LOCATION OF LARGEST ISOTHETIC EMPTY RECTANGLE AMONG ARBITRARILY ORIENTED LINE SEGMENTS

A dissertation submitted in partial fulfilment of the requirements for the M.Tech. (Computer Science) degree of the Indian Statistical Institute

By

ARANI SINHA

under the supervision of

DR. BHARGAB B. BHATTACHARYA

and

MR. SUBHAS C. NANDY

JULY 1992

INDIAN STATISTICAL INSTITUTE

203 B.T. ROAD, CALCUTTA - 700 035

ACKNOWLEDGEMENTS

I am very grateful to my guide, Dr. Bhargab. B. Bhattacharya who introduced me to the interesting field of computational geometry. I would also like to thank Mr. Subhas C. Nandy for his constant help and guidance during the course of this dissertation. Last but not the least, it is a pleasure to thank my friends Debashree Ghosh and Kaushik Dasgupta for their moral support and encouragement.

Arani Sinha

LOCATION OF LARGEST ISOTHETIC EMPTY RECTANGLE AMONG ARBITRARILY ORIENTED LINE SEGMENTS

Abstract:

In this dissertation, we consider the following problem of computational geometry which has direct application to VLSI layout design : given a set of arbitrarily oriented non-intersecting line segments in a rectangular floor, identify an isothetic empty rectangle of maximum area. An algorithm has been proposed for this problem which uses a novel data structure called line-search-tree. The time complexity of our algorithm is $O(n^2 \log^3 n)$ and uses O(nlogn) space.

Keywords

Computational geometry, VLSI layout, maximum empty rectangle, line sweep paradigm, computational complexity.

1. INTRODUCTION

Computational geometry involving isothetic rectangles plays an important role in computer-aided design, for example, VLSI layout design, operations research, computer graphics to name a few. An **isothetic rectangle** is one whose sides are parallel to the co-ordinate axes. Many algorithms now exist which recognize the largest empty rectangle in a floor with some randomly distributed point defects. The problem was first introduced by Naamad et.al. [1] and an algorithm was suggested with time complexity O(min (n^2, Rlogn)), where n is the number of points and R is the number of isothetic maximal-empty rectangles(MER) present on the floorplan. This complexity was later improved to ($R + n\log^2 n$) [2] and then to ($R + n\log n$) [3]. The algorithms given in [4] and [5] locate the maximum area empty rectangle without enumerating all MER's with time complexity $O(n\log^3 n)$ and $O(n\log^2 n)$ respectively. The problem of finding all maximal empty rectangles in an ensemblement of non-overlapping isothetic polygonal obstacles was studied in [6] and an algorithm was proposed with time complexity O(R + nlogn). In this dissertation, we have relaxed the constraint of having isothetic obstacles. Here we propose an algorithm for finding an empty isothetic rectangle of maximum area in an assemblage of arbitrarily-oriented non-intersecting straight line segments as obstacles. The time and space complexities of the algorithm are $O(n^2\log^3 n)$ and O(nlogn) respectively.

2. MAXIMUM EMPTY RECTANGLES: OBSERVATION AND CONCEPTS

Let us assume that a set of arbitrarily oriented straight line segments called "sticks" be distributed in a two-dimensional, isothetic rectangular floor. The four boundary lines of the floorplan are also considered as sticks. Without any loss of generality, the bottom_left corner of the floorplan is considered to be the origin of the co-ordinate system. Unlike the scenario for points and isothetic polygons as obstacles, in this case, the concept of the maximal empty rectangle, as we shall show shortly, is not the same. Here, we define a sort of "pseudo- maximal empty rectangles" to facilitate our search strategy.

Convention: Henceforth an MER will be denoted as R[(a,b)(c,d)] where (a,b) is the top-left corner and (c,d) is the bottom-right corner.

Definition: A stick is a straight line segment $L1[(a_1,b_1)(c_1,d_1)]$ with $b_1 >= d_1$. Throughout our discussion we shall assume the equation of the stick $L_i[(a_i,b_i)(c_i,d_i)]$ is $y = m_i.x + k_i$, where m_i is the slope of the stick and k_i is its intercept on the y-axis.

Definition: A stick is said to be a support of a rectangle if it touches the rectangle at its boundary.A support may be of two types:

1. touching a corner of an rectangle, called a flexible

support. It is so called because the corner of the rectangle can lie anywhere on the stick.

2. touching a side of an rectangle called a fixed support. It is so called because the position of the side touching this support becomes fixed.

Fig. 1 shows the two kinds of supports.

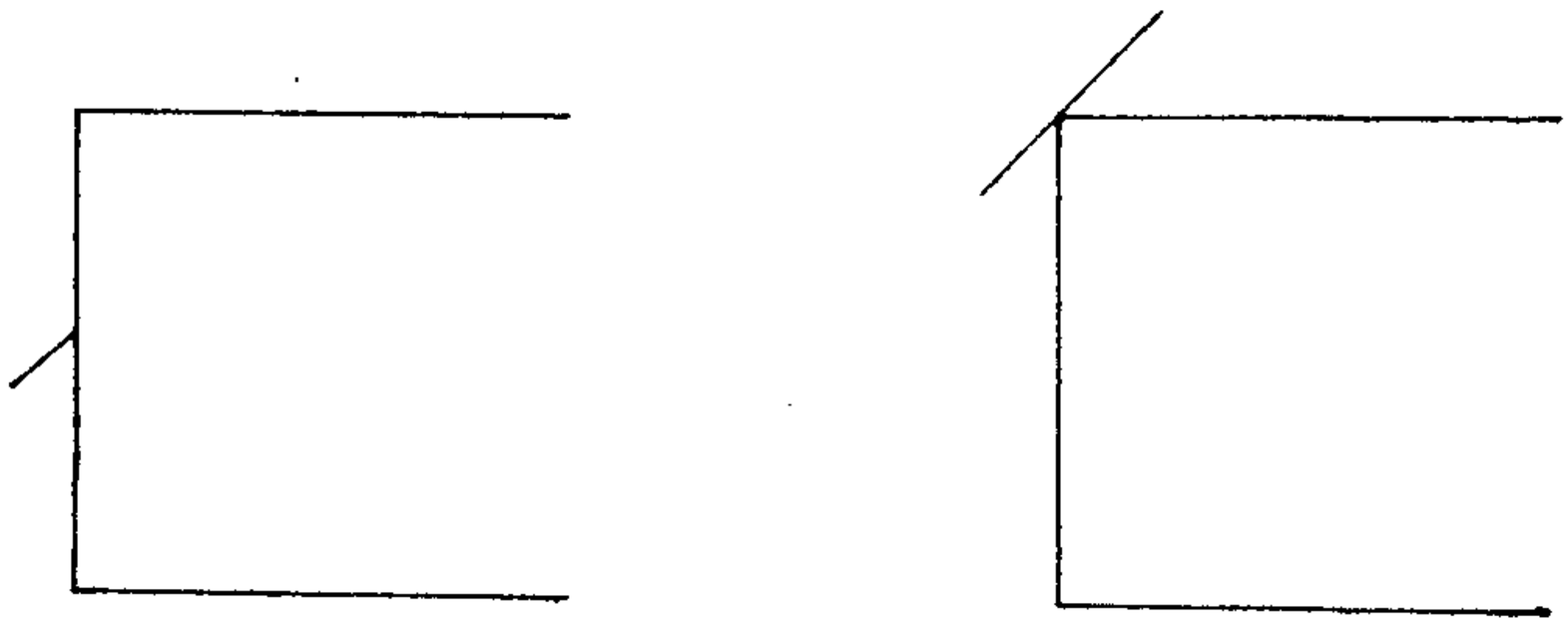


Fig. 1

Definition: A rectangle supported by a given set of sticks is said to be pseudo-maximal and empty if no empty rectangle of larger size is possible with the same set of supporting sticks. Henceforth we shall refer to a pseudo-maximal isothetic empty rectangle by the term maximal_empty_rectangle (MER).

We first investigate the nature of rectangles supported only by two flexible supports i.e. by sticks L_1 and L_2 at two opposite corners(Fig. 2).

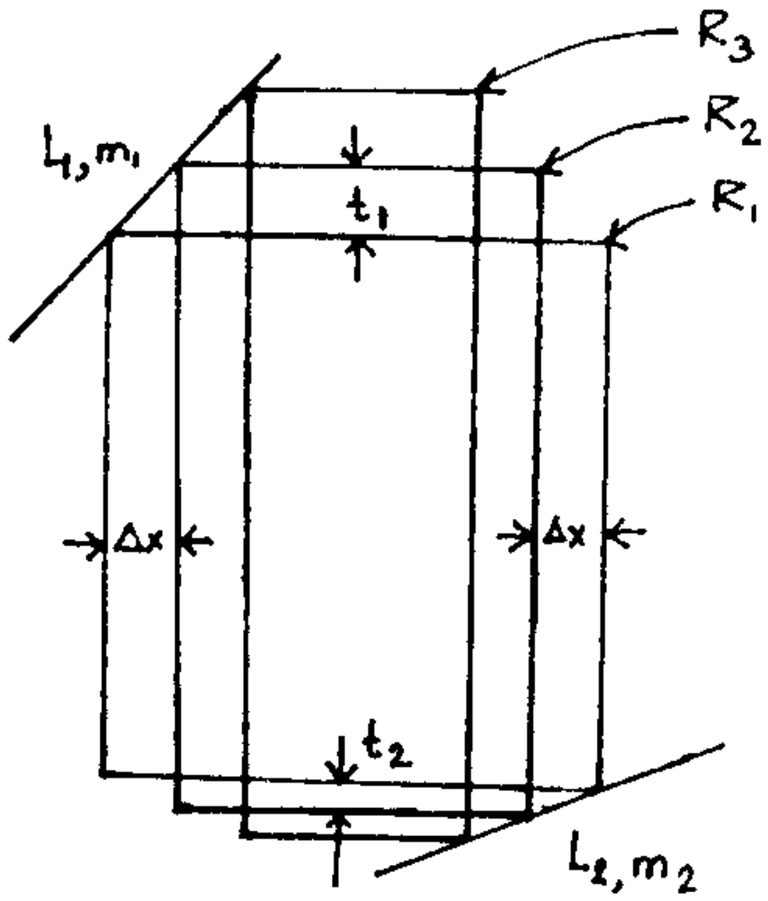


Fig. 2

The absolute values of the gradient of the lines L_1 and L_2 are m_1 and m_2 respectively. Let x_arm (y_arm) denote the horizontal (vertical) side of an arbitrary rectangle touching L_1 and L_2. Consider now three rectangles labelled R_1, R_2, R_3 as in Fig. 2. Their areas can be evaluated as follows:

Area of the rectangle R_1

Let the length of the x_arm be x and the length of the y_arm of this rectangle be y.

Then, $A(R_1) = x.y$.

Area of the rectangle R_2

Let the y_arm be displaced by a distance Δx in the x_direction, x remaining constant.

Then, $t_1/\Delta x = m_1$ or $t_1 = m_1.\Delta x$

and $t_2/\Delta x = m_2$ or $t_2 = m_2.\Delta x$.

Then, $A(R_2) = x.(y + m_1.\Delta x - m_2.\Delta x)$

Area of the rectangle R_3

Again keeping the length of the x_arm equal to x, the y_arm is shifted by a distance of Δx.

Then, $A(R_3) = x.(y + 2.m_1.\Delta x - 2.m_2.\Delta x)$

Hence, $A(R_1) - A(R_2) = x.(m_1 - m_2).\Delta x$.

Again, $A(R_2) - A(R_3) = x.(m_1 - m_2).\Delta x$.

Therefore, if $A(R_1)$ is $> (=, <) A(R_2)$, then accordingly $A(R_2) > (=, <) A(R_3)$.

This leads to the following lemma:

Lemma 1: If $m_1 > m_2 > 0$, then area of the rectangles of same width increases as the top-left corner slides upward along L_1. If $m_2 > m_1 > 0$ then the area of the rectangles of same width increases as the top-left corner slides downward along L_1(see fig 3). Hence,

Observation 1: The area of the rectangles is monotonically increasing in one of the directions for the given pair of sticks, depending on their slopes and relative positions.

Now, the maximum of these monotonically increasing rectangles touches the endpoints of one of the sticks, depending on their lengths, slopes and relative positions.

Observation 2: The rectangle touching an extreme point of one of the sticks can be further extended in one of the dotted directions(Follows from Obs. 1. See Fig. 3).

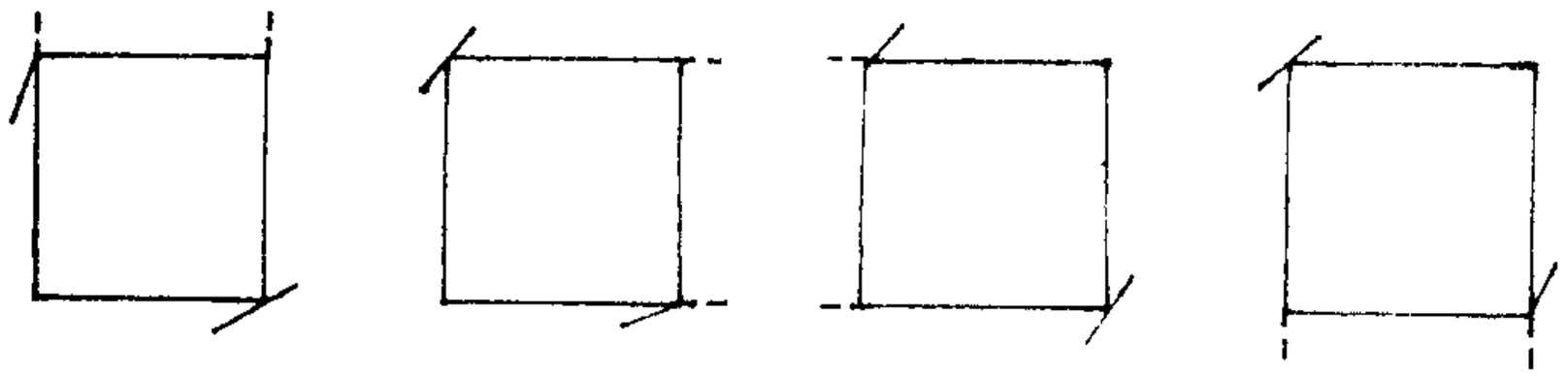


Fig. 3

Obviously, no MER can be defined by one support.

The other possible cases where an MER can have only two supports, shown in Fig. 4, along with Observation 2 lead to the following conclusion:

Lemma 2: No MER can be supported only by two or fewer sticks.

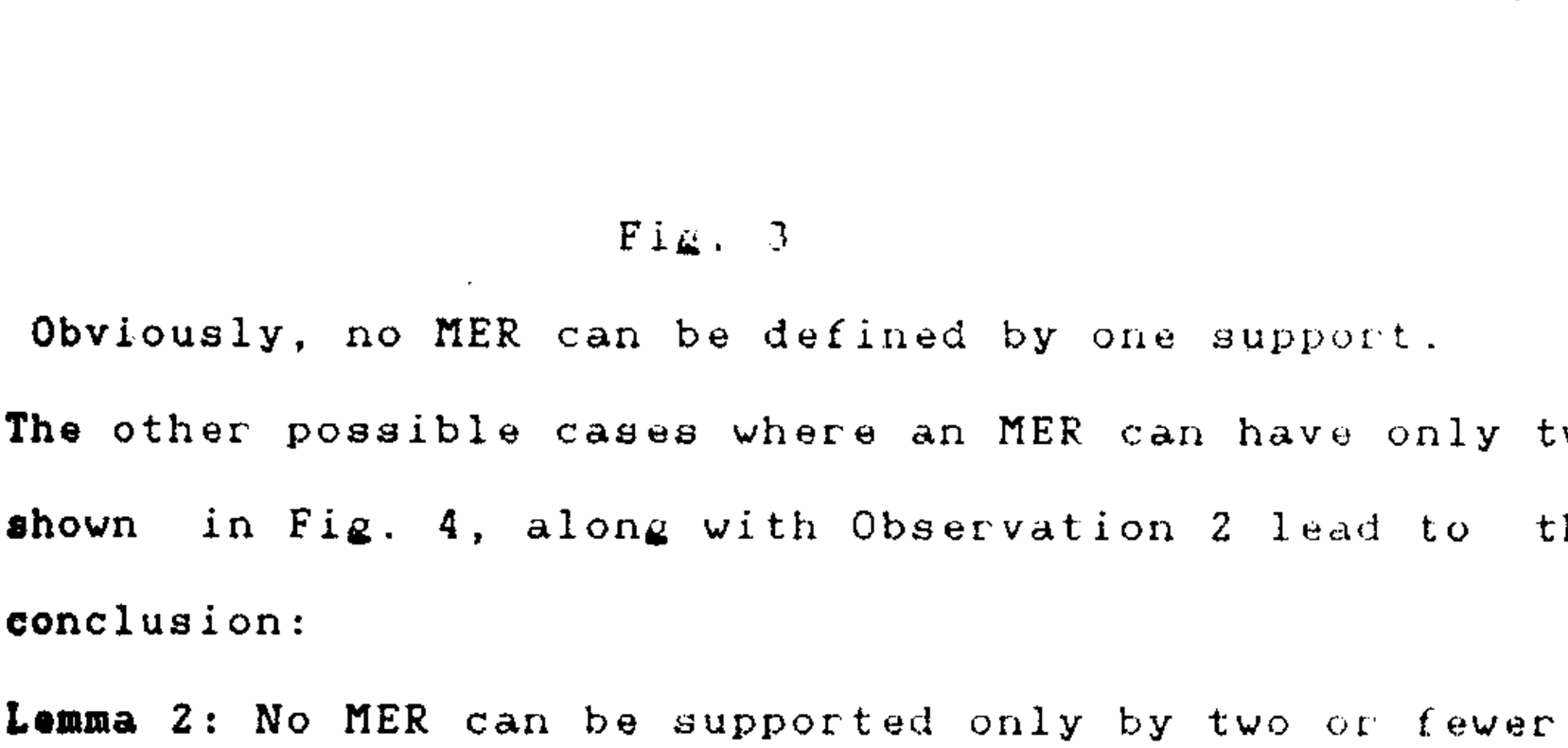


Fig. 4

Observation 4: The MER's with two flexible supports have at least another support, either fixed or flexible.

Fig 5a and 5b show the situations where

1: one more fixed support suffices.

2: one more flexible support suffices.

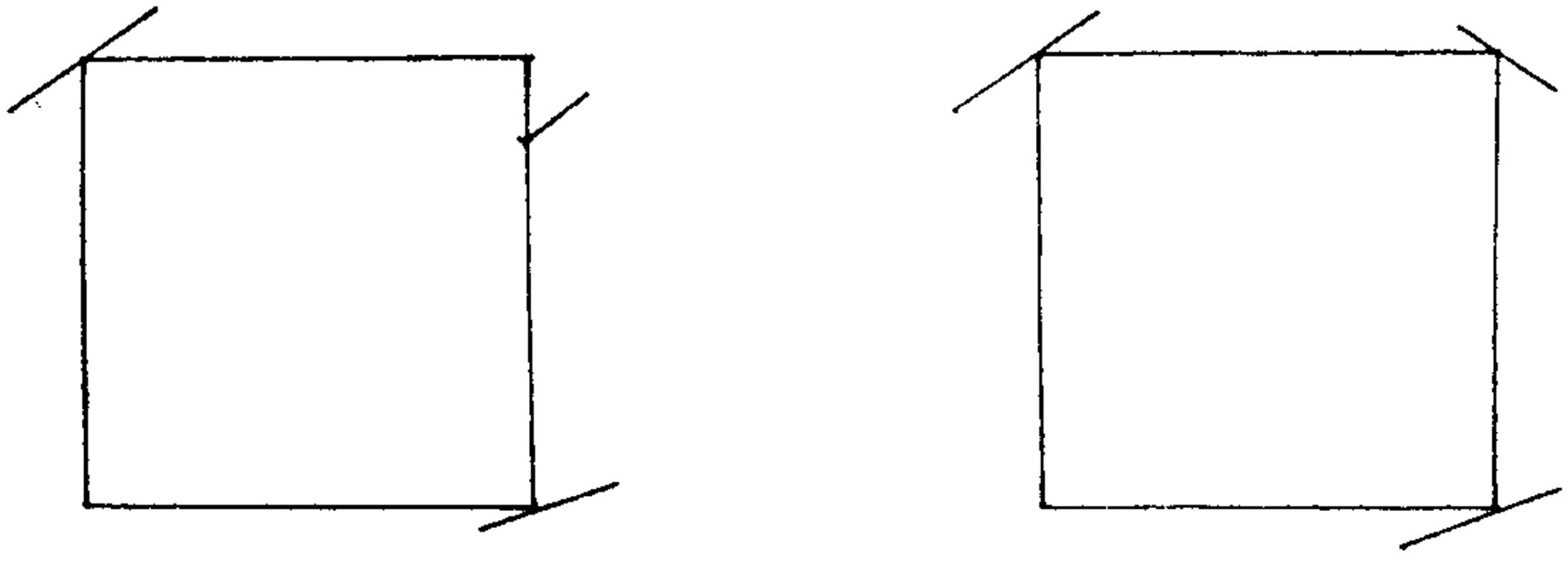


Fig. 5a

Fig. 5b

Definition: The minimum set of supporting sticks for an MER is one such that if one member of the set is removed, the remaining sticks cannot define the same MER.

Definition: An allowable combination of supporting sticks is one which is a superset of the the minimum set.

If an MER is defined by four fixed supports on four sides, the MER is unique and determining it is trivial. For determination of MER's with at least one flexible support, methods similar to those described below in Results 1 to 5 will be used.

Note that the number of rectangles having at least one flexible support is infinite. This is the rationale behind defining "pseudo-maximal empty rectangles".

In each of Results 1 to 5, the following method is used:

A variable point (x,y) is chosen on any one flexible support and an arbitrary rectangle is drawn with a corner on (x,y) which touches the minimum set of supporting set of supporting sticks for the MER. An expression for this area of this rectangle is found.

This expression is a quadratic in x and has one maximum value. The position of the maximum area rectangle is found by differentiation of the expression.

Over and above the minimum set of supporting sticks, position of other sticks could impose additional constraints on the MER and they are also considered.

Result1: Consider the set of rectangles with one flexible support $L_1[(a_1,b_1)(c_1,d_1)]$ at the top left corner and two fixed supports at points (p_1,q_1) and (p_2,q_2) on the sides adjacent to the opposite corner(Fig 6a). Let (x,y) on L_1 be the top left corner of an arbitrary rectangle. The bottom right corner is (p_1,q_2). The area of this rectangle is

$$A(x) = (p_1 - x)(y - q_2)$$

$$= (p_1 - x)(m_1.x + k_1 - q_2) \quad (1)$$

The conditions $c_1 <= x <= p_2$, $q_1 <= y <= b_1$ are to be satisfied, to ensure that the MER is supported by L1 and two fixed supports. If the value of x corresponding to the maximum area does not satisfy the conditions, no MER is reported.

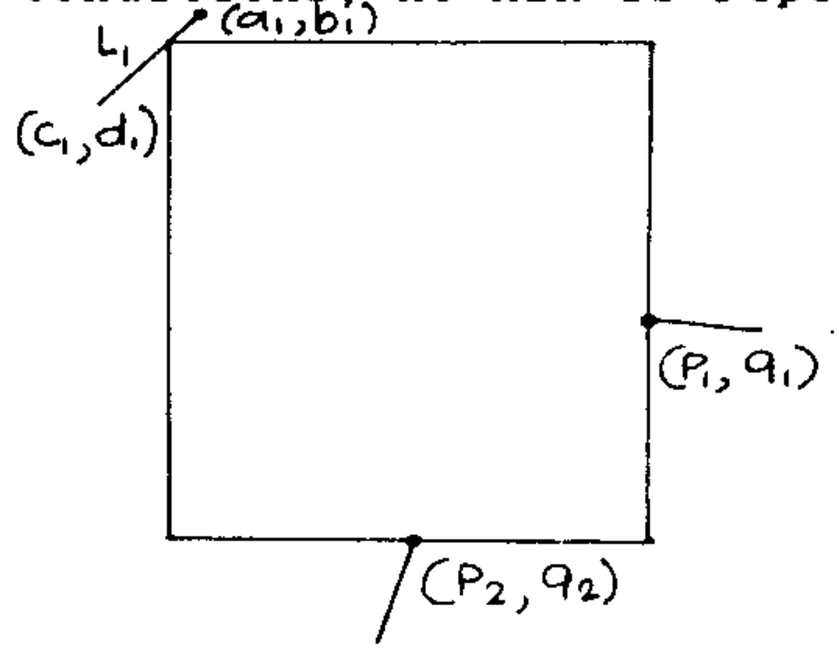


Fig. 6a

Let $L_3[(a_3,b_3)(c_3,d_3)]$ be a stick which could be a fixed support at the top or at the left side of the rectangles as shown in Fig 6b and 6c.

