DIMACS Technical Report 2002-57
December 2002

Lower bounds for the size of binary space partition of rectangles

by

Ying Liu[1,2]
RUTCOR, Rutgers University,
640 Bartholomew Road, Piscataway, NJ 08854, USA.
E-mail: yingliu@rutcor.rutgers.edu

[1]Affiliated Graduate Student Member

[2]I would like to thank Endre Boros and Vladimir Gurvich for their advice and encouragement. The financial support from DIMACS is greatfully acknowledged.

ABSTRACT

Binary space partition (BSP) tree is one of the most popular data structures in computational geometry. We proved that the lower bound on the exact size of BSP trees for a set of n isothetic rectangles in the plane is $2n - o(n)$ when the rectangles tile the underlying space, this closed the gap between the upper and lower bounds for this case. Also, in general, for a set of n isothetic rectangles in the plane, we improved the lower bound from $\frac{9}{4}n - o(n)$ to $\frac{7}{3}n - o(n)$.

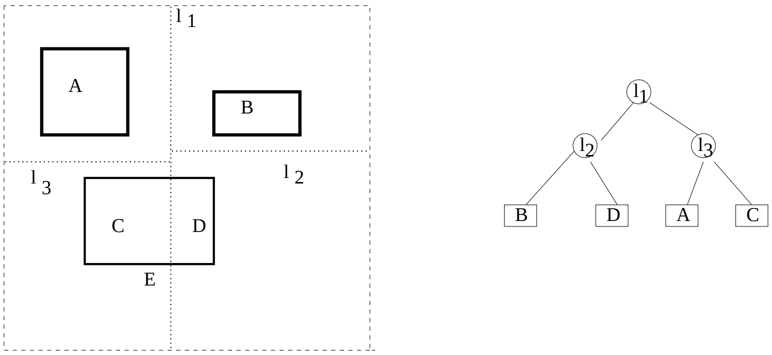

Figure 1: Example of a BSP and its BSP tree

1 Introduction

Given a set of objects S in an Euclidean space, a hyperplane h is called a *guillotine cut* if h separates S into two disjoint sets. Each fragment of the object belongs solely to one of the parts it falls in. A guillotine cut l is called *perfect* if l does not interior intersect any object of S.

A *binary space partition (BSP)* is a scheme for recursively dividing a set of objects by guillotine cuts until all objects are separated. This division process can be naturally represented as a *binary tree* (BSP tree) where a node represents a region of the plane and stores the cut that splits the plane into two parts that its two children represent; each leaf of the BSP tree represents the final partitioning of the plane and stores at most one fragment of an input object. Since a guillotine cut may split an object into two, the number of fragments in the end of the partition, equivalently the number of leaves in the BSP tree, may exceed the number of input objects.

Given a set of objects, the total number of the resulting fragments in the partition is called the *size* of the BSP.

A BSP of a set of n objects is called *perfect* if the size of the BSP is n.

A *box* in dimension d is an interval $\{\mathbf{x} : \mathbf{e} \leq \mathbf{x} \leq \mathbf{s}, \mathbf{e}, \mathbf{s} \in \mathbb{R}^d\}$. A box in $\mathbb{R}^2$ is called a *isothetic rectangle*. In this paper we simply call it a *rectangle* without confusion. A *partition* of the *box* $[0,1]^d$ is a set of boxes such that each point $\mathbf{a} \in [0,1]^d$ lies within some box and no two boxes have a common interior point. A partition in $\mathbb{R}^2$ is also called a *tiling*. A *packing* of a box B is a set of interior disjoint boxes inside B.

Figure 1 shows a BSP of three rectangles and its corresponding BSP tree.

In Figure 1, there are 3 rectangles A, B, E. The cut l_1 cuts $E(C \cup D)$ into two fragments: C and D, then the cut l_2 separates rectangles A and C, and the cut l_3 separates B and D. The corresponding BSP tree is in Figure 1. The size of this BSP tree is 4. There exists a BSP tree with size 3 for this example.

Since the introduction of BSPs, the corresponding data structure *BSP trees* in [4] have become one of the most popular data structures. They present a way to implement a geometric divide and conquer strategy. BSPs have numerous applications such as: graphics (hidden surface removal, shadow generation), computational geometry (ray-tracing, visibility problems, solid geometry), robotics (motion planning), spatial databases and approximation algorithms, see e.g. [1, 7].

