Guillotine Cuts

by
Sahil Mahajan

Under the supervision of Dr. Rajiv Raman

Indraprastha Institute of Information Technology Delhi
July, 2016

Guillotine Cuts

by
Sahil Mahajan

Submitted
in partial fulfilment of the requirements for the degree of
Master of Technology

to

Indraprastha Institute of Information Technology Delhi
July, 2016

Certificate

This is to certify that the thesis titled **Guillotine Cuts** being submitted by **Sahil Mahajan** to the Indraprastha Institute of Information Technology Delhi, for the award of Master of Technology, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations related to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

Dr. Rajiv Raman
14th July, 2016
Indraprastha Institute of Information Technology Delhi
New Delhi 110020

Acknowledgments

I would first like to thank my advisor, Dr. Rajiv, who guided me through some of the darkest times, whose insights I cherish and whose deep interest in the area kindled my own. Secondly, I would like to thank my family, who have always supported me. I would also like to acknowledge everyone who has helped me during the course of this thesis although I am unable to name all of them.

Abstract

In guillotine cut problem, we are given non-intersecting convex sets in the plane and we are allowed straight line cuts. If a cut passes through an object, it *kills* that object. Can one always save a constant fraction or $\Omega(n)$ objects? For the general case, Pach and Tardos answered in the negative. We investigate the case where we have axis-parallel rectangles and only horizontal and vertical guillotine cuts are allowed. $\Omega(n)$ bound here would have interesting implications for the Maximum Independent Set of Rectangles problem - it would imply a constant factor or $O(1)$ approximation for it. Unable to solve the problem in its totality, we discuss some approaches and small results.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

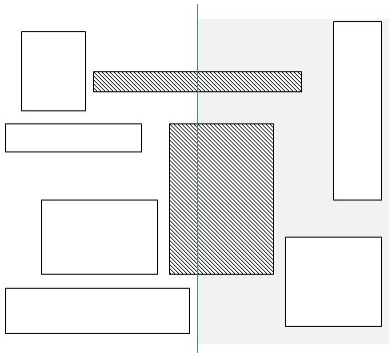


Figure 1.1: Line represents a guillotine gut on set of rectangles. Rectangles shaded dark are *killed* by the line. Initial rectangular piece P, containing whole instance, is divided into two rectangular pieces, P_1 and P_2 lying on the left and the right respectively, by the line.

Imagine a set of pairwise non-intersecting convex sets lying on a sheet of glass and a machine that can perform a straight line cut which divides a given piece of sheet into two pieces. More formally, let P be a rectangular piece in the plane and let a cut, defined by line l, divide P into two pieces i.e. P_1 and P_2(Figure 1.1). This cut is called guillotine cut. If a guillotine cut passes through a convex set, we lose that set. The sets lost are referred to as being *killed* by the cut. We want final pieces such that only one convex set lies on a piece. How many convex sets can one save? Can one always save a constant fraction i.e. $\Omega(n)$ of the them? J. Urrutia [6] first asked this question which Pach and Tardos [1] answered in the negative for the general case. They showed that their exists a family of straight line segments such that one can separate only $O(n^{\log 2/\log 3})$ of them. But for axis-parallel rectangles and for axis parallel cuts, they conjectured that one should be able to save a constant fraction but could only prove the lower bound of $\Omega(n/\log n)$. Moreover, they also proved that one can always separate a constant fraction of convex sets which are *fat* i.e. if the ratio of radius of the maximum circle which can be drawn inside any set to the radius of minimum

circle which contains the set is constant, then one can always save a constant fraction of sets where constant depends upon the *fatness* of the given instance. For a long time it was not known whether one can save a constant fraction even for arbitrary sized axis-parallel squares using axis-parallel cuts. It was recently shown to be true by Abed et al. [2]

Another interesting problem is the maximum independent set of rectangles(MISR) problem which has received a lot of attention lately([3], [5]). We have a set of axis-parallel, possibly intersecting, rectangles and we want to find largest set of pairwise non-intersecting rectangles. One of its application is the label placement problem which was discussed by Agarwal et al. [4]. Think of a map and assume one is given a large number of interesting points on map. Each interesting point has a corresponding axis-parallel rectangular label which contains information about that point. Now these labels can intersect with each other and hence one cannot display all of them. Natural thing to try is to display as many of them as possible such that no two labels intersect which is an instance of MISR problem. Agarwal et al. [4] gave an $O(\log n)$ approximation for the same. Current best known approximations for the unweighted setting is by Chalermsook et al. [5] who gave $O(\log \log n)$ approximation for the same and for weighted setting is by Chan and Har-Peled [7] who gave an $O(\log n / \log \log n)$ approximation for the same. In 2013, Adamaszek and Wiese [3] gave a QPTAS for the general case and a PTAS for the case where all rectangles are *long in at least one dimension* using a new geometric dynamic program.

An $O(1)$ approximation algorithm for MISR has been an open question for a long time. Let OPT be the optimal solution corresponding to a given MISR problem. One way to attack this problem is via structure theorem on OPT. Observe that OPT consist of non-intersecting axis-parallel rectangles. So if vertical and horizontal guillotine cuts can save $\Omega(n)$ rectangles of OPT, we get an $O(1)$ approximation for MISR. Using dynamic programming technique of Adamaszek and Wiese [3], it can be shown that one can obtain optimal sequence of vertical and horizontal guillotine cuts corresponding to non-intersecting axis parallel rectangles in $O(n^5)$ time where by optimal sequence we mean the sequence which ends up saving maximum number of rectangles. So we get a poly-time $O(1)$ approximation algorithm. This was the motivation of Abed et al. [2] to look at this problem.

