J. Symbolic Computation (1990) 10, 281-309

An Algorithm for Generalized Point Location and Its Applications

BERNARD CHAZELLE(1) AND MICHA SHARIR(2)

(1) Dept. of Computer Science, Princeton University, Princeton, NJ 08544, U.S.A.
(2) School of Mathematical Sciences, Tel Aviv University, Tel Aviv, Israel.

(Received 20 May 1988)

We show that Collins' classical quantifier elimination procedure contains most of the ingredients for an efficient point location algorithm in higher-dimensional space. This leads to a polynomial-size data structure that allows us to locate, in logarithmic time, a point among a collection of real algebraic varieties of constant maximum degree, assuming that the dimension of the ambient space is fixed. This result has theoretical bearings on a number of optimization problems posed in the literature. It also gives a method for solving multidimensional searching problems in polynomial space and logarithmic query time.

1. Introduction

The central theme of multidimensional searching is the organization of a database to which queries of a chosen type can be made. The term locus approach refers to the particular strategy that regards a query as a point in higher-dimensional space: the idea is to subdivide the query space into equivalence classes and thus reduce queryanswering to point location. This approach was followed by Dobkin & Lipton (1976), who devised an efficient searching algorithm for hyperplanes in $E^d$, for any fixed $d$. Yao & Yao (1985) have observed that the constraint manifold can usually be made linear by throwing in additional variables, if needed. Considering (as will be shown below) that the
preprocessing is doubly exponential in the number of variables, however, the linearization method may not always be so desirable. We will show here that Dobkin and Lipton's method can be generalized directly to handle arbitrary algebraic varieties. The generalization is based on Collins' cylindrical algebraic decomposition (Collins, 1975, Aarno et al., 1984a,b). The main idea is to transform Collins' algorithm into a data structure and add various bells and whistles to support fast searching.

For a precise statement of our results we need to introduce a few notions. Let $F = \{P_1, \ldots, P_n\}$ be a set of $n$ $d$-variate polynomials with rational coefficients and norm-length at most $\ell$. (The norm-length of a polynomial with integral coefficients is the number of bits needed to represent the sum of the absolute values of its coefficients. If the coefficients are rational, the norm-length is that of the integral polynomial obtained by putting all coefficients over a common denominator.) We also assume that the maximum degree of these polynomials is bounded above by a constant. Note that to allow all polynomials to be distinct, it is necessary to let $\ell$ be at least on the order of $\log n$. In practice one should expect $\ell$ to be at most polylogarithmic in $n$.

The generalized point location problem concerns the fast evaluation of the predicate 

$$\exists i \ (1 \leq i \leq n) \ P_i(x) = 0$$

for any query point $x \in E^d$. A simple true/false answer being a little too terse, we require supplementary information. If the predicate is true then some witness $i$ such that $P_i(x) = 0$ should be provided (note that requiring the reporting of all such indices might itself preclude a fast response). If, on the other hand, the predicate is false, then $x$ lies in a connected region of the open set $C = \{ x \in E^d \mid \prod_{i \leq n} P_i(x) \neq 0 \}$. In that case, the desired output is usually the value $f(c)$ of some function $f$ which is invariant over each connected region of $C$. The preprocessing will compute a distinct algebraic point, a sample, in each region of $C$ and evaluate $f$ at the sample points.

The problem is a direct generalization of the well-known planar point location problem. Previous work on point location with nonlinear boundaries has been limited to the case $d = 2$, culminating in the optimal algorithms of Cole (1986) and Edelsbrunner et al. (1986) for subdivisions with "monotone" curves, and that of Sarnak and Tarjan (1986) for more general subdivisions. As will be shown below, a Collins decomposition provides a simple framework which will describe a data structure of $O(\log n)$ time; the time needed.

These bounds hold in the worst case, but we will notice, for example, that $\ell$ is undefined. The reason is that in the complexity analysis the length of $\ell$ is ignored, regardless of the length of $\ell$. The true cost of a computation is proportional to the number of computational geometry has been shown (Hoffmann et al., 1988). In order to take these factors into account. The algebraic complexity analysis is based on the burden of storing coefficients, subresultants, and coefficients of the data.

The complexity analysis is based on the burden of storing coefficients, subresultants, and size of the coefficients of the data.

The problem is a direct generalization of the well-known planar point location problem. Previous work on point location with nonlinear boundaries has been limited to the case $d = 2$, culminating in the optimal algorithms of Cole (1986) and Edelsbrunner et al. (1986) for subdivisions with "monotone" curves, and that of Sarnak and Tarjan (1986) for more general subdivisions. As will be shown below, a Collins decomposition provides a simple framework which will describe a data structure of $O(\log n)$ time; the time needed.
Generalized Point Location 233

provides a simple framework for solving the point location problem in full generality. We will describe a data structure of size $O(n^{d-1})$ which allows us to answer any query in $O(\log n)$ time; the time needed to build the data structure is $O(n^{d+\epsilon})$.

These bounds hold in the traditional unit-cost RAM model (Aho et al. 1974). One will notice, for example, that the norm-length $\ell$ does not even appear in the bounds. The reason is that in the unit-cost model any integer operation takes constant time, regardless of the length of the integers involved. Of course, this may sometimes hide the true cost of a computation if the integers become very large. Traditionally, algorithms in computational geometry have tended to ignore the true cost of precise rational arithmetic, although this issue has recently started to gain importance (see Dobkin and Silver, 1988, Hoffmann et al., 1988). In the case of our data structure, however, this cost must be taken into account. The algorithm involves iterated computations of polynomial greatest common divisors, subresultants, Sturm sequences, etc., all of which tend to inflate the size of the coefficients of the polynomials.

The complexity analysis of our data structure follows (Collins, 1975) without the added burden of storing cylindrical algebraic samples (though such samples might be computed, used in preprocessing, and thrown away). If we start out with $d$-variate polynomials of constant degree and norm-length $\ell$, the preprocessing will only generate $k$-variate polynomials ($k \leq d$) of constant maximum degree and norm-length $O(\ell)$. Moreover, as follows from (Collins, 1975), all operations on the coefficients of the polynomials can be carried out in a number of bit operations at most cubic in their norm-length. This means that in order to obtain upper bounds on the bit complexity of the algorithm it suffices to multiply the unit-cost bounds given above by $\ell^6$. This would give us a query time of $O(\ell^6 \log n)$ and a preprocessing time of $O(\ell^{d+\epsilon})$. As long as a computer word can store up to $\ell$ bits the storage requirement is (asymptotically) the same in both models of computation. Although our underlying assumption will be the unit-cost model, we will also mention the bit complexity of an algorithm whenever there is a discrepancy between the two models. A final word concerning the dependency of the algorithm on the degree of the polynomials. We caution that our algorithm, like Collins', produces auxiliary polynomials whose maximum degrees can be truly enormous. This can add a large multiplicative factor, albeit constant, to the complexity of the algorithms: say, around $\delta^k$, where $\delta$ the maximum degree of the input polynomials. We will ignore this
Interestingly, our data structure matches Dobkin and Lipton's (1976) in terms of storage requirements, both being $O(n^{m-1})$. Although the size of our data structure is polynomial in $n$, the magnitude of the exponent puts a severe limitation on its practicality. From a theoretical standpoint, however, this result has direct application to multidimensional searching: we will discuss this relationship in some detail later. The algorithm also has somewhat unexpected ramifications. We will use it as a tool in the solution of several problems. Two of them, posed by McKenna (1986), seek (i) the longest line segment fully contained within a given $n$-gon and (ii) the minimum "vertical" distance between two collections of red and blue segments in 3-space (see section 5 for details). Another one, due to Atallah (1986), asks at which time the convex hull of $n$ points moving in the plane will first enter its final, steady configuration. Using our point location algorithm, combined with a batching technique originally proposed by Yao (1982), we are able to solve these problems, as well as other related ones, in subquadratic (almost ever so close to quadratic) time.

These are a few concrete exemplifications of a more general principle, which is one of the main consequences of this paper. Many optimization problems in computational geometry have trivial quadratic solutions. Typically these problems involve two sets $A, B$ of $n$ objects each, and ask for the pair $(a, b) \in A \times B$ that minimizes some cost function, or satisfies some predicate; think, for example, of the diameter or closest-pair problem (Preparata and Shamos, 1985). A considerable amount of recent work in computational geometry can be regarded as attempts to beat this trivial quadratic bound by building clever data structures which reduce the number of pairs $a \in A, b \in B$ that need testing (e.g., the Voronoi diagram for the closest-pair problem). Our results imply that if the interaction of a single pair $a \in A, b \in B$ can be stated as an algebraic expression (possibly involving Boolean algebraic predicates) in the real parameters specifying $a, b$, then the data structure that we develop can be used to reduce the problem complexity to subquadratic.

