TR-88-031

THE SEPARABILITY PROBLEM IN DYNAMIC COMPUTATIONAL GEOMETRY

The Separability Problem in Dynamic Computational Geometry

H. J. Hwang[1], R. C. Chang[12] and H. Y. Tu[2]

Abstract

We consider the *separability problem* in *dynamic computational geometry.* The word *dynamic* here is referred to the framework where every geometric object considered is moving in a prescribed manner. Given n red points and m blue points having k–motion in 2–D plane, we propose an $O(mN\cdot\log mN + nM\cdot\log nM + mN+nM)$ algorithm for describing the time intervals at which the red and blue points are separable, where N is $O(\lambda(n,4k))$, M is $O(\lambda(m,4k))$, and $\lambda(i,j)$ is the length of the maximal (i,j) Davenport–Schinzel sequence. If the points are moving in 1–D, the complexity can be reduced to $O(N\cdot\log N + M\cdot\log M + N+M)$ where N is $O(\lambda(n,k))$ and M is $O(\lambda(m,k))$. For a more special case, in which points are restricted to 1–motion in 1–D, we have a linear time algorithm for deciding whether they are separable.

Key words. Separability problem, Separable, Dynamic Computational Geometry.

1. Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, Republic of China

2. Institute of Information Science, Academia Sinica, Taipei, Republic of China

1. Introduction

Computational geometry refers to the design and analysis of algorithms for all kinds of geometric problems. Many fundamental problems and results in this field have been well studied [5]. Not until was Atallah's paper [1] presented, the geometric objects in the relative researches are mostly *static*. Thus, they present physical entities that do have a fixed position in space. In [1], some geometric problems were reconsidered under the assumption that the coordinate of each input point is a function of a time variable t. Atallah named them *dynamic computational geometry*. The word *dynamic* was referred to the situation when the input geometric objects are moving.

In this paper, we focus our attention on the *separability problem*. For the static case, it has been shown that linear separability is a linear programming problem, and can be solved in time $O(n)$ where n is the cardinality of the set of input points [3,4,5]. In dynamic case, the problem becomes much harder. Since the given points are moving, the separability crucially depends on the function of each moving point. In other words, the time domain can be subdivided into several time intervals in which the given two point sets are linearly separable. Given n red points and m blue points having 1–motion in plane, the obvious brute force approach gives an $O(mn(m+n)\log(m+n))$ time solution [1]. In this paper, we propose an algorithm to find the time intervals in which the red and blue points are separable. For the points having k–motion in the plane (2–D), the algorithm can be accomplished in time $O(mN\cdot\log mN + nM\cdot\log nM + mN+nM)$, where N is $O(\lambda(n,4k))$, M is $O(\lambda(m,4k)$, and $\lambda(i,j)$ is the length of the maximal (i,j) Davenport–Schinzel sequence. If the points are moving on a line (1–D) with k–motion, the complexity can be reduced to $O(N\log N+M\log M+N+M)$ where N is $O(\lambda(n,k))$ and M is $O(\lambda(m,k)$. For a more special case, in which points are restricted to 1–motion on 1–D, we have a linear time algorithm for deciding whether they are separable. Our algorithm consists of two major parts. First, we use the method proposed by Atallah [1] (which solved the dynamic convex hull problem) to obtain a set of time intervals in which the red points are not in the interior or boundary of the convex hull of blue points, and in the same way we can enumerate the time interval set of blue points. Then, in the second part of our algorithm, these two sets of time intervals will be merged to be one final set of separable time intervals.

The paper is organized as follows. After introducing our notation and definitions in section 2, we present an algorithm for separability problems with k–motion in 2–D in section 3. Section 4 discusses the special case, separability problem in 1–D. Concluding remarks are given in section 5.

2. Notation and Definitions

In this paper, the input points we consider are moving in Euclidean space. A set of points are said to be with k–motion, if every coordinate of every moving point is a polynomial function with no more than k degree in the time variable t. Thus, for general d–dimension, each input point p_i can be described as follows:

$$p_i(t) = (X_{i1}(t), X_{i2}(t), \ldots, X_{id}(t))$$

where each

$$X_{ij}(t) = C_{ijk}t^k + C_{ijk-1}t^{k-1} + \cdots + C_{ij0}t^0 \quad , 1 \leq j \leq d.$$

and C_{ijh} are constant coefficients of polynomial $X_{ij}(t)$ for $0 \leq h \leq k$. At time instance t_0, the configuration of input point set P is denoted as $P(t_0)$.

For separability, we define the follows.

Definition. Two set of points R and B in E^d are said to be *linearly separable* at time instance t_0 if there exists a $(d-1)$–dimension hyperplane L such that $R(t_0)$ and $B(t_0)$ lie on opposite sides of L.

The separability problem in dynamic computational geometry now can be stated as: given two set of points having k–motion in d–dimension, try to find the time intervals during which R and B are linearly separable. A *separable time interval* is a time interval $[t_1, t_2]$ during which R and B can be linearly separated .

In the following we would like to introduce some notations related to convex hull which will play an important role in this paper. Fig. 1 illustrates the convex hull of point set B. Let CH(B) denote this convex hull. For the vertices on the convex hull, such as v_1, v_2,v_3, v_4, and v_5 in Fig. 1, we name them *convex points*, and classify them into four types: *top*, *bottom*, *left* and *right* according to the following specifications.