A *binary space partition problem* consists in minimizing the total number of fragments generated by the partition. The binary space partition problem is a fundamental problem for applications.

In this paper we focus on rectangles in the plane because such rectangles are very important in applications and more complex objects are often replaced by their bounding rectangles. They also arise in plane tiling problems (constructing histograms in the plane), and as subproblems when higher dimensional hyper rectangles are projected onto a plane [1].

In this paper, as in [2, 5, 6], any BSP will be *auto-partition*, i.e., the cut lines will be the extensions of the input segments. Because the objects are rectangles, any cut will be either vertical or horizontal. The BSP in Figure 1 is a auto-partition.

In 1992, Paterson and Yao proved an upper bound of $12n$ for the size of BSP of a set of n rectangles [6], later improvements led to $4n$ in [2, 5], recently Berman et. al. proved upper bounds of $3n$ for n rectangles and $2n$ for tiling case [1]. The current best known lower bounds were $\frac{3}{2}n - o(n)$ for the tiling case [1] and $\frac{9}{4}n - o(n)$ for the packing case [3].

In this paper we answer the open problem in [1] to close the gap for the tiling case and improve the lower bound from $\frac{9}{4}n - o(n)$ to $\frac{7}{3}n - o(n)$ for the packing case.

2 Lower bound for the size of BSP of rectangles in tiling case

Let us associate a set of rectangles S to a simple directed graph G_S: let $L = \{l_1, l_2, \ldots, l_m\}$ be the set of the longest line segments of S. We define that l_i *cut* l_j if, by extending one side of l_i, the segment l_j is the first that is cut into two parts. Each segment l can cut at most two segments (on either side of l). We define the simple directed graph $G_S = (V, E)$: $V = L, (l_i, l_j) \in E$ if l_i cuts l_j.

For example, in Figure 2, there are 7 longest line segments for A, B: $l_1, l_2, \ldots, l_7$. Among them only line l_4 cuts l_5.

The cycles of the graph G_S can help to check if a set of rectangles has a perfect cut. Each cycle of G_S corresponds to a set of line segments and its bounded cell.

For example, in Figure 3, the lines a, b, c, d correspond to a length 4 cycle in the graph. We call a rectangle whose four sides form a cycle a *join*. A cycle (or its corresponding cell) is called *touched* if there is a guillotine cut intersecting the interior of the *cell* bounded by the lines of the cycle. The terms *join, cell, touched* were first defined in [3]. For example, in Figure 4, lines l_1, l_2, l_3, l_4 form a cycle C. The line l_5 intersects the cell of the cycle, so C is touched.

Lemma 1. *Given a set of rectangles S, there is no perfect BSP, if and only if there is an untouched cycle.*