In section 2 we review the algebraic backdrop behind the algorithms, and in section 3 we describe the point location data structure in detail. Section 4 discusses the relevance of the algorithm to multidimensional searching in general. In sections 5 we tackle McKenna's problems by reducing them to a more general optimization question of the sort just mentioned. We attack Atallah of the underlying technique is...
Lipton's (1976) in terms of size of our data structure is severe limitation on its practicality has direct application to pip in some detail later. The will use it as a tool in the McKenna (1986), seek (i) the and (ii) the minimum vertices in 3-space (see section 5) which time the convex hull of a configuration. Using our approach originally proposed by Yao related cases, in subquadratic general principle, which is one of the problems in computational these problems involve two sets $B$ that minimizes some cost of the diameter or closest amount of recent work in this trivial quadratic bound or pairs $a \in A, b \in B$ that problem. Our results imply as an algebraic expression and parameters specifying $a, b$, reduce the problem complexity algorithms, and in section 3 and 4 discusses the relevance of section 5 we tackle McKenna's question of the sort just mentioned. We attack Atallah's problem in section 6, and give a more general discussion of the underlying technique in section 7.

2. The Algebraic Machinery

Most of the algebraic notions involved in this work can be found exposed in great detail in (Collins, 1975) and (Schwartz and Sharir, 1983). We have tried to adhere to the terminology used in these papers as much as possible. The fundamental algebraic concepts can be found in van der Waerden's classic text (van der Waerden, 1953), while for the specialized treatment of resultants and subresultants used in the paper the reader should turn to (Brown and Traub, 1971).

I) Collins' Decidability Theorem: In 1948 Tarski proved that every statement in elementary algebra (which is, the elementary theory of real-closed fields) is decidable (Tarski, 1948). The non-elementary procedure given by Tarski was subsequently improved (computationally) in a number of different ways by several researchers (e.g., Siegel, 1954, Cohen, 1969, Collins, 1975, Monk and Solovay, 1974, Ben-Or et al., 1984). For the purposes of the present work, we shall use Collins' decision procedure as a guiding framework. Let a standard prenex formula be any logical sentence of the form

$$(Q_1 x_1)(Q_2 x_2) \cdots (Q_d x_d) \phi(x_1, \ldots, x_d),$$

where each $Q_i$ is a universal or existential quantifier and $\phi(x_1, \ldots, x_d)$ is a quantifier-free formula made of Boolean connectives, standard comparators, and polynomials with rational coefficients in the real variables $x_1, \ldots, x_d$. A logical sentence is called an atomic formula if it is free of quantifiers and logical connectives.

Theorem 1. (Collins, 1975) – Let $\Phi$ be an arbitrary standard prenex formula with $d$ variables, $c$ atomic formulas, $m$ polynomials of degree at most $b$ in any single variable, with all integral coefficients of length at most $\ell$. Whether $\Phi$ is true or false can be decided in at most $c\ell^2(2b)^{2mc + 2mc + 2mc}$ bit operations.

II) The Cylindrical Algebraic Decomposition: This section reviews the essential components of Collins' decomposition needed for the point location algorithm. We include this discussion to make our exposition self-contained. The reader fully familiar with Collins' work may skip the next paragraphs.
A Collins decomposition of the $d$-dimensional Euclidean space $E^d$ is a refinement of the decomposition of $E^d$ induced by a finite collection of real algebraic varieties. (Each polynomial defining a variety is sign-invariant over each region of the decomposition.) The key concept is that of a cylindrical algebraic decomposition (or 	extit{cad}, for short). A $d$-dimensional 	extit{cad} is a partitioning of $E^d$ defined inductively as follows.

(i) For $d = 1$, a 	extit{cad} is a finite set of disjoint open intervals and singletons whose union forms $E^1$. Each singleton contains an algebraic number; see a discussion later in this section on how to store algebraic numbers.

(ii) For $d > 1$, a 	extit{cad} $K$ is defined in terms of a 	extit{cad} $K'$ of $E^{d-1}$ and a $d$-variate polynomial $P(x_1, \ldots, x_{d-1}, y)$ with rational coefficients. Let $K' = \{c_1, \ldots, c_n\}$; for each $c_i \in K'$ there exists an integer $v_i$ such that for each $x = (x_1, \ldots, x_{d-1}) \in c_i$, $P(x, y)$, regarded as a polynomial in $y$, has $v_i$ real roots $f_i(x) < \cdots < f_{v_i}(x)$, each of which is a continuous function in $x$ over $c_i$. If $v_0 = 0$, set $c_{v_0} = c_i \times E^1$. If $v_0 > 0$, set $c_{v_0} = \{(x, f_{v_0}(x)) \mid x \in c_i\}$ for $1 \leq j \leq v_0$, and set $c_{v_0+1} = \{(x, y) \mid x \in c_i \text{ and } y < f_{v_0}(x)\} \cup c_{v_0+2} = \{(x, y) \mid x \in c_i \text{ and } y < f_{v_1}(x)\}$. Finally, $K$ is defined as the set of cells $\{c_1, c_2, \ldots, c_{2v_0+1}, \ldots, c_{2v_1+1}, \ldots, c_{2v_0+2}\}$.

Following (Schwartz and Sharir, 1983) we call $P$ the base polynomial of the 	extit{cad}. Informally, the cells of $K$ can be formed by considering the cylinders based at each $c \in K'$ and chopping them off with the real hypersurface $P(x_1, \ldots, x_{d-1}) = 0$. Since $K$ is defined in terms of a unique 	extit{cad} of lower dimension, by induction, it defines an induced 	extit{cad} for each $E^k$ ($1 \leq k \leq d$). Incidentally, one should note that each cell of $K$ is "well-behaved," in the sense that it is topologically equivalent to a relatively open ball of dimension at most $d$.

For our purposes the base polynomial $P$ will always be of the form $\prod_{i \leq k \leq n} P_i$, where $\mathcal{F} = \{P_1, \ldots, P_n\}$ is a collection of $d$-variate polynomials with rational coefficients. The key feature of a 	extit{cad} is that for each $c \in K$ and each $P_i \in \mathcal{F}$, the value of $P_i(x)$ is either zero over the entire cell $c$, or it keeps the same sign over the cell: a 	extit{cad} which satisfies this property is said to be $\mathcal{F}$-invariant. Besides introducing the concept itself, the main contribution of (Collins, 1975) was to prove that any collection $\mathcal{F}$ admits of an $\mathcal{F}$-invariant 	extit{cad} and that it can be constructed fairly efficiently (all things considered).

To simplify the computations, we develop the concept of a well-based decomposition of a univariate polynomial in $y$ given a value of $x$ in $E^{d-1}$ (Schwartz and Sharir, 1983) for such a root function $f_{v_0}(x)$ (defined over $c_i$). Informally, this means that $P(x, y)$ is purely a function of $x$. Therefore another solution to the search problem is possible.

Following Collins' terminology with algebraic coordinates, a 	extit{cad} is a root of a polynomial of degree $p$. (Abbreviated 	extit{cad}.) If either $a$ or $b$ is a root of a polynomial of degree $p$, then $a$ or $b$ is a root of the polynomial $f_{v_0}(x)$ for $1 \leq j \leq v_0$, and $a$ or $b$ is a root of the polynomial $f_{v_0+1}(x)$ for $1 \leq j \leq v_0$. Finally, $K$ is defined as the set of cells $\{c_1, c_2, \ldots, c_{2v_0+1}, \ldots, c_{v_1+1}, \ldots, c_{2v_0+2}\}$.

Following (Schwartz and Sharir, 1983) we call $P$ the base polynomial of the 	extit{cad}. Informally, the cells of $K$ can be formed by considering the cylinders based at each $c \in K'$ and chopping them off with the real hypersurface $P(x_1, \ldots, x_{d-1}) = 0$. Since $K$ is defined in terms of a unique 	extit{cad} of lower dimension, by induction, it defines an induced 	extit{cad} for each $E^k$ ($1 \leq k \leq d$). Incidentally, one should note that each cell of $K$ is "well-behaved," in the sense that it is topologically equivalent to a relatively open ball of dimension at most $d$.

For our purposes the base polynomial $P$ will always be of the form $\prod_{i \leq k \leq n} P_i$, where $\mathcal{F} = \{P_1, \ldots, P_n\}$ is a collection of $d$-variate polynomials with rational coefficients. The key feature of a 	extit{cad} is that for each $c \in K$ and each $P_i \in \mathcal{F}$, the value of $P_i(x)$ is either zero over the entire cell $c$, or it keeps the same sign over the cell: a 	extit{cad} which satisfies this property is said to be $\mathcal{F}$-invariant. Besides introducing the concept itself, the main contribution of (Collins, 1975) was to prove that any collection $\mathcal{F}$ admits of an $\mathcal{F}$-invariant 	extit{cad} and that it can be constructed fairly efficiently (all things considered).
To simplify the computations (as well as carry the analysis further to determine the adjacencies between the cells of a cad) Schwartz and Sharir introduce the useful concept of a well-based decomposition: $K$ is said to be well-based if, when regarded as a univariate polynomial in $y$, the base polynomial $P(x,y)$ is not identically zero for any given value of $x$ in $E^d - 1$ (Schwartz and Sharir, 1983). They show that in that case each root function $f_{ij}$ (defined over $a_i \in K'$) can be extended continuously over the closure of $a_i$. Informally, this means that every line $(x_1, \ldots, x_{d-1}) \times E$ intersects the algebraic variety $P(x,y) = 0$ only in a finite number of points. Those intersections will form the basis of the binary search underlying the point location algorithm to be presented in the next section. How can we ensure that a decomposition is well-based? Since point location is defined independently of a coordinate system, we can always modify the frame of reference to ensure this condition. As a matter of fact, it is suggested in (Schwartz and Sharir, 1983) that a few random perturbations of the original coordinate system might be the best strategy in practice. (They also give a method for checking if a cad is well-based.) For our purposes, a well-based decomposition is convenient but not necessary. Therefore another solution is, of course, not to worry about it and simply ensure that the search procedure is robust enough to handle this type of degeneracy.