(1) A convex point is a *top convex point* if no other points lie above it, i.e. the top convex point has the largest y–value among all points. We choose the rightmost one to be the top convex point if there are more than one point with this y–value. In Fig. 1, v_1 is a top convex point.

(2) A *bottom convex point* is a convex point of the smallest y–value. If there are more than one convex point with this y–value, we choose the leftmost one to be the bottom convex point. In Fig 1, v_3 is a bottom convex point.

(3) A *left convex point* is a convex point on the counterclockwise path from top convex point to bottom convex point. v_2 is a left convex point in Fig. 1.

(4) A *right convex point* is a convex point on the clockwise path from top convex point to bottom convex point. v_4 and v_5 are right convex points in Fig. 1.

Let $|U|$ be the length of an (n,s) Davenport–Schinzel sequence [2,6]. Then, the function

$$\lambda(n,s) = \text{MAX}\ \{\ |U|\ :\ U \text{ is an } (n,s) \text{ Davenport–Schinzel sequence } \}.$$

Suppose that $F=\{f_1,f_2,\ldots,f_n\}$ is a set of n real–valued continuous functions defined on a common interval I, where for every two distinct functions f_i and f_j, they intersect in at most s points. We define $l(t) = \text{MIN}\ \{f_i(t) : i=1..n \text{ and } t\in I\}$ to be the *lower envelope* of F. Note that $l(t)$ is typically made up of "pieces" each of which is a section of $f_i(t)$. The length of a lower envelope, $|l(t)|$, is the number of pieces in $l(t)$. Attalah [1] showed that $\lambda(n,s)$ is the upper bound of $|l(t)|$. Attalah [1], Sharir [6] and Hart [2] have proved the following bounds.

$\lambda(n,1) = n$, and $\lambda(n,2) = 2n-1$;

$\lambda(n,3) = \theta(n\cdot \alpha(n))$, and $\lambda(n,s) = O(n\cdot \alpha(n)^{O(\alpha(n)^{s-3})})$.

These results will be used in the rest of this paper.

3. Separability of k–motion in 2–dimension

In this section, we first state some properties of separability, and then iteratively use Attalah's method (which solved the dynamic convex hull problem) to obtain the set of separable time intervals for two given point set.

Let R and B be two given sets of points with k–motion in 2–D plane. Assume that $|R|=n$ and $|B|=m$. A crucial criterion for linear separability is provided by the following theorem.

Theorem 1. ([7] Store and Witzgal (1970), Theorem 3.3.9) Two sets of points R and B are *linearly separable* if and only if their convex hulls do not intersect and the interior of these two convex hulls are mutually disjoined.

We immediately have Corollary 2.

Corollary 2. If R and B are linearly separable at time instance t_0, then each point b_i of B is a convex vertex of the convex hull of set $R \cup b_i$ at t_0, and each point r_i of R is also a convex vertex of the convex hull of set $B \cup r_i$ at t_0, i.e.

$$b_i(t_0) \in CH(\mathrm{R}(t_0) \cup b_i(t_0)) \quad \text{for } \forall\, b_i \in \mathrm{B},$$

$$\text{and} \quad r_j(t_0) \in CH(\mathrm{B}(t_0) \cup r_j(t_0)) \quad \text{for } \forall\, r_j \in \mathrm{R}.$$

Since Corollary 2 is only a necessary condition, it is not strong enough to detect the intersection between the interiors of two convex hulls. Fig. 2 illustrates a counterexample in which R and B meet the requirement of Corollary 2 but are not linearly separable. Hence, we propose the following theorem.

Theorem 3. R and B are linearly separable at time instance t_0 *iff* one of the following conditions is true

i) each point $b_i(t_0)$ of $\mathrm{B}(t_0)$ is a *top, bottom,* or *right* convex point of $CH(\mathrm{R}(t_0) \cup b_i(t_0))$, and each point $r_j(t_0)$ of $\mathrm{R}(t_0)$ is a *top, bottom,* or *left* convex point of $CH(\mathrm{B}(t_0) \cup r_i(t_0))$;

ii) each point $b_i(t_0)$ of $\mathrm{B}(t_0)$ is a *top, bottom,* or *left* convex point of $CH(\mathrm{R}(t_0) \cup b_i(t_0))$, and each point $r_j(t_0)$ of $\mathrm{R}(t_0)$ is a *top, bottom,* or *right* convex point of $CH(\mathrm{B}(t_0) \cup r_j(t_0))$.

Proof: Here we prove condition (i) only.

If sets of points, R and B, are linearly separable at time instance t_0, by definition, there is a line L, say $y=ax+b$, such that $\mathrm{R}(t_0)$ and $\mathrm{B}(t_0)$ lie on its opposite sides. Without loss of generality, assume that the points of $\mathrm{B}(t_0)$ lie on the half–plane $y<ax+b$ while the points of $\mathrm{R}(t_0)$ lie on the half–plane $y>ax+b$. Let y_t and y_b be respectively the largest and the smallest y–values of points in $\mathrm{R}(t_0)$. It is easy to see that set $\mathrm{B}(t_0)$ can be classified into three subsets, B_t, B_r, and B_b, where

$B_t \subset \{(x,y) \mid y<ax+b \text{ and } y>y_t\}$,

$B_r \subset \{(x,y) \mid y<ax+b \text{ , } y<y_t \text{ , and } y>y_b\}$ and

$B_b \subset \{(x,y) \mid y<ax+b \text{ and } y<y_b\}$.