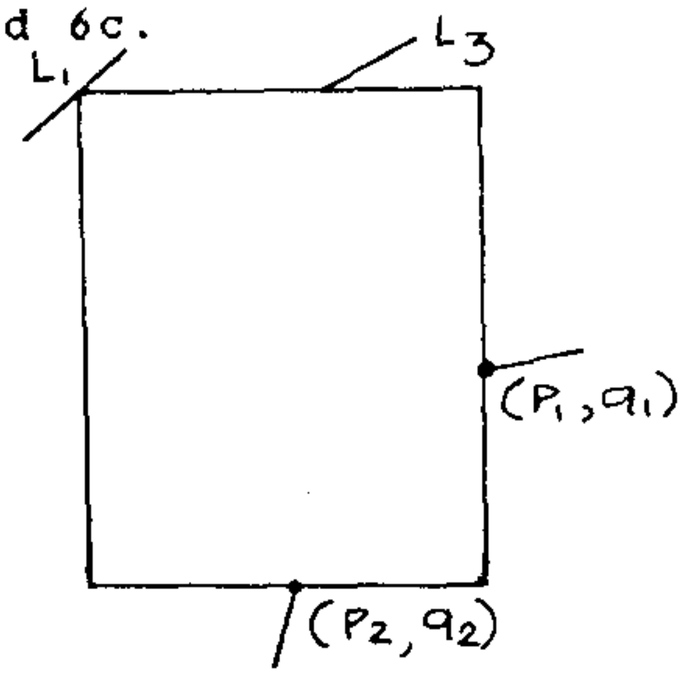


Fig. 6b

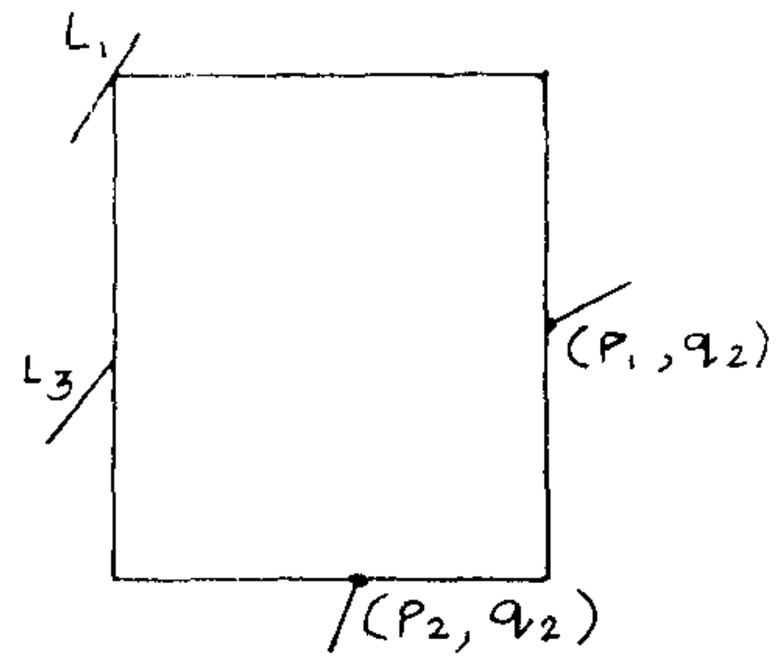


Fig. 6c

If (x_0,y_0) on L_1 is such that $A(x_0)$ is maximum and $y_0 > d_3$ (for Fig 6b), then an MER$[(m_1.d_3 + k_1, d_3)(p_1,q_2)]$ is reported. The MER for Fig 6b can be similarly determined if the constraint is violated.

Result 2: Consider the set of rectangles with two flexible supports, $L_1[(a_1,b_1)(c_1,d_1)]$ and $L_2[(a_2,b_2)(c_2,d_2)]$ lying at two adjacent corners and one fixed support at the point (p_2,q_2) on the opposite edge (Fig.7a). Let the point (x_1,y_1) on L_1 be the top-left corner of an arbitrary rectangle.

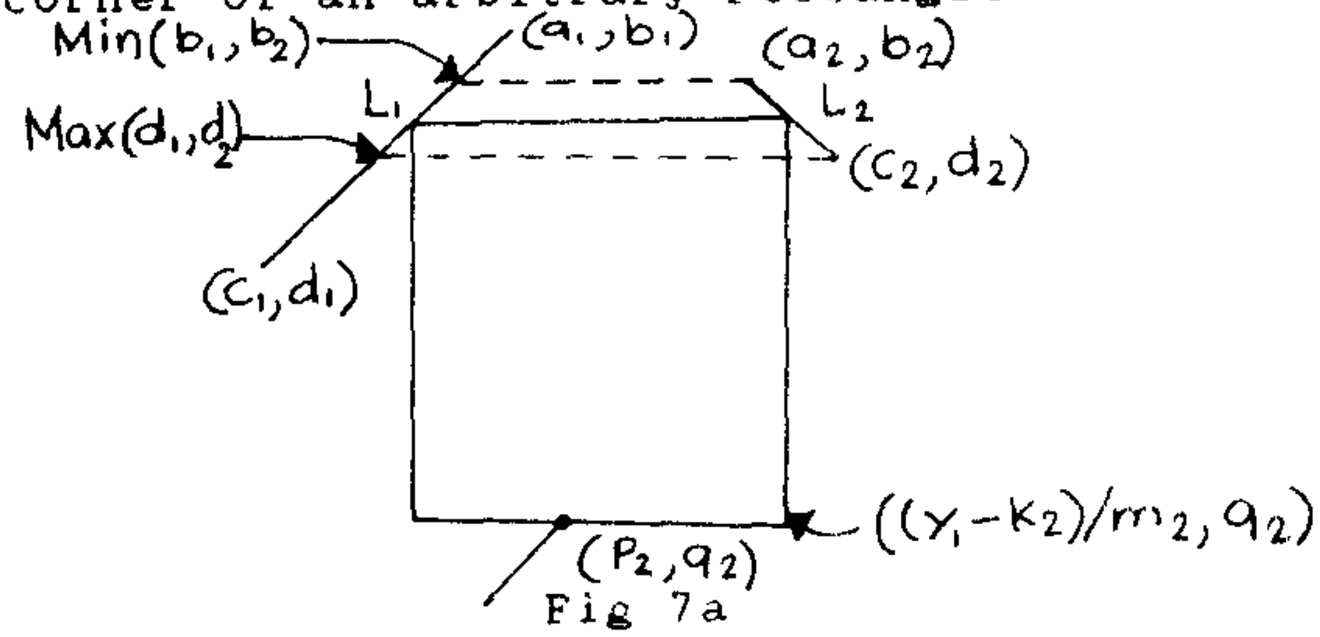


Fig 7a

Then the top-right corner is point $((y_1-k_2)/m_2, y_1)$ on L_2 and the bottom-right corner is $((y_1-k2)/m_2, q_2)$. Hence, the area of the rectangle is

$$A(x) = ((y_1-k_2)/m_2 - x_1)(y_1 - q_2)$$
$$= (((m_1.x_1+k_1-k_2)/m_2)-x_1)(m_1.x_1+k_1-q_2) \qquad (2).$$

The conditions $x_1 <= p_2$ and $\max[d_1,d_2] <= y_1 <= \min[b_1,b_2]$ being satisfied.

Constraints may be imposed by sticks as shown in Fig 7b, 7c and 7d. Here too, as in Result 1, a suitable MER touching the sticks is reported.

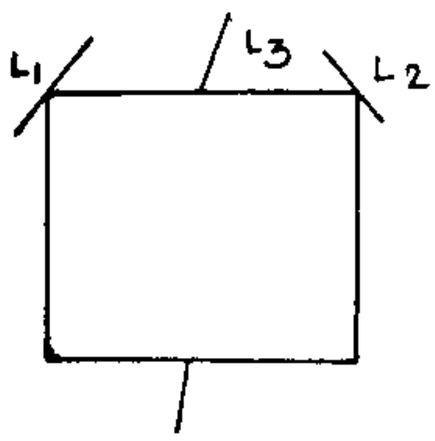


Fig. 7b

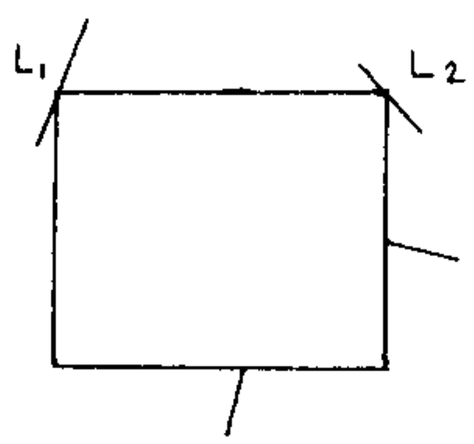


Fig. 7c

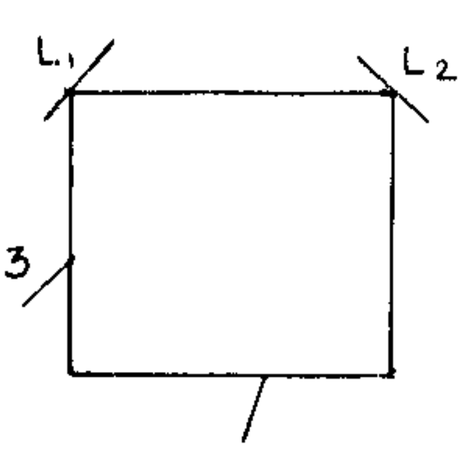


Fig. 7d

<u>Result 3:</u> Consider the set of rectangles with $L_1[(a_1,b_1)(c_1,d_1)]$ and $L_2[(a_2,b_2)(c_2,d_2)]$ as the flexible supports at the top-left and the bottom-right corners(Fig 8a).

Let $m_1 > m_2$. Thus by Lemma 1,the area of any rectangle bounded bt these supports increases as the top-left corner slides upwards along L_1. As a result, if there is a fixed support on the top at point(p,q) then the point $((q-k_1)/m_1, q)$ on L_1 is the top left corner of all rectangles possible. Let (x,y) on L_2 be the bottom

right corner of an arbitrary rectangle. The area of this rectangle is

$$A(x) = (x - (q-k_1)/m_1)(q-y)$$

$$= (x-(q-k_1)/m_1)(q-m_2.x-k_2) \qquad (3).$$

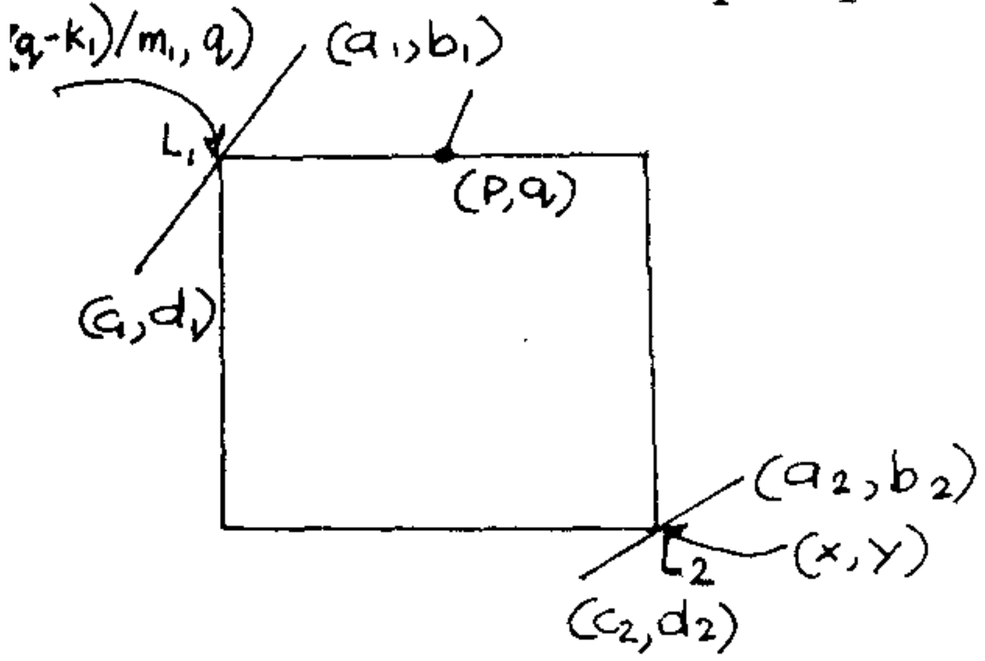


Fig 8a

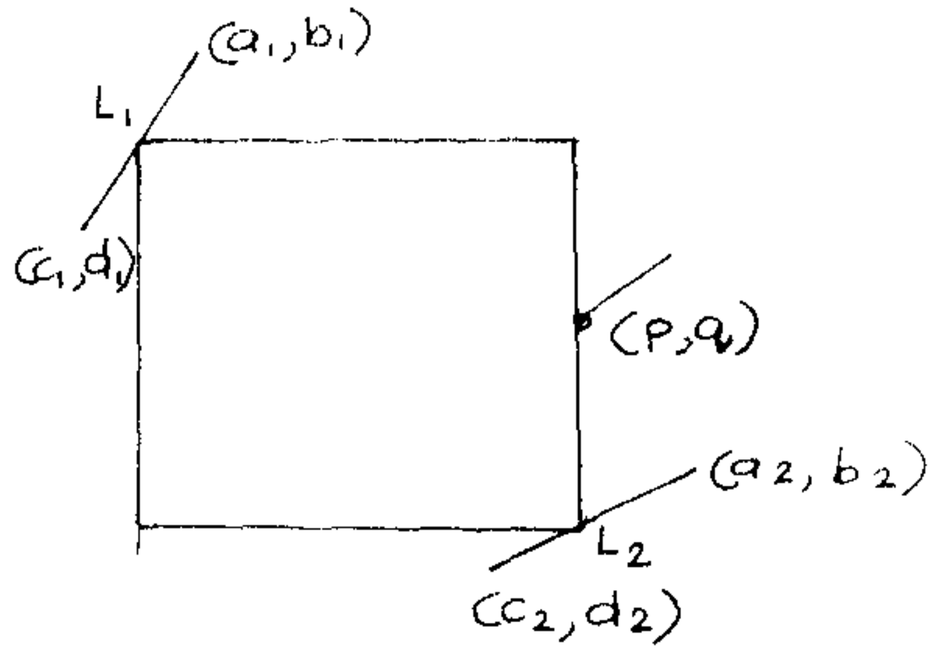


Fig 8b

The condition $(q - k_2)/m_2 <= x <= a_2$ being satisfied, failing which no MER is reported.

The case where a fixed support is present on the right side is handled similarly(Fig 8b).

In this case too, sticks shown in Fig 8c and 8d may impose constraints on the MER.A suitable MER is reported in each case.

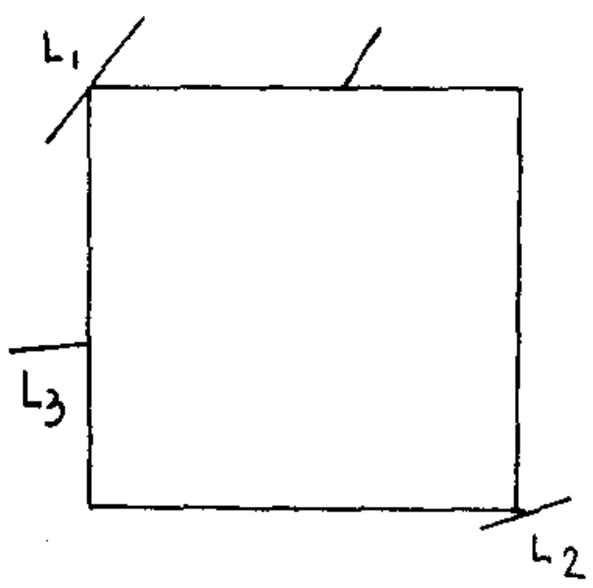


Fig. 8c

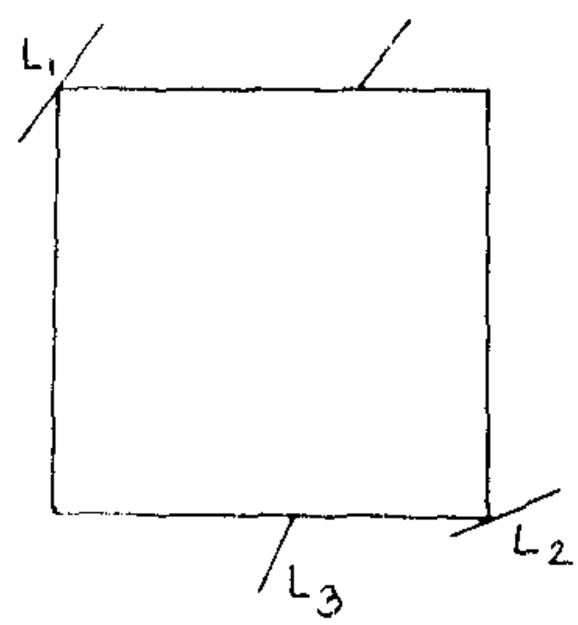


Fig. 8d

Result 4: Consider the set of rectangles with $L_1[(a_1,b_1)(c_1,d_1)]$, $L_2[(a_2,b_2)(c_2,d_2)]$, $L_3[(a_3,b_3)(c_3,d_3)]$ as the top-left, top-right and bottom-right flexible supports.(Fig 9a)

If $m_1 > m_3$ then, by Lemma 1,the area of an arbitrary rectangle increases as its top-left corner slides upwards along L_1.

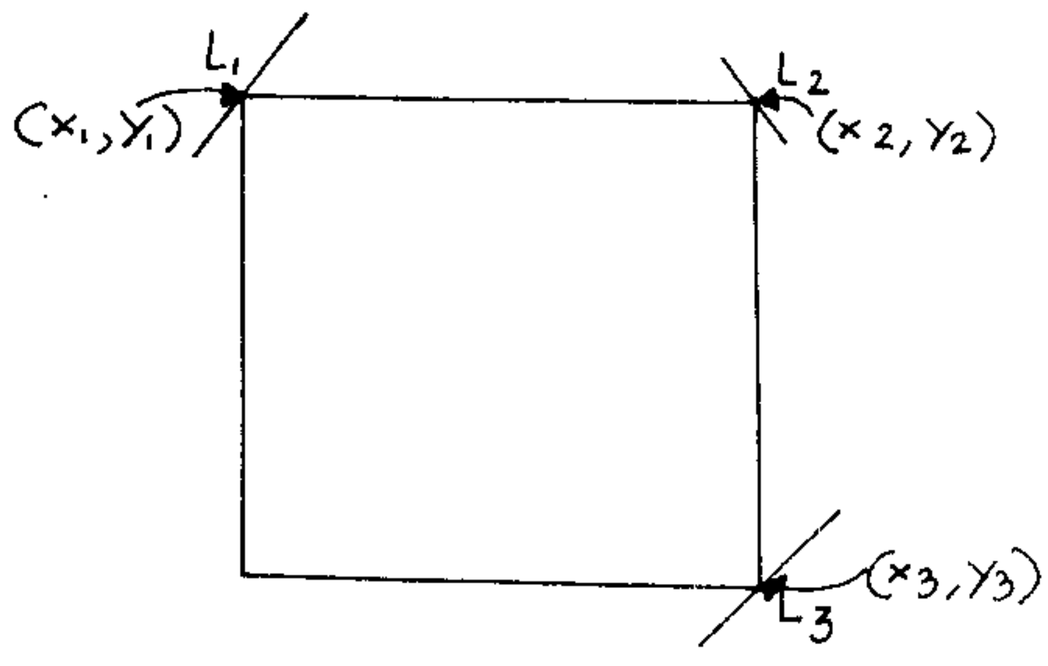


Fig. 9a

Choose a point (x_1,y_1) on L_1 as the top-left corner of an arbitrary rectangle. Then the top-right corner is (x_2,y_2) on L_2 will be given by $x_2=(y-k_2)/m_2$ and $y_2 = y_1$. The bottom-right corner is (x_3,y_3) on L_3 where $x_3 = x_2$ and $y_3 = m.x_3 + k_3$. Hence, the area

$$A(x) = (x_2 - x_1)(y_1 - y_3)$$

$$= (((m_1.x_1+k_1-k_2)/m_2-x_1).((m_1.x_1+k_1)-(m_1.x_1+k_1-k_2).(m_3/m_2)-k_3)$$

the conditions $\max[d_1,d_2] <= y_1 <= \min[b_1,b_2]$ and $d_3 <= y_3 <= b_3$ is to be satisfied.

The sticks imposing constraints may be positioned as shown in Fig 9b 9c 9d and 9e.An MER is reported in each case.

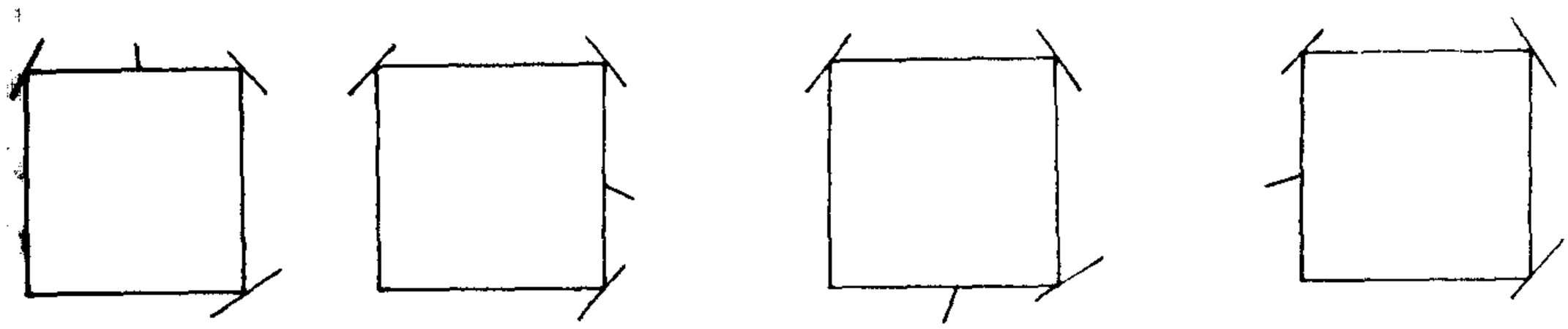


Fig. 9b Fig. 9c Fig. 9d Fig. 9e

Result 5: Consider the set of rectangles formed by four flexible supports $L_i[(a_i,b_i)(c_i,d_i)]$, i=1..4 at four corners (Fig 10 a).

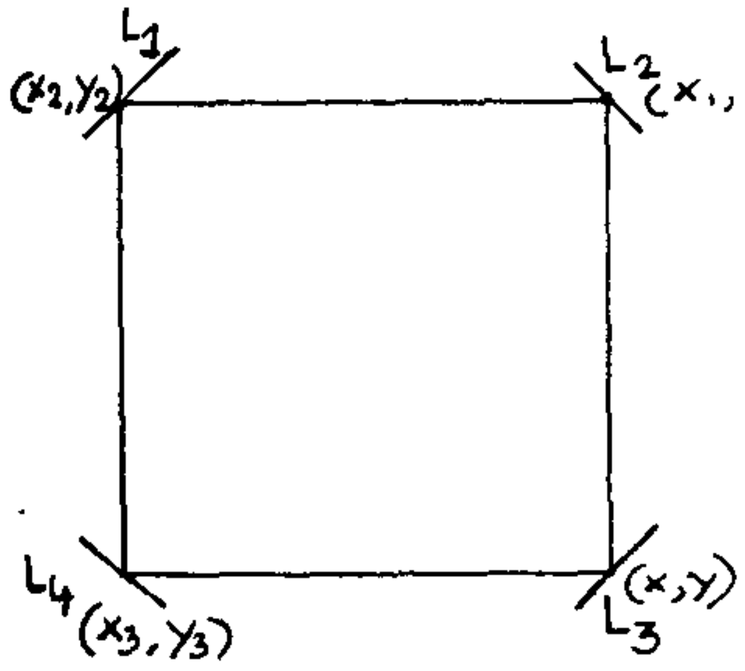


Fig. 10a

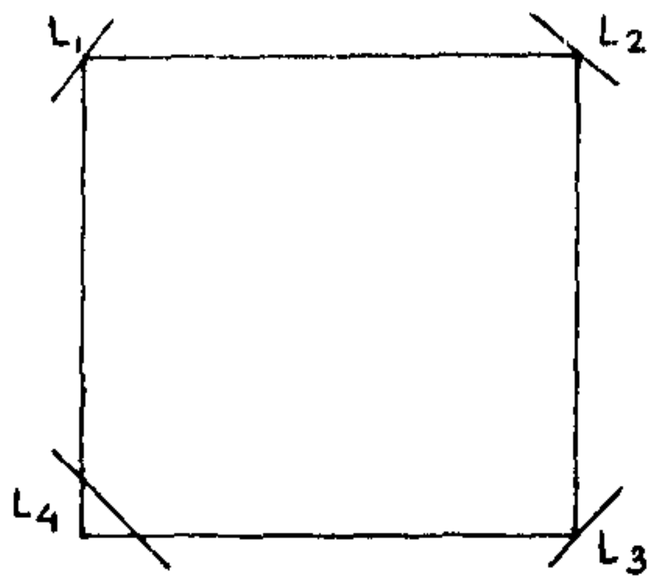


Fig. 10b

Let $m_1 > m_3$ (the other case being similar). Let (x,y) on L_3 be the bottom right corner of an arbitrary rectangle.The top right corner is (x_1,y_1) on L_2 where $x_1 = x$ and the top left corner is (x_2,y_2) on L_1 where $y_2 = y_1$.The bottom left corner is (x_3,y_3)on L_4 where $x_3 = x_2$. To ensure that the rectangle so formed is empty, we need to satisfy (to prevent erroneous reportings as shown in Fig 10 b)

the constraint $y_3 <= y$

i.e. $(m_4/m_1)(m_2.x+k_2-k_1)+k_4 <= m_3.x+k_3$

or, $x >= ((k_3 - k_4)m_1 - (k_2 - k_1)m_4)/(m_1.m_3 - m_2.m_4)$

The area of the rectangle is

$(x - x_2)(y_1 - y)$

$= (x - (m_2.x + k_2 - k_1)/m_1)(m_2.x + k_2 - y)$ (5)

The conditions are similar those in result 4.

There could be sticks imposing constraints on the four sides, the MER's are then suitably reported.

It can be shown that the area of all MER's with at least one corner on a flexible support can be calculated by using a method analogous to Result 1 through 5.

3. MER RECOGNITION

The basic idea is to consider all sets of sticks which could be supports of an MER. To this end, certain concepts are introduced.

Visibility

Definition: A stick L is left-visible (right-visible) from a point (p,q) if a horizontal ray drawn from (p,q) to the left (right) first hits L.

Clearly, two sticks can be supports of the same MER if they are visible from each other.