Following Collins' terminology, we define an algebraic sample of $K$ as a set of points with algebraic coordinates, one in each cell of $K$ (recall that a real number is algebraic if it is a root of a polynomial with integer coefficients). An algebraic sample is cylindrical (abbreviated cad) if either $d = 1$ or the set of first coordinates of each point forms a cad of $K'$. If $(x_1, \ldots, x_{d-1}, a_{d+1})$ is the set of cells of $K$ associated with the cell $a_i$ of $K'$, the sample points in each $a_i (1 \leq j \leq 2n + 1) all share the same first $d - 1$ coordinates.

III The Collins Construction: We begin with a short review of Collins' algorithm. Let $Q$ be a $d$-variate polynomial of degree $p$ with real rational coefficients. We can write $Q(x_1, \ldots, x_d)$ as a polynomial $\sum_{i \in S} a_i \cdot (x_1, \ldots, x_{d-1})^{e_i}$ of a single variable $x_d$, with coefficients in the ring of $(d - 1)$-variate polynomials with rational coefficients. Let $deg(Q) = p$ be the degree of $Q$ in $x_d$ and let $\text{deg}(Q) = Q(x_1, \ldots, x_{d-1})^{k}$. We call the leading nonzero coefficient of $Q$. We define the refiner of $Q$, denoted $\text{red}(Q)$, as the polynomial $\sum_{i \in S} a_i \cdot (x_1, \ldots, x_{d-1})^{k}$. We also introduce $\text{red}^k(Q) = Q$, and for each $k \geq 0, \text{red}^{k+1}(Q) = \text{red}^k(Q)$. Finally we let $\text{deg}(Q)$ denote the $d$th-derivative of $Q$. 

The idea is that these operations are used to simplify computations involving the function $Q$. The key to the algorithm is the fact that the degree of $Q$ is reduced by one at each step, reducing the complexity of the problem. The algorithm proceeds by repeatedly applying the refiner operation until a sufficiently simple polynomial is obtained. Once the simplified polynomial is found, the value of $Q$ at any point can be computed efficiently.
Let \( A(x) = \sum_{i=0}^{a} a_i x^i \) and \( B(x) = \sum_{j=0}^{b} b_j x^j \) be two polynomials in the real variable \( x \) with \( \deg(A) = a \) and \( \deg(B) = b \). The Sylvester matrix of \( A \) and \( B \) is

\[
\begin{pmatrix}
  a_a & a_{a-1} & \cdots & a_0 \\
  a_{a-1} & a_{a-2} & \cdots & a_1 \\
  \vdots & \vdots & \ddots & \vdots \\
  a_1 & a_0 & \cdots & a_{a-2} \\
  b_b & b_{b-1} & \cdots & b_0 \\
  b_{b-1} & b_{b-2} & \cdots & b_1 \\
  \vdots & \vdots & \ddots & \vdots \\
  b_1 & b_0 & \cdots & b_{b-2}
\end{pmatrix}
\]

The determinant of \( M \) is the resultant of \( A \) and \( B \). For \( 0 \leq j \leq \min(a, b) \), \( M_j \) is the matrix obtained by deleting the last \( j \) rows of \( A \) coefficients, the last \( j \) rows of \( B \) coefficients, and all the last \( 2j \) columns. We can then define \( \text{res}^j(A, B) \) (the \( j \)th principal subresultant coefficient of \( A \) and \( B \)) as the determinant of \( M_j \) (see Collins, 1975, for details). As is shown in (Brown and Traub, 1971), the unique factorization theorem for polynomials implies that \( A \) and \( B \) have exactly \( j \) common roots (i.e., \( j \) is the degree of their greatest common divisor) if and only if \( j \) is the last index \( k \) for which \( \text{res}^k(A, B) \neq 0 \). This result is at the basis of the recursive construction of a \( \text{cad} \).

Let \( \mathcal{F} = \{ F_1, \ldots, F_t \} \) be a set of \( d \)-variate polynomials \( (d \geq 2) \) with rational coefficients and norm-length \( \leq \ell \). As expected, the construction of an \( \mathcal{F} \)-invariant \( K \) proceeds recursively. It suffices to specify what arguments should be passed to the decomposition algorithm at the first recursive call. To do so, we define the projection of \( \mathcal{F} \), denoted \( \mathcal{G} \), as the union of \( G_1, G_2, \text{ and } G_3 \), with

(i) \( G_1 = \{ \text{red}^k(P) \mid P \in \mathcal{F} \text{ and } k \geq 0 \text{ and } \deg(\text{red}^k(P)) \geq 1 \} \),

(ii) \( G_2 = \{ \text{lcd}(P) \mid P \in G_1 \} \),

(iii) \( G_3 = \{ \text{res}^k(P, \text{der}(P)) \mid P \in G_1 \text{ and } 0 \leq k < \deg(\text{der}(P)) \} \),

(iv) \( G_4 = \{ \text{res}^k(P, Q) \mid P, Q \in G_1 \text{ and } 0 \leq k < \min(\deg(P), \deg(Q)) \} \).

The notion of a projection generalizes the idea of (Dobkin and Lipson, 1976) of pairing up hyperplanes: the pairing takes place in \( G_4 \); the set \( G_2 \) accounts for multiple roots (for polynomials in \( x_0 \) roots can be caused by a base let \( K' \) be a \( \mathcal{F} \)-invariant \( \text{cad} \) distinct real roots of the pol constant as \( (x_1, \ldots, x_{d-1}) \) are functions over \( a_1, \ldots, a_{d-1} \) roots ever coincide over \( a_1 \). For \( x_{d-1} \) by the hypersurfaces \( x_d = j \) \( \text{cad} \) of \( F' \).

This provides a recursive algorithm takes \( \mathcal{F} \) and \( d \) as \( \mathcal{F} \), and \( d-1 \) as arguments. The formulas defining each cell \( G_i \)

For our application the definition correspondence between same, \( \text{res} \) each \( K \) has a base \( \text{cad} \) \( K' \) of \( K \), computed recurrence points of the \( \text{cad} \) of \( K \), orders of the point \( \beta \). Each point \( \beta \)

for \( P(x) = 0 \). Let \( m_i \) be \( i \), \( \mathcal{F} \) as the number of distinct \( Q(y) = P_{a_1}(a_1, \ldots, a_{d-1}, y) \) the \( (a_1, \ldots, a_{d-1}, y) \)

will be necessary later on in construction. Let \( (a, \beta) \) be a \( \text{cad} \). Trivially, we can test the pairings.
the two polynomials in the real
characteristic matrix of A and B is
coefficients of the polynomials
consecutive rows of M. with

\[
\begin{pmatrix}
\alpha_0 \\
\beta_0
\end{pmatrix}
\]

For \(3 \leq j \leq \min(a, b)\), \(M_j\) coefficients, the last \(j\) rows of \(\Gamma^t(A, B)\) (the \(j\)th
minant of \(M_j\) (see Collins, 71), the unique factorization
by \(j\) common roots (i.e., \(j\) is
is the least index \(k\) for which
construction of a cad.

nominals \((d \geq 2)\) with rational
raction of an \(\mathcal{F}\)-invariant \(K\)
nts should be passed to the
o, we define the projection of

\[
\beta_j (p) \cup \{p \in \mathbb{Q} | \deg(Q) \}
\]

Robbins and Liston, 1976) of
et \(G_2\) accounts for multiple
roots (for polynomials in \(x_d\), while \(G_2\) is included because a change in the number of
roots can be caused by a loss of degree. The following result is proven in (Collins, 1975):
let \(K'\) be a \(G\)-invariant cad of \(E^{d-1}\) and let \(c_i\) be any cell of \(K'\); the total number of
distinct real roots of the polynomials in \(x_d\), \(P_1(x_1, \ldots, x_d), \ldots, P_n(x_1, \ldots, x_d)\), remains
constant as \((x_1, \ldots, x_{d-1})\) varies in \(c_i\). These roots form a well-ordered set of continuous
functions over \(c_i: f_1(x_1, \ldots, x_{d-1}), \ldots, f_n(x_1, \ldots, x_{d-1})\); in particular, no two such
roots ever coincide over \(c_i\). As a result, for each \(c_i \in K'\), the partition of \(c_i \times E^1\) induced
by the hypersurfaces \(x_d = f_i(x), \ldots, x_d = f_n(x) (x \in E^{d-1})\) defines an \(\mathcal{F}\)-invariant

cad of \(E^d\).