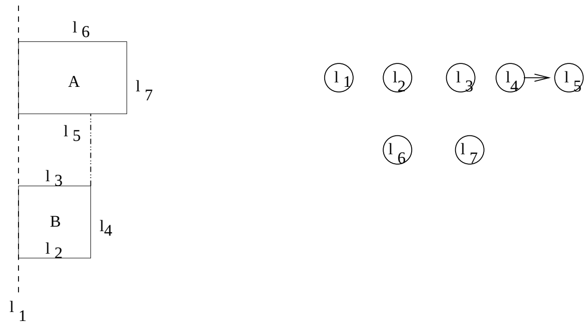

Figure 2: Line segments of A, B and its directed graph

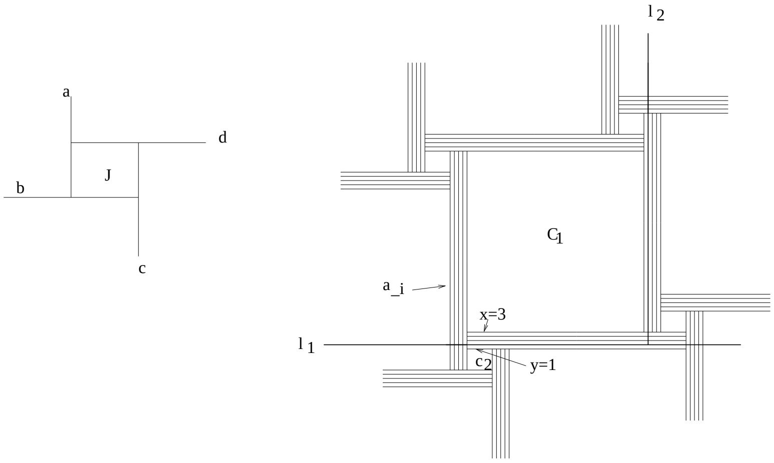

Figure 3: Joins with thickness 1 and 5

Proof. First let us prove the necessity. Suppose $S' \subseteq S$ is the smallest subset of S such that there is no perfect guillotine cut. Then each segment cuts some other segments. Start from any segment l_1, there is a segment cut by l_1, denoted by l_2, there is a segment cut by l_2, denoted by l_3, …, by the finiteness of the segments, there is a cycle in G_S.

Let us prove the sufficiency. For a cycle c in G_S, let us denote A to be the set of segments corresponding to c. Because there is no segment which is a perfect guillotine cut through it, the set A can not be divided into two sets without cutting segments, i.e. generating extra rectangles. So the result is proved.

□

Our construction for obtaining the best lower bound is as follows. In Figure 5, there are 3×3 small joins and 2×2 large joins, and each segment corresponds to 2 parallel segments of equal length. We call this configuration 3×3 2-grid. In general, a $k \times k$ m-grid is a configuration as in Figure 5 in which there are k^2 small joins and $(k-1)^2$ large joins, and every two parallel segments with equal length in Figure 5 is replaced by m parallel segments

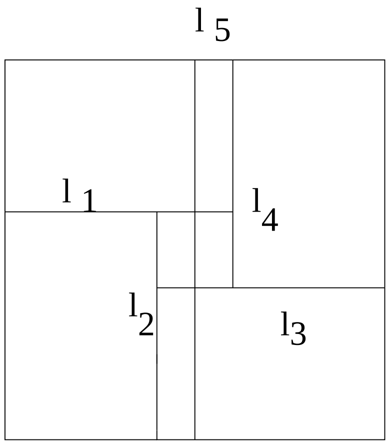

Figure 4: Example of a touched cycle

with equal length. So in a $k \times k$ m-grid, there are $e = k^2 + (k-1)^2$ joins, the total number of segments is $n = 2mk(k+1)$, the total number of rectangles is $n - 1 = 2mk(k+1) - 1$.

Let us define the *thickness* of a join c, denoted by $t(c)$, to be the minimum number of rectangles that one needs to cut so that c is not a join any more. In each side of a join c, because there are at most m segments, so we denote $A(c)$ to be the set of at most $4m$ segments associated to c. In fact, $t(c)$ is equivalent to the minimum number of segments of $A(c)$ needed to be cut such that c will not be a join, because in our configuration each cut has to cut a rectangle first before it cuts a segment of $A(c)$. For example, in Figure 3, before the cuts l_1 and l_2, the thickness of C_1 is 5. After the cut l_1, $t(C_1) = 3$ and after another cut l_2, $t(C_1) = 1$.

By Lemma 1, any BSP of a $k \times k$ m-grid has to touch all those $k^2 + (k-1)^2 = e$ joins. Let c_n denote the number of cuts that generate extra rectangles in the partition, i.e. c_n is the number of extra rectangles generated, and the size of BSP is $n + c_n$. We claim that $c_n \geq (m - \frac{1}{2})e$.

We employ a charging scheme that charges each join of the configuration with at least $m - \frac{1}{2}$. It maintains the following invariant: if the thickness of a join is m, i.e. $t(C) = m$, then the charge $w(C) = 0$; otherwise $w(C) = m - t(C) - \frac{1}{2}$. By this way, the charge of each cycle (join) counts the number of extra rectangles generated by decreasing the thickness of the cycle. By Lemma 1, in the end of any BSP, the thickness of any cycle has to be 0, so each join will be charged at least $m - \frac{1}{2}$.