Observation 5: If gradient of stick L is positive,then no MER is possible with top right corner on L. Similarly, for a stick with negative gradient, no MER with top left corner on L is possible (Fig 11).

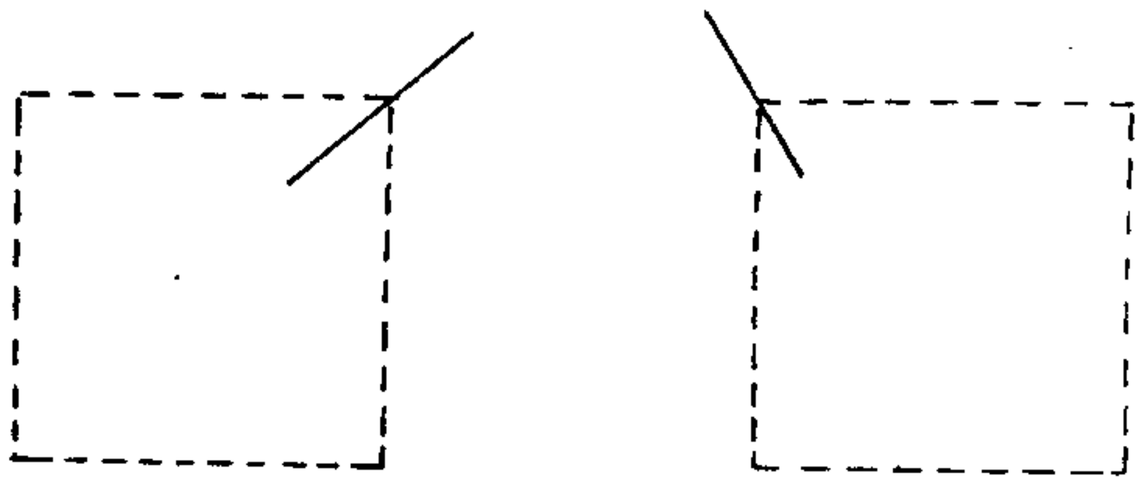


Fig. 11

To capture the visibility information, a visibility list is constructed.

Definition: A visibility list consists of the set of end points of all sticks stored in decreasing order of their y-co-ordinates. The following information is stored with each end-point:

Case 1: The point is the top end point of a stick

Case 1.1: m > 0: Store the end points of the right visible stick.

Case 1.2: m < 0: Store the end points of the left visible stick.

Case 1.3: m = infinity: Store the end points of both left visible and right visible sticks.

This is a consequence of Obs. 5.

Case 2: The point is the bottom end point of a stick

Store the end points of both the right visible and left visible sticks.

No separate action is to be taken for horizontal sticks.

Construction of the visibility list:

For each end point of a stick, the required left and/or right visibile stick can be found by considering each of the other sticks, in O(n) time. Thus the visibility list is constructed in $O(n^2)$ time.

The algorithm uses the line sweep technique [7]. The event points, i.e. the points at which the sweep line stops are the end points of the sticks. At each event point, a search space is generated.

Search Space

A search space is a 2-d region where restricted search for a stick is carried out and is denoted by a 6-tuple [s,t,TR,TL,BL,BR].

TR may be a fixed point or a line segment (a stick or part of it) on which the top right corner of the MER in this 2-d region might lie. In the latter case, TR is called flexible and is suffixed by an asterisk(*). TL is similarly defined for top left corner. BL and BR are also similarly defined for the bottom left

and bottom right corners respectively, except that they can only be flexible.

The symbol "s" denotes the span of search and is given by the distance between two vertical straight lines passing through the bottom end points of TL and TR. The search for a stick is confined within this span.

The symbol "t" denotes the height below which the search is to be carried out.

When the search space is initiated, BL and BR contain null values. As search progresses they may contain non-null values and this will be discussed during a description of the algorithm.

There can be 16 types of search spaces, depending on whether one or both of BL and BR are null or not and whether TL and TR are fixed or flexible.

The 16 types of search spaces are shown below (Fig 12). Types 1 through 4 are the only search spaces possible when both BL and BR are null; types 5 through 8 are the only types possible with BL defined; types 9 through 12 are the only ones that arise when only BR is defined and finally types 13 through 16 come into the picture when both of BL and BR are non-null. Note that BR and BL cannot be fixed.

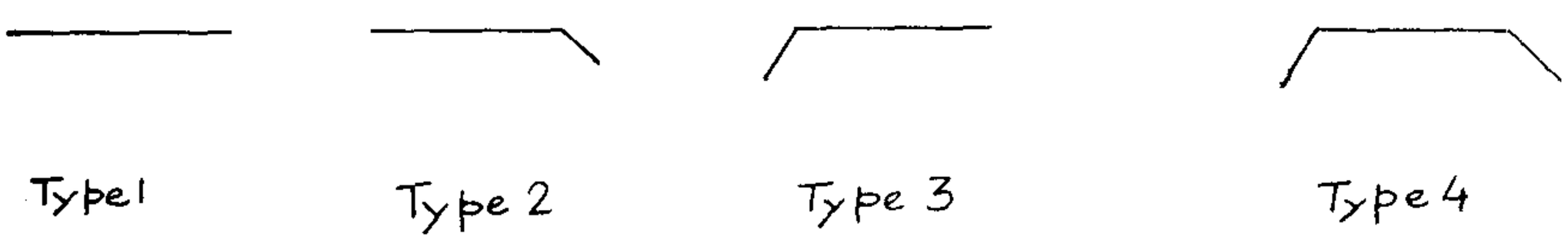


Fig. 12

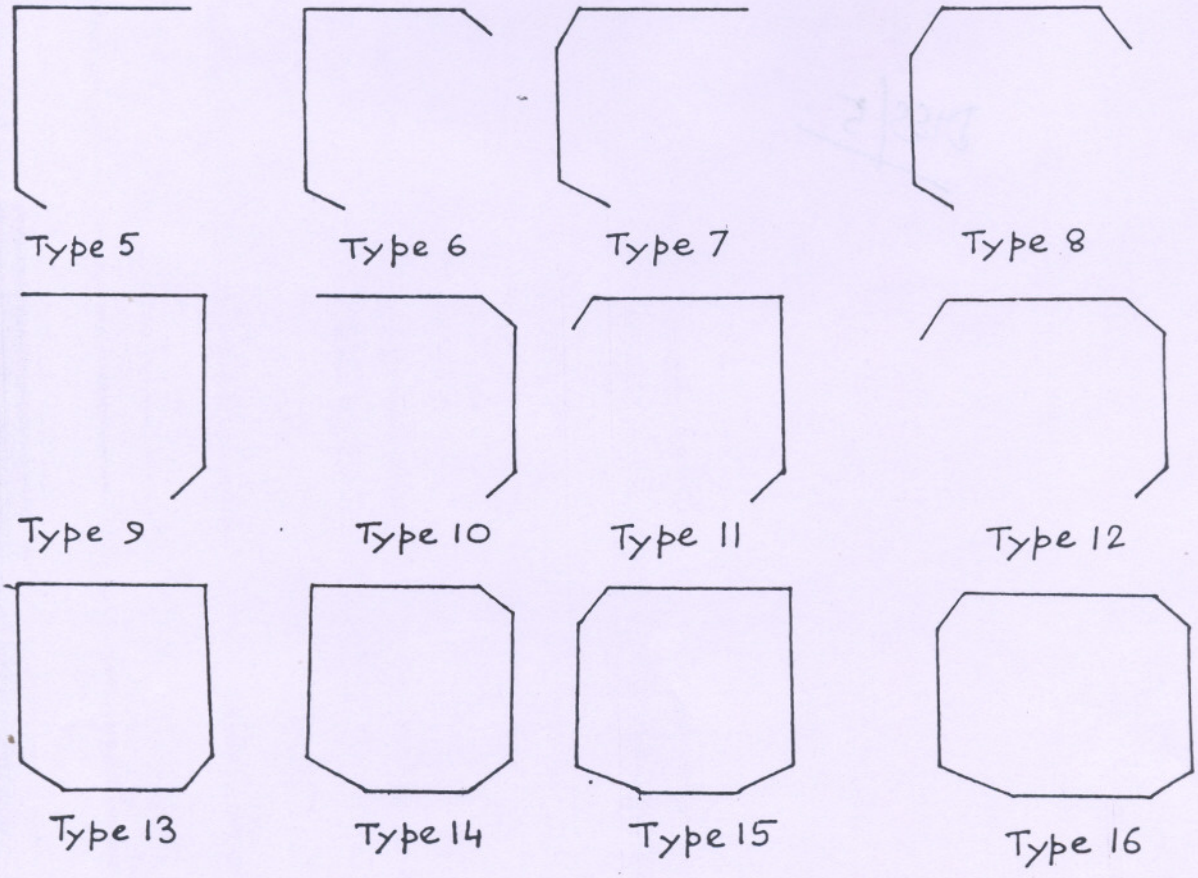


Fig. 12

Determination of an MER from a search space

A set of 8 parameters $C_1, C_2, C_3, C_4, S_1, S_2, S_3, S_4$ is obtained from a search space as discussed in the algorithm. C_1, C_2, C_3, C_4 are the flexible supports at the top right, top left, bottom left and bottom right corners. S_1, S_2, S_3, S_4 are fixed supports at top, left, bottom and right sides of the MER. One or more of these 8 parameters may be undefined for certain search spaces. In this framework, the MER can be found using one of Results 1 through 5 as shown in the table below.

# of flexible sticks defined	Result to be used
1	1
2(adjacent)	2
2(opposite)	3
3	4
4	5

Now, we consider the following Lemma:

Lemma 3: The number of allowable combinations of supporting sticks is finite.

Proof: Fig. 13 shows the set of all possible supports for an MER.

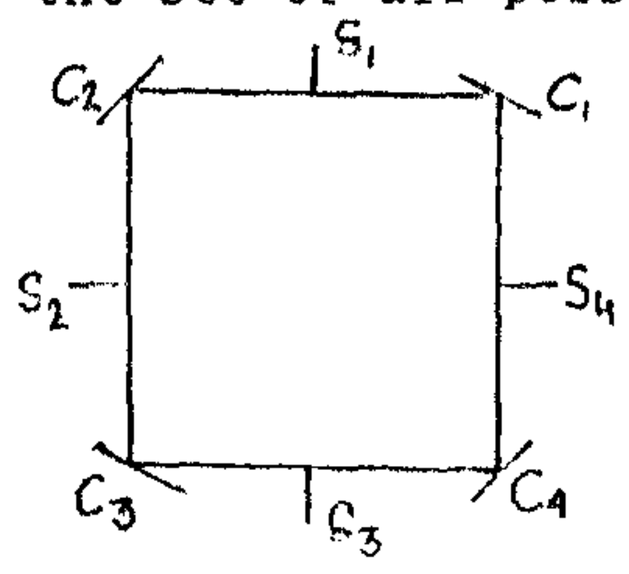


Fig. 13

Since the number of all possible combinations of the supporting sticks is 2^8, and the set of allowable combinations of sticks is a subset of the set of all possible combinations, the number of allowable combination of supporting sticks is finite.

4.ALGORITHM DESCRIPTION

As mentioned earlier, the algorithm is based on the line sweep technique. There are 4 passes, one pass each for a north-south sweep, a south-north sweep, a west-east sweep and an east-west sweep. The previous definitions and concepts have been developed for the north-south sweep and we describe the same. The other sweeps are similar.

The top and bottom end points of all sticks are processed in decreasing order of their y-co-ordinates.

The algorithm consists of two processings for each stick:

1. the one, initiated from the top of a stick is called top processing and
2. the other, initiated from the bottom of a stick is called bottom processing.

Initiation of search_space

Initiation of a search space for top processing of a stick L_1 with slope greater than 0, is discussed here. An analogous situation arises if the slope of the stick is less than 0. If the stick is vertical, two search spaces are initiated in succession, one for the right visible stick and another for the left visible stick.

The visibility list contains information pertaining to the right visible stick L_2 for the top end point p of L_1. An x-axis parallel straight line is drawn from p to the right which

intersects L_2 at point q. A search space is initiated with TL = L_1, TR = the part of L_2 below point q, s = $[c_1,c_2]$, t = b_1, BL = 0, BR = 0.

Of the parameters to be obtained from this search space,two are assigned values at this stage, $C_1 = L_2$; $C_2 = L_1$.

For formation of a search space for bottom processing of a stick L, the right and the left visible sticks for the bottom end point p of L, say L_1 and L_2, are obtained from the visibility list. Suppose the horizontal straight line drawn through p intersects L_1 and L_2 at points q and r respectively.For the search space initiated in this case, TL = the part of L_1 below the point q; TR = the part of L_2 below the point r, s = $[c_1,c_2]$, t = d, BL = 0, BR = 0.

Also, at this stage, $C_1 = L_2$; $C_2 = L_1$; $S_1 = L$.

The point p is called the reference point of the search space and is preserved. The significance of this point will be discussed later .

Finding the sticks that intersect the search space

With the search space initiated by top or bottom processing, an appropriate data structure, discussed later, is searched to find a stick L_3 that intersects the span s of the search space at the maximum y-coordinate below the height t.

The position of L_3 can lead to three situations as shown in Fig 14.

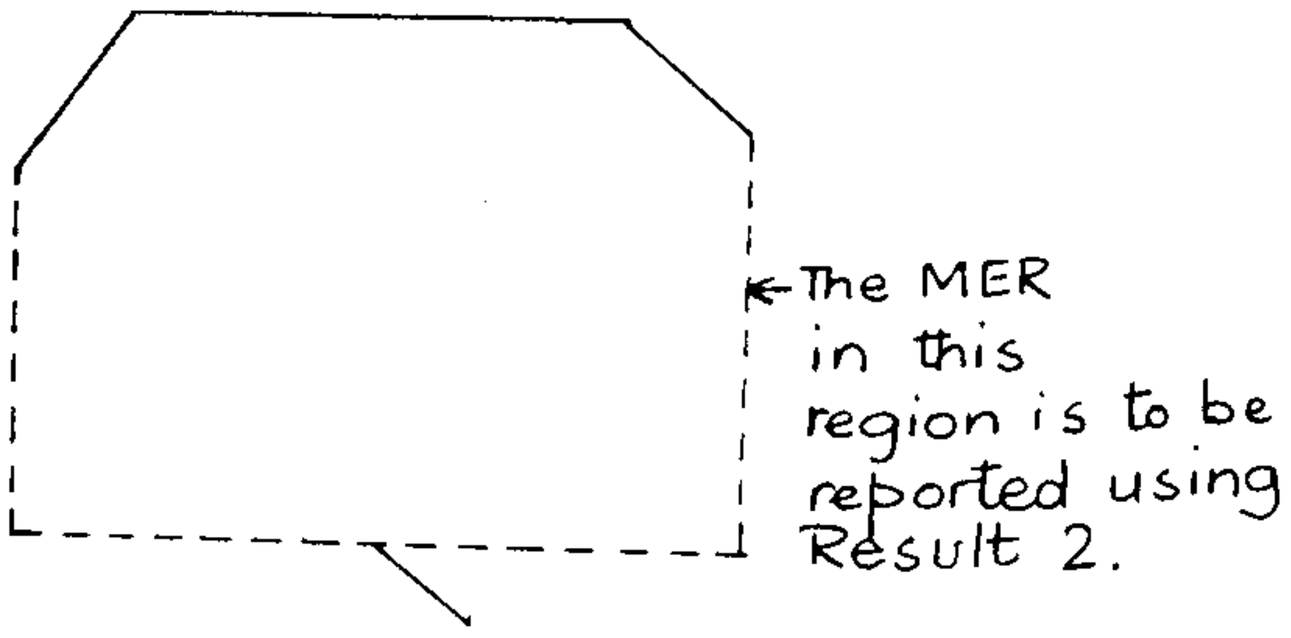


Fig. 14a

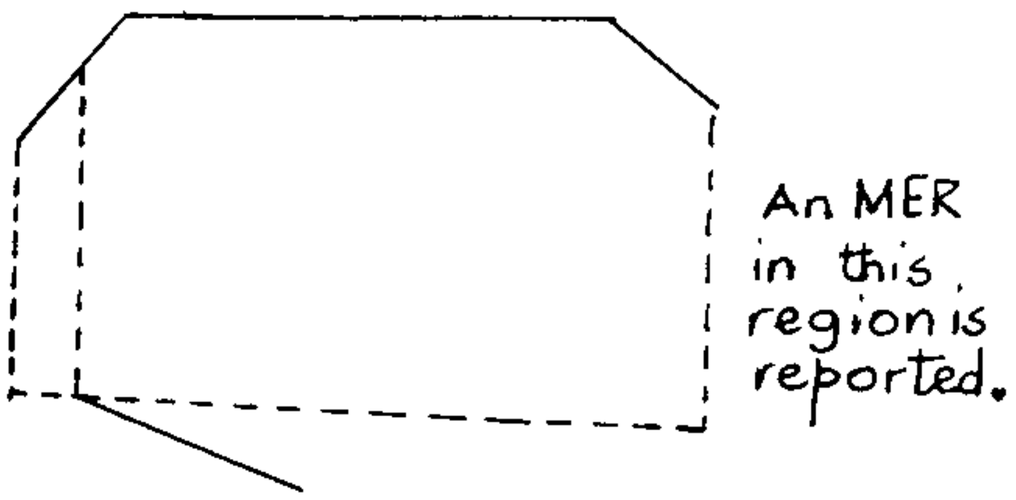


Fig. 14b

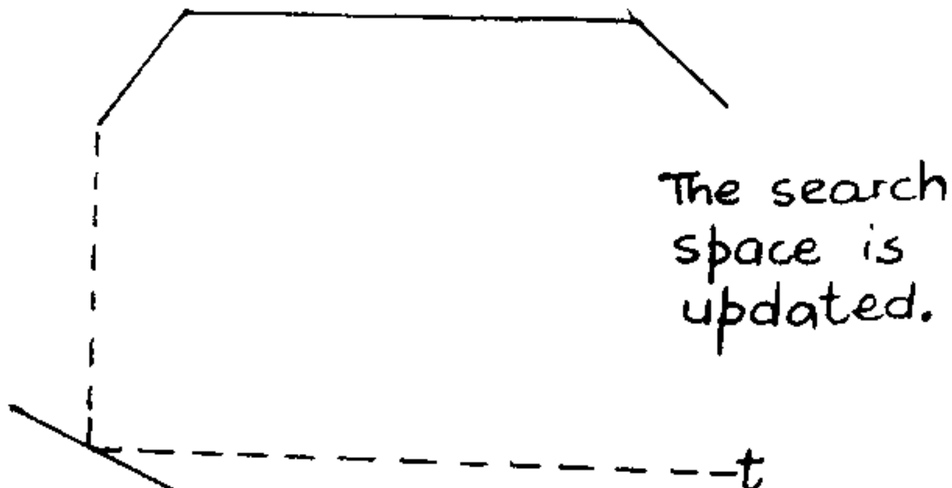


Fig. 14c

In the situation shown in Fig14(a), the parameter S_3 is set to L_3 and the MER is found using Result 2.

In the second situation (Fig 14b), TL is updated so that that the x-coordinate of the bottom end point of TL and the x-coordinate of the top end point of BL are the same. BL and C_3 are both set equal to L_3. s is changed to the new span, t is set to the y-co-ordinate of the top end point of BL. An MER is also found by Result 3.

In the third situation (Fig 14c), BL is set equal to the part of L_3 lying to the right of the vertical line drawn through the bottom end point of L_1. C_3 is set to L_3. t is set to the y-co-ordinate of the top end point of BL.

The search space now is of the type shown in Fig 15.

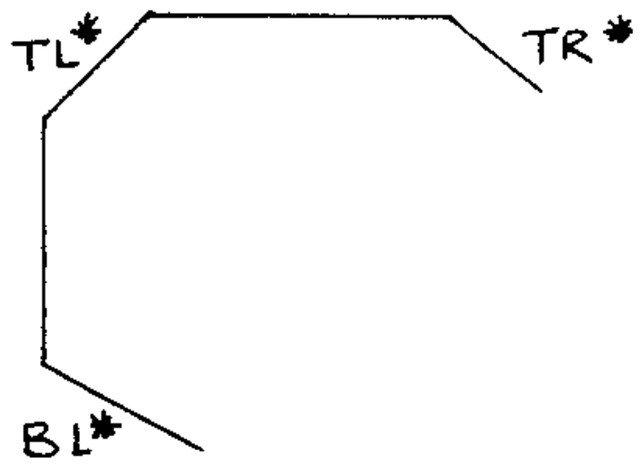


Fig. 15

In the new search space, the search is for a stick L_4. This might lead to four cases as shown in Fig 16.

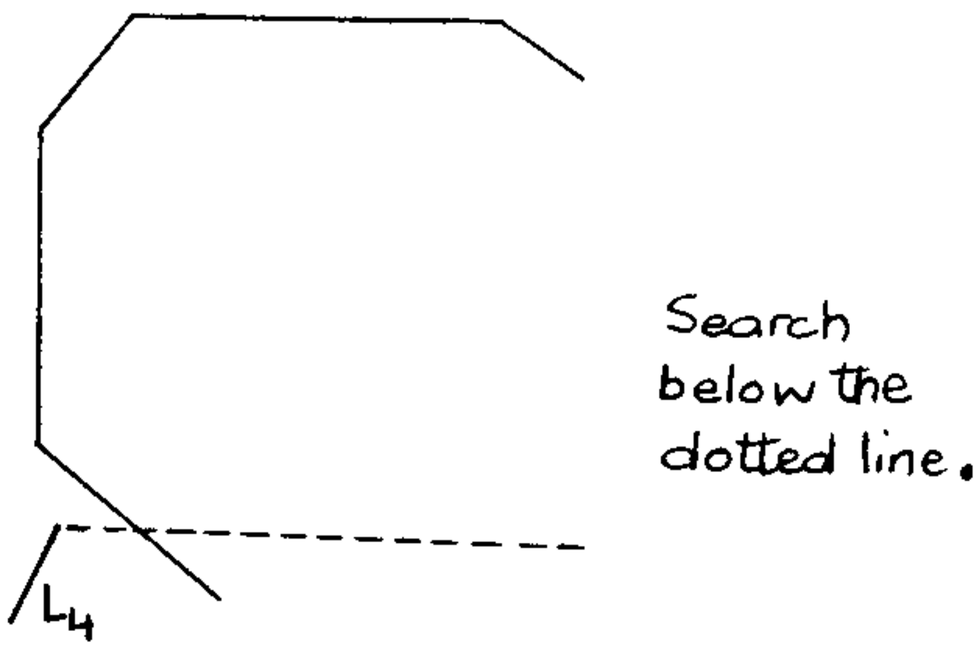


Fig. 16a

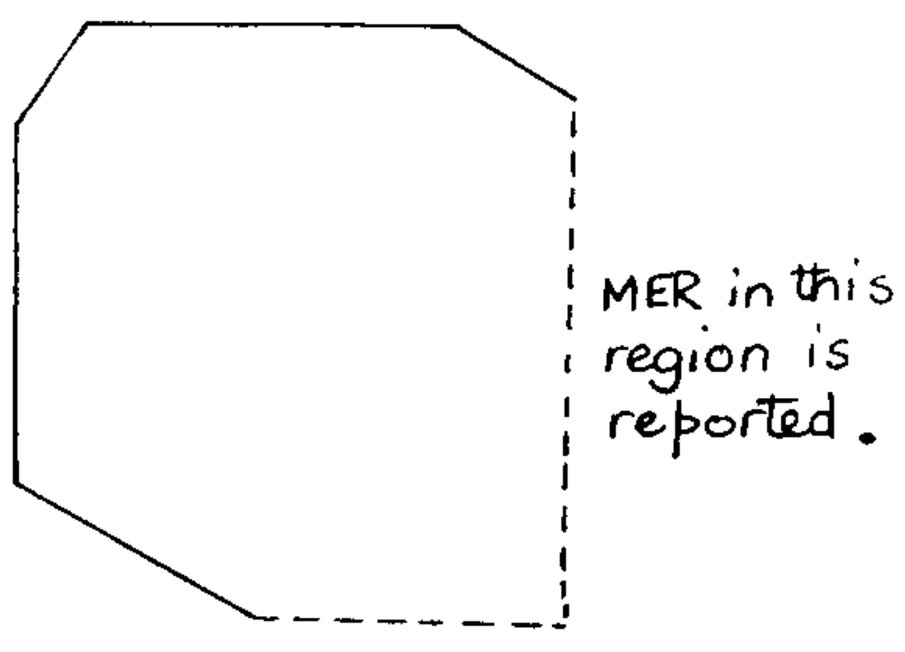


Fig. 16b

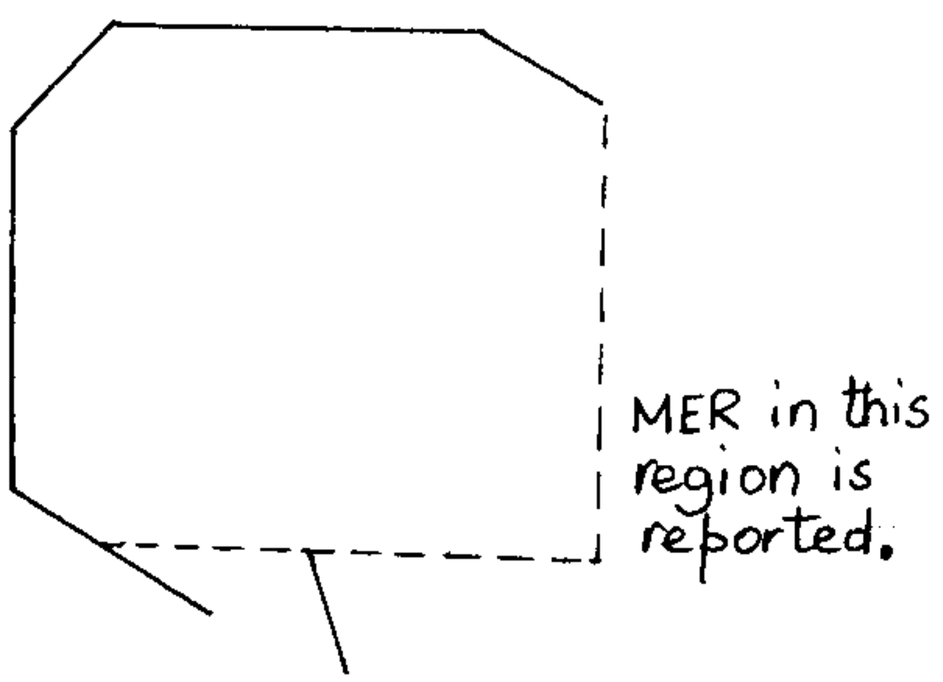


Fig. 16c

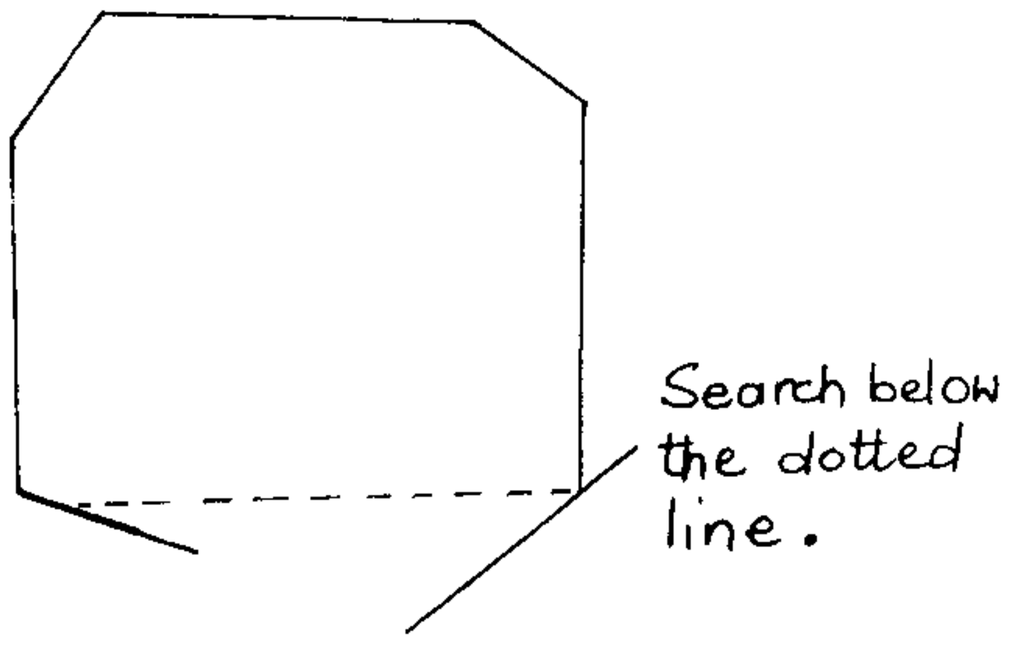


Fig. 16d

In the first situation shown, as L_4 lies to the left of BL, another stick is searched in the same search space which might lie to the right of BL.