This provides a recursive scheme for computing an \(\mathcal{F}\)-invariant cad \(K\) of \(E^d\). The algorithm
takes \(\mathcal{F}\) and \(d\) as input and recovers by calling itself with \(G\), the projection of
\(\mathcal{F}\), and \(d-1\) as arguments. The output of Collins’ construction includes (i) quantifier-free
formulas defining each cell of \(K\) and (ii) a cad of \(K\) of the form \(\{\beta_1, \ldots, \beta_n\}\), where for
each \(i = 1, \ldots, n\), each coordinate of \(\beta_i \in E^d\) is represented by a quantifier-free formula.
For our application the definitions of the cells are not really needed: instead, we need a
 correspondence between sample points and their associated polynomials in \(\mathcal{F}\). If \(d > 1\)
then \(K\) has a base cad \(K' = \{c_1, \ldots, c_n\}\) which is \(G\)-invariant. Let \(\{\beta_1, \ldots, \beta_n\}\) be the
cas of \(K'\), computed recursively. For each \(i = 1, \ldots, n\), let \(\{\beta_{i,1}, \ldots, \beta_{i,3n+1}\}\) be the
points of the cad of \(K'\), ordered in ascending \(x_d\)-order, whose first \(d-1\) coordinates form
the point \(\beta_i\). Each point \(\beta_{i,j} (1 \leq j \leq n)\) lies on at least one algebraic variety of
form \(P_i(x) = 0\). Let \(k_i\) be any such value of \(l\) and let \(\beta_{i,j} = (a_1, \ldots, a_d)\); we define
\(m_{ij}\) as the number of distinct real roots of \(Q_j(y)\) that are strictly smaller than \(a_j\), where
\(Q_j(y) = P_i(a_1, \ldots, a_{d-1}, y)\) is regarded as a polynomial in \(y\). As part of the output, we
require the sequence \(\{m_{i,1}, \ldots, m_{i,3n+1}\}\) for each \(i = 1, \ldots, n\). This sequence
will be necessary later on in order to carry out the binary searches underlying the point
location algorithm.

The next step is to show how to derive these sequences from \(\{\beta_{i,1}, \ldots, \beta_{i,3n+1}\}\)
(1 \leq i \leq \mu). Recall that the latter sequences are provided directly by the Collins
construction. Let \(\phi(x)\) be the quantifier-free defining formula for \(\beta_{i,j} (1 \leq j \leq n)\).
Trivially, we can test the predicate

\[ \exists y \in E^d \mid \phi(x) \land P_i(x) = 0 \]
for each $l = 1, \ldots, n$, and pick as $i_l$, say, the first value of $l$ found to satisfy the predicate. To obtain $m_{l,i}$ it suffices to express with a prefix formula the proposition, denoted $F_{l,i}$, that $z$ is a root of $Q$ and $Q(z) = P_{l,i}(z_1, \ldots, z_{l-1}, z)$ has exactly $k$ distinct roots strictly smaller than $z$. In the spirit of (Arnow, 1981) we express $F_{l,i}$ with the formula

$$F_{l,i} = \bigwedge_{y_1, \ldots, y_{k+1}, y_{k+2}} \left( Q(z) + Q'(y_1) + \ldots + Q'(y_{k+1}) = 0 \right) \wedge \left( y_1 < \cdots < y_{k+1} < z \right) \wedge \left( Q(z) \neq 0 \right) \wedge \left( z \preceq x \right) \wedge \left( \prod_{1 \leq i \leq k+2} (y_i - z) = 0 \right).$$

The value of $m_{l,i}$ is then given by the unique index $k$ for which $\mu_{l,i-1,k}(x)$ is true, with $\mu_{l,i} = (\mu_{l,i}, x)$.

IV) Complexity Analysis. We assume that only rational symbolic calculations are used during the course of the computation. The following complexity results are derived from (Collins, 1975). Let $b$ be the maximum degree of any polynomial in $F$ in any variable. Recall that all the polynomials in $F$ have norm-length at most $l$. We assume that $d$ and $b$ (but not necessarily $l$) are constants. The $F$-invariant set produced by the Collins construction consists of $O(2l^{b^2+1})$ elements. (Collins’ paper actually states a slightly larger bound, but the one above easily follows from his derivations.) The total number of polynomials defined in the various projections introduced in the decomposition is bounded above by $O(2l^{b^2+1}) = O(2^{b^2+1})$ and the maximum degree of each polynomial in any variable is at most $\frac{1}{b}(2l^{b^2+1}) = O(1)$. The norm-length of each polynomial is at most $(2l)^{b^2} = O(l)$.

Consider now the complexity of the decomposition. Each algebraic point is represented by its coordinates. Collins uses two different representations of real algebraic numbers. One is the traditional root isolation method: the number $a$ is the unique real root of an integral polynomial falling in some interval $I$, whose endpoints are rational of the form $n/2^m$. In the other representation, a real algebraic number $\beta$ will appear as an element of the algebraic number field $Q(a)$ (i.e., the smallest subfield of $\mathbb{R}$ that contains both $Q$ and $a$). In this case, we represent $\beta$ by a rational polynomial $B(x)$ with $\beta = B(a)$. The degree of each polynomial used in the definition of the $m_{l,i}$ is dominated by $(2l)^{b^2} = O(1)$ and their norm-length is at most $l(2l)^{b^2} = O(l^{b^2+1})$. Implementing the

Collins construction proper requires $O(l^{b^2+1})$ time asymptotically dominated by $(l^{b^2+1}) = O(l^{b^2+1})$.

3. The Geometric Structure of the Composition

Most of the ingredients have been introduced. The data structure consists of

(i) $D(G)$, where $G$ is the preimage
(ii) a set of $K$:
(iii) a set of $v$ one-word monomials (the cells of $K$).

Let $C_1, \ldots, C_l$ be the membranes with the cells of $K = \{v_1, \ldots, v_k\}$ present in $2n + 1$ cells of $K$ (each represents the corresponding memory $\{k_1, \ldots, k_{n+1}\}$ as an ordered set). This set form a sequence of $2n + 1$ correspondence with $W_i$. The algorithm.

The input is a family of $k$-relations described. The generalized $G_i$ is the following: given a query point $y_i$, $C_i$ corresponds to the unique polynomials of $F$, the index $i$ has the other hand, $C_i$ lies in one of the access to the sample points $y_i$. But often the data structures are interested in evaluating $y_i$.
Collins construction proper requires $O((p(2p))^{2\alpha+\epsilon} n^{\alpha+\epsilon}) = O(p(2p)^{\epsilon} n^{\epsilon})$ bit operations. Using Theorem 1 and the previous upper bounds, it is easy to see that this running time asymptotically dominates the overhead of computing the sequences of the form \((l_{i,1}, m_{i,1}), \ldots, (l_{i,\alpha}, m_{i,\alpha})\). In the unit-cost model, this gives us a total running time of $O(n^{2\alpha+\epsilon})$.

3. The Generalized Point Location Algorithm

Most of the ingredients entering the composition of the algorithm have already been introduced. The data structure $D(F)$ is defined recursively as follows: it includes

(i) $D(p)$, where $p$ is the projection of $F$;

(ii) a case of $K$;

(iii) a set of $v$ one-word memory cells $C_1, \ldots, C_v$ (which we conveniently associate with the cells of $K$).

Let $C_1', \ldots, C_v'$ be the memory cells associated with $D(p)$ (in one-to-one correspondence with the cells of $K' = \{c_1, \ldots, c_v\}$). Each cell $C_i'$ (1 $\leq i \leq v$) stores a pointer to the sequence \((l_{i,1}, \ldots, l_{i,\alpha})\) previously defined. Recall that the cell $C_i'$ is associated with $2n_i + 1$ cells of $K$ (each projecting exactly onto $c_i$). Let $W_i = \{C_{i,1}, \ldots, C_{i,2n_i+1}\}$ be the corresponding memory cells in ascending $x_2$-order. Consider the sequence $S_i = (l_{i,1}, \ldots, l_{i,\alpha})$ as an ordered set of keys. The possible outcomes of a binary search in this set form a sequence of $2n_i + 1$ keys and open intervals, which we put in one-to-one correspondence with $W_i$. The data structure is now complete, so we can describe the algorithm.

The input is a family of polynomials $F$, assumed to be preprocessed as previously described. The generalized point location problem defined earlier can be reduced to the following: given a query point $x = (x_1, \ldots, x_d) \in E^d$, compute the index $i$ such that $C_i$ corresponds to the unique cell of $K$ that contains $x$. If $x$ is a zero of one of several polynomials of $F$, the index of one of them will be directly available from $C_i$. If, on the other hand, $C_i$ lies in one of the connected regions of $E = \{ x \in E^d | \prod_{j \in S_i} p_j(x) \neq 0 \}$, access to the sample points provided by the case of $K$ would provide the desired answer. But often the data structure does not need to store the sample points. Recall that we are interested in evaluating a particular function $f$ which is invariant over the regions of
4. Point Location

Multidimensional search
retrieve information of a point $q$ in a finite set $V$, a query space function $p: Q \times V \rightarrow \{0,1\}$ for an arbitrary element $q \in Q$ and an example is orthogonal range searching in the $d$-dimensional isothetic hyper-rectangle $q$. How to evaluate $p$ at $q$?