Let us examine the first touch for a join. As in Figure 3, suppose l_1 is the first cut for the join C_1. l_1 cuts through segments $a_1, \ldots, a_m$ by cutting m rectangles, we define the joins C_1 and C_2 to be the *neighbors* of l_1 on $a_1, \ldots, a_m$. There are at most two neighbors of each cut (if both C_1, C_2 are joins, then there are two neighbors, otherwise, there is at most one). After the cut l_1, the thickness of C_1 decreases from m to x, and the thickness of C_2 decreases from m to y, and $x + y = m - 1$.

The charge of C_1 becomes $w(C_1) = m - t(C_1) - \frac{1}{2}$ and the charge of C_2 becomes $w(C_2) =$

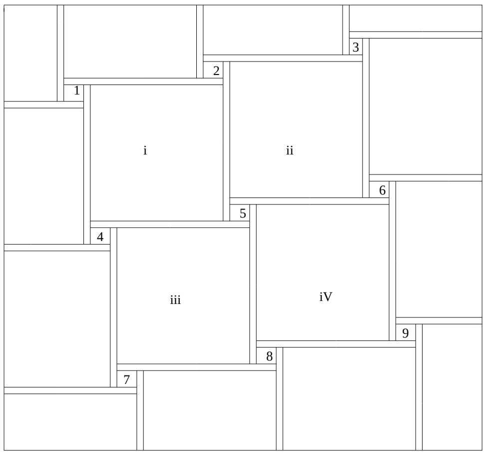

Figure 5: 3×3 2-grid

$m - t(C_2) - \frac{1}{2}$. $(m - t(C_1) - \frac{1}{2}) + (m - t(C_2) - \frac{1}{2}) = m$. So the charge is valid because l_1 has cut through m rectangles to decrease the thickness of C_1, C_2.

Because there are no aligned segments, so each cut is to extend a segment either anchored on a side of the region or is in the center of the region.

When a cut l extends a segment in the center of the region, as in Figure 6, l cuts rectangles B_1, B_2 to decrease the thickness of joins J_4 and J_i from 2 to 0 and 1 respectively. The charges on them are $\frac{1}{2}$ and $\frac{3}{2}$ respectively, so the total charge is 2. Then, the line l cuts the rectangles B_3 and J_{iV} to decrease the thickness of rectangles 5 and iii, finally, line l cuts the rectangles B_4, B_5 to decrease the thickness of the rectangle J_{iV}. It can be checked that the charge is valid and the invariant is kept for each join.

When a cut l is to extend a segment anchored on a side of the region, as in Figure 7, l cuts the rectangles B_1 and J_{iii} to decrease the thickness of rectangle 4 (there is only one neighbor in this case). Then, l cuts through the rectangles B_2 and J_{iV} to decrease the thickness of rectangle J_{iii}, finally, l cuts through the rectangles B_3, B_4 to decrease the thickness of J_{iV}. It can be checked that the charge is valid and the invariant is kept for each join.

So the invariant holds after the first cut that touches C_1. After the first cut, there are two cases:

Case 1: If a cut on a set of rectangles reduced only the thickness of one join. By the definition of thickness, the reduced number is the minimum number of rectangles needed to be cut. So the invariant holds.

Case 2: If a cut l on a set of rectangles reduced the thickness of two neighbors: join C_1 and C_2. Let us denote $r_1 = t(C_1) - t(C_1)'$, $r_2 = t(C_2) - t(C_2)'$ where $t(C_1)'$ and $t(C_2)'$ are the thickness of C_1, C_2 after the cut. We only need to show that the number of rectangles cut by l, denoted by p, satisfies $p \geq r_1 + r_2$ to maintain the invariant. Since $t(C_1)' + t(C_2)' = m - 1$, and $p \geq t(C_2)$, $m - 1 \geq t(C_1)$, the invariant still holds.

When we choose $m = k$, we obtain a lower bound of $2n - o(n)$ on the size of any BSP