So can one always save $\Omega(n)$ axis-parallel non-intersecting rectangles using vertical and horizontal guillotine cuts? We tried to solve this problem but unfortunately our efforts were futile. It appears to be notoriously hard. We show some results for some greedy approaches and we also obtained an upper bound on the number of axis-parallel unit length line segments which can be saved using vertical and horizontal guillotine cuts. One can always save at least half - pick all the vertical or horizontal line segments. We are able to show that the bound is tight i.e. there exists family of graphs where one cannot save more than $1/2 + o(1)$ line segments. Abed et al. [2] proved the same for unit squares but there argument uses recursive arguments which are not exactly intuitive. We show very simple argument works for line segments. We also ran some experiments to gain insights into the nature of solution returned by the greedy approach.

This thesis is organised as follows: in chapter 2 we discuss previous work like $\Omega(n/\log n)$ bound on rectangles using guillotine cuts, how to save $\Omega(n)$ squares and connection between guillotine cuts and MISR problem. In chapter 3, we investigate some greedy approaches and prove the upper bound for unit length line segments. In chapter 4, we give results of few experiments we ran. In chapter 5, we conclude.

Chapter 2

Previous Work

From now on whenever we mention objects like - rectangles, squares and line segments, we assume them to be axis-parallel and non-intersecting, unless otherwise mentioned. Also whenever we say guillotine cuts, we mean vertical and horizontal guillotine cuts. We will first give overview of $\Omega(n/\log n)$ lower bound for rectangles using guillotine cuts. We will discuss the approach of Agarwal et al. [4] which they gave for approximation of MISR. It is directly extendible to guillotine cut problem as well. Then we will discuss the proof given by Abed el at. [2] which shows that one can always save $\Omega(n)$ squares using guillotine cuts. Then we will briefly see how using geometric dynamic program of Adamaszek and Wiese [3], it is possible to get a $O(1)$ approximation to the MISR problem if guillotine cuts can save $\Omega(n)$ rectangles. This has been discussed by Abed et al. [2].

2.1 Saving $\Omega(n/\log n)$ rectangles using guillotine cuts

Pach and Tardos [1] discuss how to save $\Omega(n/\log n)$ rectangles using guillotine cuts. But here we discuss the algorithm given by Agarwal et al. [4] which gives $O(\log n)$ approximation for MISR as it is much more intuitive and can be extended directly to show that one can always save $\Omega(n/\log n)$ rectangles using guillotine cuts. Input instance contains a set of non-intersecting rectangles $\mathcal{R}$ where $|\mathcal{R}| = n$. The algorithm, corresponding to guillotine cuts and rectangles, is as follows:

1. Pick median x-coordinate and name it x_{mid}.

2. Let the vertical line through x_{mid} be l. Now l divides initial set of rectangles R into three sets - R_1 which lie to the left of l, R_2 which lie to the right of l and R_{12} which intersect with l.

3. Now compute OPT both R_1 and R_2 recursively using this approach. Let it be named G_1 and G_2 respectively.

4. If $|R_{12}| \geq |G_1 + G_2|$, consider it as the solution. Otherwise $|G_1 + G_2|$ is our solution.

We will now prove the $\Omega(n/\log n)$ bound using induction on the number rectangles in an instance. Let it be true for instances containing $k < n$ rectangles. Let G be the solution corresponding to the whole instance. Notice that if $|R_{12}| \geq n/\log n$ rectangles, we are done as we can save all of them using horizontal guillotine cuts. Otherwise, consider the solutions G_1 and G_2 corresponding to R_1 and R_2. By induction, we know that $G_1 > |R_1|/\log(n/2)$ and $G_2 > |R_2|/\log(n/2)$. Now, if $R_{12} < n/\log n$,

$$\begin{aligned}
|G| &= \frac{|R_1|}{\log(n/2)} + \frac{|R_2|}{\log(n/2)} \\
&= \frac{|R_1| + |R_2|}{\log(n/2)} \\
&= \frac{|\mathcal{R}| - |R_{12}|}{\log n - 1} \\
&> \frac{n - n/\log n}{\log n - 1} \text{ (as } R_{12} < n/\log n) \\
&= \frac{n}{\log n}
\end{aligned}$$

2.2 Saving constant fraction or $\Omega(n)$ squares

Here we discuss the proof presented by Abed et al. [2] which shows that one can always save a constant fraction i.e. $\Omega(n)$ squares of arbitrary size using vertical and horizontal guillotine cuts. We will only discuss the unweighted case, although the proof is extendible to weighted case as well. Input instance contains a set of non-intersecting squares $\mathcal{R}$ where $|\mathcal{R}| = n$. Let P be a rectangular *piece* in the plane and let a vertical or horizontal line l divide P into two pieces i.e. P_1 and P_2. A rectangular piece is just an axis-parallel rectangle in the plane, possibly containing, overlapping or intersecting with one the squares $R \in \mathcal{R}$. A cutting strategy is represented by a binary tree $\mathcal{T}$ where each non-leaf node $v \in V(\mathcal{T})$ is represented by a piece P_v and a line l_v where $P \cap l_v$ gives us two pieces P_{v_1} and P_{v_2} where v_1 and v_2 are the children of v. For set of squares $\mathcal{R}$, we say that $\mathcal{T}$ separates $\mathcal{R}$ and $\mathcal{R}$ is guillotine separable if

- Piece P_r belonging to root node of $\mathcal{T}$ contains all the squares $R \in \mathcal{R}$
- For each non-leaf node v, l_v does not intersects any square in P_v