Elements of both $V$ and $q$ of the predicate $p(q, x)$ is evaluated at $q$. The family of polynomials of constant maximum degree, multidimensional searching approach: subdivide $\mathbb{E}^d$ via the corresponding (constant) $x$.

Two implementations of point location algorithm (Cormen and Yao, 1980), is to linear example: the variety in $\mathbb{E}^3$, $x$ can be replaced by the hyperplane $h$.

Searching for the location of a point $p = (x, y, z) \in \mathbb{E}^3$, where $k$ is the maximum depth, we have direct consequences, as the case and doubly exponential.
4. Point Location and Multidimensional Searching

Multidimensional searching refers to the general task of querying a database to retrieve information of a particular nature. This can be defined formally by introducing a finite set $V$, a query space $Q$, and a response domain $R$. We also need a predicate function $p : Q \times V \rightarrow \{0, 1\}$ and an evaluation function $e : 2^V \rightarrow R$. A query is an arbitrary element $q \in Q$ and its output is the value of $e\{\{x \in V | p(x, q)\}\}$. A classical example is orthogonal range searching: $V$ is a set of points in $\mathbb{R}^d$, $Q$ is the set of all $d$-dimensional isothetic hyperrectangles, $p(x, q)$ is true if and only if the point $x$ lies in the hyperrectangle $q$, and $e$ returns the cardinality of the input set in the counting version of the problem; in the reporting version, $e$ is the identity function.

Elements of both $V$ and $Q$ are expressed as points in Euclidean space, and the value of the predicate $p(x, q)$ is determined by the signs of certain polynomials $P_x, Q_x, \ldots$, evaluated at $q$. The family $\mathcal{S} = \{P_x, Q_x, \ldots | x \in V\}$ is assumed to consist of $d$-variate polynomials of constant maximum degree with rational coefficients. The reduction of multidimensional searching to point location is now obvious. This is called the locus approach: subdivide $\mathbb{R}^d$ via the varieties defined by $\mathcal{S}$ and assign to each resulting cell the corresponding (constant) value of the function $e$.

Two implementations of the locus approach suggest themselves. One is to apply the point location algorithm of the previous section. The other approach, suggested in (Yao and Yao, 1985), is to linearize the polynomials by throwing in additional variables. For example, the variety in $\mathbb{R}^3$,

$$2y^5z^3 + 2x^2y^3 - 3x^4 + z^2y + z = 0$$

can be replaced by the hyperplane in $\mathbb{R}^6$

$$z_1 + 2z_2 - 3z_3 + z_4 + z_5 = 0.$$

Searching for the location of the point $(x, y, z)$ is thus reduced to the point location of $(x^2, z^2, y^2, x^4, x^2y, z) \in \mathbb{R}^6$ with respect to a linear variety. The obvious disadvantage of the latter method is that the number of variables may jump from, say, $d$, to $(d + 1)^4$, where $d$ is the maximum degree in a single variable of any polynomial of $\mathcal{S}$. This can have dire consequences, as the preprocessing cost will be doubly exponential in $d$ in one case and doubly exponential in $(d + 1)^4$ in the other.
Let us illustrate our point location approach on a specific example. Let \( V \) be a set of \( n \) points in \( E^d \) and \( Q \) be the set of \( d \)-variate polynomials of degree at most \( b \) with rational coefficients. Given a query polynomial \( f \in Q \), count the number of points \( x \in V \) such that \( f(x) > 0 \). In our framework the query \( q \) can be regarded as a rational point in \( E^d \), where \( c \) is the dimension of the vector space \( Q \). It is well-known that \( c = \binom{d+b}{b} \). (The dimension \( c \) is equal to the number of ways one can assign exponents to \( x_1, x_2, \ldots, x_d \) adding up to at most \( b \).) The family \( F \) consists of \( n \) linear forms of \( c \) variables. A query is answered in time \( O(\log n) \) at the cost of \( O(n^{d+1}) \) space.

Consider now the case where \( V \) is a set of \( n \) points in \( E^d \) and a query is a pair \((q,r)\) consisting of a point \( q \in E^d \) and a positive rational number \( r \). The response to the query is the number of points in \( V \) lying within a distance \( r \) of \( q \). The family \( F \) consists of \( n \) \((d+1)\)-variate polynomials \( p(x_1, \ldots, x_{d+1}) \) of the form \((x_1-a_1)^2 + \cdots + (x_d-a_d)^2 - x_{d+1}^2 \). A query is answered in time \( O(\log n) \) at the cost of \( O(n^{d+1}) \) space.

5. Biggest Stick, Segment Shifting, and Other Related Problems

In this section we concern ourselves with the following class of problems: given two collections \( A, B \) of \( n \) objects each and a real-valued function \( F \) defined on \( A \times B \), compute the minimum of \( F \) over \( A \times B \). If the function \( F \) can be evaluated anywhere in constant time, problems of this type always have trivial \( O(n^2) \) solutions. Note that many common problems fall in this category, e.g., Bogcraf's problem (given a collection of lines and points in the plane, determine whether any line passes through any point) and the diameter problem in \( E^3 \) (given a three-dimensional polytope, compute the largest interdistance between any two vertices).

We will give a method for solving these problems in subquadratic time. The technique is very general, and will always work as long as a fixed number of rational parameters are needed to represent objects in \( A \) or \( B \), and the expression \( F(a, b) \) can be specified by a straight-line program of constant length involving algebraic functions in the parameters specifying \( a \) and \( b \) of bounded degree. Rather than describing the method in full generality we will illustrate it by looking at two problems posed in the literature. In a different context the next section will also provide an example of the same basic technique.

Here is a problem pose nonvertical segments in \( E^3 \), \( xz\)-plane (resp. the \( yz\)-plane) the largest distance \( d \) by which rephrase this problem in the more difficult to solve the more its weaker restriction on the orientation.

As a starter, we consider both \( F \) and \( G \) are collection \( a(x + b) \), for \( i = 1, \ldots, m \), and \( j = 1, \ldots, n \). In this particular

Put \( u_i = (a_i, b_i, y_i, 1) \), for \( i = 1, \ldots, m \), \( v_i \) must now compute \( \min \left\{ u_i \right\} \).

Given two sets of vectors \( \{u_i\} \) and \( \{v_j\} \).

Clearly the minimum must fall in \( U \) and \( V \), respectively. So, on the corresponding hulls shifting problem, computational \( O(n \log n) \), respectively, because of \( E^4 \). Next, without loss of minimum of \( u_i \)'s is attained. a hyperplane whose inward a data structure that supports in \( E^2 \) and \( E^3 \). In \( E^3 \), for example also known as the normal d \( O(n) \) regions, so that for each planes with inward drawn
specific example. Let \( V \) be a set of points in \( \mathbb{R}^3 \). Consider the closed interval \( [0,1] \) and let \( x \) be a point in \( [0,1] \). Define \( f(x) = x^2 \). Then, \( f(x) \) is a convex function, and the set of points \( \{ f(x) : x \in [0,1] \} \) is a convex set in \( \mathbb{R}^2 \).

Here is a problem posed in (McKenna, 1986): given two collections \( F \) and \( G \) of nonvertical segments in \( \mathbb{R}^3 \), such that each segment in \( F \) (resp. \( G \)) is parallel to the \( xy \)-plane (resp. the \( yz \)-plane) and each segment in \( F \) lies above every segment in \( G \), find the largest distance \( d \) by which \( F \) can be shifted downwards until it hits \( G \). It is easy to rephrase this problem in the framework outlined above. As it turns out, it is not much more difficult to solve the more general segment shifting problem obtained by removing any restriction on the orientation of the segments.

As a starter, we consider a special case of the problem, restricted line shifting, where both \( F \) and \( G \) are collections of infinite lines. Each line of \( F \) is of the form \( y = y_i, z = a_i x + b_i \), for \( i = 1, \ldots, m \), and each line of \( G \) is of the form \( y = x_j, z = c_j y + d_j \), for \( j = 1, \ldots, n \). This particular case we want to compute

\[
\min_{i,j} (a_i x_j + b_i - c_j y_i - d_j).
\]

Put \( u_i = (a_i, b_i, 1, 1) \), for \( i = 1, \ldots, m \), and \( v_j = (x_j, 1, -c_j, -d_j) \), for \( j = 1, \ldots, n \). We must now compute \( \min_{i,j} u_i \cdot v_j \), which gives us a new problem.