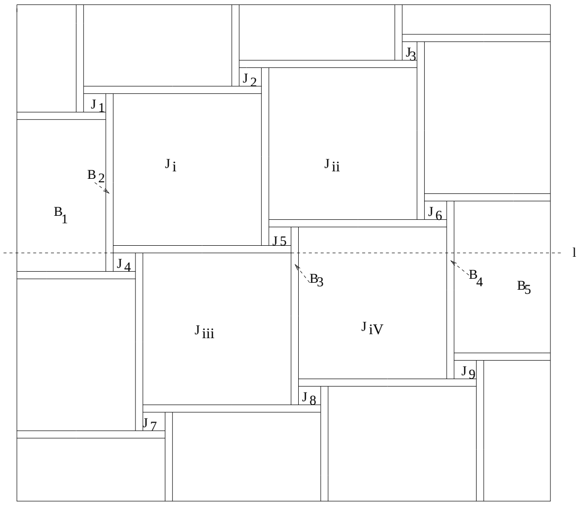

Figure 6: Example of extending a line in the center of the region

for the configuration when $k \to \infty$:

$$\frac{c_n + n - 1}{n - 1} = \frac{(k - \frac{1}{2})(k^2 + (k-1)^2)}{2mk(k+1) - 1} \to 2.$$

By the previous analysis, we obtain the following Lemma:

Lemma 2. *There exists a set S of n rectangles in tiling case with the property that any binary space auto-partition of S has size at least $2n - o(n)$.*

Let S be a set of rectangles which is a tiling. It was proved in [1] that by applying the algorithm provided in [2], an upper bound of $2n$ was obtained for any set of rectangles in tiling case. So we have established that the upper bound is tight.

Remark The construction of $k \times k$ m grid is a modification of the construction of paper [3], where it was used to establish the tightness of the upper bound $2n$ for the size of BSP of a set of n disjoint axis-parallel line segments.

3 Lower bound for the size of BSP of rectangles in packing case

For a set of rectangles in packing case, the best known upper bound is $\frac{9}{4}n - o(n)$ [3]. We shall use the same configuration as in [3], but analyze it in more detail to obtain the better lower bound $\frac{7}{3}n - o(n)$.

Consider a configuration as in Figure 8: there are $k \times k$ squares packing together, and among 4 squares as in Figure 8, among the squares 1, 2, 3, 4, there is an empty space called

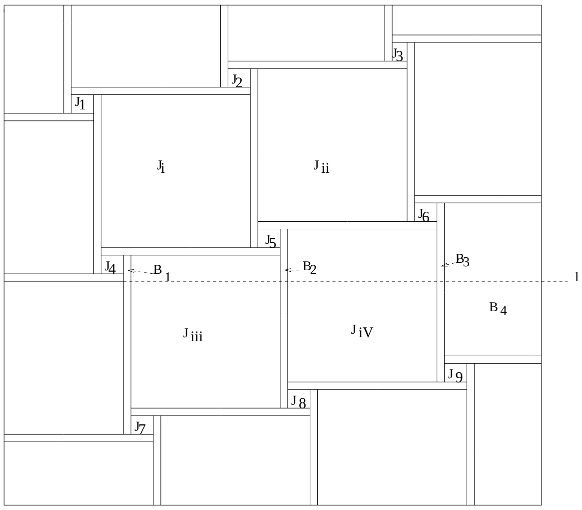

Figure 7: Example of extending a line anchored aside

cell. We call this configuration $k \times k$ *array of squares* or $(k-1) \times (k-1)$ *array of cells*. This configuration is similar to $k \times km$ grid, but it is different. Because $k \times k$ array of squares is to pack k^2 squares together, and $k \times km$ grid is to partition a rectangle.

In a $k \times k$ array of squares, each cell corresponds to a cycle in G_S. There are $(k-1)^2$ cells totally. Consider any cut l of any BSP, as in Figure 8, there are three parts of the cut: (i) border part a_1, which is on the border of some square. The length of a_1 is $1 + \epsilon$; (ii) aligned part b_1, b_2: they are adjacent to the border part whenever there is a cell with a_1 as a side. There are at most two aligned parts, each with length 1; (iii) the middle part, which is the rest of the line l. The middle part cuts through squares such that the cut line is not adjacent to any side of the squares or cells.

Let us denote c_n to be the number of extra rectangles generated in the partition, then the size of the BSP is $n + c_n$. By Lemma 1, each cell will create at least one extra cut which is counted by the aligned parts of cuts. We need to count the length of middle parts. We denote $T(i, j)$ to be the length of the middle parts of the cuts of a BSP for our configuration with $i \times j$ array of cells. A lower bound for the size of a BSP tree for a $k \times k$ array of squares is:

$$n + c_n = k^2 + (k-1)^2 + T(k-1, k-1).$$

We shall prove the following:

Lemma 3. *For $i, j \geq 3$, $T(i, j) \geq \frac{(i-2)(j-2)}{3}$.*

Proof. It is easy to check that $T(3, 3) \geq 1$, so the base is true.