- Each leaf node contains exactly one square $R \in \mathcal{R}$

One crucial observation here is that if there exists a cutting strategy $\mathcal{T}$ for a set of squares $\mathcal{R}$ such that at each non-leaf node $v \in V(\mathcal{T})$, line l_v intersects at most k squares in P_v such that each resulting piece has at least one complete square, then $\mathcal{R}$ is guillotine separable where at least $n/(k+1)$ squares survive. Call such a cutting strategy k-good. The proof depends on the fact that a proper binary tree $\mathcal{T}$ having x nodes and y leaf nodes satisfies $x = 2y$ - 1. Therefore

$$\begin{aligned}\text{Squares saved} &= \text{Number of leaf nodes in tree} - 3 \times \text{Number of internal nodes} \\ &= y + k \cdot (x - y) \\ &= y + k \cdot (y - 1) \\ &\leq (k+1) \cdot x\end{aligned}$$

The proof overview is as follows: we first use hierarchical decomposition to remove a constant fraction of squares, then we remove some more squares so that we can define a k-good cutting strategy on the pruned instance. As a k-good sequence further only destroys a constant fraction, we are able to save a constant fraction of squares in the end.

We will now discuss the details. First we place a multi-level grid lines on our input graph which will be used to define our cuts later. A square $R \in \mathcal{R}$ belongs to level-i if its side length belongs to the interval $(N/2^{i+1}, N/2^i]$, where N is a positive integer used for normalization so that level-0 contains the largest squares. We also pick up two numbers $x, y \in [0, N)$ to define a random shift of the grid. For level-i, vertical grid lines are drawn at intervals x, $x + N/2^i$, $x + 2 \cdot N/2^i$.... and horizontal lines at intervals y, $y + N/2^i$, $y + 2 \cdot N/2^i$.... where we wrap around the lines appropriately or think of a very large grid placed over squares so that wrapping is not required. A grid cell at level-i defined by consecutive horizontal and vertical line segments belonging to level $i-1$ and therefore has side length $2 \cdot N/2^i$. We remove all squares $R \in \mathcal{R}$ which intersects with lines at level lower than themselves. Let $\mathcal{R}_1$ be the set of squares remaining. Probability that a square $R \in \mathcal{R}$ having side length l_R is also present in $\mathcal{R}_1$ is at most $(1 - \mu_R)^2 \geq 1/4$ where $\mu_R = l_R 2^{i-1}/N$.

Now we sample squares $R \in \mathcal{R}_1$ to get a set $\mathcal{R}_2$. Notice that a grid cell C at level-i can contain at most 9 squares at level-i (grid cell at level i is defines by lines at level $i-1$. Let these square be represented by R_1^C. We want to retain at most one of these squares. For easier understanding, one can imagine removing all but one of the level-i 9 squares present in the grid cell at level-i but then we would end up saving only $\frac{1}{36}$ squares from $\mathcal{R}$ in $\mathcal{R}_2$. So we sample squares a bit more carefully with probability

$$\frac{1}{(1-\mu_R)^2 \cdot M_c}, \text{ for } M_C = \sum_{S \in \mathcal{R}_1^C} \frac{1}{(1-\mu_S)^2}$$

Now probability that a square $R \in \mathcal{R}$ is also present in $\mathcal{R}_2$ is

$$\begin{aligned}\mathbf{Pr}[R \in \mathcal{R}_1] \cdot \mathbf{Pr}[R \in \mathcal{R}_2 | R \in \mathcal{R}_1] &= (1-\mu_R)^2 \cdot \frac{1}{(1-\mu_R)^2 \cdot M_C} \\ &= 1/M_C\end{aligned}$$

It can be proved that $M_C \leq 81/4$ which we skip here for readability. Intuitively, one can think of distribution used to generate $\mathcal{R}_2$ it as follows: if probability of losing a level-i square while getting $\mathcal{R}_1$ is large, then not many such squares can be inside a grid cell at level-i. Whereas if probability of losing a level-i square while getting $\mathcal{R}_1$ is small, then one can have many(at most 9) such squares inside grid cell at level-i. Probability distribution we use to get $\mathcal{R}_2$ from $\mathcal{R}_1$ captures this notion and hence the bound we get is $\frac{4}{81}$ instead of crude $\frac{1}{36}$.

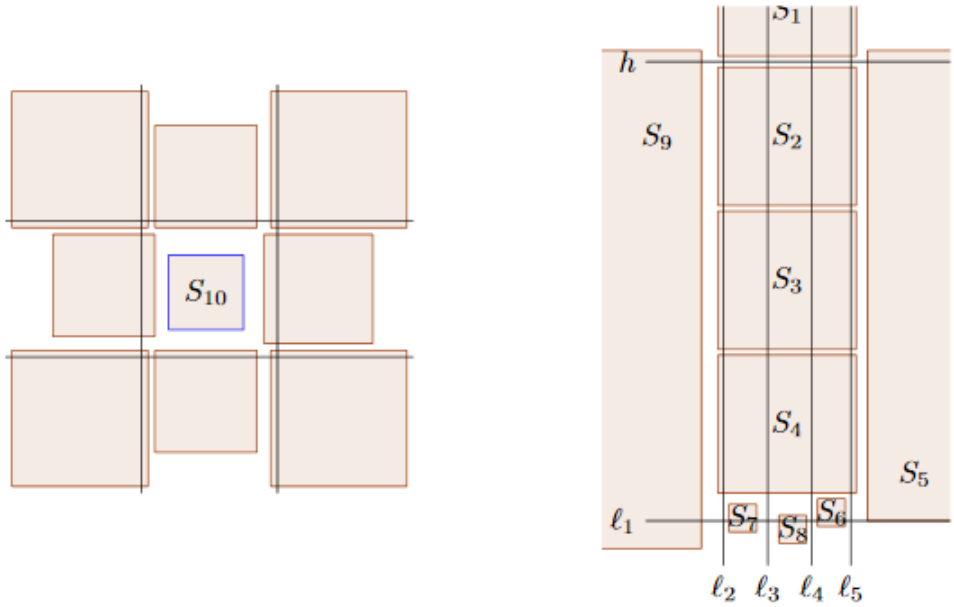


Figure 2.1: Cases arising while drawing k-good tree over sampled set R_2. Taken from [2]

