Rectangular Partitions: Applications

- Connect all the points labeled 1 together by wires located inside the large rectangle and on the outside of the interior rectangles.

- Do the same for the points labeled 2, labeled 3, and so on.

- Problem is computationally intractable.
Reduce to Channel Routing Problems

- Define a set of channels and reduce the global routing problem into a set of channel routing problems each of which can be solved independently.
Defining Channel Routing Problems

- Replace each interior rectangle by its four corner points.
- Partition a rectangle with interior points so that all the points end up on a partitioning line.
- Then delete the segments inside the interior rectangles to obtain the individual channels.
Replacing Rectangles by Corner Points
Partitioning into Rectangles
Resulting Set of Channels After Deleting Segments Inside Rectangles
Minimum Edge Length

Rectangular Partition

- Let $P$ be a set of points inside rectangle $R$ with height $Y$ and width $X$.
- $RG - P$ problem: Introduce inside $R$ a set of (orthogonal) line segments with least total length that partitions $R$ into rectangles without elements of $P$ inside them.
- $RG - P$ is NP-hard.
- There are several Approximation algorithms for this problem.
- We present a dynamic programming and a divide-and-conquer approximation algorithms.
## Approximation Algorithms

<table>
<thead>
<tr>
<th>Method</th>
<th>Time Complexity</th>
<th>Approx Bound</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic Programming</td>
<td>$O(n^5)$</td>
<td>2 (simple proof)</td>
<td>1.75 (complex proof)</td>
</tr>
<tr>
<td>Problem Transformation</td>
<td>$O(n^3)$</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Divide &amp; Conquer</td>
<td>$O(n \log n)$</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

- First and last algorithm generalizes to any number of dimensions.
- Approx Bound depends on number of dimensions.
- T.C. depends on number of dimensions.
Guillotine Partition

- $E(I)$: Rectangular partition for $I = (R, P)$.
- $E(I)$ has a guillotine cut if there is a line segment in $E(I)$ that partitions $R$ into two rectangles.
- $E(I)$ is a guillotine partition if either $E(I)$ is empty or $E(I)$ has a guillotine cut that partitions $R$ into $R_1$ and $R_2$, and both $E(I_1)$ (edges from $E(I)$ in $R_1$) and $E(I_2)$ (edges from $E(I)$ in $R_2$) are guillotine partitions for $I_1 = (R_1, P_1)$ and $I_2 = (R_2, P_2)$, respectively.
- A guillotine partition is a rectangular partition, but a rectangular partition is not necessarily a guillotine partition.
No Guillotine Cut

Guillotine Cut & Guillotine Partition

Guillotine Cut but not Guillotine Partition
Dynamic Programming

- The minimum edge length guillotine partitioning problem consists of finding a guillotine partition of least total edge length.
- An optimal guillotine partition can be constructed in $O(n^5)$ time via dynamic programming.
Dynamic Programming

"Guillotine Partition"

\[ \text{OPT}(i_2, i_r, \ldots, i_u): \text{Value of optimal solution for problem inside the box (shaded region)} \]

How to compute \( \text{OPT} \)?

\[ \text{OPT}(1, 6, 1, 6) = \min \{ x + \text{OPT}(1, 6, 2, 6), \]
\[ x + \text{OPT}(1, 6, 3, 6), \]
\[ x + \text{OPT}(1, 6, 5, 6), \]
\[ y + \text{OPT}(2, 6, 1, 6), \]
\[ y + \text{OPT}(3, 6, 1, 6) + \text{OPT}(1, 5, 1, 6) \} \]
Approximation Algorithm

- Return a minimum edge length Guillotine Partition as an approximation to the minimum edge rectangular partition.

- Let $G(I)$ be an optimal guillotine partition and $L(G(I))$ be the edge length of $G(I)$.

- Let $E_{opt}(I)$ be an optimal rectangular partition and $L(E_{opt}(I))$ be the edge length of $E_{opt}(I)$.

- Claim: $L(G(I)) \leq 2L(E_{opt}(I))$

- We prove that $L(G(I)) \leq 2L(E(I))$, where $E(I)$ is any rectangular partition. We prove this by introducing a set of line segments $E'(I)$ so that $E(I) \cup E'(I)$ is a guillotine partition and $L(E'(I)) \leq L(E(I))$.

- $L(G(I)) \leq L(E(I) \cup E'(I)) \leq 2L(E(I))$
Horizontal and Vertical Cuts

- Let $E_h(I)$ ($E_v(I)$) be the set of horizontal (vertical) line segments in $E(I)$

- A horizontal (vertical) cut is a horizontal (vertical) line segment that partitions rectangle $R$ into two rectangles.