In the second situation, as only C_1, C_2 and C_3 are defined, the MER is found using Result 4.

In the third situation, S_3 is set to L_4 and the MER is found using Result 4.

In the fourth situation, BR will be defined as a segment of L_4 and TR and BL may have to be updated such that the x-co-ordinate of the bottom end point of TR and that of top end point of BR are equal to the minimum of the two and the y co-ordinates of the bottom end points of BL and BR are equal to maximum of the two. t is set to the y-co-ordinate of the top end point of BR. If TR is updated, then s is changed to the new span. The situation is depicted in Fig 17.

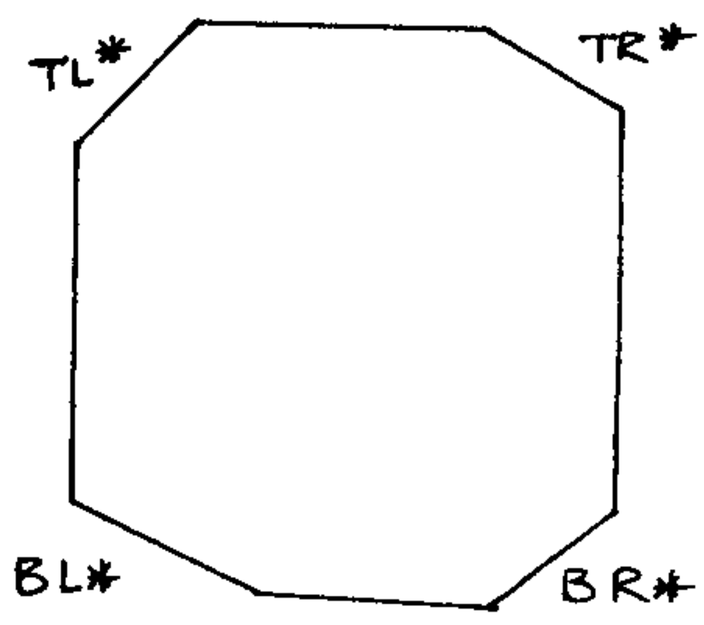


Fig. 17

The search for a stick L_5 with the new search space can lead to one of the four situations as shown in Fig. 18.

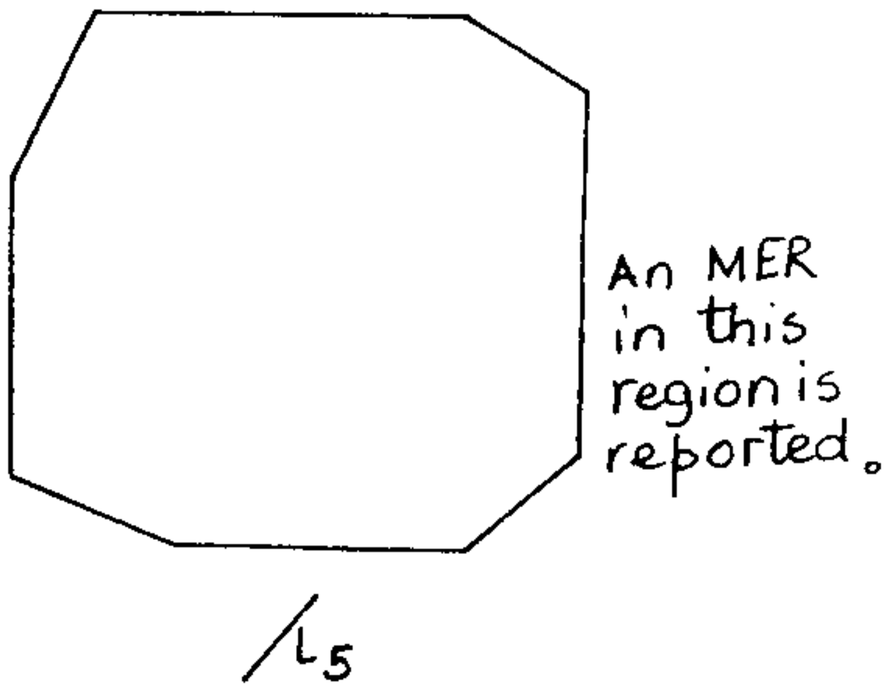


Fig. 18a

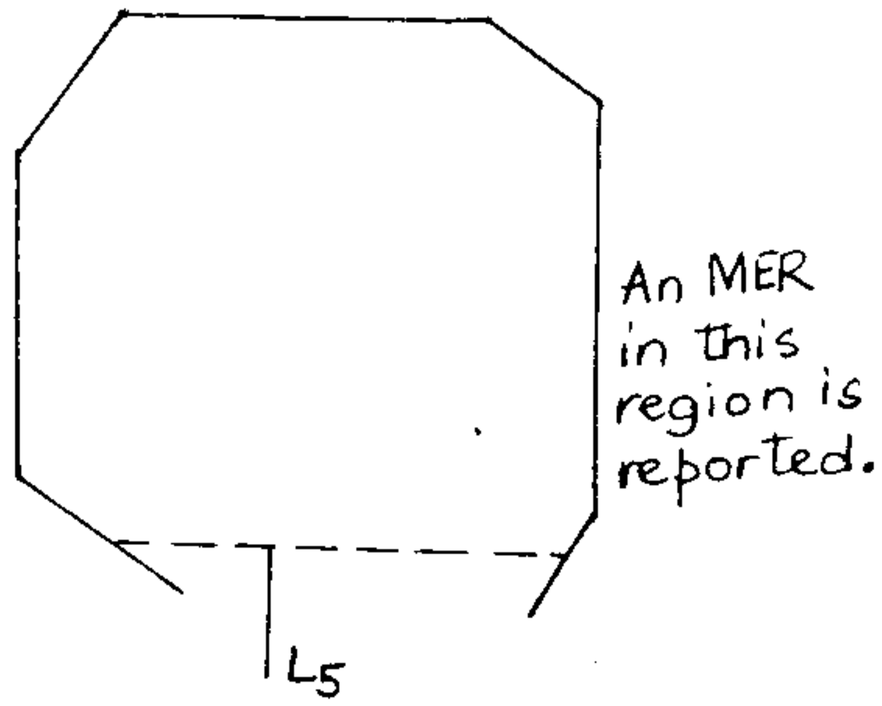


Fig. 18b

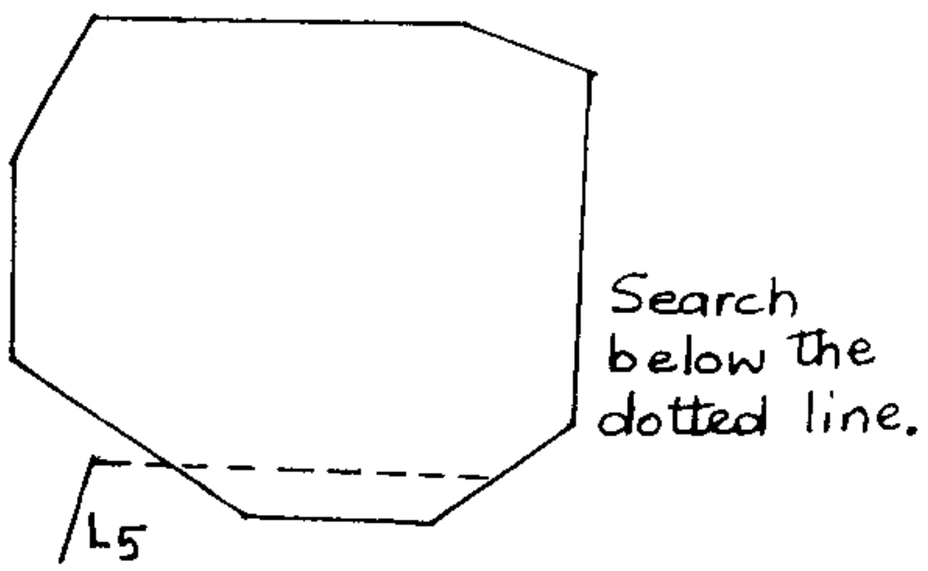


Fig. 18c

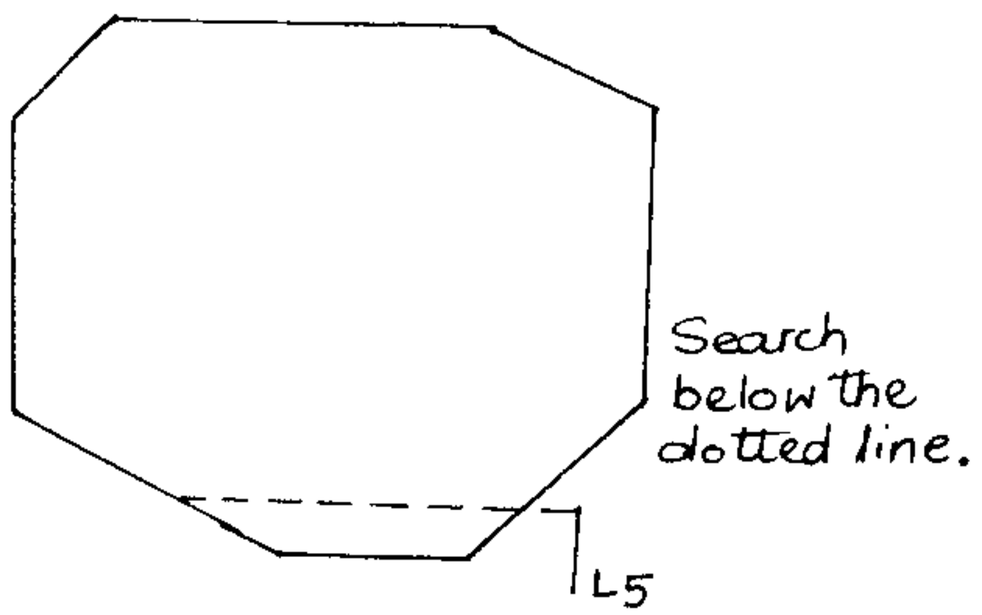


Fig. 18d

In the first two cases, C_1, C_2, C_3, C_4 are defined and the resulting MER can be found by Result 4. In the third case, since L_5 lies to the right of BR, with the same search space, continue search for a stick which might lie to the left of BR. In the fourth case, a similar action is taken.

There may be several MER's whose top is constrained by the same fixed support(i.e. their top edges are at the same height) but whose other supports vary. Fig 19 shows a set of such MER's.

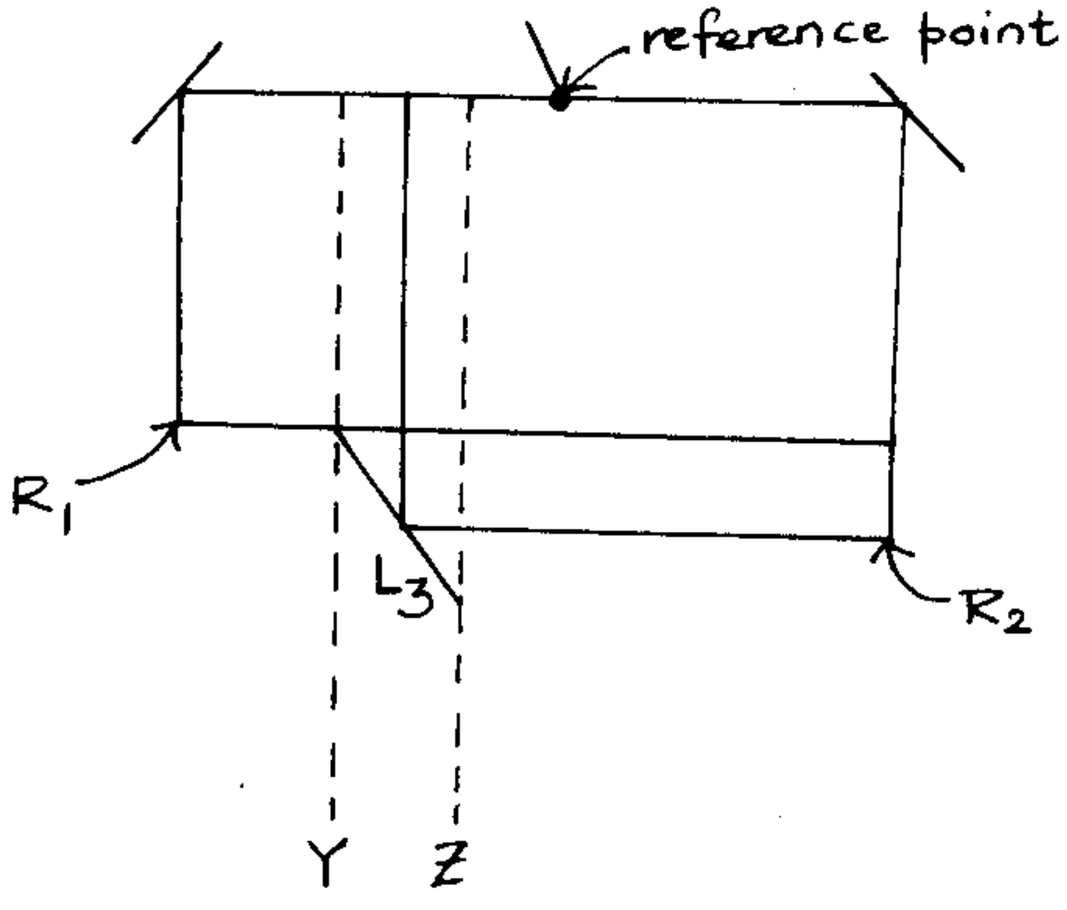


Fig. 19

Consider the situation depicted in Fig. 19. Draw a vertical line Y passing through the top end point of L_3. Once MER R_1 has been generated, no other MER can be formed which has L as a fixed support and part of which lies to the left of Y. The search space is thus split into two parts, the one to the left of Y is rejected, the one to the right of Y touches the reference point and should be further examined. A new search space is formed which descends from the same height but which reflects this change. At this point, the set of 8 parameters are accordingly adjusted e.g. in this case C_2 is set to null and C_3 is set to L_3. For a similar reason, once R_2 has been reported the search space is split along Z passing through the bottom end point of L_3 and the parameters are suitably adjusted.

Such reduction of search space is called splitting.

Search space initiation, splitting and MER generation will be discussed in details in the appendix.

Suppose bottom processing of stick L_1 is to be done. Draw a vertical straight line X from the bottom end point of L_1 to the bottom boundary of the floor.

Observation 6: If a stick L_1 intersects X and all search spaces with L_1 and touching the reference point have been generated, then any stick not yet considered cannot form a search space with the reference point.

Clear from Fig. 20.

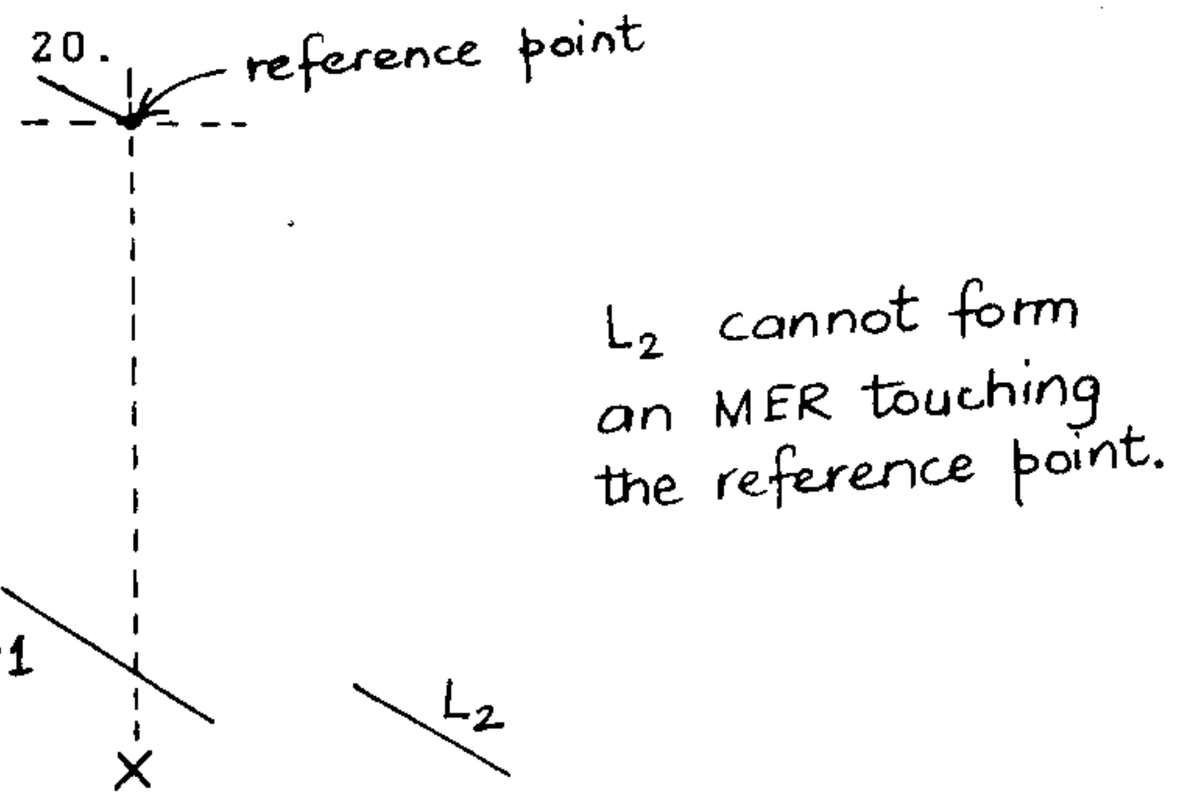


Fig. 20

Now, we prove the following lemma.

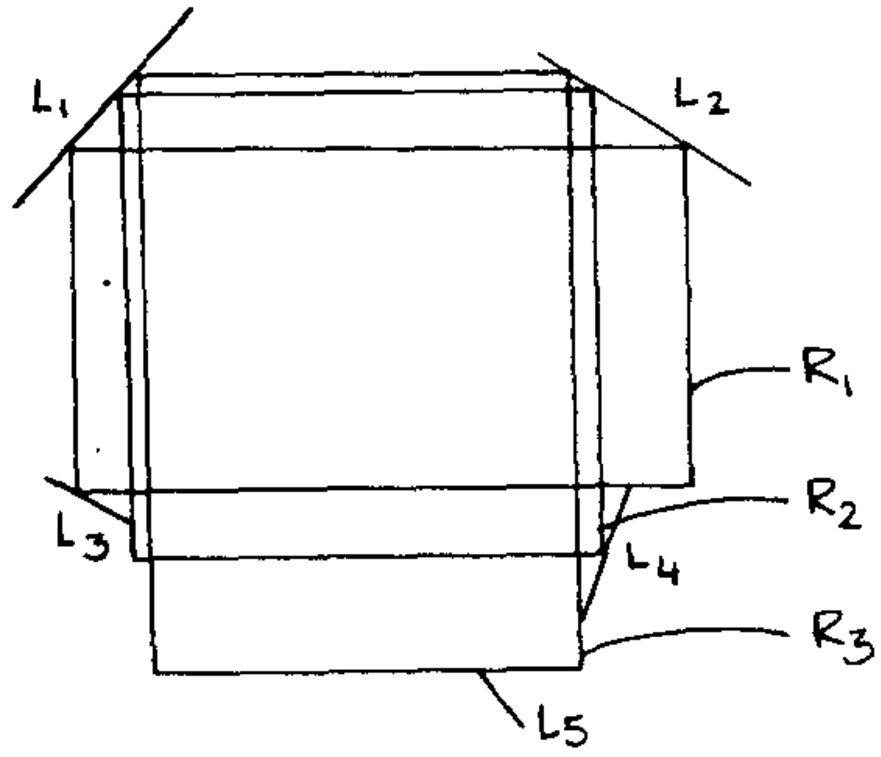
Lemma 4: A search space can be split at most O(n) times.

Proof: To prove this lemma, the following facts are considered.

1. Any line which lies outside the span or above the height of the search space presently considered, will never intercept a search space formed from the present one, in future, by splitting.

2. During the splitting process, a split may occur at the top end point of a stick and another split may occur at the bottom end point of the same stick, generating different search spaces.

Since there are n sticks, there can be atmost 2n splittings and hence the number of times the a search space can be split is O(n).



R_2 & R_3 is supported by at least one fixed support, R_1 is supported only by flexible supports.

Fig. 21

Top processing finds out MER's whose top left and top right corners lie on flexible supports. There may be a class of MER's whose top left and top right corners lie on the same flexible supports (see Fig 21). But of this class, all but one should have at least one fixed support and these MER's can be detected during bottom processing of this fixed support in one of the four sweeps. Therefore,

Observation 7: Provided that splitting is done during bottom processing, it is not required during top processing.

Observation 8: MER's which have more than one fixed support, will be detected during bottom processing of all these fixed supports in the corresponding sweeps. Hence, an MER may be reported more than once.

Now, we prove the following theorem:

Theorem: All MER's are reported by the MER recognition algorithm.

Proof: Suppose there exists an MER M which has not been reported. If one side of M touches an end point of some stick then M must have been reported during bottom processing of that stick in one of the passes.

If M has been defined with 3 or more corners on flexible supports, then it has been reported during top processing of one of these supports. Otherwise, the implication is that there is a stick L, as shown in Fig. 22, in which case M is not an MER.

Hence, the theorem is proved by contradiction.