Now we prove that there exists a 3-good tree for the sampled set $\mathcal{R}_2$. Given initial piece P_0, we show how to find a 3-good separator iteratively for any piece P' i.e. we give a constructive proof. Two cases arise(See figure 2.1). First, when P' contains at least 10 squares. Arrange squares $R \in P'$ in non-increasing order $S_1, S_2, \cdots$. Let S_{10} be at level i and consider grid cell C at level i containing S_{10}. By definition of sample R_2, C contains at most one square $R \in \mathcal{R}$ inside it at level i. Sides of C can contain at most 8 squares as all these are at a level lower than i and size length of C is not big enough to accommodate more of them. Therefore, one of the grid lines corresponding to cell cuts at most 3 squares and separates S_{10} from one the squares in $S_1, \cdots, S_9$ as at least one of them lies completely outside C.

When P' contains 9 or less squares, we order the squares by non-increasing order of their bottom y-coordinate. Now observe line l that touches bottom boundary of S_5. If it intersects with at most 3 squares out of S_6, S_7, S_8, S_9, we are done. Otherwise, observe that l passes though S_6, S_7, S_8, S_9 and touches bottom of S_5. Now there are 4 combinatorially different vertical lines possible such that each of them partitions S_5, S_6, S_7, S_8, S_9 in a different way. If any of these lines intersects with at most three of S_1, S_2, S_3, S_4, we are done. Else, observe horizontal line touching bottom of S_1. This line would intersect with at most two squares from S_5, S_6, S_7, S_8, S_9.

So in all we save at least $4/81$ squares in $\mathcal{R}_2$ and $1/4$ to get a k-good tree and hence are able to save $1/81$ fraction of squares.

2.3 Guillotine cuts and Maximum Independent Set of Rectangles

Here we will discuss how using dynamic program by Adamaszek and Wiese [3], if one could save $\Omega(n)$ rectangles using guillotine cuts, one could get $O(1)$ approximation to the MISR problem. We will first discuss the how to get optimal sequence of guillotine cuts for, possibly intersecting, rectangles. We would again like to have final pieces such that only one rectangle lies on a piece. We will discuss the unweighted case but it is directly extendible to weighted case as well. Let $\mathcal{R}$ be the set of, possibly overlapping, axis-parallel rectangles. Let $|\mathcal{R}| = n$ where n is a positive integer. Observe that w.l.o.g. we can draw $\mathcal{R}$ in a grid of size $[2n \times 2n]$ such that for all rectangles $R \in \mathcal{R}$, end points of R have integer coordinates in $\{0, 1, ...2n\}$(to see it more clearly, consider the vertical projection of rectangles on x-axis. It forms an interval graph. Now assign leftmost point coordinate 0, next point coordinate 1 and so on. Now all the end points of intervals are in $\{0, 1, ...2n\}$ and moreover all the intersection relations with respect to vertical cuts are still same and can be covered by vertical cuts having vertical coordinates in $\{0, 1, ...2n\}$. Similar case for the horizontal cuts). We assume that input instance of MISR contains no two rectangles R, R' such that $R \subseteq R'$ and all rectangles are considered as open sets. Algorithm considers every rectangular piece inside the grid with end points in $\{0, 1, ...2n\}$. Notice that there are $O(n^4)$ such rectangular pieces. To see it consider the fact that intersection of a pair of vertical and a pair of horizontal lines uniquely identifies a rectangular piece in the grid.

For computing an optimal solution corresponding to a rectangular piece, the dynamic program works as follows: If a piece contains no rectangle $R \in \mathcal{R}$ inside it or a piece is completely inside a rectangle $R \in \mathcal{R}$, we assign its solution as empty set. Else if a rectangular piece exactly coincides with a rectangle $R \in \mathcal{R}$, we assign R to be its solution. Otherwise, we try to divide a piece in all possible ways using vertical or horizontal guillotine cuts such that vertical or horizontal coordinate lies in $\{0, 1, ...2n\}$. We can do this because all rectangles have end points in $\{0, 1, ...2n\}$(see above). Let a guillotine cut divide a piece P into pieces P_1 and P_2. For every possible cut, we add the

solutions of corresponding P_1 and P_2 and we set the maximum over all cuts as the solution of piece P. There are $O(n)$ such cuts possible and as total number of rectangles are $O(n^4)$, this DP runs in $O(n^5)$.

To see why this DP returns an optimal solution, consider induction over rectangular pieces according their size. For a piece P of size $m \times n$, assume DP computes optimal solution for all pieces smaller that P i.e. none of their dimension is greater than P and at least of the dimension is less than P. As we consider all the cuts possible for P, we would consider one corresponding to the optimal cut and as we know DP already computes optimal solution for the pieces smaller than P, we are done.