Condition (i) is a immediate consequence of above statements and Corollary 2 (see Fig. 3).

Conversely, if condition (i) holds, we will claim that the convex hull of set $R(t_0)$ and the convex hull of $B(t_0)$ do not intersect, and the interiors of these two convex hulls are mutually disjoined. Let y_t and y_b be respectively the largest and the smallest y–values of points in $R(t_0)$. Since condition (i) holds, once again we can subdivide set B into three subsets, say B_t, B_b, and B_r, where

$B_t \subset \{(x,y) \mid y>y_t\}$,

$B_r \subset \{(x,y) \mid y<y_t \text{ and } y>y_b\}$ and

$B_b \subset \{(x,y) \mid y<y_b\}$.

Suppose that $\overline{b_i(t_0)b_j(t_0)}$ is an edge of $CH(B(t_0))$, which intersects $CH(R(t_0))$. By the fact that $b_i(t_0)$ and $b_j(t_0)$ are the *top, bottom,* or *right* convex points, $\overline{b_i(t_0)b_j(t_0)}$ should intersects $CH(R(t_0))$ on two and exactly two edges (see Fig. 4). This implies that there exists a point $r_k(t_0)$ of $R(t_0)$ where

(a) its y–value is between the y–values of $b_i(t_0)$ and $b_i(t_0)$, and

(b) it is located on the right side of $\overline{b_i(t_0)b_j(t_0)}$.

(a) and (b) obviously contradict to condition (i), "point $r_k(t_0)$ is a *top, bottom,* or *left* convex point of $CH(B(t_0) \cup r_k(t_0))$." Thus, $CH(B(t_0))$ should not intersect $CH(R(t_0))$. For the interiors, if these two convex hulls do not intersect with each other, but parts of their interior are in common, there is only one case meeting this requirement : one convex hull with its interior is properly contained in the other one. This also contradicts to the assumption of convex points. Thus, R and B are linearly separable at time instance t_0. ■

Our next goal is to compute the time intervals during which a given point belongs to the convex hull.

Let $\theta_{ij}(t)$ be the angle determined by $\langle\overline{b_i(t)r_j(t)}\rangle$ and x–axis at time t where $-\pi<\theta_{ij}(t)\leq+\pi$. Define α_i, β_i, γ_i, and δ_i as follows [1]:

$$\alpha_i(t)=\underset{j}{\mathrm{MIN}}\{\ \theta_{ij}(t)\ |\ \theta_{ij}(t)\geq 0\ \}$$
$$\beta_i(t)=\underset{j}{\mathrm{MAX}}\{\ \theta_{ij}(t)\ |\ \theta_{ij}(t)\geq 0\ \}$$
$$\gamma_i(t)=\underset{j}{\mathrm{MIN}}\{\ \theta_{ij}(t)\ |\ \theta_{ij}(t)<0\ \}$$
$$\delta_i(t)=\underset{j}{\mathrm{MAX}}\{\ \theta_{ij}(t)\ |\ \theta_{ij}(t)<0\ \}$$

If all $\theta_{ij}(t)$ are negative, $\alpha_i(t)$ and $\beta_i(t)$ are both undefined. Similarly, if all $\theta_{ij}(t)$ are positive, $\gamma_i(t)$ and $\delta_i(t)$ are both undefined.

Lemma 4. For $CH($ R$(t) \cup b_i(t)\ )$, point $b_i(t)$ is

a right convex point *iff* $\alpha_i(t)-\delta_i(t)\geq\pi$,

a left convex point *iff* $\beta_i(t)-\gamma_i(t)\leq\pi$,

a top convex point *iff* $\alpha_i(t)$ and $\beta_i(t)$ are undefined, and

a bottom convex point *iff* $\gamma_i(t)$ and $\delta_i(t)$ are undefined.

Proof: Directly derived from Lemma 4.7 in [1].

Note that each of the functions α_i, β_i, γ_i, and δ_i contains $O(n)$ transitions and jump discontinuities, hence, each of them has no more than $\lambda(n,4k)$ pieces. If we count the time needed to find the roots of polynomial with $O(k)$ degree as $O(1)$, a divide–and–conquer algorithm, in time $O(\lambda(n,4k)\log\lambda(n,4k))$, can compute the set of time intervals during which $b_i(t)$ is a top, bottom, left, or right convex point of $CH($ R(t) $\cup$ $b_i(t)$) [1]. Denote the set of these time intervals as I_{bi}. After executing the computations for all $b_i(t)$, we have n sets of time intervals, say I_{b1}, I_{b2},..,I_{bn} (see Fig. 5). Now intersect these n sets to be a new set of time intervals I_b in the following way.

(1) A time interval $[t_1,t_2]$ belongs to I_b if $[t_1,t_2]\subset[a,b]$, where $[a,b]\in I_{bi}$ for i=1,2,..,n.