Given two sets of vectors \( U = (u_1, \ldots, u_m) \) and \( V = (v_1, \ldots, v_n) \) in \( \mathbb{R}^4 \), find

\[
\min_{i,j} |u_i \cdot v_j| = \min_{i,j} (u_i \cdot v_j) = \min_{i,j} \left( \sum_{k=1}^{4} u_{ik} v_{jk} \right).
\]

Clearly the minimum must be attained at points \( u_i \) and \( v_j \) lying on the convex hulls of \( U \) and \( V \), respectively. So, without loss of generality, suppose that all points \( u_i, v_j \) lie on the corresponding hulls. (Note that for the set of vectors arising in the restricted line shifting problem, computation of the convex hulls can be done in time \( O(m \log m) \) and \( O(n \log n) \), respectively, because each of these sets lies in a 3-dimensional cross-section of \( \mathbb{R}^4 \).) Next, without loss of generality, assume that \( m \leq n \). For each vector \( v \), the minimum of \( u_i \cdot v \) is attained at that point (or points) \( u_i \in U \) at which \( U \) is supported by a hyperplane whose inward drawn normal is \( v \). Therefore, we need to preprocess \( U \) into a data structure that supports queries of the above form. This is reasonably easy to do in \( \mathbb{E}^3 \) and \( \mathbb{E}^4 \). In \( \mathbb{E}^3 \), for example, we use the standard Gaussian sphere representation, also known as the normal diagram of \( U \). That is, we define a map on the sphere \( S^2 \) with \( O(n) \) regions, so that for each region \( R_i \), there corresponds a vector \( u_i \in U \) such that all planes with inward drawn normals in \( R_i \) support \( U \) at \( u_i \). Next, we construct a data
structure which supports \(O(\log n)\) point location queries in this spherical map. Such a data structure can be obtained in \(O(m \log \log m)\) preprocessing. We can now determine \(\min(v_i, -v_i)\) in logarithmic time by simply locating \(v\) in the map. In a total time of \(O((m + n) \log \min(m, n))\) we can thus find the required minimum.

Note that this approach is also applicable to the restricted line-shifting problem, because the underlying set is essentially 3-dimensional. We can thus preprocess \(U\) as above, given any vector \(v\) we simply remove its fourth coordinate and find the plane supporting \(U\) whose inward normal is precisely the truncated \(v\). This gives us an \(O((m + n) \log \min(m, n))\) time solution to McKenna’s restricted line-shifting problem.

Next, we turn to the general segment-shifting problem. Let \(A\) and \(B\) be two collections, each consisting of \(n\) nonvertical segments in \(E^3\). The problem is to find the smallest positive vertical distance \(F(a, b)\) between any pair of segments \(a \in A\) and \(b \in B\).

The function \(F(a, b)\) is defined as follows: if the projections of \(a\) and \(b\) onto the \(xy\)-plane intersect at some point \(c\), then \(F(a, b)\) is equal to \(z_a - z_b\), where \(z_a\) (resp. \(z_b\)) is the \(z\)-coordinate of the point of \(a\) (resp. \(b\)) projecting into \(c\); otherwise \(F(a, b) = +\infty\).

Fix two segments \(a \in A\), \(b \in B\). Each of them can be specified by six parameters, e.g., the coordinates of its two endpoints. Let \(a_1, a_2\) be the endpoints of \(a\), and \(b_1, b_2\) the endpoints of \(b\). Let \(a_1, a_2, b_1, b_2\) be the projections of these four points onto the \(xy\)-plane. We first find the two rational parameters \(\alpha, \beta\), satisfying

\[
a_1 \cdot \alpha + (a_2 - a_1) = b_1 \cdot \beta + (b_2 - b_1).
\]

For the projections of \(a\) and \(b\) to intersect, it is necessary and sufficient that \(0 \leq \alpha \leq 1\) and \(0 \leq \beta \leq 1\). Clearly both \(\alpha\) and \(\beta\) are rational functions of \(a_1, a_2, b_1, b_2\). Once \(\alpha\) and \(\beta\) have been found (and lie between 0 and 1), the desired \(F(a, b)\) is equal to

\[
Z(a, b) = a_1 \cdot \alpha + (a_2 - a_1) - b_1 \cdot \beta - (b_2 - b_1),
\]

where \(a_1, a_2, b_1, b_2\) are the \(z\)-coordinates of the four corresponding points.

In other words, regarding each segment \(b \in B\) as a point in \(E^3\), for each \(a \in A\), we can express \(F(a, b)\) as follows:

\[
F(a, b) = \begin{cases} 
Z(a, b), & \text{if } 0 \leq \alpha(a, b) \leq 1 \text{ and } 0 \leq \beta(a, b) \leq 1; \\
+\infty, & \text{otherwise,}
\end{cases}
\]

where \(\alpha(a, b), \beta(a, b), \text{ and } Z\) are rational functions. The collection \(F\) of \(O(n^2)\) rationals and \(Z(a, b) - Z(a', b')\) for \(a, a' \in A\), \(b, b' \in B\), we will see that the collection \(F\) will be able to generate a Collins cell.

What does this presuppose? One should consist of polynomials of any rational \(P/Q\) can be represented by doubling the size of the collection \(F\) between each cell of the corresponding segment to \(a \in A\) (if any). To obtain one, the \(P\) can be a segment in \(E^3\) and \(Q\): the complexity of the algebraic points, so labeling \(O(n^{-1})\) time. This is again given by Theorem 2, that is, we can now use these segment shifting problem.

We reduce the two-dimensional minimum span problem of the FOR algorithm to see also (Chazelle, 1987) that for collection \(A\) into \([n^{-1/2}]\) entire set \(A\) for point location for every segment \(x \in B\) within \(1\) to \(1\). Repeating this operation \(O(n^{-1/2} \log n) = O(n^3)\) time.

Theorem 3. The segment shifting.

We shall follow a similar outline (1986): given a simple polygon, the closure of \(P\) in \(\mathbb{R}^2\)
\begin{itemize}
\item \textbf{Theorem 1.} \textit{The segment shifting problem on n line segments can be solved in} \( O(n^{1.99678}) \textit{ time.} \)
\end{itemize}

We shall follow a similar approach to solve another problem posed by (McKenna, 1986): given a simple \( v \)-gon \( P \), what is the longest line segment that can be drawn in the closure of \( P \)? In McKenna's terminology the segment is called the \textit{biggest stick of} \( P \).
Obviously, the biggest stick is not necessarily unique (think of a regular n-gon), so the term actually refers to any segment with the characteristics described above. A simple yet crucial observation of McKenna is that any biggest stick must pass through two distinct vertices of P. Going through n iterations of a linear vertex-visibility algorithm (Ellingham and Avis, 1981), an O(n^2) solution to the biggest stick problem follows readily. If the vertex-visibility graph is G(n) then we can obtain better results by using output-sensitive methods for computing vertex visibilities, as in (Bresenham, 1987, Ghosh and Mount, 1987, Kapoor and Maheshwari, 1988). Often, in the worst-case the complexity is still quadratic. Can one do better?

To begin with, we set the stage for divide-and-conquer by applying the polygon-cutting theorem (Chazelle, 1982). In O(n log n) time we find a diagonal c which partitions P into two subpolygons, P1 and P2, each of size at least roughly n/3. The diagonal c is a line segment inside P joining two of its vertices. Next, we call the algorithm recursively to determine a biggest stick is each of P1 and P2. What remains to be done is to find a biggest stick crossing the diagonal c and keep the biggest of all three as the output. To look at the problem in dual space will clarify some of the issues.

Consider the dual mapping which puts in one-to-one correspondence the point (a, b) and the line ax + by + 1 = 0. If d is the distance from the origin O to the point p, the dual of p is the line perpendicular to Op at distance 1/d from O and placed on the other side of O. Any line ℓ crossing c is thus mapped to a point ℓ∗ in the dual plane. This point lies in the double wedge W (not containing the origin) formed by the dual lines of the endpoints of c. For each such line ℓ and ℓ∗, let F1(ℓ) be the length of the connected portion of ℓ ∩ P1, one of whose endpoints lies on c. Clearly the length of a biggest stick s∗ through c is given by

\[
\max \{ F1(ℓ∗) + F2(ℓ∗) | ℓ∗ crosses c \}.
\]

As shown in (Chazelle and Guibas, 1989), each of the functions F1 and F2 can be represented as a piecewise smooth function such that the projection of its smooth portions forms a straight-edge convex subdivision of W. The domain of the functions can be extended to the whole plane (setting F1 to 0 outside of W), which gives us two convex subdivisions M and N of the plane. These subdivisions encode the set of boundary points of P that are visible from the diagonal c. More precisely, an edge of M (resp. N) encodes the visibility of a vertex or an edge of P between those three intersecting edges of M and N: any O(n) number of edges which intersect between an edge c of M or N.

Given two bivariate functions F1 and F2, each region of M (resp. N) can be tested in O(log n) time to see if the maximum of F1 + F2 in that region is greater than the maximum of F1 + F2 in the preceding region. This property holds for the maximum of a linear function over a convex polygon. Therefore, by repeatedly applying this technique to the two convex subdivisions M and N of the plane, we can test for the existence of a biggest stick in O(log n) time. Therefore, the algorithm can be applied in O(n log n) time and space to find the biggest stick in any case.

It should be clear that computing the maximum of F1 + F2 over a convex polygon is a matter of finding the maximum of a linear function over a convex polygon. Therefore, we can compute the maximum of F1 + F2 over a convex polygon in O(log n) time for any convex polygon. Therefore, the algorithm can be applied in O(n log n) time and space to find the biggest stick in any case.
of a regular \( n \)-gon), so the algorithm described above. A simple yet effective approach is to use a visibility algorithm for polygons. The problem follows readily. If the visibility of a point \( p \) is computed, then the complexity is still \( O(n \log n) \) for the case where \( p \) is on the boundary of \( G \).