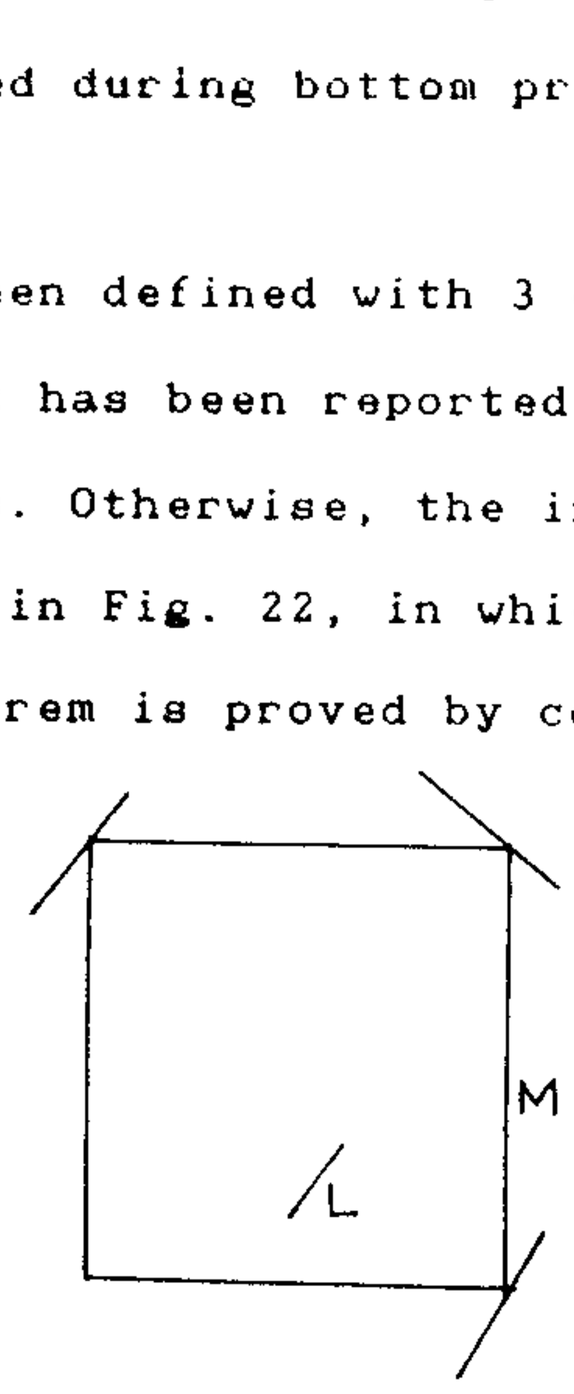


Fig. 22

5. RECOGNITION OF THE TOPMOST STICK WITHIN THE SEARCH SPACE

The MER generation method is based on recognising the stick which intersects the span of the search space at a maximum height. In this section we introduce a data structure and an algorithm based on this data structure which efficiently determines such a stick. The complexity of this search with respect to a given search space is also mentioned.

Data structure :Line search tree

The set of x-coordinates of the end points of the sticks is partitioned into two equal parts by a suitable median finding algorithm. Suppose the partitioning is along a vertical line Y whose equation is X = x.

The line Y divides the set of lines into three parts V_1,V_2 and V_3.(See Fig 23).The set V_1 (resp. V_3) contains the lines both of whose end points lie to the left(resp. right) of Y.The set V_2 contains the remaining lines which have the left end point to the left of Y and the right end point to the right of Y.

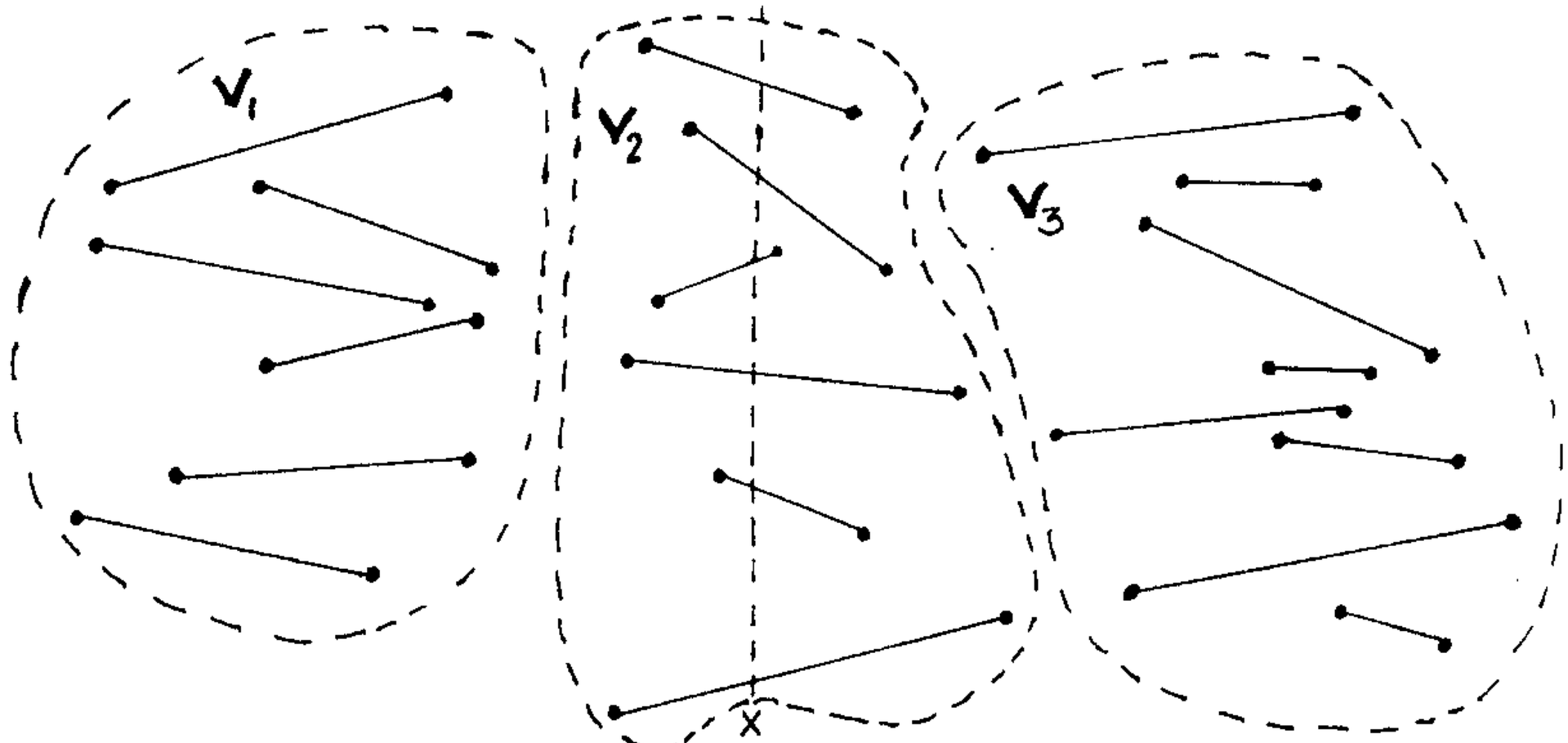


Fig. 23

The root of the line search tree may be empty(for an empty point set) or may contain the following fields:

(i) discriminant value x

(ii) L_TREE: a pointer to an associated structure for the left end points of sticks in V_2.

(iii) R_TREE: a pointer to an associated structure for the right end points of sticks in V_2.

(iv) L_PTR: a pointer to the root of the line search tree for the set V_1.

(v) R_PTR: a pointer to the root of the line search tree for the set V_3.

Observation 9: Since the number of end points is even,

$$|V_1| = |V_3|.$$

So the tree is weight balanced and the length of the path from the root to a leaf can be O(log n) in the worst case. The worst case situation is attained if each partition line intersects atmost one stick along the path.

Associated structure of a node

As already discussed, each node of the line-search tree contains two associated structures, L-TREE and R-TREE. We shall discuss the structure of R_TREE of a node. L-TREE can be similarly constructed.

Construct a balanced search tree with the x-coordinates of the right end points of all sticks in V_2. Each node (v) of this tree contains the following information.

(i) Discriminant : The x value which guides the search path;

(ii) STICK : The stick, the x-coordinate of whose right end point is x;

(iii) L-LINK : Pointer to the root of the left subtree;

(iv) R-LINK : Pointer to the root of the right subtree;

(v) Y-TREE : A balanced binary tree on the y coordinates of the right end points of the sticks that are stored in the tree rooted at node x.

(vi) INT-TREE : Consider a vertical line L_x at the point x. The sticks in V_2 whose endpoints are in the right subtree of node v cut L_x. With the ordinates of these points of intersection, this binary tree is constructed.

The significance of Y-TREE and INT-TREE will be discussed when we shall explain the search strategy.

Search strategy

Given a search space with span s = [a1,a2] and t = b, the method of finding the stick which hits the search space first is described below.

Initially, associate two more boolean fields [lb,rb] with the search space. The field lb (resp rb) = 1 indicates that the left (resp right) bound of the span of the search space lies along a partition line of the line-search tree. Otherwise it contains the value 0(see Fig. 24).

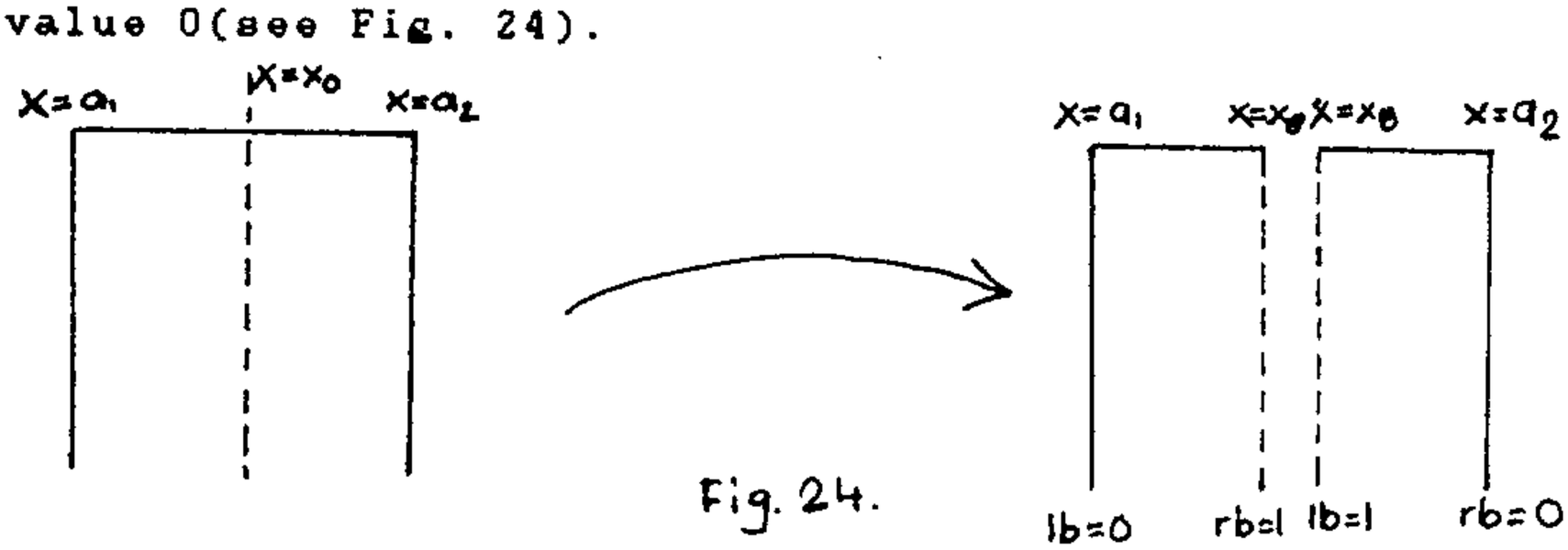


Fig. 24.

Two global locations (1)Max_l [(a,b)(c,d)] and (2) MAX are maintained to retain information about the stick which has hit the search space at the highest point till the present. Max_l contains the end points of such a stick and MAX the y-coordinate of highest point on the stick which lies inside the span of the search space.

The search is initiated with the horizontal span $[a_1, a_2]$ at a height b from the root of the line search tree.

Search in the main tree

At the time of processing a node v of the line search tree, the following situations may arise.

Case 1: The entire horizontal span lies to the right of the discriminant value x of v. In this case the right associated structure and the right subtree of v are to be processed. The need to process the right subtree is obvious. The right associated structure is processed due to the fact that a stick in the set V_2 corresponding to the partition line of node v may hit the span of the search space first. (See Fig 25).

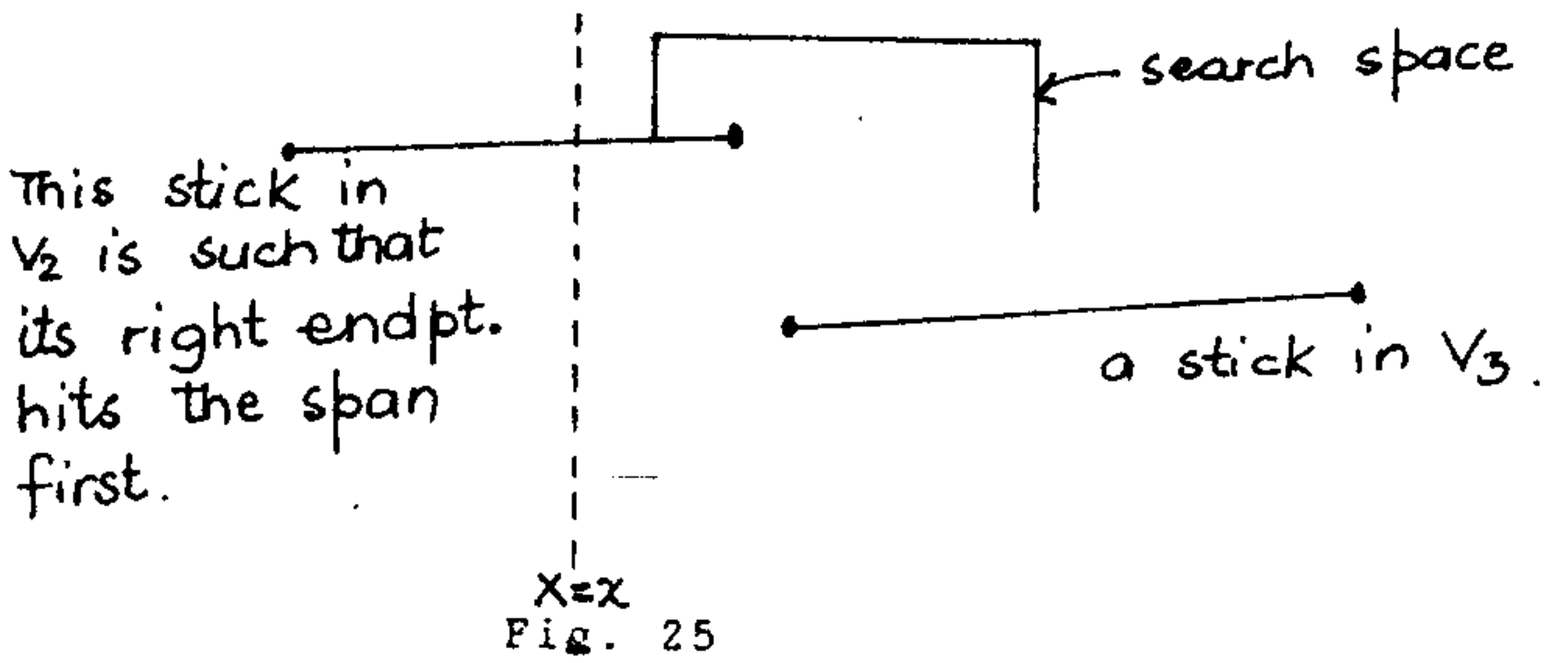


Fig. 25

Case 2:The entire horizontal span $[a_1,a_2]$ lies to the left to x.In this case the left subtree and the left associated structure of node v are to be considered for the search.

Case 3:The discriminant x splits the span into two parts. Here, two situations may arise.

Case 3.1: Both lb and rb associated with the span contain 1 which implies that the span is bounded to the left and right. In this case, all the sticks stored in the line-search tree rooted at this node and lying entirely below the height of the search space (b) are to be considered. Of these, the one whose top end point is at a maximum height will hit the span first and can be found from the Y-TREE corresponding to the root node of both the associated structures of the node v. Let Max-y be the y-coordinate of the top end point of the stick which hits the span first. Then Max-y is compared with the global location MAX and if it is found to be greater,it replaces the old value in MAX. Also,the stick associated with it is stored in the location Max_L.

Case 3.2: The horizontal span is unbounded to the left or to the right or both. In this situation, split the span into two parts.

(a) $[a_1,x]$,the lb value of this span is the same as that of $[a_1,a_2]$ and the rb value is 1.

(b) $[x,a_2]$, the rb value of this span is the same as that of $[a_1,a_2]$ and the lb value is 1.

Search the left associated structure and the left subtree of node v with the span $[a_1,x]$. Search the right associated structure and the right subtree of node v with the span $[x,a_2]$.

Search in the associated structure of a node

Let $[a_1,a_2]$ be the horizontal span initiated at height b which falls to the right of the discriminant value x corresponding to node v of the line-search tree. It is observed from Fig 25 that the sticks corresponding to the points in the right associated structure may hit the window first.

Consider the right associated structure of node v. Trace down the search tree to find a node whose discriminant lies in the span $[a_1,a_2]$.

Case 1: There is no such node (i.e. all the sticks whose right end points are in the tree have right end points to the left of a_1), then ignore node v and proceed searching in the line search tree.

Case 2: Let u be the node found.Then, all nodes which lie on the path from the root of the associated structure to node u are to be considered. Let w be such a node. If the search path goes to the right from node w, then ignore the left

subtree of node w. If the search path goes to the left from node w, then the sticks which cross the vertical line X = x, where x is the discriminant of node w, and intersect the span $[a_1, a_2]$ below height b are to be considered. Of these, the one that intersects the span at a maximum height is to found. This is done by searching the INT-TREE corresponding to node w. The highest point of intersection of the span with the stick found, is compared with the global location MAX to decide whether the stick is to be retained or not. The point of intersection of the stick (STICK) corresponding to the node u with the span $[a_1, a_2]$ is also computed to check whether this stick hits the span at a maximum height.

From node u, two search paths, L-path and R-path to the left end point of the span (a_1) and right end point of the span (a_2) are considered. Consider a node v^* lying on L-path. If the search path from v^* moves to the right, ignore the left subtree rooted at v^*. If the search path moves to the left then consider the right end point of STICK corresponding to v^* and check whether it hits the span first. Also, all the sticks whose top end point is below b and whose right end point is stored in the right subtree of v^*, are considered and the one whose right end point has the maximum y-coordinate is selected by searching Y-TREE of the root of the right subtree. The y-

coordinate of the right end point of the selected stick is compared with the location MAX, and the stick is retained, if necessary.

For v^* lying on the R-path, recall that searching the right associated structure is being described. If the search path goes to the right, the STICK of node v^* and Y_TREE of the root of the left subtree of v^* is to be searched. If the search path goes to the left, then the STICK and the INT_TREE associated with node v^* are to be considered.

6. ALGORITHM MER

Input: A set of arbitrarily oriented, non-intersecting line segments in a rectangular floor.

Output: The maximum empty isothetic rectangular area in this floor.

Algorithm: The algorithm consists of four identical sweeps -- a north-south sweep, one from east to west, another from west to east and finally a south to north sweep. For each sweep the following two data structures are constructed:

(i) The visibility list.

(ii) The line search tree.

1. North-South sweep

For each point in the visibility list do

(i) Determine the visible sticks from this point from the visibility list.

(ii) Initiate a search space with the sticks so found.

(iii) Search the line search tree for a stick that intercepts the search space at a maximum height. If BL & BR is null, go to step (iv).When BL and/or BR is non-null, it might so happen that the highest point of a stick within the span does not intercept the interval, then go to step (iv). Else go to (iii).

(iv) Report the MER using one of Results 1 through 5.

(v) If the end point is a bottom end point and a new search space can be formed by splitting from the current one which touches the reference point then form the new search space and go to (iii)

endfor.

2. East-West Sweep.

3. West-East Sweep.

4. South-North Sweep.

5. Of the MER's so reported, find out the maximum.

7. COMPLEXITY ANALYSIS

Time Complexity

Consider a stick L. For the top end point of L, one search space is generated and is not split. At the time of processing its bottom end point, a search space is generated and it is split by the sticks that hit it. Thus, by lemma 4, the total number of search spaces generated from the end point of L can be atmost O(n). This implies that the total number of search spaces that are to be processed for a stick is O(n) in worst case. The stick that intersects the span of a search space at a maximum height is obtained from the line-search tree. Thus the complexity of the algorithm depends on the search time of the line-search tree for each search space.

Our algorithm suggests that for each search space at most O(logn) nodes of the line-serach tree are to be checked. For each node (v) in the path of traversal, the associated structure(s) is(are) to be visited. An associated structure is also a balanced search tree and its height is $O(\log k_v)$ in worst case where k_v is the number of sticks in the set V_2 corresponding to the partition line of node v of the line-search tree. Thus the number of nodes visited in this associated structure is $O(\log k_v)$. In each node of the associated structure, the stick which hits the search space at a maximum height can be obtained from either INT-TREE or Y-TREE by $O(\log k_v)$

comparisons. Thus the worst case total time complexity of processing a single search space is $O(\log^3 n)$ [since $k_v < n$]. As we have mentioned earlier, the total number of search spaces to be processed can be $O(n^2)$ in worst case. So, the total time complexity of the algorithm is $O(n^2 \log^3 n)$.

Space Complexity

Regarding space complexity, first of all, it is to be noted that a stick is present in exactly one node of the line-search tree. Suppose the node v of the line search tree is associated with k_v sticks. Thus each of the two associated structures, one to the left and one to the right, will have k_v nodes. Again each of the k_v sticks is present in $O(\log k_v)$ nodes of the associated structure. This indicates that the total size of the associated structures corresponding to node v is $O(k_v \log k_v)$. Thus the total space required for this data structure is $\sum_{v=1}^{p}(k_v \log k_v)$, where p is the total number of nodes in the line-search tree. It is easy to see that this space complexity is much less than $O(n\log n)$. Thus the worst case space complexity of our algorithm is $O(n\log n)$.

8. CONCLUSION

The problem of this dissertation was to find a largest empty isothetic rectangular region within a set of randomly distributed line segment obstacles of arbitrary orientation. Introducing the concept of pseudo-maximal rectangles, we restrict our search to these rectangles.We have not been able to give an upper bound for the number of such rectangles.The time complexity of the algorithm developed is $O(n^2 \log^3 n)$ and the space complexity is $O(n \log n)$.The problem has several applications in VLSI layout design and operations reearch.

APPENDIX

Here, we will consider three representative cases and show MER generation and search space formation by splitting in each case.

Let us assume that the reference point is given by (c,d).
Also that the perpendicular line from (c,d) to the bottom boundary is denoted by X.

A split search space is generated only during bottom processing. The initial MER generation being same for both top and bottom processing, we will assume that bottom processing is being done and show the formation of newer search spaces, always working under the assumption that the last encountered stick does not intersect X.

1: Processing search spaces of types 1 through 4

Only the processing of type 4 search space will be described. Processing of the other types are similar.

Processing a type 4 search space

Suppose the type 4 search space is given by [s=[c1,c2],t=b,TR*= [(a2,b2)(c2,d2)],TL*=[(a1,b1)(c1,d1)],0,0] where b1 = b2.C1 = L2; C2 = L1. S1 may or may not be defined, depending on whether the search space is initiated during bottom processing or top processing.

Let L3[(a3,b3)(c3,d3)] be the new stick intercepted within this span (Fig 1.1).

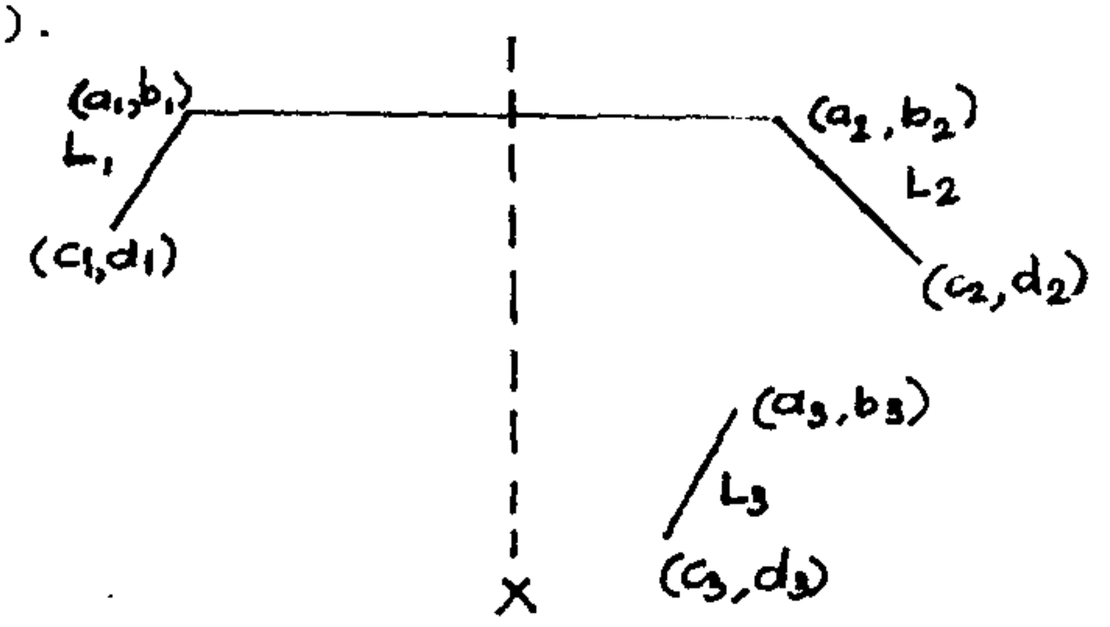


Fig 1.1

Case 1:L3 lies to the right of X.

Case 1.1:m3 > 0.

Case 1.1.1:a3 <= a2 (Fig 1.2).

S3 = L3. As C1 and C2 are the only flexible sticks defined, an MER is obtained using Result 2.

Generate the type 11 search space

[s=(c1,a3),t=b3,TR=(a3,b1),TL*,0,BR*=[(a3,b3)(c3,d3)]].