(2) A time interval $[t_1,t_2]$ in I_b is assigned to "R–type" if every $b_i(t)$ is either a top, bottom, or right convex point of $CH($ R$(t) \cup b_i(t)$) where $t\in[t_1,t_2]$.

(3) A time interval $[t_1,t_2]$ in I_b is assigned to "L–type" if every $b_i(t)$ is either a top, bottom, or left convex point of $CH($ R$(t) \cup b_i(t)$) where $t\in[t_1,t_2]$.

Fig. 6 illustrates the set I_b obtained by intersecting all I_{bi}'s according to above steps. In the same way, we can compute I_{rj} for each $r_j(t)$, and get their intersection I_r.

By Theorem 3, it is easy to understand that the final result I, the set of separable time intervals for sets R and B, can be obtained by intersecting I_b and I_r in the following way (see Fig. 7):

a time interval $[t_1,t_2]\in I$ if $[t_1,t_2]\subset[a,b]$ and $[t_1,t_2]\subset[c,d]$ where $[a,b]\in I_b$ and $[c,d]\in I_r$.

In the following, we shall summarize the algorithm, and give an analysis of its time complexity.

Algorithm 1. Given two sets of points R=$\{r_1,r_2,..,r_n\}$ and B=$\{b_1,b_2,..,b_m\}$ having k–motion in 2–D. We compute the set of separable time intervals.

Step (1) For each b_i, compute I_{bi}, the set of time intervals during which b_i is a convex point of $CH($ R$(t) \cup b_i(t)$);

Step (2) Intersect all I_{bi}'s to get I_b, the set of time intervals with either R–type or L–type;

Step (3) For each r_j, compute I_{rj}, the set of time intervals during which r_j is a convex point of $CH($ B$(t) \cup r_j(t)$);

Step (4) Intersect all I_{rj}'s to get I_r, the set of time intervals with either R–type or L–type;

Step (5) Intersect I_b and I_r to get the final result I, the set of separable time intervals.

The running time of this algorithm is clearly polynomial in n and m. Let M be $O(\lambda(m,4k))$ and N be $O(\lambda(n,4k))$. Specifically, Step (1) can be accomplished, for each b_i in time $O(M\log N)$, in overall time $O(m\cdot M\log N)$ [1]. In Step (2), we need to merge m sets of time intervals produced in Step (1), and identify the type (R–type or L–type) for each time interval. Since each I_{b_i} is a sorted list with $O(N)$ time intervals, this step can be carried out by a linear merging algorithm for m sorted lists in time $O(m\cdot N\cdot\log m)$. In the same way, we know that Step (3) and Step (4) need $O(n\cdot M\log M)$ and $O(n\cdot M\cdot\log n)$ time respectively. Since list I_b has $O(m\cdot N)$ sorted elements while there are $O(n\cdot M)$ sorted elements in list I_r, Step (5) can be performed by a linear merging procedure in time $O(mN+nM)$. Hence, totally Algorithm 1 can be accomplished in time $O(mN\cdot\log mN + nM\cdot\log nM + mN+nM)$.

In [6], Sharir gave the almost linear upper bounds for N and M:

$M = O(m\cdot\alpha(m)^{O(\alpha(m)^{4k-3})})$ and $N = O(n\cdot\alpha(n)^{O(\alpha(n)^{4k-3})})$, where $\alpha(i)$ is the functional inverse of Ackermann's function. The function $\alpha(i)$ is very slowly growing, but tends to infinity with i. Note that $\alpha(i)\leq 4$ for all $i\leq A$ which is a tower with 65536 2's,

i.e. $A = 2^{2^{\cdot^{\cdot^{\cdot 2}}}} \longrightarrow$ with 65536 2's in the exponential tower.

Thus $\alpha(i)\leq 4$ is suitable for all practical purposes. Assume that the sizes of sets R and B are almost equal. The formula of time complexity for Algorithm 1 can be simplified to $O(C\cdot n^2\cdot\log n)$ where C is a function of $\alpha(n)$.

4. Separability of k–motion in 1–dimension

We now consider a more special case, where input points are specified to be k–motion in 1–D. Suppose that one point can run over another point without collisions. Each point p_i can be described as follows.

$p_i(t) = C_{ik}t^k + C_{ik-1}t^{k-1} + \cdots + C_{i0}t^0$

where each C_{ih} is a constant coefficient. At time t, define

$$b_{max}(t) = \underset{i}{MAX}\{b_i(t)\},$$
$$b_{min}(t) = \underset{i}{MIN}\{b_i(t)\},$$
$$r_{max}(t) = \underset{j}{MAX}\{r_j(t)\} \text{ and}$$
$$r_{min}(t) = \underset{j}{MIN}\{r_j(t)\}.$$

Since $CH(\mathrm{R}(t)) = \{r_{max}(t), r_{min}(t)\}$, $b_i(t)$ is a convex point of $CH(\mathrm{R}(t) \cup b_i(t))$ if and only if $b_i(t) < r_{min}(t)$ or $b_i(t) > r_{max}(t)$, where the first condition specifies $b_i(t)$ to be a left convex point while the second condition classifies $b_i(t)$ into the set of right convex points. Hence, we can immediately deduce a more simple property from Lemma 4.