- A rectangular partition $E(I)$ has a half horizontal overlapping cut $l$ if $l$ is a horizontal cut such that $L(l \cap E(I)) \geq 0.5X$.

- We say that $E(I)$ has a horizontal guillotine cut, $l$, if $l$ is a horizontal cut such that $L(l \cap E(I)) = X$.

- We define vertical guillotine cuts similarly.

- When $R$ is partitioned by a cut into two rectangles $R_1$ and $R_2$, we use $E(I_1)$ and $E(I_2)$ as the set of line segments $E(I)$ inside $R_1$ and $R_2$, respectively.
Transformation Idea

- Coloring the right hand side of a rectangle means that we have introduced a line segment (in $E'_v(I)$) of that length that have not been accounted by a vertical segment ($E_v(I)$).

- Taking a colored line segment away means that such segment has been accounted by segments from $E_v(I)$. (This is used to show, $E'_v(I) \leq E_v(I)$).

- The new horizontal segments will be introduced along half horizontal horizontal cuts. Therefore, $E'_h(I) \leq E_h(I)$.

- Empty problem instances will not have their right side colored.
Initially $E(I)$ is not empty and $R$ is not colored.

**PROCEDURE E_TO_G** $(I, E(I))$

begin
if $(E(I) == \text{Empty})$ return; // $R$ is not colored

\text{case}

: $E(I)$ has a half horizontal overlapping cut:
  Add line segment along cut;
  (Color $R_1$ and color $R_2$) if $R$ was colored;

: $E(I)$ has a vertical guillotine cut:
  Add line segment along cut;
  ($R_1$ and $R_2$) are not colored;

: else: Introduce mid-cut. I.e. vertical cut
  that intersecting the center of $R$;
  (Color $R_1$) and (color $R_2$ if $R$ was colored);

endcase

$E\_TO\_G$ $(I_1, E(I_1))$;
$E\_TO\_G$ $(I_2, E(I_2))$;
end
Lemma 1

Every invocation made to procedure $E\_TO\_G(I, E(I))$, satisfies the following conditions:

(a) $E(I)$ is empty and $R$ is not colored; or

(b) $E(I)$ is non-empty and $E(I)$ has a horizontal guillotine cut, then the right side of $R$ is not colored; or

(c) $E(I)$ is non-empty and $E(I)$ does not have a horizontal guillotine cut, then the only side of $R$ which could be colored is its right side.

Proof: Obviously true for the first invocation.

We prove that if upon entrance to the procedure the conditions are satisfied, then the invocations made directly from it will also satisfy (a), (b), or (c).

There are three cases.
Case 1: $E(I)$ is Empty. Then the procedure does not make any further invocations.

Case 2: $E(I)$ is not Empty and procedure $E\_TO\_G$ partitions $R$ along a half horizontal overlapping cut.

- $E(I)$ has a horizontal guillotine cut: Then $R$ is not colored and nothing gets colored at this step. Therefore, all invocations satisfy (a), (b), or (c).

- $E(I)$ does not have a horizontal guillotine cut: Then the right side of $R$ may be colored. Since there is no horizontal guillotine cut there are vertical line segments on both sides of the half horizontal overlapping cut, so $E(I_1)$ and $E(I_2)$ must be non-empty. Neither of these two partitions has a horizontal guillotine cut, therefore each invocation satisfy properties (b) or (c).
Case 3: \( E(I) \) is not Empty and procedure \( E\_TO\_G \) partitions \( R \) along a vertical guillotine cut.

- Just after partitioning \( R \) rectangles \( R_1 \) and \( R_2 \) will not be colored.
- Since \( R_1 \) and \( R_2 \) are not colored, the calls satisfy (a), (b) or (c).
Case 4: \( E(I) \) is not Empty and procedure \( E\_TO\_G \) introduces a mid-cut.

- Since \( E(I) \) does not have a half horizontal overlapping cut, each of the resulting rectangular partitions has at least one vertical line segment and there are no horizontal guillotine cuts on the two resulting problems.

- The right side of \( R_1 \) gets colored. The right side of \( R_2 \) gets colored when the right side of \( R \) was colored.

- Both problem instances are non-empty and do not have horizontal guillotine cuts. Therefore, the calls satisfy (c).
Lemma 2

For any non-empty rectangular partition $E(I)$ of any instance $I$ of the $RG - P$ problem, procedure $E_{TO}G$ generates a set $E'(I)$ of line segments such that $L(E'_h(I)) \leq L(E_h(I))$, and $L(E'_v(I)) \leq L(E_v(I))$.