Now if one could always save a constant fraction of non-overlapping axis parallel rectangles using guillotine cuts, that would imply $O(1)$ or constant size approximation to the MISR problem. It is because optimal solution of the MISR problem would consist of pairwise non-intersecting rectangles and if guillotine cuts could save a constant fraction of them, above DP would return a solution that is at least as good.

Chapter 3

Guillotine Cuts: Things tried and results

Implications of saving a constant fraction of axis-parallel non-intersecting rectangles using guillotine cuts encouraged us to try and solve it. Unable to directly attack the problem, we tried using obvious approaches like trying to see if simple greedy algorithms work or not. Surprisingly, we are unable to say much about even greedy approaches. Arguing about guillotine cut problem seems hard as every guillotine completely destroys structure of the problem and gives two disconnect pieces. It is also hard to think about instances containing more than a handful of rectangles manually and recursive structures also do not seem to help much.

We also tried to think about if one can say something about structure of such graphs. One natural way is to think of notion of vertical or horizontal non-separability i.e. two rectangles $R_1, R_2 \in \mathcal{R}$ are vertically non-separable if there exists no vertical line l which divides the piece P containing R_1 and R_2 into two pieces P_1 and P_2 such that, w.l.o.g., R_1 lies in P_1 and R_2 lies in P_2. Similarly for horizontal non-separability and horizontal lines. This way one gets two interval graphs, one corresponding to vertical non-separability and other to horizontal non-separability both having same set of vertices with different intersection relations. Question then becomes when two different interval graphs having same set of vertices with possibly different intersection relations can be drawn as axis-parallel non-intersecting rectangles in the plane. The solution turns out to be quite uninteresting - if no two vertices are both horizontal and vertically non-separable, then one can always draw the two interval graphs as non-intersecting axis-parallel rectangles lying in the 2-d plane. This approach again fails to tell us about any property of the graphs which we can use for our problem.

We also thought about what can be said about non-intersecting horizontal or vertically line segments - essentially rectangles with no width. For unit case, we are able to prove that there exists family

of graphs such that one cannot save more than $\frac{1}{2} + o(1)$ segments thereby making the bound for unit length line segments tight. Abed et.al. proved the same for unit squares. We now discuss greedy approaches and other results in detail.

3.1 Greedy: Cut with smallest size

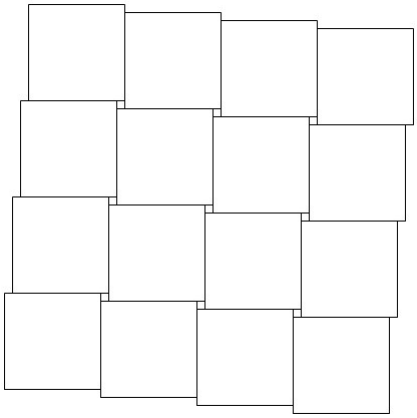


Figure 3.1: A square grid

We first consider the naive approach - a guillotine cut whose size is smallest i.e. which kills the least amount of objects. We show that that this approach fails for squares and rectangles. Consider a grid of squares. A cut just cutting a square on the extremities will kill one square with all remaining squares sitting on the same side. We can keep on finding such a cut and thereby reducing total number of squares left until only one square is left. So for arbitrary size grid, we can end up with only one square and hence this greedy cannot save a constant fraction.

3.2 Greedy: Cut where ratio of objects killed to the objects on smaller side is smallest

Here we are trying to get *most balanced guillotine cut which cuts the least.* It can be easily show that this approach is not optimal for both squares and rectangles. Consider matrix of $3 \times k$ squares where $k = (2^n - 1)$.

A horizontal cut through the middle row would lose one-third squares and save two-third. To bound the greedy, we can construct a recurrence relation. Let $f(n)$ represent number of squares cut in a $3 \times k$ grid. Then

$$f(k) = 2f(\frac{k-1}{2}) + 3$$

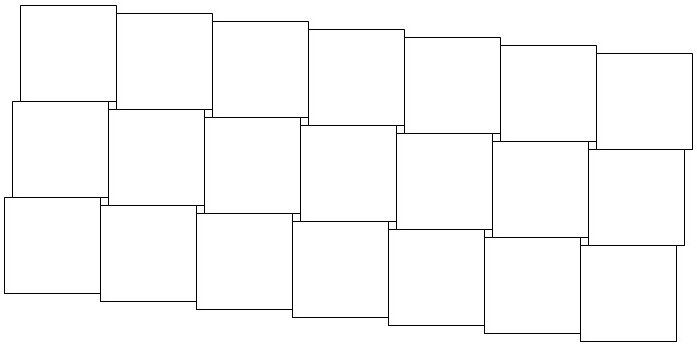
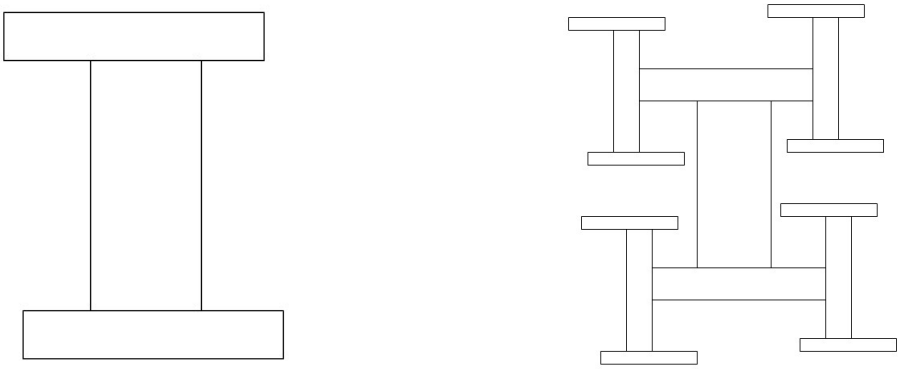


Figure 3.2: A $[3 \times 7]$ matrix of squares

is the required recurrence relation with $f(1) = 1$. Solving it, one gets $f(N) \geq \frac{N}{2} - O(1)$ where N is the total number of squares in our instance.