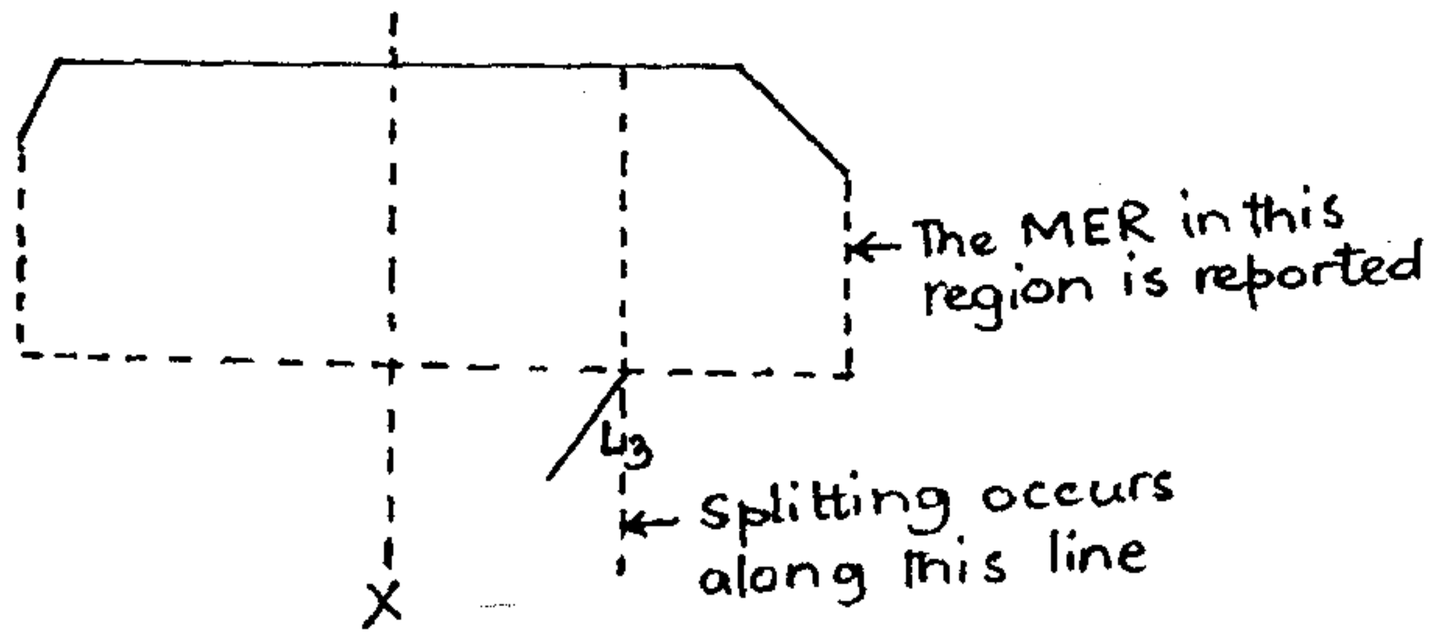


Fig 1.2

Case 1.1.2:a2 < a3 <= c2 (Fig 1.3).

Set S3 = L3 and generate an MER by Result 2.

Generate the type 12 search space

[s=(c1,a3),t=b3,TR*=[(a2,b1)(a3,a3.m3+k3)],TL*,0,

BR*=[(a3,b3)(c3,d3)].

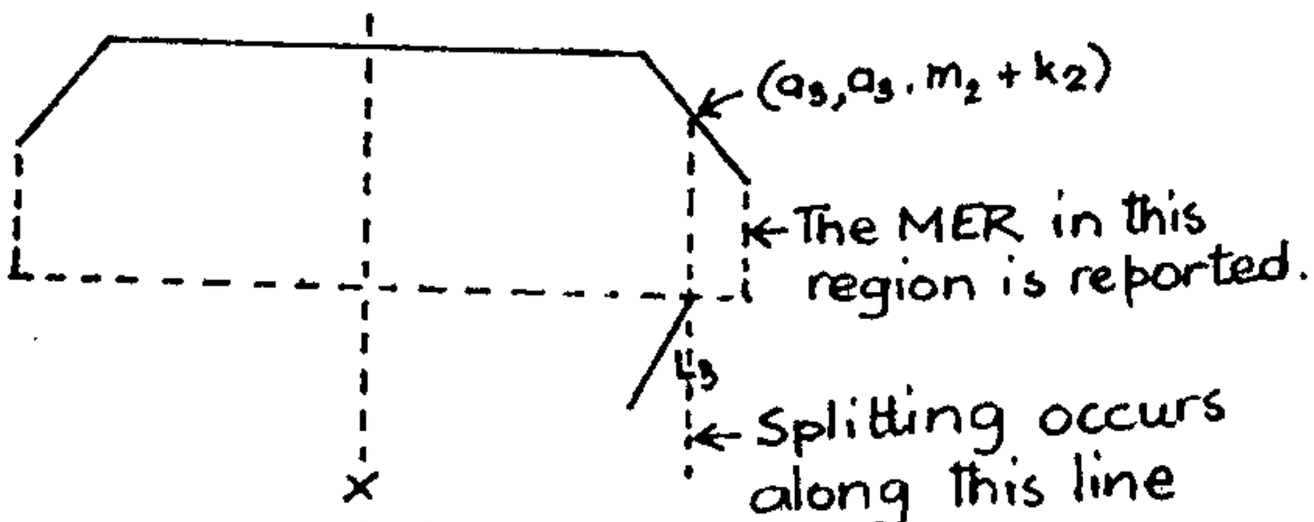


Fig 1.3

Case 1.1.3:a3 > c2.(Fig 1.4)

Generate the type 12 search space [s,t,TR*,TL*,0,

BR*=[(c2,c2.m3+k3)(c3,d3)].

In this case, no MER is reported.

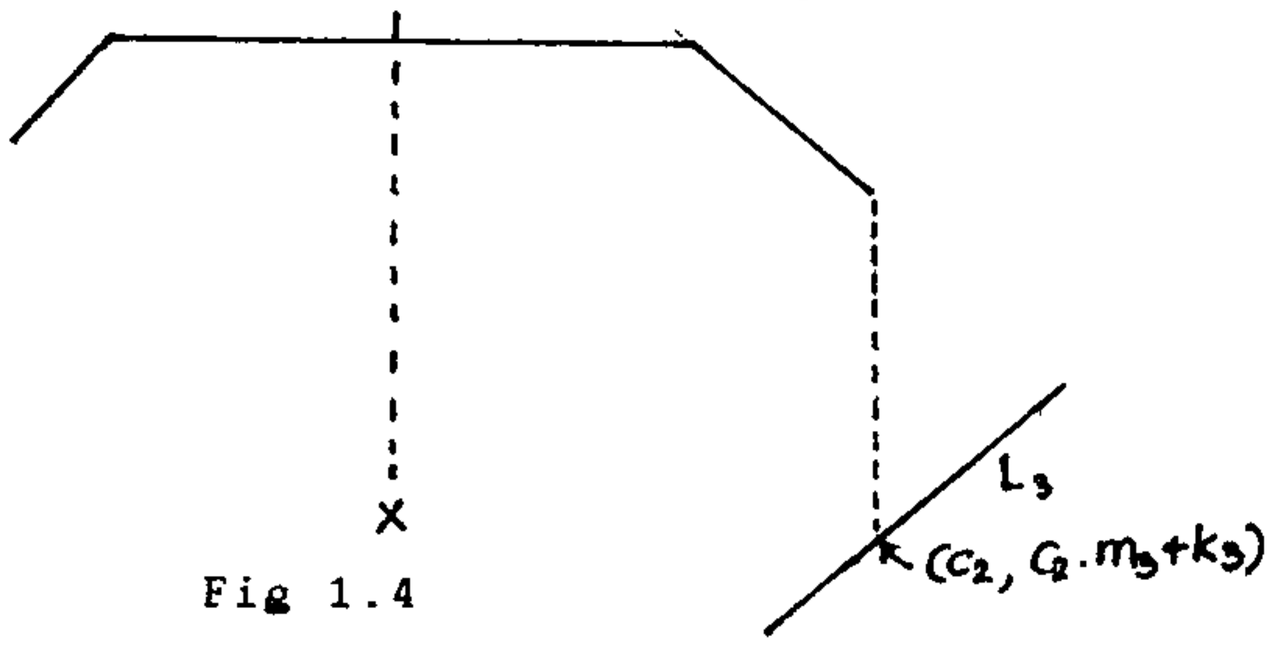


Fig 1.4

Case 1.2: m3 < 0.

Case 1.2.1: a3 <= a2 (Fig 1.5)

S3 = L3. Determine MER using Result 2.

Generate the type 3 search space

[s=(c1,a3),t,TR=(a3,b),TL*,0,0].

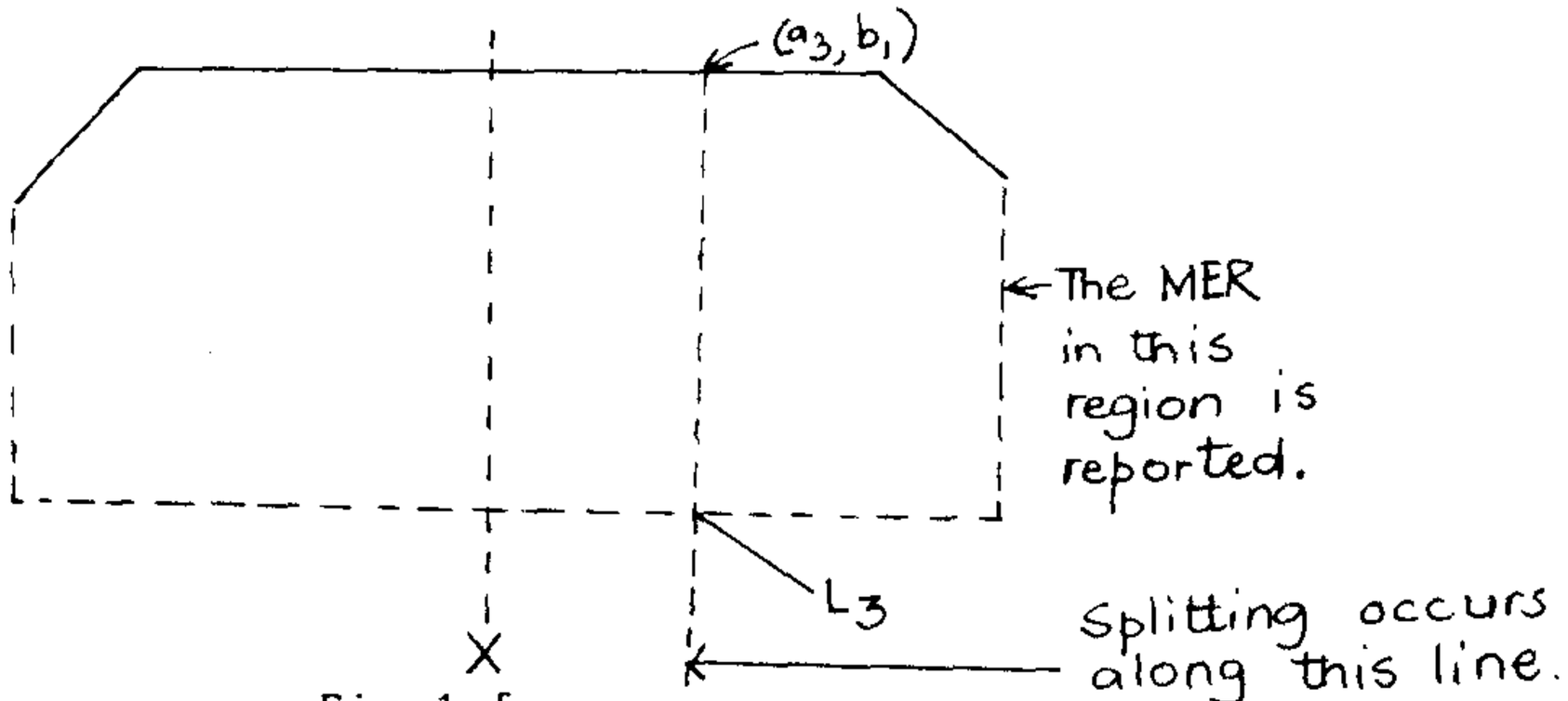


Fig 1.5

Case 1.2.2: a3 > a2(Fig 1.6).

S3 = L3. Determine MER using Result 2.

Generate the type 4 search space [s=(c1,a3),t,

TR*= [(a1,b)(a3,a3.m2+k2)],TL*,0,0].

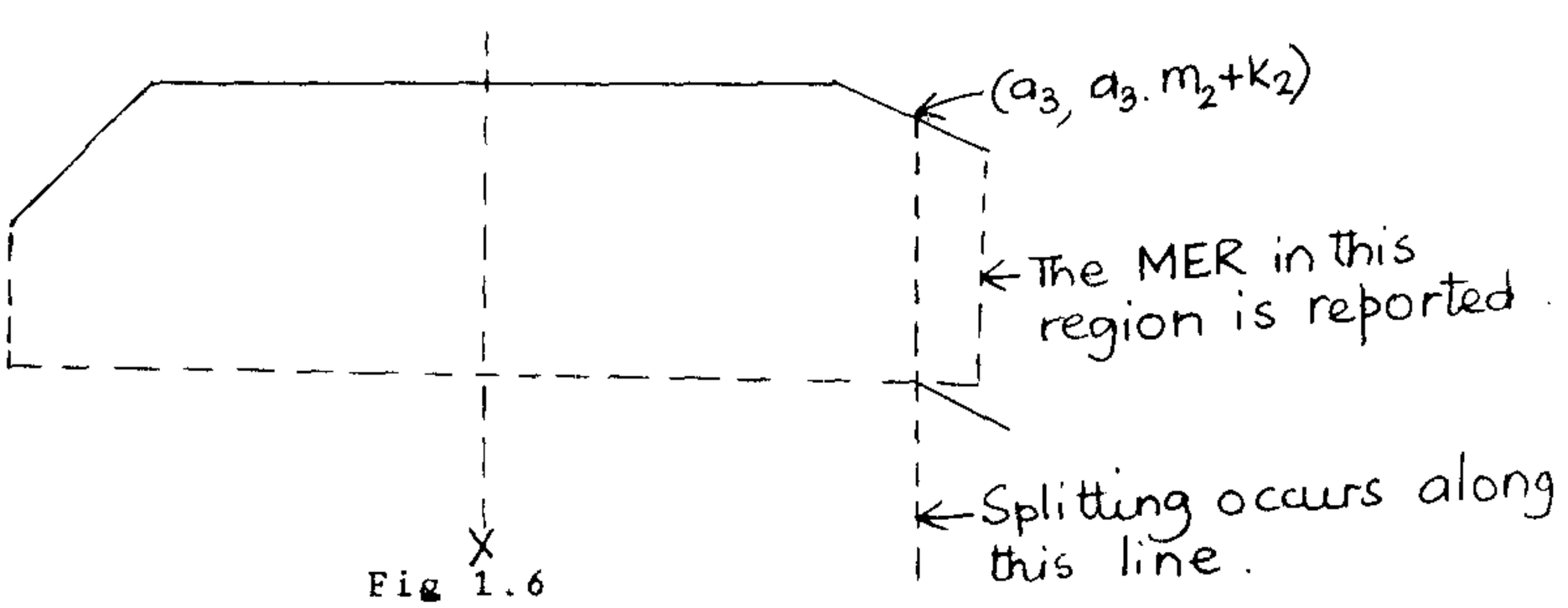


Fig 1.6

Case 2: L3 lies to the left of X.

Exactly similar to Case 1.

2: Processing search spaces of types 5 through 12

For types 5 through 12, only processing type 8 search space will be described. Processing the rest are similar.

Processing type 8 search space

A type 8 search space is given by [s=(c1,c2),t=b3,TR*-[(a2,b2)(c2,d2)],TL*=[(a1,b1)(c1,d1)],BL*=[(a3,b3)(c3,d3)]] where b1 = b2 and c1 = a3.

Let L4[(a4,b4)(c4,d4)] be the new stick intercepted (Fig. 2.1).

Here, C1 = L2, C2 = L1, C3 = L3.

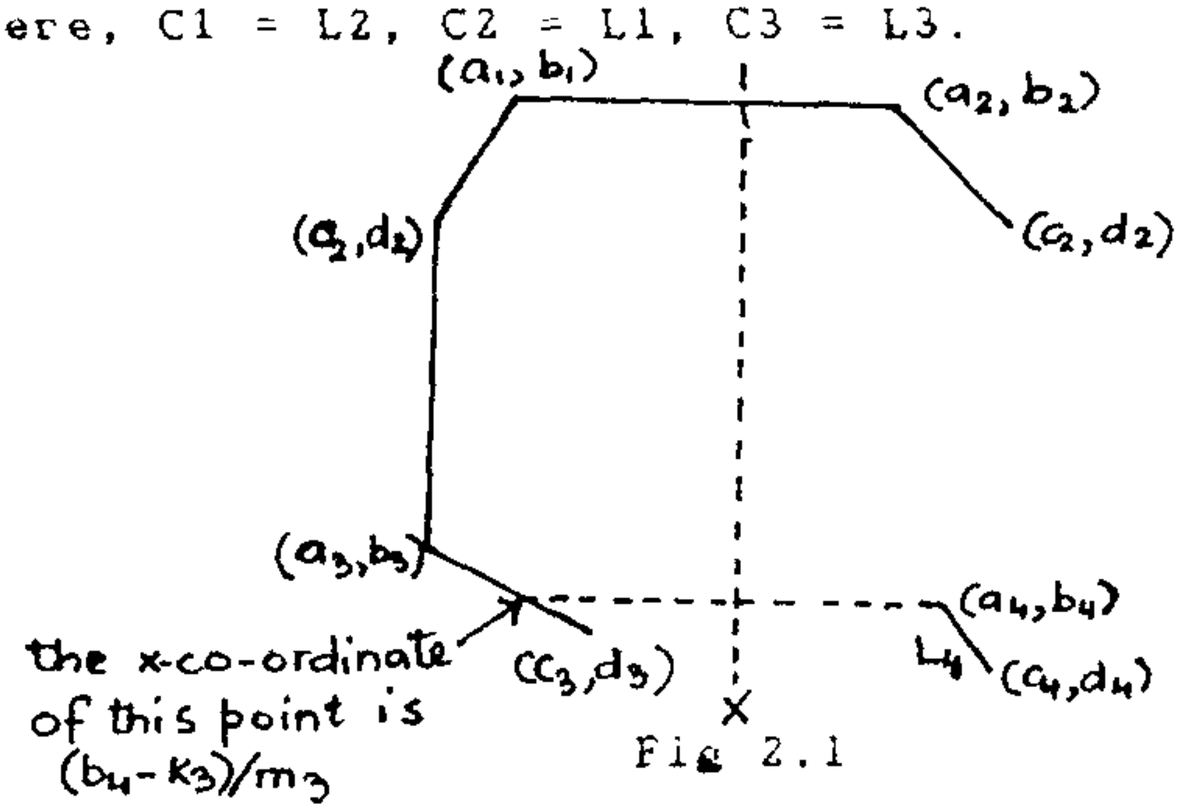


Fig 2.1

Case 1: b4 > d3 and (b4-k3)/m3 <= a4 <= c2.

Case 1.1: L4 lies to the right of X.

Case 1.1.1: a4 <= a2, m4 > 0(Fig 2.2).

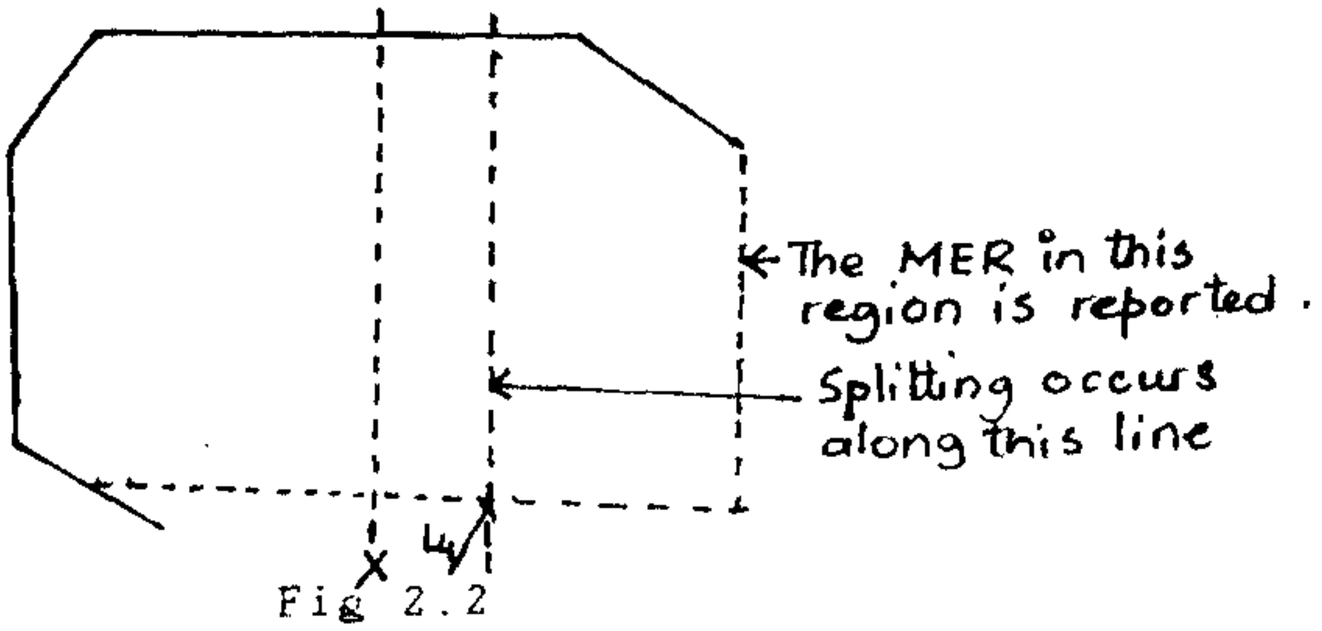


Fig 2.2

S3 = L4. Report an MER using Result 4 as only 3 flexible supports are obtained.

Generate a type 15 search space [s = [c1,a4],t,TR = (a4,b1), TL* = [(a1,b1)((b4-k3)/m3,(b4-k3)(m1/m3)+k1)], BL* = [(b4-k3)/m3,b4)(c3,d3)], BR*= [(a4,b4)(c4,d4)]] if (b4-k3)/m3 <= a3, otherwise generate a type 13 search space with the same set of tuple as above replacing only TL by ((b4-k3)/m3, b1).

Case 1.1.2: a4 <= a2, m4 < 0 (fig. 2.3).

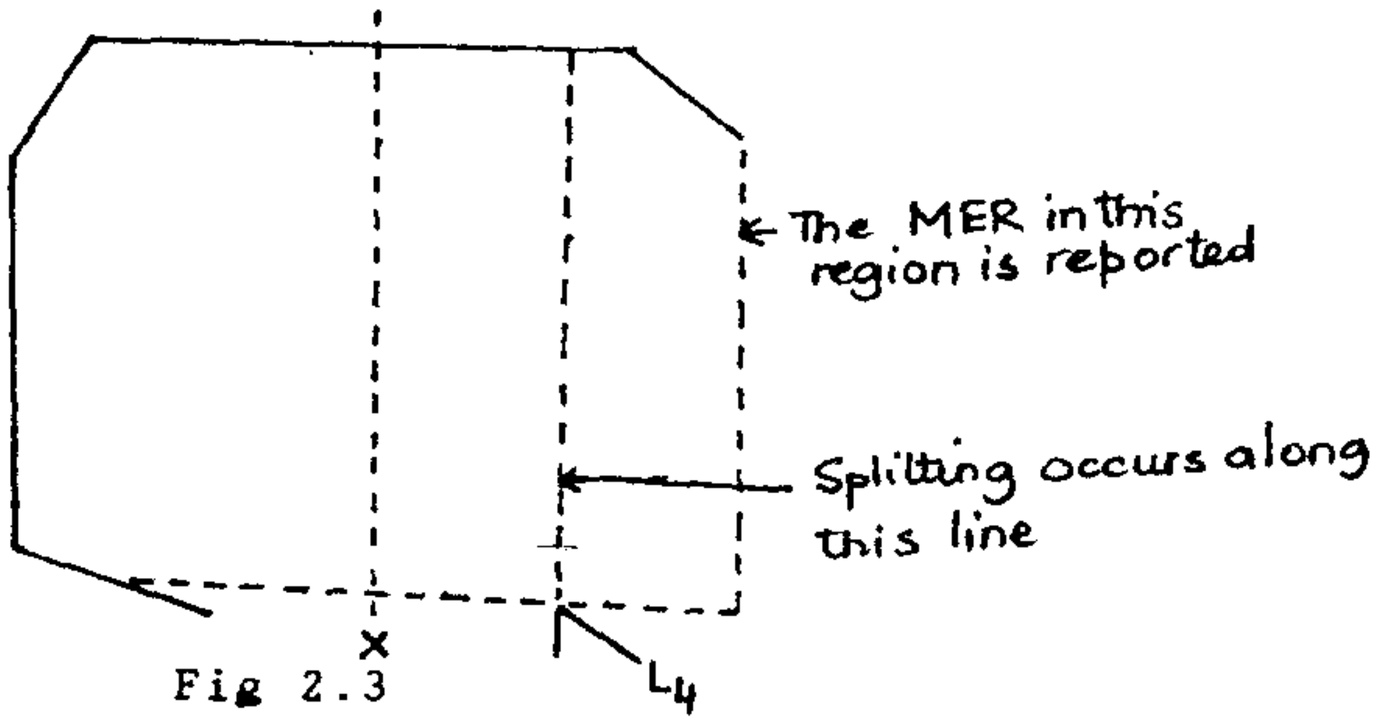


Fig 2.3

The MER is reported as in previous case.

If (b3-k3)/m3 <= a3 then generate a type 7 search space [s=[c1,a4],t,TR =(a4,b1),TL* =((a1,b1)((b4-k3)/m3, (b4-k3))(m1/m3)+k1)],BL*=[(b4-k3)/m3,b4)(c3,d3)],0] otherwise generate a type 5 search space with all parameters remaining same excepting TL which is equal to ((b4-k3)/m3,b1).

Case 1.1.3: a4 > a2, m4 < 0 (fig 2.4).