Lemma 5. Assume that the points in R and B are k–motion in 1–D. At time t_0, $\mathrm{R}(t_0)$ and $\mathrm{B}(t_0)$ are linearly separable *iff* one of the following conditions is true
(i) $b_{max}(t_0) < r_{min}(t_0)$ or (ii) $b_{min}(t_0) > r_{max}(t_0)$.

By Lemma 5, Algorithm 1 can be simplified to the following version.

Algorithm 2. Given two sets of points $\mathrm{R}=\{r_1, r_2, .., r_n\}$ and $\mathrm{B}=\{b_1, b_2, .., b_m\}$ having k–motion in 1–D. We compute the set of separable time intervals.

Step (1) Compute $r_{min}(t)$ and $r_{max}(t)$ for $t \in [0,\infty)$, the lower envelope and upper envelope of $\mathrm{R}(t)$ respectively;

Step (2) Compute $b_{min}(t)$ and $b_{max}(t)$ for $t \in [0,\infty)$, the lower envelope and upper envelope of $\mathrm{B}(t)$ respectively;

Step (3) Comput I_{rb}, the set of time intervals during which $r_{max}(t) < b_{min}(t)$;

Step (4) Comput I_{br}, the set of time intervals during which $b_{max}(t) < r_{min}(t)$;

Step (5) Merge I_{rb} and I_{br} to be the final result I, the set of separable time intervals.

Since the envelope functions of $R(t)$ and $B(t)$ respectively have N and M pieces of polynomial functions, where N is $O(\lambda(n,k))$ and M is $O(\lambda(m,k))$, Step (1) and Step (2) can be performed by a simple divide–and–conquer technique in time $O(M\log N + M\log M)$ [1]. To accomplish Step (3) and Step (4), we need only $O(M+N)$ time by executing a standard merging procedure. For Step (5), the time to combine I_{rb} and I_{br} can be dominated by previous time order. Hence, the time complexity of Algorithm 2 is $O(N\log N + M\log M + M+N)$. Assume that R and B have the equal size, n. We have a more simple formula, $O(C\cdot n\log n)$, where C is a function of $\alpha(n)$.

For another kind of special case of separability problem, we consider the problem of which points are moving in 1–D with 1–motion. The position of each p_i now is a linear function of time t,

$p_i(t) = C_{i1}t^1 + C_{i0}t^0$ where C_{i1} and C_{i0} are all constant coefficients.

Hence, the upper envelope of $R(t)$, say $r_{\max}(t)$, is a concave piecewise linear function while its lower envelope, $r_{\min}(t)$, is a convex one. Since there is at most two separable time intervals in this simple case, separability here can be transformed to a linear programming problem as follows. Specifically, given two sets of points having 1–motion in 1–D, $R=\{r_1,r_2,..,r_n\}$ and $B=\{b_1,b_2,..,b_m\}$, we therefore seek the two optimal solutions, one for "Maximize t" and the other for "Minimize t", satisfying the conditions

$$(1)\qquad \begin{array}{lll} y \leq r_i(t) & & \text{where } i=1,2,\ldots,n \\ y \geq b_j(t) & & \text{where } j=1,2,\ldots,m \\ t \geq 0 & & \end{array}$$

and we also find another pair of minimum and maximum of t meeting the requirements

$$(2)\qquad \begin{array}{lll} y \geq r_i(t) & & \text{where } i=1,2,\ldots,n \\ y \leq b_j(t) & & \text{where } j=1,2,\ldots,m \\ t \geq 0 & & \end{array}$$

Without loss of generality we assume that the optimal solutions of (1) and (2) are respectively $t_{\max 1}$, $t_{\min 1}$, $t_{\max 2}$, and $t_{\min 2}$. It is easy to understand that $[t_{\min 1}, t_{\max 1}]$ and $[t_{\min 2}, t_{\max 2}]$ are separable time intervals. For the time complexity, since Megiddo [3] has proposed a linear time algorithm to solve linear programming problems with two variables, the linearly separable problem with 1–motion in 1–D can be solved in time $O(m+n)$.

5. Concluding Remarks

In this paper we proposed an algorithm to solve the *separability problem* in dynamic computational geometry. For input points having k–motion in 2–D, our algorithm can be accomplished in time $O(mN\cdot\log mN + nM\cdot\log nM + mN+nM)$, where N and M is almost linearly proportional to the size of input set. In the special case, the points are k–motion in 1–D, the complexity can be reduced to $O(N\cdot\log N + M\cdot\log M + N+M)$. If the points are restricted to 1–motion in 1–D, our algorithm needs only linear time $O(m+n)$.