Given two bivariate functions \( F(x,y) \) and \( G(x,y) \) and two associated convex planar subdivisions \( M \) and \( N \), such that \( F \) (resp. \( G \)) is smooth (or continuous, or convex) over each region of \( M \) (resp. \( N \)), and such that \( F + G \) attains its maximum either at vertices of \( M \) or at intersections of edges of \( M \) with edges of \( N \), is it possible to determine the maximum of \( F + G \) in \( O(m \log m) \) time, where \( m \) (resp. \( n \)) denotes the number of vertices of \( M \) (resp. \( N \))? By preprocessing each subdivision for point location we can evaluate \( F \) and \( G \) at any point in \( O(n \log n + m \log m) \) time. This allows us to evaluate \( F + G \) at all the vertices of \( M \) and \( N \) in \( O(n \log n + m \log m) \) time. There now remains the more difficult task of testing edges against each other.

It should be clear that our previous batching technique can be applied in much the same way. A few differences are worth noticing, however. Let us discuss the problem of computing \( \max_{x,y \in G} F(x,y) + G(x,y) \) in logarithmic time, given some edge \( e \) of, say, \( N \). First, we must recall the geometric meaning of the subdivision \( N \). With each edge \( e \) of \( N \) is associated a vertex \( v \) of \( P \) as well as a line segment \( t \) on the boundary of \( P \) (Figure 1).
Therefore we can use the sul
s = 3.2

Returning now to the b
again the cost of the precompu
t by the construction of the da
O(n^2) functions, therefore it
leads to the following result.

Theorem 4. Computing the

Note that the same basic
functions over more general
particular definitions of the

Let L be the line dual to the point (p(a, z), q(a, z)) and let A (resp. B) be the intersections of L with the portion of the boundary of P1 (resp. P2) associated with the edge a (resp. e(z)). If H(a, z) is not zero then it is equal to the length of the segment |AB|. The idea is to include all functions H(a, z) - H(b, z) in S, for all edges a, b ∈ M. A minor problem is that H(a, z) is not a polynomial, but the square root of a rational function, and it is not even continuous. It is easy to extend H continuously, however, by defining it as P(x, y) + O(x, y), where (x, y) is now the intersection of the lines supporting a and e(z). Then, we include in S the rational functions (H(a, z))^2 - (H(b, z))^2. We also add to S the determinants that test for intersections between e(z) and edges of M.

Let us mention in passing a general method for dealing with sets T that include arbitrary algebraic functions (not just polynomial). Suppose that we have a function f(x1, ..., xk) which is expressed as a root of a univariate polynomial P(z) whose coefficients are polynomials in x1, ..., xk. Any zero (x1, ..., xk) of f is also a zero of the polynomial P(0), so we can replace f by P(0) in T. As long as the queries are not zeros of P(0) themselves the Collins decomposition obtained after the replacement will work just fine. For example, let f(x, y, z) = \sqrt{2} - 2y + 3z. Using repeated squaring, we derive the identity

\[ ((f^2 + x - 4y - 9z)^2 - 4xy^2 - 14yz)^2 = 2304y^2 z^2.\]

Atallah has posed the f
moving in the plane, with the
of the time. What is the f
i.e., a combinatorially stable
degree bounded above by a
number, we will content our
polynomial along with an in
the steady convex hull in O

time each point will achieve

Let e be an edge of the hull:
(if any) at which p lies on the
obtained by this process. To
of the convex hull. Can the
indeed—well, at least there
produce an O(n^2) time.
Therefore we can use the substitute function

\[ z^2 + 16y^2 + 8y^3 - 8y - 72yz - 18xz. \]

Returning now to the biggest stick problem, we are about ready to conclude. Once again the cost of precomputing the function on the algebraic points of the case is dominated by the construction of the data structure. As in the segment shifting problem, \( \mathcal{F} \) contains \( O(n^3) \) functions, therefore the preprocessing requires \( O(n^{3+\epsilon}) \) time. The batching trick leads to the following result.

**Theorem 4.** Computing the biggest stick of a simple \( n \)-gon can be done in \( O(n^{3+\epsilon}) \) time.

Note that the same basic technique can be applied to handle the sum of two bivariate functions over more general planar maps. The complexity, of course, will depend on the particular definitions of these functions.

6. A Problem on Points in Motion

Atallah has posed the following problem (Atallah, 1985): Suppose that \( n \) points are moving in the plane, with the trajectory of each point described by a polynomial function of the time. What is the first instant at which their convex hull will enter a steady-state, i.e., a combinatorially stable configuration? We shall assume that each polynomial has degree bounded above by a constant. Since the output can be a fairly arbitrary algebraic number, we will content ourselves with a description of that number involving a defining polynomial along with an isolating interval. The naive algorithm consists of computing the steady convex hull in \( O(n \log n) \) time (Atallah, 1985), and then determining the first time each point will achieve its steady positioning with respect to each edge on the hull. Let \( e \) be an edge of the hull; for each moving point \( p \) it suffices to compute the last instant (if any) at which \( p \) lies on the line supporting the edge \( e \). Let \( t(e) \) be the maximum value obtained by this process. The desired answer is the maximum value of \( t(e) \) over all edges \( e \) of the convex hull. Can this quadratic algorithm be improved? We will show that it can indeed—well, at least theoretically. We will use our generalized point location algorithm to produce an \( O(n^{2+\epsilon}) \) time algorithm, for some small positive constant \( \epsilon \).
Let \( V = \{p_1, \ldots, p_n\} \) be a set of \( n > 2 \) moving points in the Euclidean plane. We assume the existence of \( 2n \) univariate polynomials \( x_1, y_1, \ldots, x_n, y_n \) of degree \( d \) with rational coefficients, such that for each \( i \) \((1 \leq i \leq n)\), \( x_i(t) \) and \( y_i(t) \) are respectively the \( x \) and \( y \) coordinates of \( p_i \) at time \( t \geq 0 \). Let \( p_1, \ldots, p_n \) be the points on the boundary of the convex hull of \( V \) at time \( t \), given in clockwise order, with \( i_1 < \min(i_2, i_3) \). If somehow \( p_{i_1}, \ldots, p_{i_r} \) coincide for some \( j, j' \) then their indices should appear in the order \( i_j < \ldots < i_{j'} \). Let \( H(t) \) be the uniquely defined sequence \((i_1, i_2, \ldots)\). It is clear that \( H(t) \) converges, as \( t \) grows to infinity \([A]\). We define the threshold of \( H(\infty) \) as the smallest value of \( t \geq 0 \) such that \( (\forall \epsilon > 0) \ H(t) = H(\epsilon) \).

Let \( x_i(t) = \sum_{j \leq i} a_{ij} t^j \) and \( y_i(t) = \sum_{j \leq i} b_{ij} t^j \), for \( i = 1, \ldots, n \). Without loss of generality, assume that \( H(\infty) \) is the sequence \((1, \ldots, n)\), for \( m \leq n \), and that all \( n \) points \((a_{1,0}, \ldots, a_{1,n}, b_{1,0}, \ldots, b_{1,n})\) of \( \mathbb{Q}^{2n+2} \) are pairwise distinct. In \( O(n \log n) \) time compute \( H(\infty) \) \([A]\) and check all pairs \( \{p_i, p_{i+1}\} \) \((1 \leq i \leq m)\) in order to determine the largest \( t_0 \) \( \geq 0 \) such that, for some \( i \), we have \( x_i(t_0) = x_{i+1}(t_0) \) and \( y_i(t_0) = y_{i+1}(t_0) \); if not defined, set \( t_0 = -\infty \). Here \( (\text{as in the following}) \), index arithmetic is taken mod \( m \). Similarly, we ensure the convexity of the polygon \( \{p_1, \ldots, p_n\} \) by considering the function

\[
 f_i(q) = (\delta_i(t) - \delta_{i+1}(t)) t_i + (x_{i+1}(t) - x_i(t)) \eta_i + x_i(t) \delta_{i+1}(t) - y_i(t) x_{i+1}(t).
\]

The point \( q = (x_i, y_i) \in \mathbb{E}^2 \) lies to the right (resp. on or to the left) of the oriented line \( (p_i, p_{i+1}) \) iff \( f_i(q) < 0 \) (resp. \( f_i(q) = 0 \) or \( f_i(q) > 0 \)). For each \( p_i \) \((1 \leq i \leq m)\) compute the largest real root of \( f_{i+1} \) as a polynomial in \( t \); discard every case where the polynomial is identically zero. Let \( t_i \) be the largest value thus obtained (or \(-\infty \) if there is none), and let \( t_f = \max\{0, t_0, t_1\} \). Once \( H(\infty) \) is available, \( t_f \) can be easily computed in \( O(n) \) time. All that remains to be done is to compute the first instant at which each \( p_i \) \((m < j \leq n)\) lies inside \( H(\infty) \) for good. To accomplish this, we allow ourselves some preprocessing. Let \( g_i(t) = (\varphi_i(t), \psi_i(t)) \) be a time-varying point in \( \mathbb{E}^2 \) with \( \varphi_i(t) = \sum_{j \leq i} a_{ij} t^j \) and \( \psi_i(t) = \sum_{j \leq i} b_{ij} t^j \). The point \( \chi = (\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \varphi_7, \varphi_8, \varphi_9, \varphi_{10}, \varphi_{11}, \varphi_{12}) \) belongs to \( \mathbb{E}^{2n+2} \) and is independent of \( n \). Let sign \( A = -1 \) (resp. \( 0, 1 \)) if \( A < 0 \) (resp. \( A = 0, > 0 \)). We define

\[
 t(\chi) = \min\{t \in \mathbb{R} \mid t \geq t_f \text{ and } (\forall \epsilon, 1 \leq i \leq m)(\forall t' > t) \text{sign} f_i(q(t')) = \text{sign} f_i(q(t))\}.
\]

Clearly, \( t(\chi) \) can be computed by computing \( f_i(q(t)) \) based on \( p_i \).