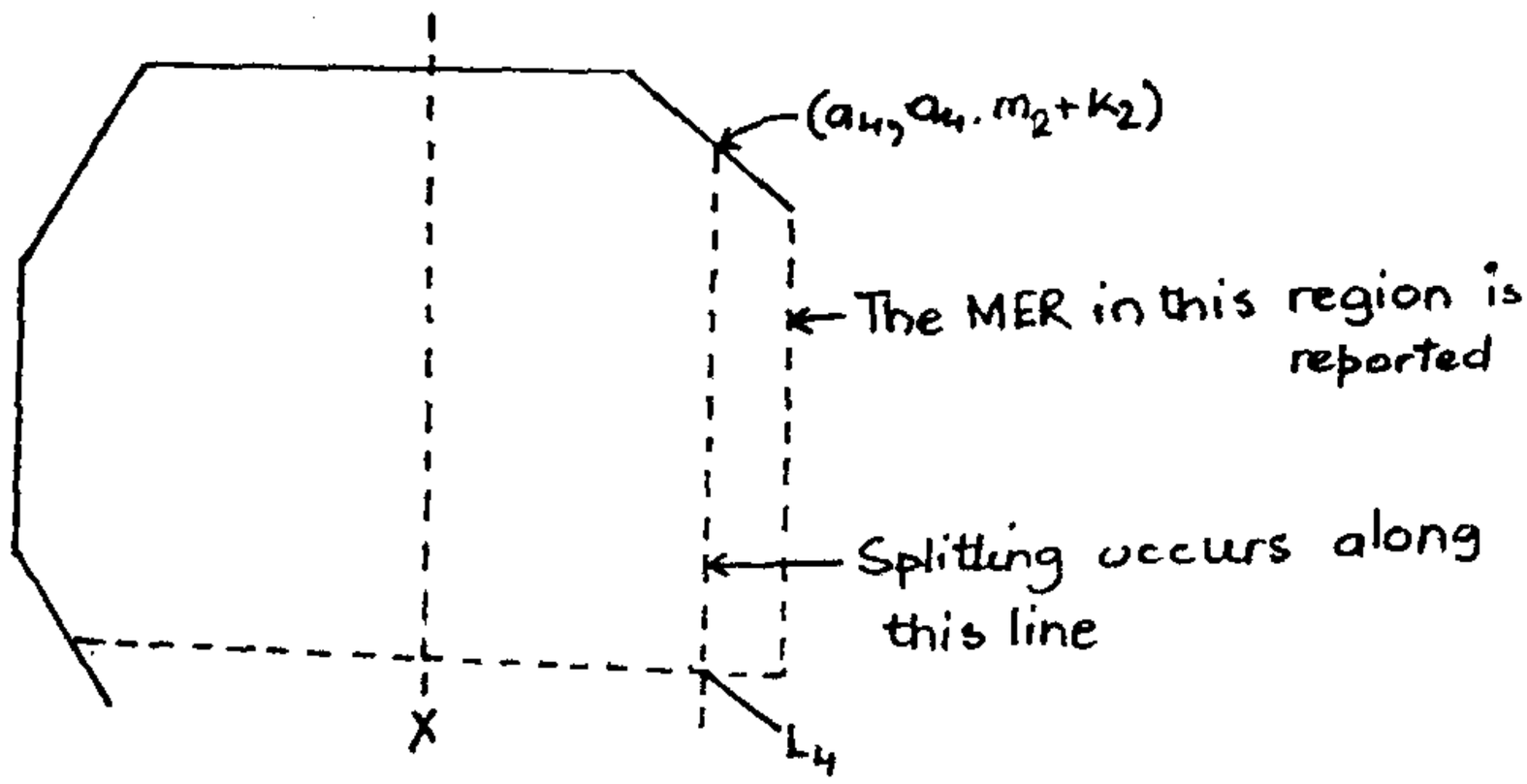


Fig 2.4

The MER is reported as above.

If (b4-k3)/m3 <= a3 then generate a type 8 search space [s=[c1,a4],t,TR* =[(a2,b2)(a4,a4.m2+k2)],TL* =((a1,b1) ((b4-k3)/m3,(b4-k3))(m1/m3)+k1)],BL*=[(b4-k3)/m3,b4) (c3,d3)],0] otherwise type 6 search space is generated which has the same set of tuples except that TL is given by ((b4-k3)/m3,b4).

Case 1.1.4: a4 > a2, m4 > 0(Fig. 2.5).

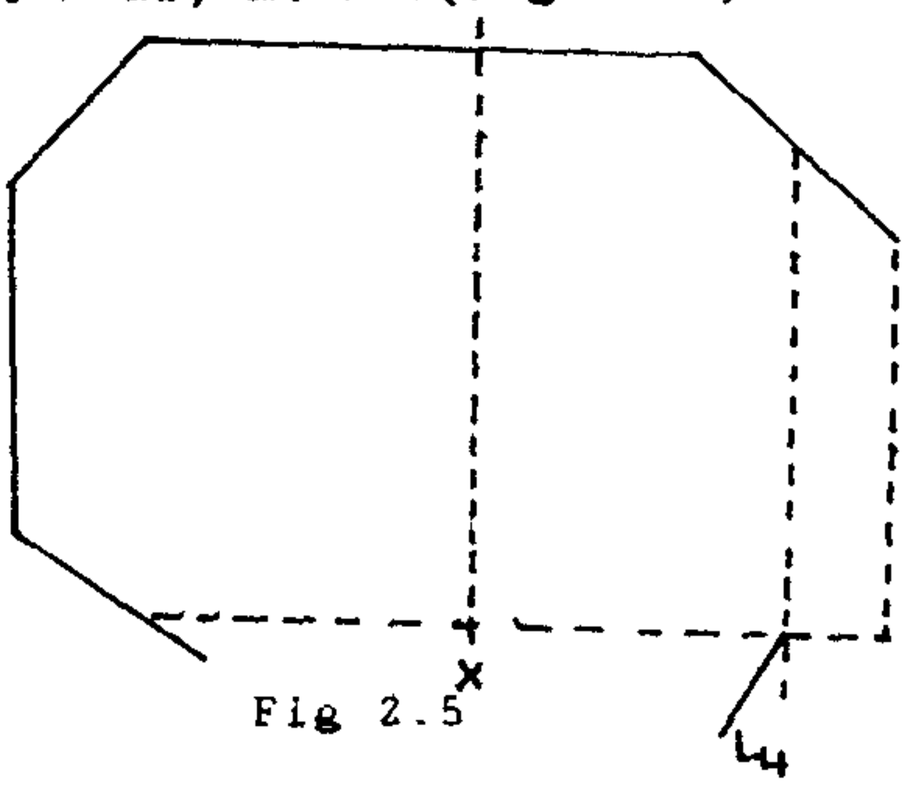


Fig 2.5

Let z = max(d3,d4).

The MER is reported as above.

Generate a type 16 search space given by [s = (c1,a4),t,TR* = [(a2,b2)(a4.m2+k2)], TL*, BL* = [(a3,b3)((z-k3)/m3,z)],BR* = [(a4,b4)((z-k4)/m4,z)].

Case 1.1.5: a4 > c2, m4 > 0(Fig 2.6).

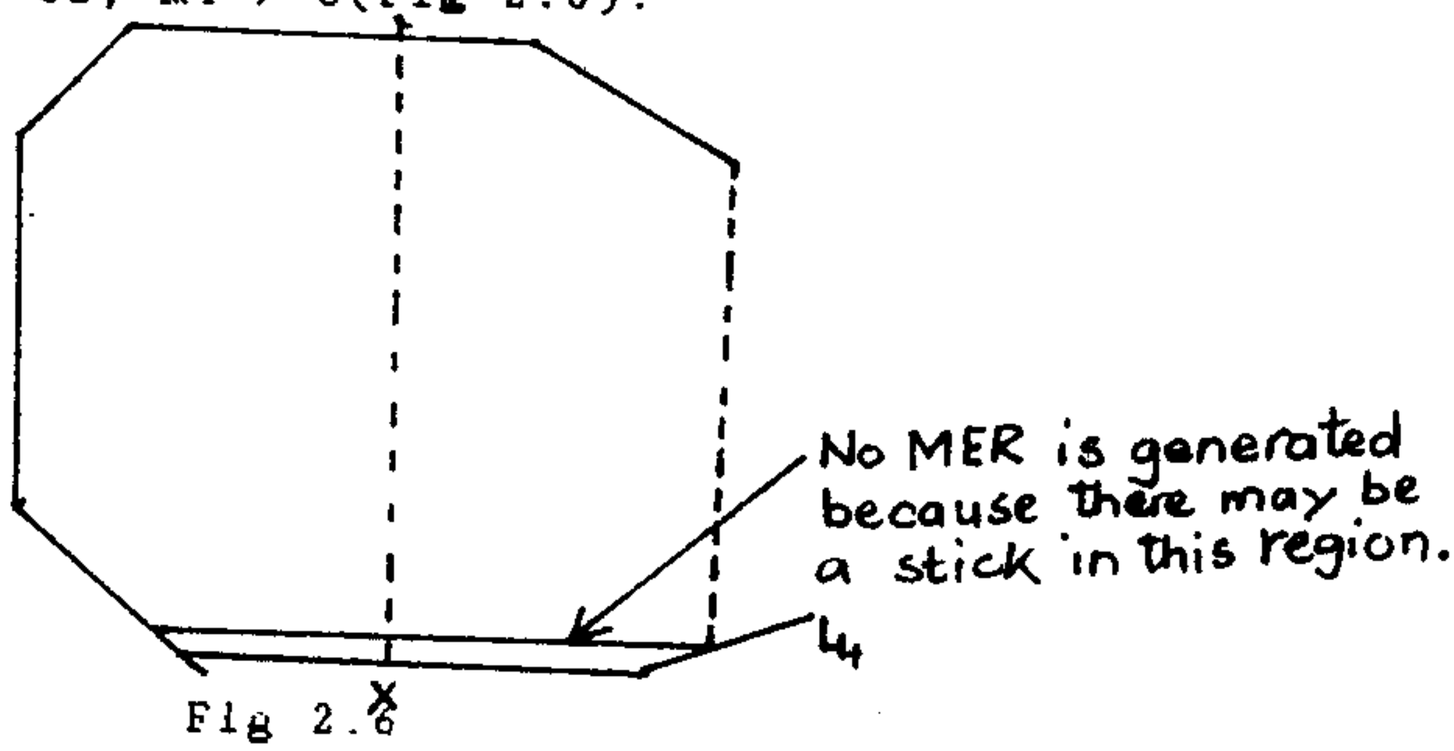


Fig 2.6

No MER is reported this case.

Generate a type 16 search space given by [s,t = b4, TR*, TL*, BL*= [(a3,b3)((z-k3)/m3,z)], BR* = [(c2,c2.m4+k4)((d3-k4)/m4,d3)]].

For the following 6 cases assume that a3 < a1.

Case 1.2: L4 lies to the left of X.

Case 1.2.1: a4 < a1, m4 > 0(Fig. 2.7).

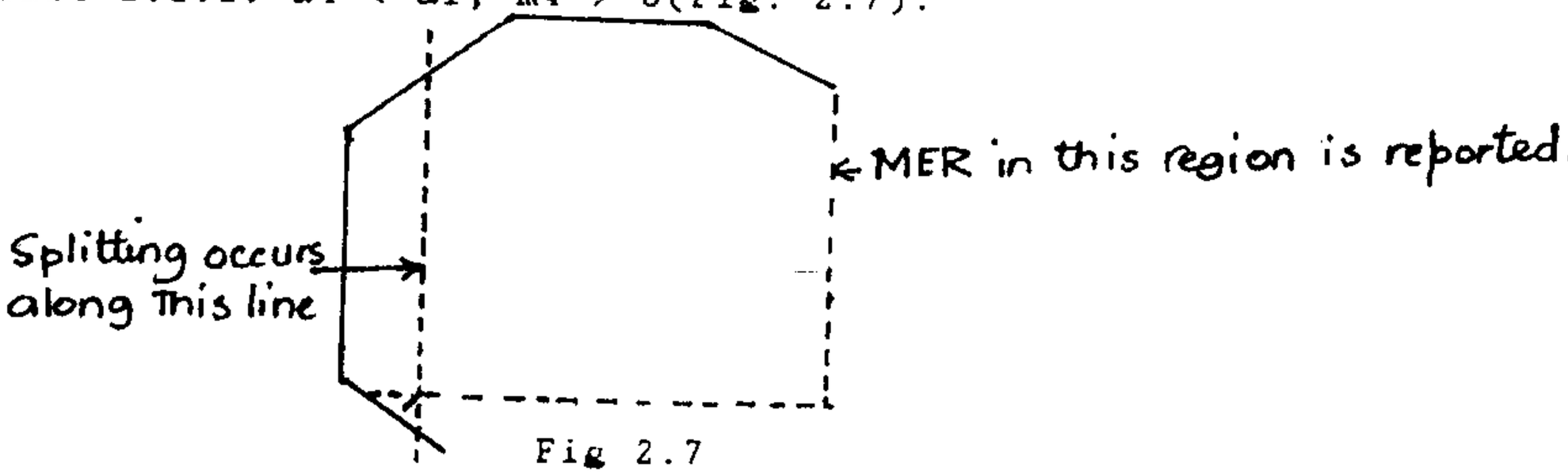


Fig 2.7

The MER is reported as above.

Generate a type 4 search space given by [s = [a4,c2],t, TR*,

TL* = [(a1,b1)(a4,a4.m2+k2)],0,0].

Case 1.2.2: a4 < a1,m4 < 0(Fig. 2.8).

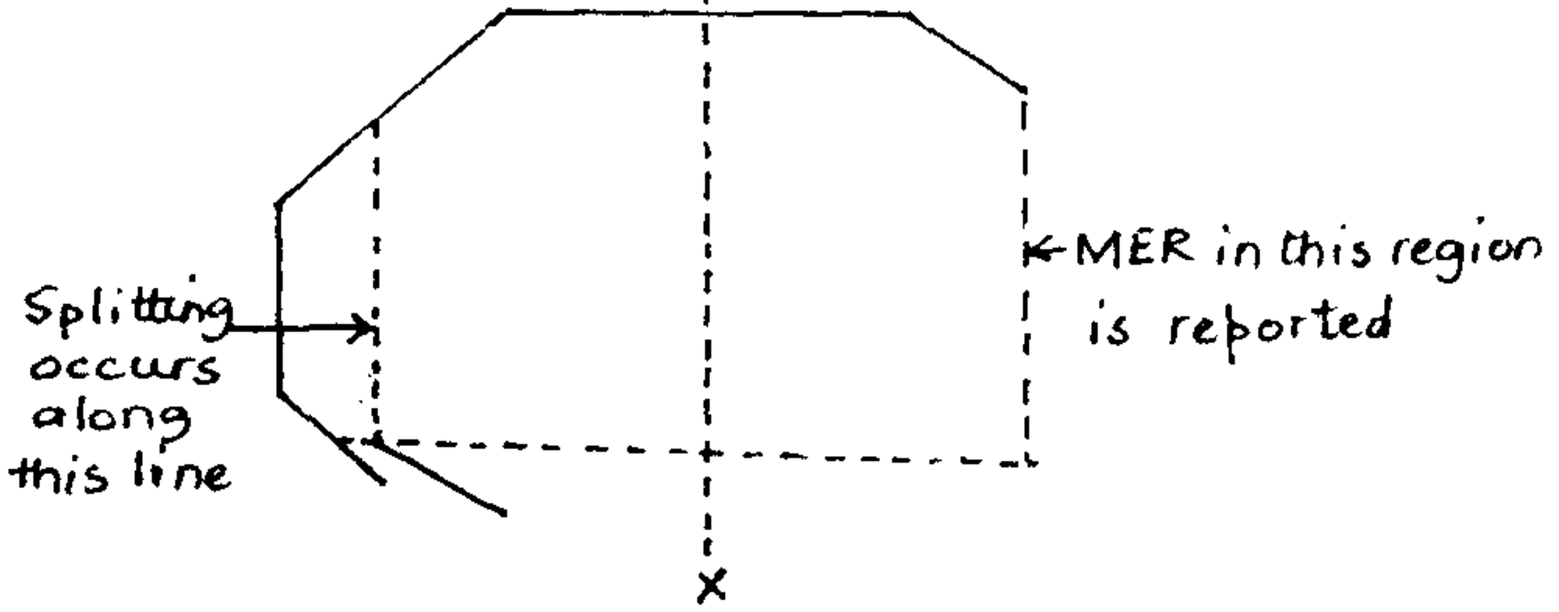


Fig 2.8

The MER is reported as above.

Generate a type 8 search space given by [s = [a4,c2],t,

TR*,TL* = [(a1,b1)(a4,a4.m1+k1)],BL*= [(a4,b4)(c4,d4)].

Case 1.2.3: a4 > a1, m4 > 0(Fig. 2.9).

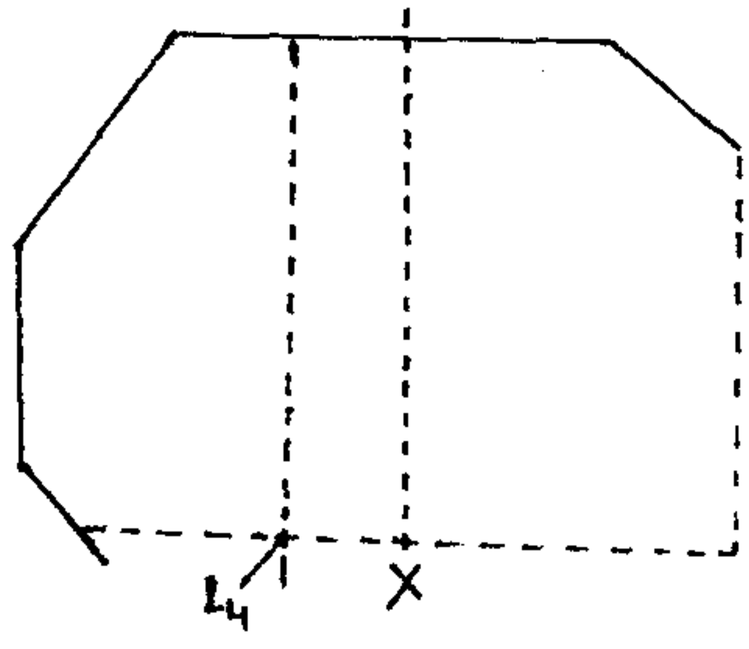


Fig 2.9

The MER is reported as above.

Generate a type 2 search space given by [s=[a4,c1],t,TR*,

TL = (a4,b2),0,0].

Case 1.2.4: a4 >= a1, m4 < 0(Fig 2.10).

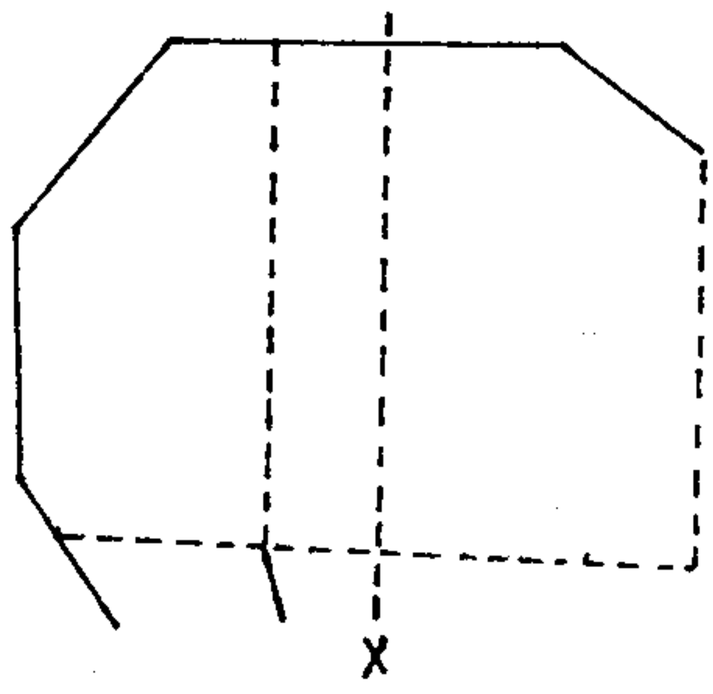


Fig 2.10

The MER is reported as above.

Generate a type 6 search space given by [s = [a4,c2],

t,TR*,TL = (a4,b2),BR* = [(a4,b4)(c4,d4)],0].

Case 1.2.5: a4 < a1, a4 < a3, m4 > 0 (Fig. 2.11).

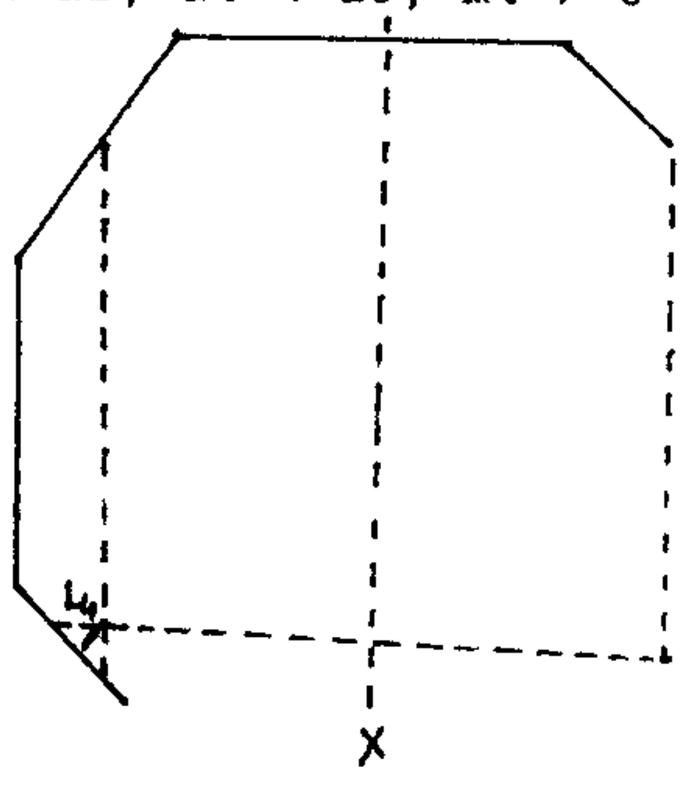


Fig 2.11

Generate an MER by Rsult 4.

Form a type 8 search space [s = [a4,c2],t,TR*,TL* =

[(a1,b1)(a4.m2+k2)],BL*=[(a4,a4.m3+k3)(c3,d3),0].

Case 1.2.6: a4 < a1, a4 < a3, m4 < 0 (Fig 2.12).

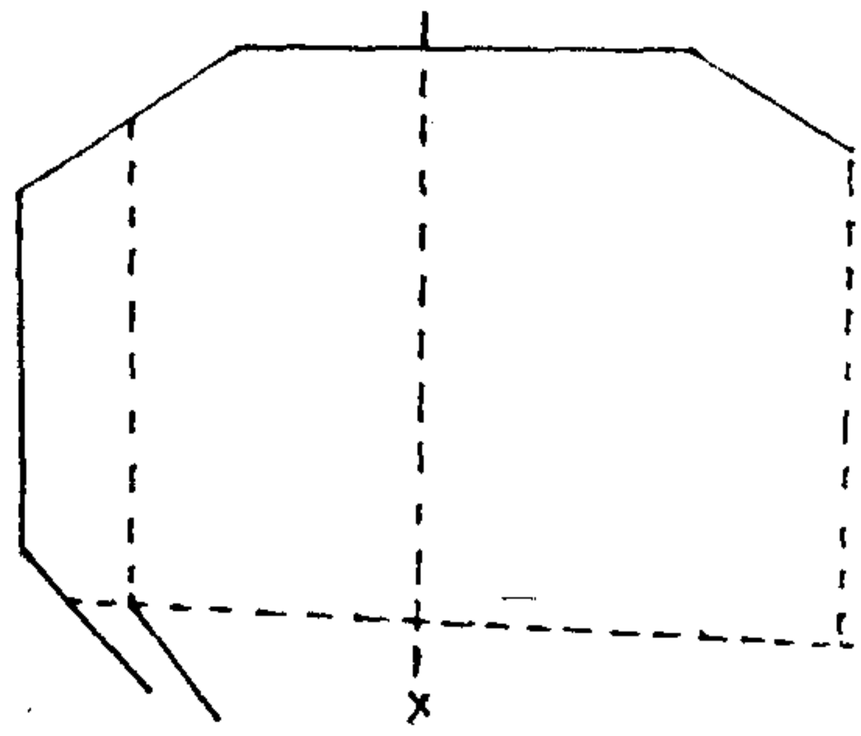


Fig 2.12

Generate MER as above.

Form a type 8 search space [s = [a4,c2],t,TR*, TL* = [(a1,b1)(a4,a4.m2+k2)], BL* = [(a4,b4)(c4,d4)],0].

For the following two cases assume that a1 < a3.

Case 1.2.7: a1 < a4 < a3, m4 > 0 (Fig 2.13).

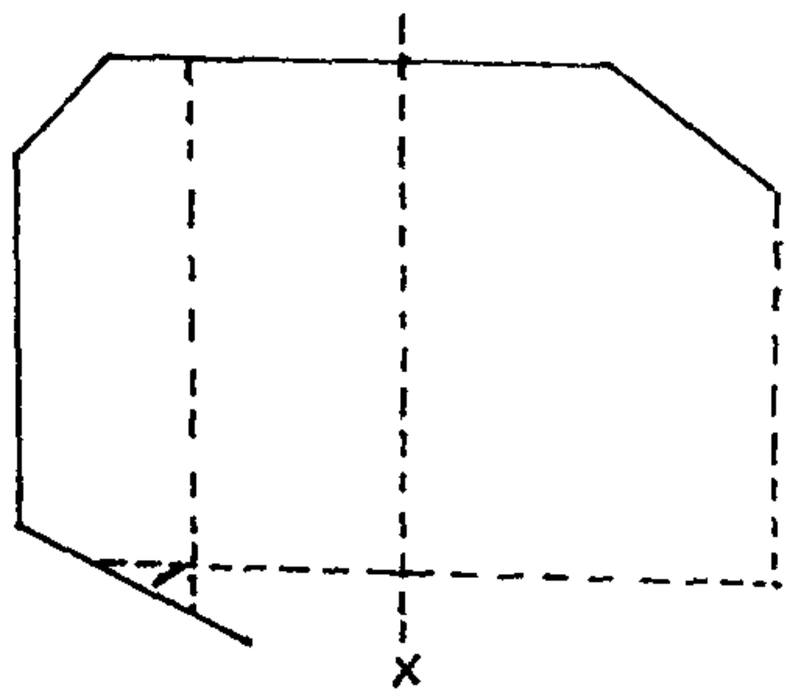


Fig 2.13

Use Result 4 for reporting MER.