In d–dimension, the dynamic separability problems are much harder than static ones. For static version, linear separability can always be transformed to a linear programming problem [6]. Specifically, given two sets of points in d–dimension, say $R=\{r_1,r_2,\ldots,r_n\}$ and $B=\{b_1,b_2,\ldots,b_m\}$, we seek a $(d-1)$–dimension hyperplane

$$\sum_i C_i X_i = 0$$

satisfying the conditions

$$\sum_i C_i X_{ir_j} \leq 0 \quad \text{for } r_j \in R \quad \text{and} \quad \sum_i C_i X_{ib_j} \geq 0 \quad \text{for } b_j \in B.$$

This is clearly a linear programming, which can be solved in linear time by Megiddo's algorithm [4]. On the contrary, the dynamic linear separability is not a simple "Yes" or "No" problem. We need describe the whole set of separable time intervals. For the problems moving in 1–D and 2–D, by Lemma 4 and lemma 5, an efficient algorithm can be designed successfully. Although Theorem 1 holds for any fixed d–dimension, it is still very difficult to use this *iff* condition to construct an algorithm, even for 3–D case. It could be an interesting future research topic to find good characteristics of linear separability in dynamic computational geometry for high dimension.

References

[1] M. J. Atallah, Dynamic computational geometry, *Proceedings of 24th IEEE Annual Symposium Foundations of Computer Science*, (1983), pp. 92–99.

[2] S. Hart and M. Sharir, Nonlinearity of Davenport–Schinzel sequences and of generalized path compression schemes, *Combinatorica*, 6(2), (1986), pp. 151–17

[3] N. Megiddo, Linear–time algorithms for linear programming in R^3 and related problems, *SIAM J. Comput.*, 12(4), (1983), pp. 759–776.

[4] N. Megiddo, Linear–programming in linear time when the dimension is fixed, *J. ACM*, 31(1), (1984), pp. 114–127.

[5] F. P. Preparata and M. I. Shamos, *Computational Geometry*, Springer–Verlag, New York, 1985.

[6] M. Sharir, Almost linear upper bounds on the length of general Davenport–Schinzel sequences, *Combinatorica*, 7(1), (1987), pp. 131–143.

[7] J. Stoer and C. Witzgall, *Convexity and optimization in finite dimensions I*, Springer–Verlag, New York, 1970.

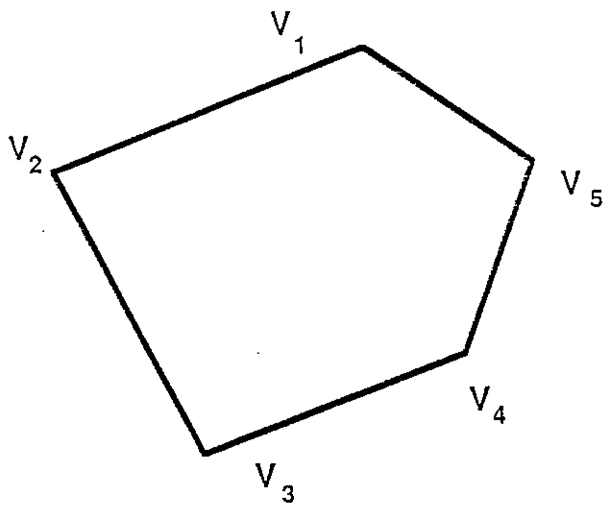

Fig. 1. Convex hull CH(B) of set B.

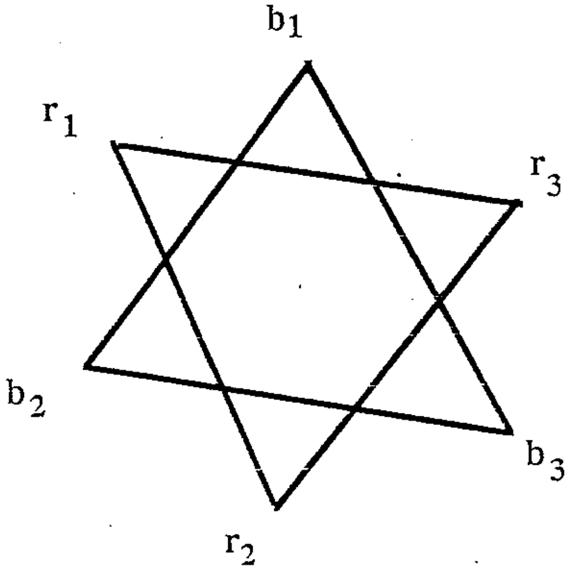

Fig. 2. A counterexample of Corollary 2.