- Horizontal segments are introduced only along half guillotine cuts.

- Vertical segments are introduced as mid-cuts. All are accounted for by other vertical guillotine cuts, because when $I_1$ or $I_2$ are empty the corresponding rectangles are not colored.
Theorem 1

Let $E(I)$ be an optimal rectangular partition for any instance $I$ of the $RG - P$ problem, and let $G(I)$ be an optimal guillotine partition for $I$. Then $L(G(I)) \leq 2L(E(I))$.

$L(E(I) \cup E'(I)) \leq 2L(E_h(I)) + 2L(E_v(I))$

$= 2L(E(I))$.

Since $E(I) \cup E'(I)$ is a guillotine partition for $I$, $L(G(I)) \leq L(E(I) \cup E'(I))$.

Therefore, $L(G(I)) \leq 2L(E(I))$. 
Divide-and-Conquer

- $O(n \log n)$-time divide-and-conquer approximation algorithm
- $L(I)$ represents the total length the line segments in the solution for problem instance $I$ generated by our algorithm, and $opt(I)$ represents the corresponding one in an optimal solution.
- Generates solutions within 4 times the optimal solution value, i.e. for every $I$, $L(I) \leq 4 \, opt(I)$. 
Definitions

- Problem instance: \( I = ((x, y), (X, Y), P) \), where \((x, y)\) and \((X, Y)\) define a rectangle \( R \) \((x, y)\) is the lower-left corner, and \((X, Y)\) are the dimensions), and \( P = \{p_1, p_2, \ldots, p_n\} \) is a nonempty set of points inside \( R \).

- A *mid-cut* is a line segment orthogonal to the x-axis that intersects the center of the rectangle (i.e., it includes point \((x + \frac{X}{2}, y + \frac{Y}{2})\)).

- An *end-cut* is a line segment orthogonal to the x-axis that contains either one of the “leftmost” or the “rightmost” points in \( P \).
PROCEDURE PARTITION((x,y) , (X,Y) , P)

begin
    Relabel the dimensions so that $X \geq Y$;
    \[ P_1 \leftarrow \{ p_k \mid p_k \in P \text{ and } x_k < x + \frac{X}{2}, \text{ where } p_k = (x_k, y_k) \}; \]
    \[ P_2 \leftarrow \{ p_k \mid p_k \in P \text{ and } x_k > x + \frac{X}{2}, \text{ where } p_k = (x_k, y_k) \}; \]
    case
    
    :$P_1 \neq \emptyset$ and $P_2 \neq \emptyset$: /* introduce a mid-cut */
    Introduce the line segment orthogonal to the x-axis that partitions $R$ through its center;
    PARTITION(((x, y), (\frac{X}{2}, Y), P_1));
    PARTITION(((x + \frac{X}{2}, y), (\frac{X}{2}, Y), P_2));
:else: /* introduce an end-cut */

Let $c$ be the coordinate value along the x-axis of a point in $P$ with smallest $|c - (x + \frac{X}{2})|$;

Introduce the line segment orthogonal to the x-axis that partitions $R$ through the points with x-coordinate value equal to $c$;

Delete from $P_1$ and $P_2$ all the points located along the end-cut;

if $P_1 \neq \emptyset$ then PARTITION($((x, y), (c - x, Y), P_1)$);
if $P_2 \neq \emptyset$ then PARTITION($((c, y), (X - (c - x), Y), P_2)$);

endcase

end
• It is easy to verify that the figure represents all the possible outcomes of one step in the recursive process of our algorithm. A region with dashed lines represents a sub-instance without interior points.

• We use \(X'\) and \(X''\) to represent the length along the x-axis of the two resulting sub-instances \((I_1 \text{ and } I_2)\), respectively.
Lower Bound Function

\[
LB(I) = \begin{cases} 
Y & \text{(1) An end-cut is introduced,} \\
LB(I_1) + LB(I_2) & \text{(2) A mid-cut is introduced,} \\
LB(I_1) + \min\{Y, X''\} & \text{(3) An end-cut is introduced,} \\
LB(I_2) + \min\{Y, X'\} & \text{(4) An end-cut is introduced,}
\end{cases}
\]

\[
P_1 = \emptyset \text{ and } P_2 = \emptyset. \\
P_1 \neq \emptyset \text{ and } P_2 \neq \emptyset. \\
P_1 \neq \emptyset \text{ and } P_2 = \emptyset. \\
P_1 = \emptyset \text{ and } P_2 \neq \emptyset.
\]
Lemma 1: For any problem instance $I$, $LB(I) \leq opt(I)$.