But we are currently neither able to prove whether this greedy approach always saves a constant fraction or can find a family of examples where it cannot save constant fraction of rectangles. Even using recursive structures does not seem to help. For eg - consider structure in figure 3.3(a) and recursively apply this structure to the rectangle present on the extremity. Cutting strategy where one always cut the middle rectangle corresponding to recursive substructure give us a k-good tree with $k = 1$ and hence we end up saving half rectangles.



(a) Basic recursive structure (b) Recursive structure applied twice.

Figure 3.3: Recursive structure

To try to see what this greedy exactly does and how bad it is compared to the optimal solution, we ran some experiments which we will discuss in more detail in the next chapter.

3.3 Upper bound for unit length line segments

In this section, we prove the existence of the family of graphs containing unit length horizontal and vertical line segments where one cannot save more than $\frac{1}{2} + o(1)$ segments using guillotine cuts. Graph can be stretched in one dimension i.e. one of the vertical or horizontal line segments can be longer than the other but still they cannot be of arbitrary sizes and hence this proof holds for only unit length. Abed et al. [2] proved the same for unit squares. But there argument is not intuitively clear. The proof uses the structure depicted in figure 3.1 i.e. a grid on squares. They define sub-wall as the set of squares completely contained inside a rectangle drawn over this grid. Then they define the following lemma the following using induction

$$S(q) \leq \left\lceil \frac{2A(q) + P(q) - 4}{4} \right\rceil$$

where $S(q)$ is the number of squares that can be saved using guillotine cuts, $A(q)$ is the number of squares a sub-wall q contains and $P(q)$ is the outer perimeter of the sub-wall and prove it using induction on $A(q)$. We show that the proof for line segments in much more simpler. The proof depends upon the following basic structure, a *square* structure, for the line segments

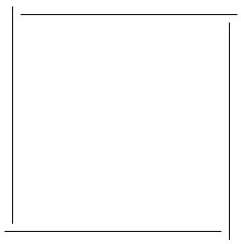


Figure 3.4: A *square* structure

Observe that one will always have to lose one of the line segments and at most 3 can be saved. Intuitively proof can be seen as follows: one can form a *chain* of such *square* structures, sharing segments among them and then extend the idea to form a grid of *square* structures. In a grid one can imagine each *square* structure sharing its two line segments with other *square* structures. So we have approximately twice the number of line segments as the *square* structures. For each *square* structure, we will have to lose one line segment and thus we will have kill at least half of the line segments. Now we will show it formally.

We say an instance containing 'square' structures is valid if it has following properties:

1. All *square* structures are connected that is if there is more than one such structure, then each structure shares at least one line segment with another structure.

2. Moving in clockwise direction, gap between two consecutive line segments of the square structure is $< 1/(2 \times$ Number of line segments$)$

Theorem 1*: Given an instance of vertical and horizontal unit length line segments, one can save at most* $1/2 + o(1)$ *line segments using vertical and horizontal guillotine cuts.*

To prove above theorem, we would first introduce folllowing lemma:

Lemma 2*: Given a valid instance with square structures, every guillotine cut, that cuts a line segment, kills as many line segments as the* square *structures it destroys.*

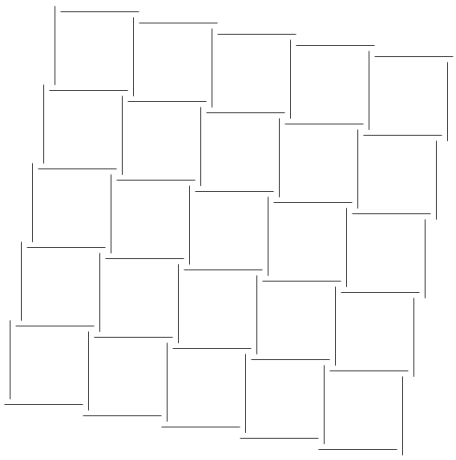


Figure 3.5: A grid of *square* structures

Proof of theorem 1: Assuming above lemma is true, consider a grid of these *square* structures (see figure 3.5) having n^2 squares and $2n^2 + 2n$ line segments with gaps in the *square* structure less than $1/(2 \times$ Number of line segments$)$. This grid is a valid instance. Moreover, every sequence of guillotines cuts produces instances that are valid as *square* structures in new instances are always connected and gap size remains same although number of line segments decreases. Therefore only $n^2 + 2n$ line segments can be saved at most fulfilling $\frac{1}{2} + o(1)$ bound. $\square$