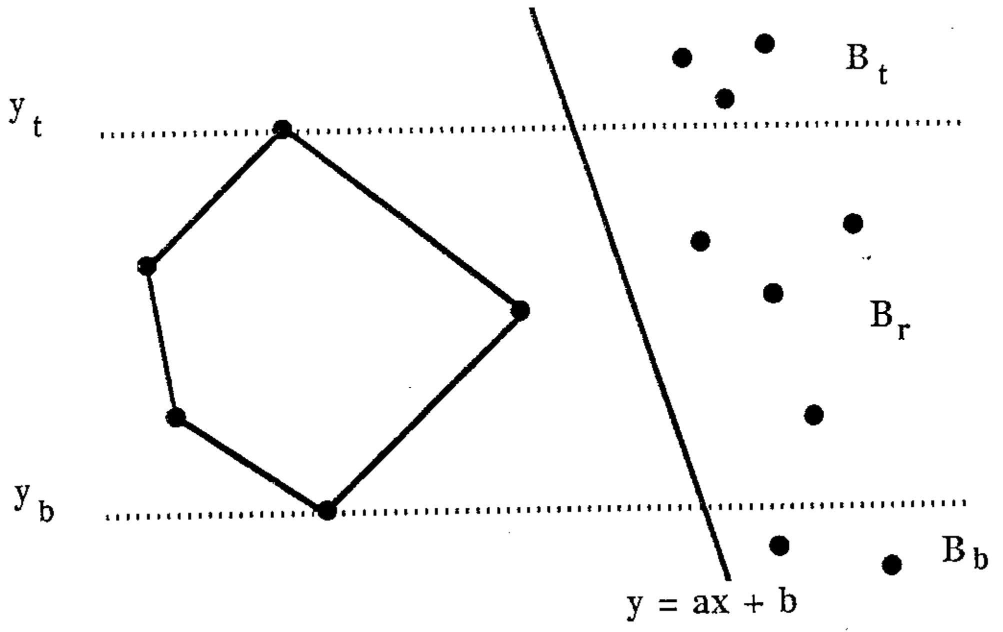

Fig. 3. A classification of set B.

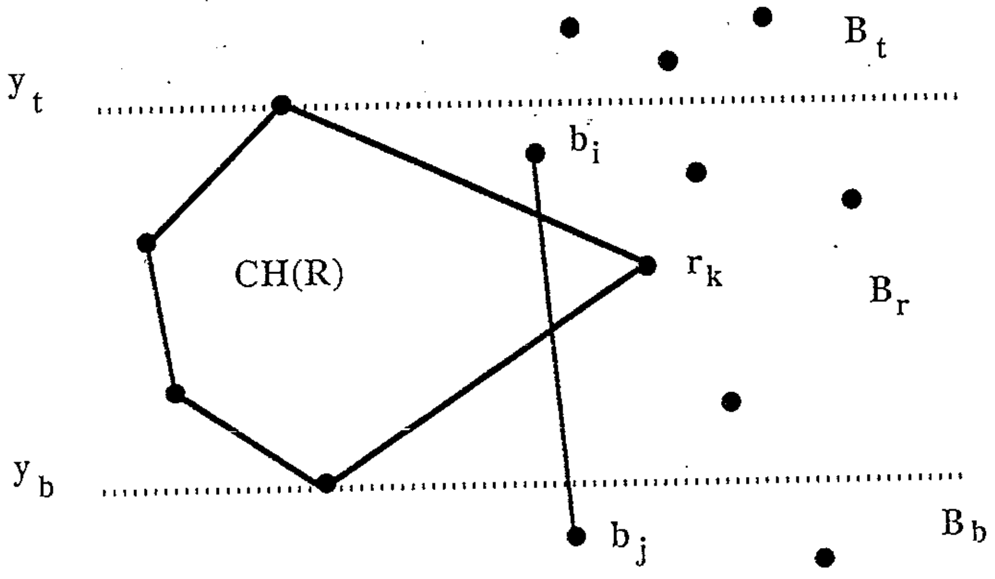

Fig. 4. Segment $\overline{b_i b_j}$ intersects CH(R) on two and exactly two point.

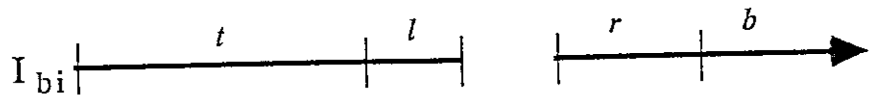

Fig. 5. The list of I_{bi}.

