plane) are asymptotically the same as those given in Theorem 4.3.

As an interesting application, consider the problem where we are given c arbitrary polygonal regions in the plane with a total of n edges, and wish to report all intersecting pairs of these regions. We note that a pair of regions intersect if either an edge of one of them intersects an edge of the other, or a vertex of one of them is contained in a triangle of some triangulation of the other. We then transform these two latter problems into CI problems, where we assign to each region a distinct color, and color all its edges, vertices, and triangles by that color. We obtain two colored intersection searching problems, one involving n line segments in the plane with c colors, and the other involving n points and $O(n)$ triangles, again with c colors. It is easy to see that both problems can be solved within the same time bounds as in Theorem 4.3, so we can report all intersecting pairs of the given polygons within a time bound as given in Theorem 4.3. A similar technique solves the problem of reporting all intersecting pairs in a collection of c arbitrary polyhedral regions in $\mathbb{R}^3$ with a total of n facets. We defer the detailed study of this and other extensions to the full version of the paper.

5. CONCLUSION

In this paper we have shown how to use efficient algorithms for matrix multiplication, combined with geometric decomposition techniques, to obtain efficient algorithms for batched colored intersection problems.

One obvious question raised by our study is whether our bounds for the iso-oriented CI problems and non-iso-oriented CI problems are optimal. An immediate way to improve our bounds is to improve the bounds for $SRMM$, or, more profoundly, for the raw matrix multiplication problem itself. The (near-)equivalence of the CI problem on the line and $SRMM$ suggests that a faster algorithm for CI may lead to a faster algorithm for $SRMM$. It would indeed be very interesting if new direct geometric methods could yield an improved solution for $SRMM$.

To get a better feeling for the complexity of $SRMM$, let us assume that $\omega = 2$. That is, assume that multiplying two $n \times n$ matrices takes $\Theta(n^2)$ time, and multiplying an $n_1 \times n_2$ matrix by an $n_2 \times n_3$ matrix takes $\Theta(n_1n_2 + n_2n_3 + n_1n_3)$ time, which is proportional to the sum of the sizes of the input and output. However, even under this optimistic assumption, our algorithm for $SRMM$ only gives $SRMM_{Bool}(n, *, n, m) = \tilde{O}(m\sqrt{n} + n^2)$, as opposed to the lower bound of $\Omega(m + n^2)$ which is the sum of the sizes of the input and output. An interesting open question is whether a matching lower bound $\Omega(m\sqrt{n} + n^2)$ can be established in some computational model. It may be helpful, as a first step, to assume that any column has exactly $\sqrt{n}$ 1s. Such a lower bound may support the hypothesis that the dependence of our algorithm for $SRMM$ on ω is optimal.

One might also like to consider the output-sensitive variant of CI where we want an algorithm which runs faster if the number of intersecting pairs of colors is small (see [3]). To use our techniques for this problem, a natural subtask is to develop an output-sensitive algorithm for $SRMM$, which runs faster when the output matrix has a small number of non-zero entries.

6. ACKNOWLEDGEMENTS

We thank Yoav Giyora and Uri Zwick for helpful discussions.

7. REFERENCES

[1] P. K. Agarwal, S. Bereg, O. Daescu, H. Kaplan, S. Ntafos, and B. Zhu. Guarding a terrain by two watchtowers. In *Proc. 21st Annu. ACM Sympos. Computational Geometry*, pages 346–355, New York, NY, USA, 2005. ACM Press.

[2] B. Ben-Moshe, P. Carmi, and M. J. Katz, Computing all large sums-of-pairs in $\mathbb{R}^n$ and the discrete planar two-watchtower problem, *Inform. Process. Lett.* 89 (2004), 137–139,

[3] P. Bozanis, N. Kitsios, C. Makris, and A. K. Tsakalidis. Red-blue intersection reporting for objects of non-constant size. *Comput. J.*, 39(6):541–546, 1996.

[4] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9(2):145–158, 1993.

[5] B. Chazelle. *Handbook of Data Structures and Applications*, chapter 25, Cuttings, pages 25.1–25.10. CRC Press, 2005.

[6] H. Cohn, R. D. Kleinberg, B. Szegedy, and C. Umans. Group-theoretic algorithms for matrix multiplication. In *Proc. 46th Annu. IEEE Sympos. Foundat. Comput. Sci. (FOCS '05)*, pages 379–388.

[7] H. Cohn and C. Umans. A group-theoretic approach to fast matrix multiplication. In *Proc. 44th Annu. IEEE Sympos. Foundat. Comput. Sci. (FoOCS '03)*, pages 438–449.

[8] D. Coppersmith. Rectangular matrix multiplication revisited. *J. Complexity*, 13(1):42–49, 1997.

[9] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990.

[10] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications.* 2nd Edition, Springer-Verlag, Berlin, 2000.

[11] J. Erickson. New lower bounds for hopcroft's problem. *Discrete and Computational Geometry*, 16(4):389–418, 1996.

[12] P. Gupta, R. Janardan, and M. Smid. *Handbook of Data Structures and Applications*, chapter 64, Computational geometry: generalized intersection searching, pages 64.1–64.17. CRC Press, 2005.

[13] X. Huang and V.Y. Pan. Fast rectangular matrix multiplication and applications. *J. Complexity*, 14(2):257–299, 1998.

[14] E. Kaltofen. An output-sensitive variant of the baby steps/giant steps determinant algorithm. In *Proc. Internat. Sympos. Symbolic Algeb. Comput. (ISSAC '02)*, 2002, pages 138–144.

[15] D. Kratsch and J. Spinrad. Between $o(nm)$ and $o(n^{\alpha})$. In *Proc. 14th Annu. ACM-SIAM Sympos. Discrete Algo. (SODA '03)*, 2003, pages 709–716.

[16] L. Lee. Fast context-free grammar parsing requires fast boolean matrix multiplication. *J. ACM*, 49(1):1–15, 2002.

[17] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.*, 10(2):157–182, 1993.

[18] E. Mossel, R. O'Donnell, and R. A. Servedio. Learning functions of k relevant variables. *J. Comput. Syst. Sci.*, 69(1):421–434, 2004.

[19] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

[20] R. Yuster and U. Zwick. Fast sparse matrix multiplication. In *Proc. Europ. Sympos. Algorithms (ESA)*, pages 604–615, 2004.

[21] R. Yuster and U. Zwick. Answering distance queries in directed graphs using fast matrix multiplication. In *Proc. 46th Annu. IEEE Sympos. Foundat. Comput. Sci. (FOCS '05)*, 2005, pages 389–396.