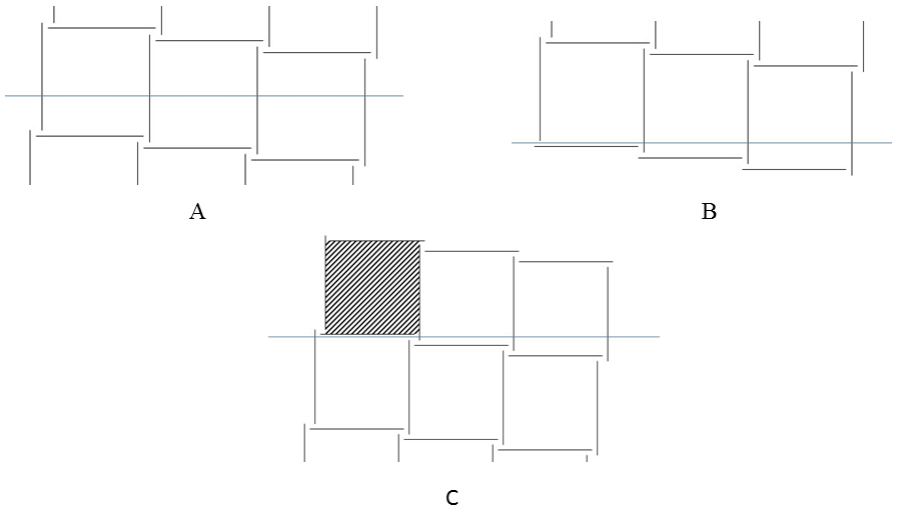


Figure 3.6: All three cases which can arise

Proof of lemma 2: Consider any horizontal or vertical cut that destroys a *square* structure. Let total number of *square* structures destroyed by cut be k. As the gap in the *square* structure is $< 1/(2 \times$ Number of line segments), any cut can only pass through one such gap. Now three cases arise (See figure 3.6). Case A arises when no gap is hit at all. In this case, guillotine cut hits $k + 1$ line segments. For the cut at the extremity, it is possible to destroy one *square* structure by hitting only one line segment but the bound still holds. Case B arises when gap is present in the extremities. In this case cut hits k line segments. Case C arises when gap is inside the structure of squares. If line segment hit just after passing through gap is shared by two square structures, then we hit k line segments. Otherwise, $k + 1$ line segments. So in all cases we lose as many line segments as the number of square structures. Hence proved. □

Chapter 4

Some experiments

To try to understand the way greedy works, we decided to run some experiments. We randomly generated non-intersecting objects(rectangles, squares and line segments) and computed rectangles saved by optimal sequence of guillotine cuts using dynamic program discussed in section 2.3 and rectangles saved by greedy approach discussed in section 3.2. Observe that although dynamic program in section 2.3 is discussed with respect to MISR problem, it directly extends to guillotine cut problem as well as input to the guillotine cut problem can be seen as special case of input to the MISR problem. We wanted to see how many rectangles can be save by both of the approaches and how they compare to each other. As in the case of DP, the objects lie in $[2n \times 2n]$ grid.

To generate random objects, we try two different approaches. In general, we take two random points in case of rectangle, two random points and the shorter side as the side length in case of squares and a random point along with length and orientation for line segment. In the first approach, we randomly generate an object as described above and place it on the grid if it does not intersects with any previously placed object. Else, we drop that particular object and again keep on continuing till we have the required number of objects in our instance. If even after large number of tries we are unable to place an object, we drop the current instance and again generate an instance. We call this approach basic.

In the second approach, the *tetris* method, we ignore the y-coordinate of the point/s closer to x-axis and let the object *fall* from top the grid, as in the game tetris, and stop if moving it further down would result in an intersection or bottom is reached. As in previous case, if object does not fit i.e. there is not enough space to bring object completely down, then we again generate the object and if the same persists for large number of tries, we regenerate the instance.

We also consider the weighted setting as algorithm directly scales for it. Weight for each rectangle ranged from 0 to 199. We did 30 iterations for following object sizes: 10, 50 and 100.

Experiments were ran on a machine having Intel Core™ i5-2410M processor and 8 GB RAM. For implementation Java SE 8 was used.

As for dynamic program implementation, recursive approach turns out to be significantly faster than iterative approach. Observe that if a rectangular piece has less than or equal to three rectangles inside it, one can always save all the rectangles and hence does not need to compute solution for that piece. Recursive approach can use this fact and speed up obtained is significant. For 100 objects, recursive approach takes about 15 minutes to compute the solution whereas iterative approach takes about 45 minutes.

There are several downsides to the experimental way. First, it is currently not known how to generate randomly distributed non-intersecting objects(rectangles, squares, line segments). Both of our approached do not provide any theoretical bound. Second, dynamic program requires $O(n^5)$ time and $O(n^4)$ space which is infeasible even for as few as 1000 rectangles. Hence we can only test small instances. Last, even for small instances having 100 rectangles, solving one instance requires about 15 minutes of time. Hence testing large number of instances is also not feasible. Keeping these downsides in mind would help one better to interpret any data.

Objects	Random Method	Saved by optimal	Saved by Greedy	Total Objects
Rectangles	Basic	9.8	9.8	10
Rectangles	Tetris	9.87	9.87	10
Squares	Basic	9.97	9.97	10
Squares	Tetris	9.97	9.97	10
Line segments	Basic	10	10	10
Line segments	Tetris	10	10	10

Table 4.1: Objects per instance = 10, Unweighted

Objects	Random Method	Saved by optimal	%	Saved by Greedy	%	Total Weight
Rectangles	Basic	993.13	99.49	991.83	99.36	998.16
Rectangles	Tetris	1006.77	99.76	1006.77	99.76	1009.16
Squares	Basic	1041.77	99.89	1041.77	99.89	1042.94
Squares	Tetris	1016.4	100	1016.4	100	1016.4
Line segments	Basic	984.87	99.9	983.3	99.74	985.87
Line segments	Tetris	981.2	99.67	979.53	99.50	984.43