Without loss of generality, let us assume the orthogonal cut splits the rectangle containing the $i \times j$ array of cells into 2 subrectangles containing $i \times j_1$ and $i \times j_2$ arrays of cells, with $j - 1 = j_1 + j_2, j_1 \leq j_2$. Then

$$T(i, j) \geq T(i, j_1) + T(i, j_2) + i - 2$$

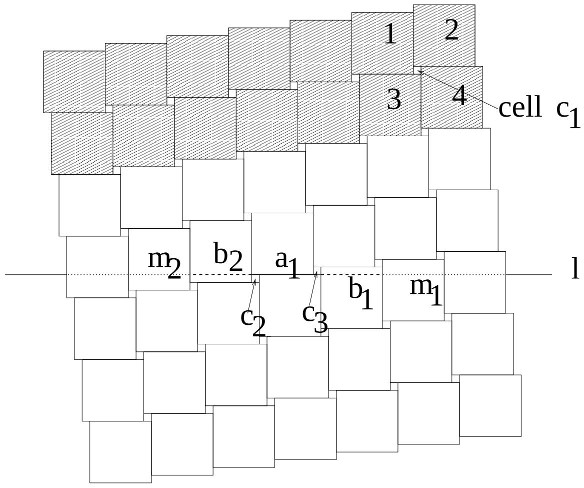

Figure 8: A set of unit squares in packing case

There are three cases:

Case 1. $j \leq 5$. Then

$$T(i,j) \geq i-2 \geq \frac{(i-2)(j-2)}{3}$$

Case 2. $j \geq 6, j_1 \leq 2$. Since $j_2 \geq j-3 \geq 3$, we can use the inductive bound on $T(i,j_2)$.

$$T(i,j) \geq T(i,j_2)+i-2 \geq T(i,j-3)+i-2$$

$$\geq \frac{(i-2)(j-5)}{3}+i-2 = \frac{(i-2)(j-2)}{3}$$

Case 3. $j \geq 6, j_1, j_2 \geq 3$. Using the inductive bounds on both terms:

$$T(i,j) \geq T(i,j_1)+T(i,j_2)+i-2$$

$$\geq \frac{(i-2)(j_1-2)}{3}+\frac{(i-2)(j_2-2)}{3}+i-2$$

$$= \frac{(i-2)(j_1+j_2-1)}{3} = \frac{(i-2)(j-2)}{3}$$

$\square$

So a lower bound on the size of the BSP is:

$$k^2+(k-1)^2+\frac{(k-3)^2}{3} = \frac{7n}{3}-o(n).$$

Lemma 4. *There exists a set S of n interior disjoint rectangles with the property that any binary space partition of S has size at least $\frac{7}{3}-o(n)$.*

References

[1] P. Berman, B. DasGupta, and S. Muthukrishnan. On the exact size of the binary space partitioning of sets of isothetic rectangles with applications. *SIAM Journal of Discrete Mathematics*, 15(2):252–267, 2002.

[2] F. d'Amore and P. Franciosa. On the optimal binary plane partition for sets of isothetic rectangles. *Information Proc. Letters*, 44:255–259, 1992.

[3] A. Dumitrescu, J. Mitchell, and M. Sharir. Binary space partitions for axis-parallel segments, rectangles, and hyperrectangle. In *Proc. 17th ACM Symp. On Computational Geometry*, pages 141–150. 2001.

[4] H. Fuchs, Z. Kedem, and B. Naylor. On visible surface generation by a priori tree structures. *Comput. Graph.*, 14(3):124–133, 1980.

[5] V. H. Nguyen and P. Widmayer. Binary space partitions for sets of hyperrectangles. *LNCS*, page 1023, 1995.

[6] M. Paterson and F. Yao. Optimal binary space partitions for orthogonal objects. *J. Algorithms*, 13:99–113, 1992.

[7] W. C. Thibault and B. F. Naylor. Set operations on polyhedra using binary space partitioning trees. *Comput. Graph*, 21(4), 1987.