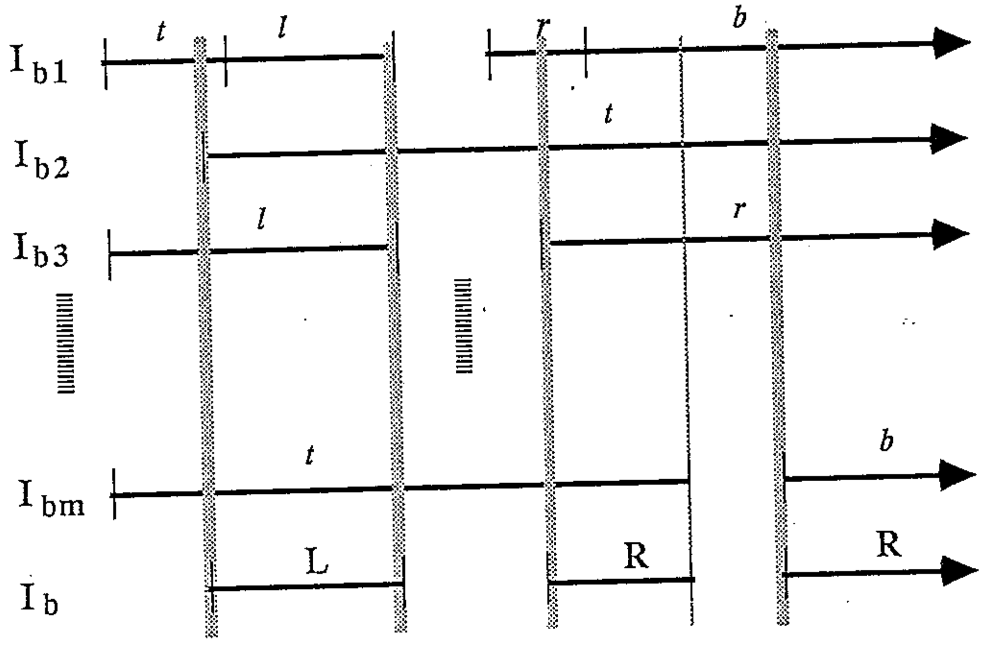

Fig. 6. Intersect all I_{bi}'s to be one I_b

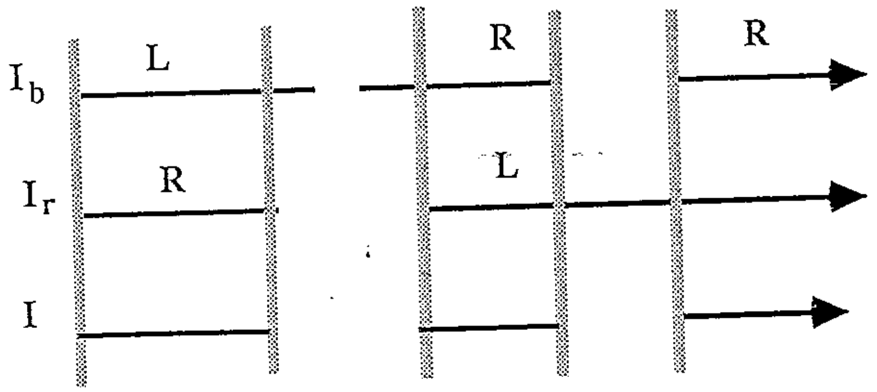

Fig. 7. Merge I_b with I_r to I.

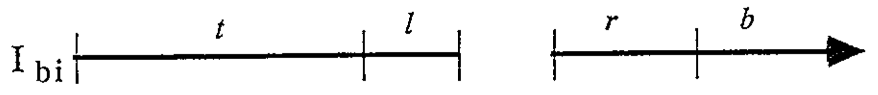

Fig. 5. The list of I_{bi}.

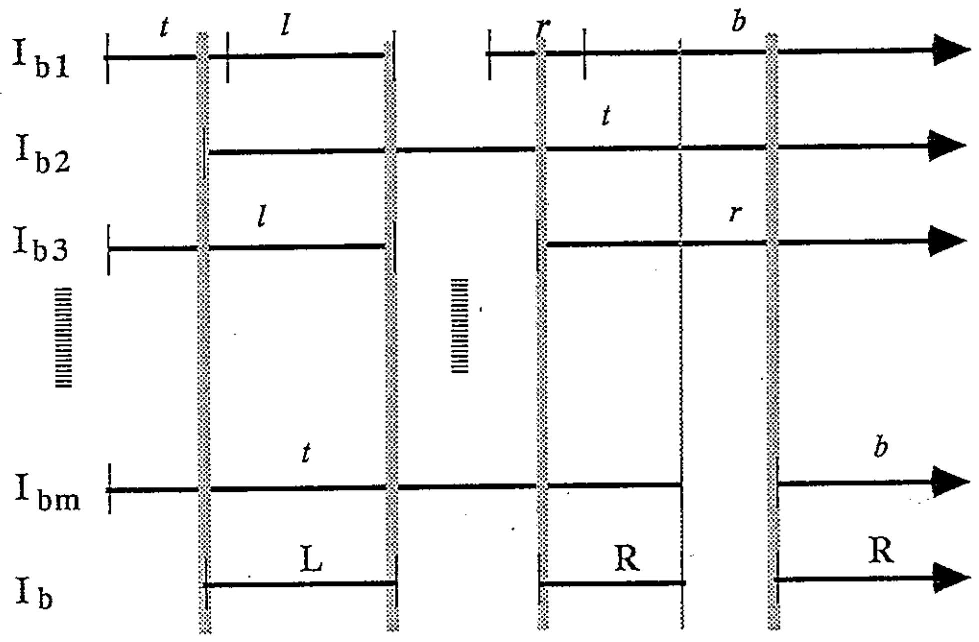

Fig. 6. Intersect all I_{bi}'s to be one I_b

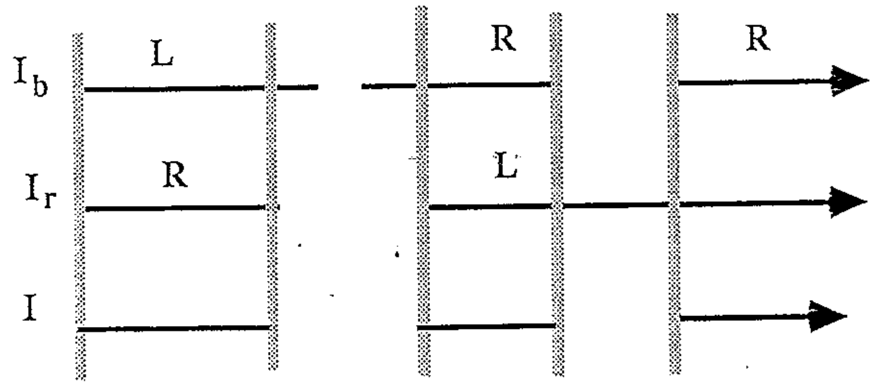

Fig. 7. Merge I_b with I_r to I.