Generate a type 6 search space [s = [a4,c2],t, TR*,TL = (a4,b2), BL* = [(a4,a4.m3+k3)(c3,d3),0].

Case 1.2.8: a1 < a4 < a3, m4 < 0 (fig. 2.14).

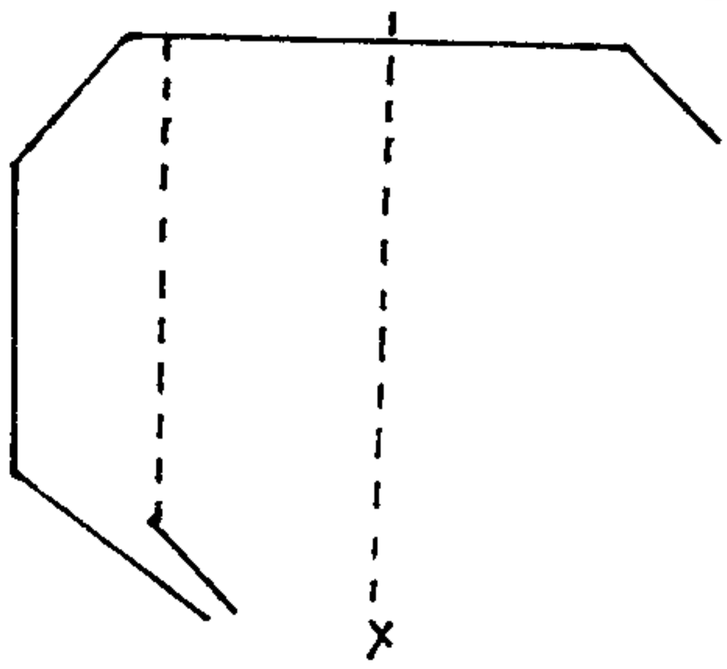


Fig 2.14

Use Result 4 for reporting MER.

Generate a type 5 search space [s = [a4,c2],t,TR*,

TL = (a4,b2), BL* = [(a4,b4)(c4,d4)],0].

Case 2: a4 < d3 (Fig. 2.15).

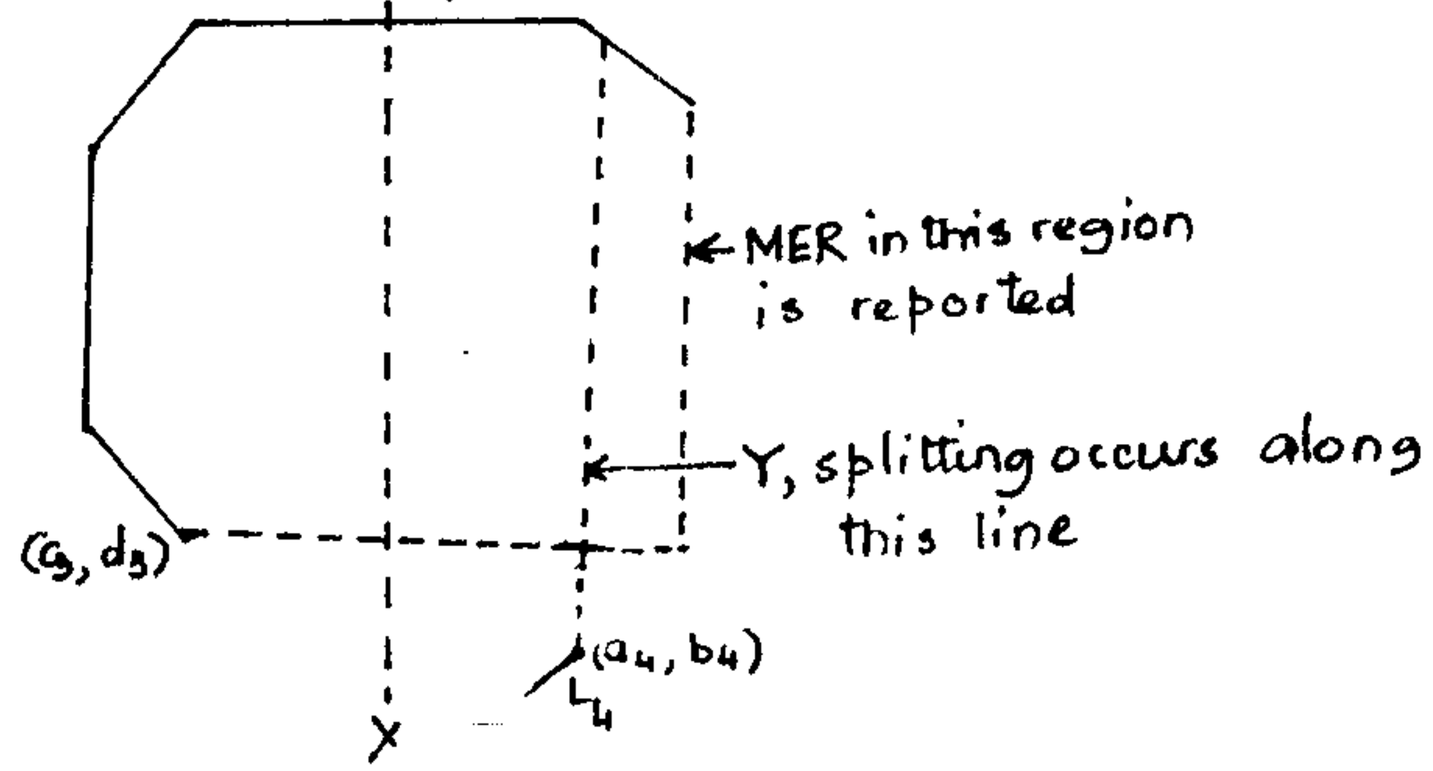


Fig 2.15

In this case Y denotes the line along which splitting occurs.

Case 3: L4 is situated to the left of L3 (Fig 2.16).

Another stick is searched until the situation is like the one mentioned in Case 1 or Case 2.

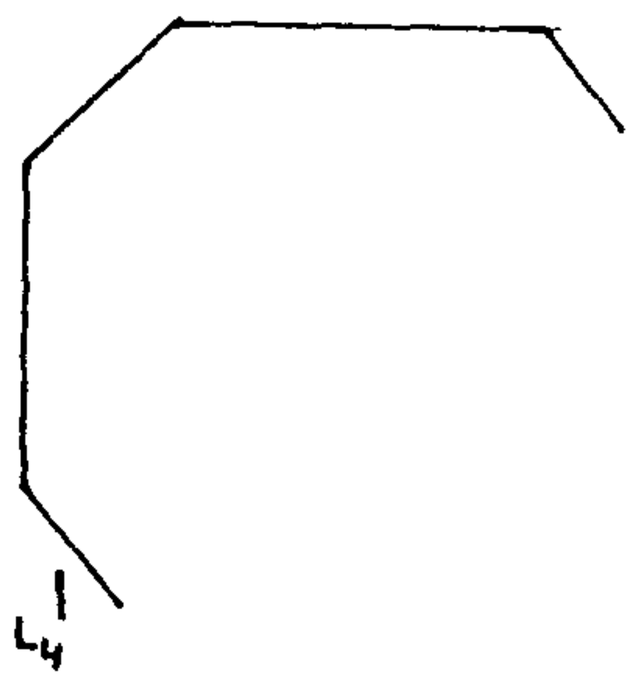


Fig 2.16

3: Processing search spaces of types 13 through 16

For types 13 through 16, only processing type 16 search space will be described. Processing the rest are similar.

Processing the type 16 search space

The type 16 search space is given by [s=[c1,c2],t = b1, TR* = [(a2,b2)(c2,d2)], TL* = [(a1,b1)(c1,d1)], BL* = [(a3,b3)(c3,d3)], BR* = [(a4,b4)(c4,d4)] where b1 = b2; d3 = d4; c1 = a3 and c2 = a4 (Fig. 3.1)

Assume that a2 < c4 < c2 and c1 < c3 < a2. Other situations may arise when c2 < c4 < a2 or a1 < c3 < c1.

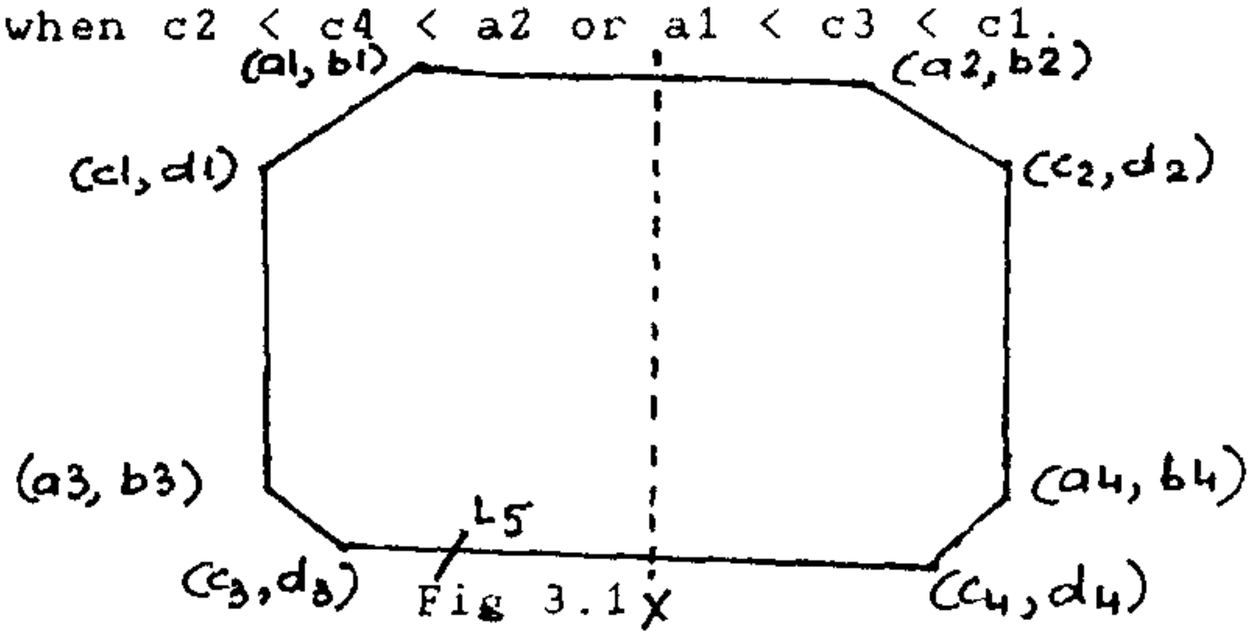


Fig 3.1

Case 1: L5 lies to the right of X.

Case 1.1: m5 > 0.

Case 1.1.1: a5 > a2 and a5 < (b5 - k4)/m4 (Fig.3.2).

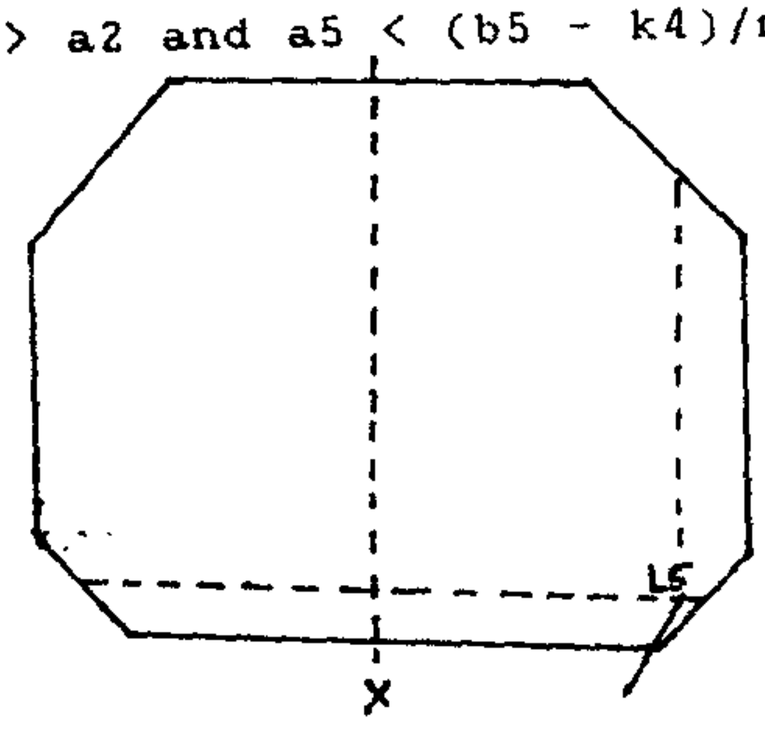


Fig 3.2

C1 = L2, C2 = L1, C3 = L3, C4 = L4, S3 = L5.Report MER using Result 5.

Let z = max(d3,d5).

Generate the type 16 search space [s = ((b5-k3)/m3,a5),t, TR*=[(a2,b2)(a5.m2+k2)], TL* = [(a1,b1)((b5-k3)/m3, ((b5-k3)(m1/m3)+k1)],BL* = [((b5-k3)/m3,b5)((z-k3)/m3,z)], BR*= [(a5,b5)(c5, (z- k5)/m5,z)]].

Case 1.1.2: a5 <= a2 (Fig. 3.3).

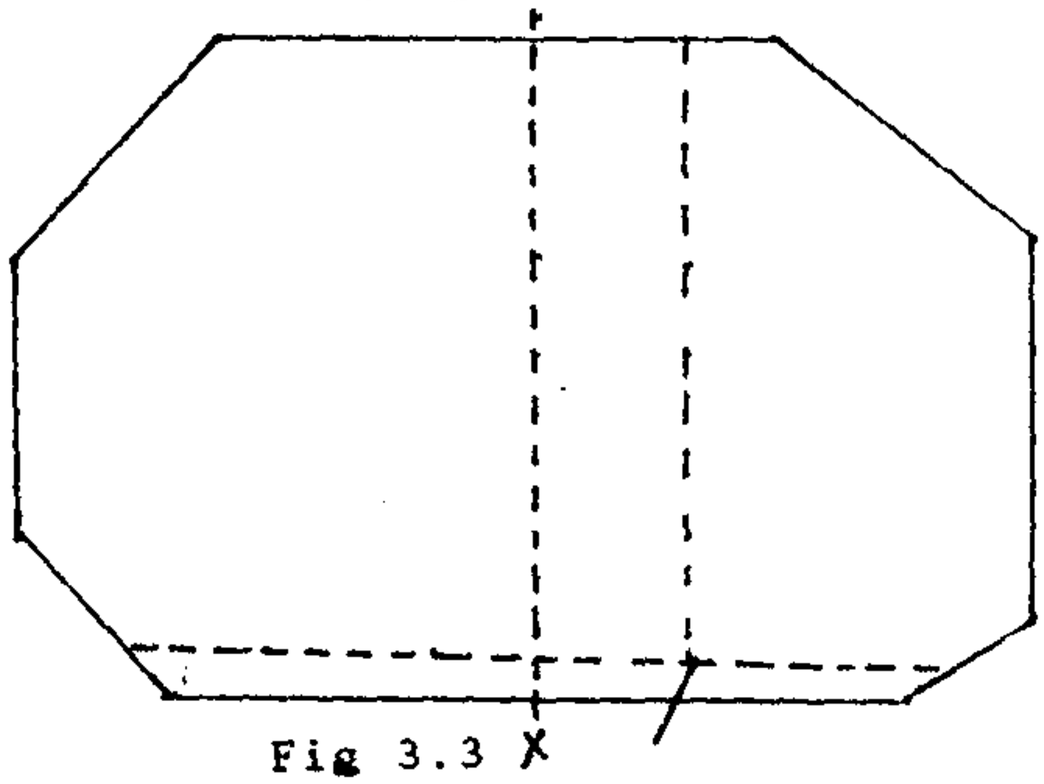


Fig 3.3 X

Report MER using Result 5.

Generate the type 15 search space given by [s=(c1,a5),
t,TR = (a5,b1), TL*, BL*=[(a3,b3)((z-k3)/m3,z)],
BR* = [(a5,b5)(z-k5)/m5,z)]].

Case 1.2: m5 < 0.

Case 1.2.1: a2 < a5 <= c2 (Fig. 3.4).

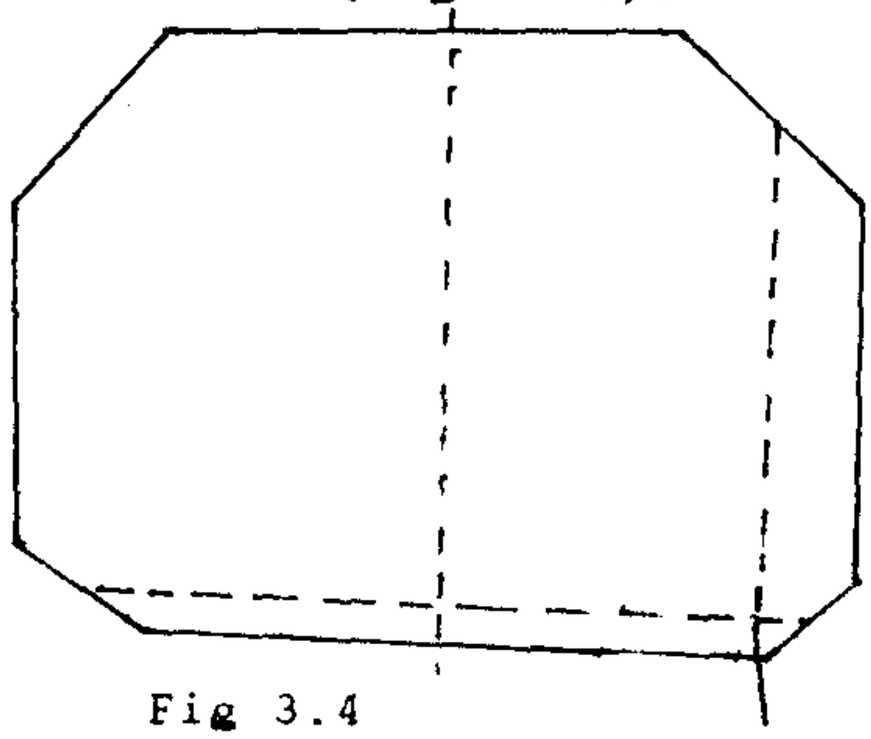


Fig 3.4

Report MER using Result 5.

Generate the type 8 search space given by [s = (c1,a5),
t, TR* = [(a2,b2)(a5.m3+k3)],TL*,BL*,0].

Case 1.2.3:a5 <= a2 (Fig. 3.5).

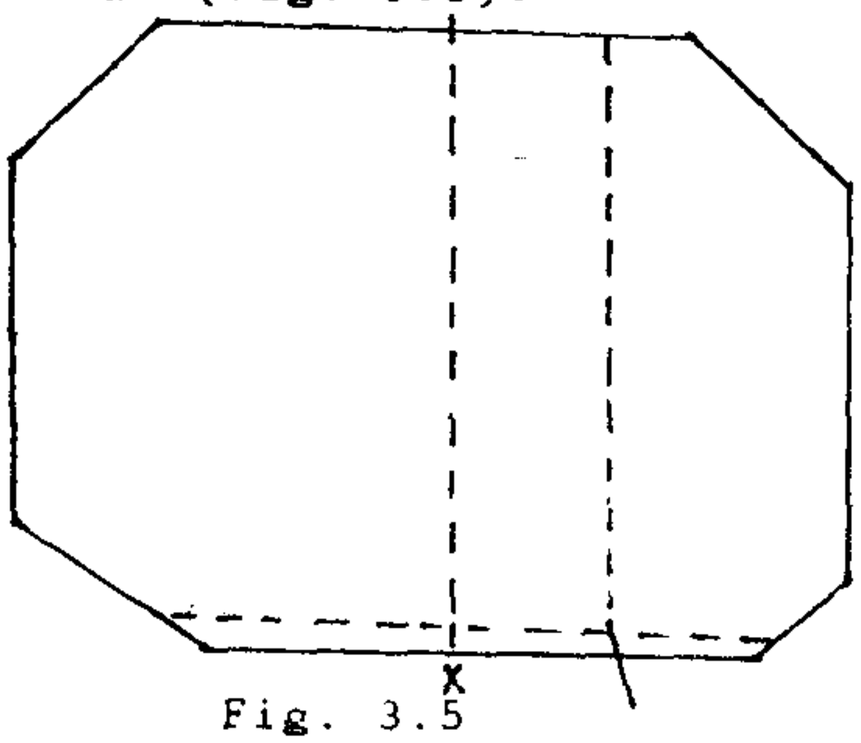


Fig. 3.5

Report MER using Result 5.

Generate the type 15 search space given by [s=[c1,a5],t, TR = (a5,b1),TL* = [(a1,b1)((b5-k3)/m3,(b5-k3)(m1/m3)+k1),BL* = [((b5-k3)/m3,b5)(c3,d3)],TR* = [(a5,b5)(c5,d5)].

Case 2: L5 lies to the left of X.

This is similar to Case 1.

Case 3: a5 < max(d3,d4)(Fig. 3.6)

In this case, splitting occurs along Y1 and Y2.

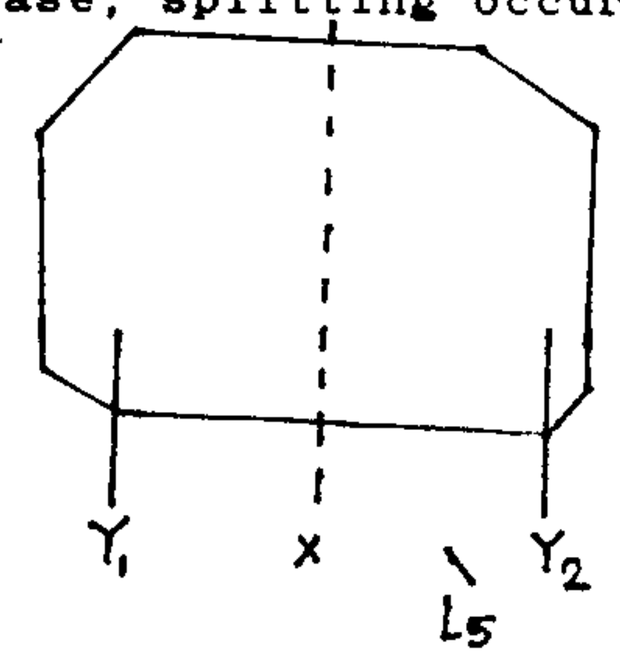


Fig. 3.6

Case 4: L5 lies outside the search space (Fig 3.7).

Another stick is searched until the situation becomes like the one mentioned in one of the previous cases.

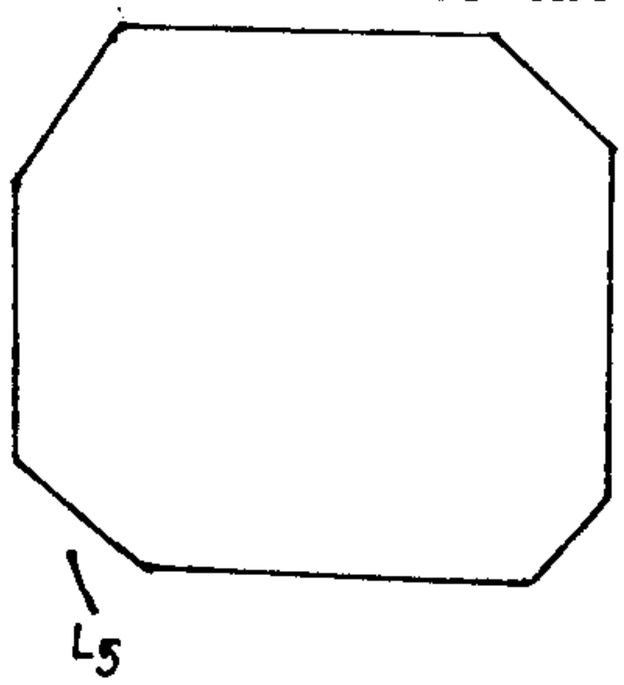


Fig 3.7

9. REFERENCES:

1. Naamad, A., Lee, D.T. and Hsu, .L., "On the maximum empty rectangle problem", Discrete Applied Mathematics, Vol. 8, 1984, pp. 267-277.

2. Atallah, M.J., and Frederickson, G.N., "A note on finding the maximum empty rectangle", Discrete Applied Math., Vol. 13, pp 87-91, 1986.

3. Orlowski, M, " A new algorithm for largest empty rectangle problem", Algorithmica, Vol. 5, pp. 65-73, 1990.

4. Chazelle, B., Drysdale, R.L. and Lee, D.T., "Computing the largest empty rectangle", SIAM Journal of Computing, Vol. 15, 1986, pp. 300-315.

5. Aggarwal, A. and Suri, S., "Fast algorithm for computing the largest empty rectangle", Proc. 3rd Annual ACM Symposium on Computational Geometry", 1987, pp. 278-290.

6. Nandy, S.C., Bhattacharya, B.B. and Ray, S., "Efficient algorithms for identifying all maximal isothetic empty rectangles in VLSI layout design", Proc. FSTTCS - 10, 1990, pp. 255-269.

7.Preparata, F.P., and Shamos, M.L.,"Computational Geometry: An Introduction",Springer Verlag,NY,1984,pp 352-355.