Table 4.2: Objects per instance = 10, Weighted

Objects	Random Method	Saved by optimal	Saved by Greedy	Total Objects
Rectangles	Basic	47.07	46.4	50
Rectangles	Tetris	48.37	48.07	50
Squares	Basic	49.13	49.0	50
Squares	Tetris	49.37	49.37	50
Line segments	Basic	49.47	49.2	50
Line segments	Tetris	49.53	49.47	50

Table 4.3: Objects per instance = 50, Unweighted

Objects	Random Method	Saved by optimal	%	Saved by Greedy	%	Total Weight
Rectangles	Basic	4767.13	96.83	4743.1	96.34	4922.8
Rectangles	Tetris	4826.96	98.27	4796.7	97.66	4911.63
Squares	Basic	4969.3	99.32	4962.93	99.20	5003.17
Squares	Tetris	4986.7	99.27	4979.46	99.12	5023.46
Line segments	Basic	4858.46	99.6	4819.1	98.79	4878.0
Line segments	Tetris	4953.17	99.39	4924.5	98.17	4983.43

Table 4.4: Objects per instance = 50, Weighted

Objects	Random Method	Saved by optimal	Saved by Greedy	Total Objects
Rectangles	Basic	92.5	90.36	100
Rectangles	Tetris	96.7	95.56	100
Squares	Basic	97.6	97.2	100
Squares	Tetris	98.73	98.64	100
Line segments	Basic	99.13	98.96	100
Line segments	Tetris	98.8	98.27	100

Table 4.5: Objects per instance = 100, Unweighted

Objects	Random Method	Saved by optimal	%	Saved by Greedy	%	Total Weight
Rectangles	Basic	9413.63	95.58	9244.24	93.86	9848.74
Rectangles	Tetris	9688.4	98.04	9605.3	97.20	9881.34
Squares	Basic	9754.47	98.59	9729.1	98.34	9893.07
Squares	Tetris	10052.13	99.50	10040.5	99.39	10102.24
Line segments	Basic	9913.76	99.17	9810.9	98.14	9996.34
Line segments	Tetris	9912.63	99.33	9872.4	98.92	9979.92

Table 4.6: Objects per instance = 100, Weighted

Firstly, one thing which can be observed from the tables is that *tetris* approach for random rectangles produces poor instances i.e. we are able to save too many. Secondly, when the number of objects in an instance is small, it appears shape of objects does not really matter. As number of

objects began increasing, it appears rectangles are hardest to save followed by squares and line segments. Thirdly, it seems that greedy's performance is consistently close to the optimal.

We also ran instances containing unit squares, squares and rectangles with large area. Unit squares had side length 14 and area equal to 196 units, squares had side length from 13 to 17 and area from 169 to 289 whereas area for rectangles ranged from 151 to 300. We only ran it for basic method as *tetris* method would not work with unit squares and keeping previous data in mind, we only ran it for case where number of object = 100. Here are the results

Objects	Random Method	Saved by optimal	Saved by Greedy	Average Area
Rectangles	Basic	93.9	92.5	252.73
Squares	Basic	88.04	84.93	215.26
Unit Squares	Basic	89.8	86.94	196

Table 4.7: Number of objects = 100, Large objects(area large), Unweighted

Now this result came out to be opposite of previous results - large unit squares and squares are more difficult to save than large rectangles even when average area of rectangles is larger. Result for large rectangles is even poorer than result for rectangles in general. So to say anything about objects in general seems hard. Maybe our random instances are not good or we are looking at the from a completely wrong way. Large squares also seems to be worst for greedy - it's difference with the optimal is largest for them.

We also wanted to see if there exist cases with small number of rectangles where one cannot save more than half of the rectangles to try to potentially prove an upper bound as even that is not known currently. We tested for 2 million instances of size 10 using basic approach of random rectangles generation but at most 2 rectangles were killed in each cases i.e. we were always able to save at least 8 rectangles and hence this also could not help us.

Chapter 5

Conclusion

Question first asked by J. Urrutia still remains open for axis-parallel rectangles with axis-parallel guillotine cuts. We currently even do not know how to possibly think about and solve this problem. Even saying anything about simple greedy strategies seems hard. Experimental results too turned out to be a dead end and lead us nowhere. This problem turned out to be much more difficult than we initially anticipated and we were unable to solve it.

Bibliography

[1] Pach, Jnos, and Gbor Tardos. "Cutting glass." In Proceedings of the sixteenth annual symposium on Computational geometry, pp. 360-369. ACM, 2000.

[2] Abed, Fidaa, Parinya Chalermsook, Jos Correa, Andreas Karrenbauer, Pablo Prez-Lantero, Jos A. Soto, and Andreas Wiese. On guillotine cutting sequences. Vol. 40. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[3] Adamaszek, Anna, and Andreas Wiese. "Approximation schemes for maximum weight independent set of rectangles." In Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on, pp. 400-409. IEEE, 2013.

[4] Agarwal, Pankaj K., Marc Van Kreveld, and Subhash Suri. "Label placement by maximum independent set in rectangles." Computational Geometry 11, no. 3 (1998): 209-218.

[5] Chalermsook, Parinya, and Julia Chuzhoy. "Maximum independent set of rectangles." In Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 892-901. Society for Industrial and Applied Mathematics, 2009.

[6] Jorge Urrutia. Problem presented at ACCOTA96: Combinatorial and Computational Aspects of Optimization, Topology, and Algebra, Taxco, Mexico, 1996

[7] Chan, Timothy M., and Sariel Har-Peled. "Approximation algorithms for maximum independent set of pseudo-disks." Discrete and Computational Geometry 48, no. 2 (2012): 373-392.