- Construct rectangular partition by using procedure MOD-PARTITION, similar to PARTITION.

- Difference is:
  - When a mid-cut is introduced by PARTITION and $P_1 = P_2 = \emptyset$, MOD-PARTITION returns without introducing a cut
  - When PARTITION introduces an end-cut and $P_1 \cup P_2 \neq \emptyset$, MOD-PARTITION associates the points in the end-cut with the empty rectangle generated at this step.
• MOD-PARTITION constructs a rectangular partition in which
  – Each rectangle either contains a set of points inside it, all of which are located on a line orthogonal to one of the axes; or
  – the rectangle contains no interior points, but there is at least one point on its boundary associated with the rectangle.
Observations:

- In the former case it is simple to see that any rectangular partition must have a line segment inside that rectangle including the interior points with length at least equal to the lower bound given in case (1) of the definition of $LB$;

- In the latter case any rectangular partition must have a line segment inside or on the sides of the rectangle that includes the point associated with it with length at least equal to the lower bound given in case (3) or (4) in the definition of $LB$. 
Definitions

- $USE(I)$: length of the line segments introduced during the first call to procedure $PARTITION(I)$.
- If $P = \emptyset$ then $USE(I) = 0$; otherwise, $USE(I) = L(I) - L(I_1) - L(I_2)$.
- $I_1, I_2, \ldots, I_m$: problem instances encountered at all levels of the recursive process.
- $L(I) = \sum_{j=1}^{m} USE(I_j)$. 
Definitions: Regular / Irregular

- $I$ is said to be *irregular* if $X > 2Y$, and *regular* otherwise (i.e., $X \leq 2Y$).

$$C(I) = \begin{cases} 3Y & \text{if } I \text{ is irregular,} \\ X + Y & \text{if } I \text{ is regular.} \end{cases}$$
Proof Idea

- Whenever a line segment is introduced by the algorithm (*mid-cut* or *end-cut*) it is colored red.

- When a lower bound from $LB(I)$ is "identified" a line segment with such length is marked blue.

- Bound the sum of the length of all the red segments by 4 times the sum of the length of the blue segments.

- Carry function $C$ corresponds to the length of red segments introduced at previous steps which have not yet been accounted for by blue segments.
Proofs

Lemma 1.2: \( C(I) \leq 3 \cdot Y \).

Lemma 1.3: \( L(I) + C(I) \leq 4 \cdot LB(I) \).

Basis

- Set \( P \) contains exactly one point
- Single line segment is added.
- Clearly, \( L(I) + C(I) \leq 4 \cdot LB(I) \).

Induction hypothesis

- Assume the lemma holds when the number of points in \( P \) is less than \( m \).
Induction Step

- Prove the lemma holds when the number of points in $P$ is $m > 1$.

Case 1

- A mid-cut is introduced and both $I_1$ and $I_2$ contain at least one point.
- Applying the induction hypothesis we know
  \[ L(I_1) + L(I_2) + C(I_1) + C(I_2) \leq 4LB(I_1) + 4LB(I_2) \]
- By definition we know that
  \[ L(I) = L(I_1) + L(I_2) + USE(I), \]
  \[ LB(I) = LB(I_1) + LB(I_2). \]
- Therefore, we just need to prove that
  \[ Y + C(I) \leq C(I_1) + C(I_2). \]

Subcase 1.1: $I$ is regular.
• By definition,

\[ Y + C(I) = X + 2Y. \]

• Both \( I_1 \) and \( I_2 \) are also regular. Therefore,

\[ Y + C(I) = X + 2Y = C(I_1) + C(I_2). \]

**Subcase 1.2: \( I \) is irregular.**

• By definition of the carry function, we know that \( Y + C(I) = 4Y \).

• By the definition of the carry function we know that \( Y + C(I) = 4Y < C(I_1) + C(I_2) \).
Case 2

- An *end-cut* is introduced and exactly one of the two resulting subproblems has no interior points.

See proof in paper.
Case 3

- The algorithm introduces an end-cut, and $I_1$ and $I_2$ contain no points.
- The procedure introduces a line segment that intersects all the points in $P$.
- So $L(I) = USE(I) = Y$.
- By the previous lemma $C(I) \leq 3Y$.
- Since $LB(I) = Y$, it then follows that $L(I) + C(I) \leq 4LB(I)$.

Theorem

- For any instance of the $RG$-$P$ problem, algorithm PARTITION generates a solution such that $L(I) \leq 4opt(I)$. 