Let \( F = \{f_1, f_2, \ldots\} \) be the set of all \( \mathbb{Q} \)-linear functions of degree \( 2d + 1 \), with \( t \) being the \((2d + 1)\)-th coefficient, by the procedure described in \([B]\). A sequence of indices (possibly for any given \( \chi \in \mathbb{E}^n \) the left univariate polynomials is trivial: it gives the index of the trajectory of \( \chi \) in \( \mathbb{Q}^n \); if \( \chi \) is empty then \( \chi \) never intersects \( \mathbb{Q}^n \) and \( \mathbb{Q}^n \). Hence, we are ready to a new, independent proof of the previous section. This is the last part of the proof. We omit the details.

**Theorem 5.** In \( O(n^{n+1} \log n) \), the steady-state convex decompositions are obtained by the functions of time of maxima.
Clearly, \( t(x) \) can be computed in \( O(m) \) time. Next, we describe a fast algorithm for computing \( t(x) \) based on point location.

Let \( \mathcal{F} = \{ \phi_1(x,t), \ldots, \phi_n(x,t) \} \), where \( \phi_i(x,t) \) denotes the \((2d+3)\)-variate polynomial of degree \( 2d + 1 \),

\[
\begin{align*}
\phi_i(x,t) = \sum_{j=0}^{2d+4} q_{ij} t^j,
\end{align*}
\]

with \( t \) being the \((2d+3)\)rd coordinate. Let \( \mathcal{K} \) be the \( \mathcal{F} \)-invariant cad of \( E^{2d+2} \) provided by the procedure described in section 2, and let \( \mathcal{K}' = \{ a_1, a_2, \ldots, a_m \} \) be its base cad (i.e., the induced cad of \( E^{2d+2} \)). Recall that, for each \( a_i \in \mathcal{K}' \), the procedure provides us with a sequence of indices (possibly empty) \( S_i = \{ i_1, i_2, \ldots, i_n \} \) with the following meaning: for any given \( x \in a_i \) the line \( x \times E^1 \) contains an increasing sequence of real roots for the univariate polynomials \( \phi_{i_1}(x,t), \ldots, \phi_{i_n}(x,t) \). The interpretation of this sequence is trivial: it gives the indices of the lines supporting \( p_t a_i \) that are intersected by the trajectory of \( x \) in chronological order (from \( t = -\infty \) to \( t = +\infty \)). If the sequence is empty then \( x \) never intersects such a line. Once \( \mathcal{K}' \) has been preprocessed for point location, computing \( t(x) \) is straightforward.

To accomplish this, we allow time-varying point in \( E^1 \), with \( t \times x = (x_1, x_2, x_3, \ldots, x_n) \) (resp. \( x = (0, 0, 0) \) if \( A < 0 \) (resp. \( A = x(t) = \max(t,x_t) \).

From Theorem 2 we immediately conclude that in \( O(n^{2d+4} \log n) \) time it is possible to construct a data structure so that the function \( t(x) \) can be evaluated at any point \( x \in E^{2d+2} \) in \( O(n \log n) \) time.

We are now ready to attack Atallah's problem, using the same batching trick used in the previous section. This leads to an algorithm with a running time of \( O(n^{2d+1} \log n) \).

We omit the details.

**Theorem 5.** In \( O(n^{2d+1} \log n) \) time it is possible to compute the threshold time of the steady-state convex hull of \( n \) points moving in the plane according to polynomial functions of time of maximum degree \( d \).

We close this section with a few remarks about Atallah's problem. Our technique clearly is general enough to be applied to other problems (e.g., closest/farthest pairs). An
interesting question is to determine whether a more ad hoc treatment of these problems might lead to a more efficient solution. For example, a continuity argument easily shows that ensuring the local coherence of the steady-state Voronoi diagram is sufficient to compute its threshold (e.g., checking the nonzero length of its edges). It is then fairly simple to devise an $O(n \log n)$ algorithm for computing the steady-state Voronoi diagram of $n$ moving points as well as its threshold. Note that the same argument can be made for convex hulls if all the points are guaranteed to lie on it. One essential feature of these easy cases is that the output contains all the input. Is this in general a necessary condition of efficiency?

7. A Discussion of the Batched Technique

In the preceding sections we have given a few examples of techniques for obtaining (slightly) subquadratic solutions to a large variety of geometric problems which admit trivial exhaustive quadratic solutions. Many other problems yield to our technique. For example,

(i) Given a set of $m$ red objects (algebraic curves, surface patches, etc.) and $n$ blue objects, do any red object intersect any blue object? Hopcroft's problem, mentioned above, is such a problem; detecting intersection between a collection of red segments and a collection of blue segments is another example.

(ii) Given $m$ rays and $n$ triangles in 3-space, find the first triangle hit by each of the rays, or alternatively, find the number of triangles stabbed by each ray.

(iii) Given a collection of $n$ (disjoint) triangles in three dimensions, find all pairs of mutually visible vertices. Here, we regard each pair of vertices as a query point $z$ in $\mathbb{R}^3$; each triangle $\Delta$ corresponds to a Boolean predicate that expresses the fact that $z$ is not blocked by $\Delta$. By batching the triangles using our technique, we can obtain a subcubic solution (a cubic solution being trivial).

Reflecting on our results, we can see a whole spectrum of problems amenable to our techniques.

(A) The simplest of them admit linear or near-linear data structures and can be solved in $O(n \log n)$ time, e.g., the closest-pair problem in the plane.

(B) Next in line, we have structures and for which diameter of $n$ points in

(C) Then we have the problems as to offer no obvious theory, that is.

(B) Finally, we have problem decomposition of the form problem (ii) above in which a point of one type can be solved in subcubic time, currently known runs in $O(n^2)$.

A final comment concerns the work described in this paper. Such a method, 1987, Clarkson 1987, 1988, described as a divide-and-conquer algorithm, is known in advance which partitions them among the objects with a small number of objects, blending this randomized method (slightly) improved probability.

We wish to thank John for his work. We also thank the reviewers for their helpful comments on the presentation of this paper.
Generalized Point Location

(B) Next in line, we have problems for which we have efficient, polynomial-size data structures and for which the batching technique is very effective, e.g., computing the diameter of a point set $\mathcal{E}$, Hopcroft’s problem (section 5).

(C) Then we have the problems of the type discussed in this paper: sufficiently complicated as to offer no obvious alternatives but computing Collins decompositions (in theory, that is).

(D) Finally, we have problems for which it is not clear even how to obtain a Collins decomposition of the form used in this paper. For example, consider a variant of problem (iii) above in which we want all pairs of mutually visible triangles (i.e., pairs where a point of one triangle can see some point of the other one.) Can this problem be solved in subcubic time using our technique? As it turns out, the best solution currently known runs in time $O(n^3 \log n)$ (McKenna and O’Rourke, 1988).

A final comment concerns the use of probabilistic algorithms for the problems studied in this paper. Such a method has been developed recently in (Haussler and Wegel, 1987, Clarkson 1987, 1988, Edelsbrunner, Guibas, and Sharir, 1988). It can be roughly described as a divide-and-conquer paradigm which uses random sampling of a small subset of the input objects to obtain a cell decomposition such that, with high probability, each cell contains (or intersects) only a small number of the given objects. If the queries are known in advance (which is the case in the problems studied in section 5) then we can partition them among the cells of the decomposition, so that each point can intersect only with a small number of objects. We believe that one can develop a general framework for blending this randomized method with our Collins decomposition technique, and obtain (slightly) improved probabilistic algorithms for such problems. We leave this as an open problem.

Acknowledgments

We wish to thank Johan Swekner for providing us with useful comments about this work. We also thank the referees for their diligence and scrutiny, which helped improve the presentation of this paper.
Bernard Chazelle wishes to acknowledge the National Science Foundation for supporting this research in part under Grant CCR-8700917. Michal Sharir is pleased to acknowledge the support of the Office of Naval Research under Grant N00014-87-K-0129, the National Science Foundation under Grant DCR-83-20085, the Digital Equipment Corporation, the IBM Corporation, and the NCRD — the Israeli Council on Research and Development.

References


Hershberger, J. (1987). Fan is proportional to its size, Proc. 3rd


Towards Implementing Robust

as for Euclidean shortest path

polynomial, Michigan Math.

computational Geometry Day,

in 3-space: a data structure
Geom., 371-380.

Proc. Amer. Math. Soc. 15,

via, PhD thesis, University of

using persistent search trees,

any algebra, Annals of Math.,

* problem. II: General tech-
tic manifolds, Adv. in Appl.

New York.

in k-dimensional space and