STOCKMEYER ALGORITHM

ECE 6133 : FINAL PROJECT

SUJAY KHOLE

WHAT DOES STOCKMEYER ALGORITHM DO?

1. Given a slicing floor-plan, find out the optimal orientation of each module with a goal of minimizing the area of the floor-plan.

2. Can be used to improve the quality of certain algorithms which only determine the relative positions of modules without considering their orientations.

3. Used mostly as a post-processing step. Improvement depends upon whether the critical path is affected. (Explained in conjunction with the results)

OBJECTIVES OF THE PROJECT

1. Develop a C++ Implementation of the Stockmeyer's Algorithm.

2. Five initial floorplans provided. 5_block.ple, 10_block.ple, 30_block.ple, 100_block.ple, 150_block.ple.

3. Using C++ code, determine the optimal rotation of each block to reduce the total area of the floorplan.

4. Determine the statistics such as Area improvement, number of block rotated and runtime.

5. Redirect the output into a .txt file in a suitable format such that MATLAB can be invoked to display the initial and the final floorplans in a GUI.

DATA-STRUCTURES USED

1. C++ was used because of the powerful Standard Template Library.

2. **Stack:** A Stack was used to push and pop the nodes of a parse tree while it was being generated from the Polish Expression.

3. **Vectors**: Used to store the dimension list of each node in the parse tree. Provides combined features of an array and the queue data-structure.

4. **Pairs:** A natural choice available in the STL to store the width-height pairs associated with every module. Forms an element of a nodes dimension list.

5. **Binary Parse Tree:** To process the given Polish expression

PARSING THE INPUT .ple FILE

1. If there are N modules, modules are numbered from 0 to N-1

2. The first line contains the Polish Expression. It is read into a String variable so that a parse tree can be generated later.

3. Line 2 onwards contains the width height pair separated by a space . There is one width height pair per line

   For example

   Line 2 has width height of module 0

   Line 3 has width height of module 1

   Line N+1 has width height of last module N-1

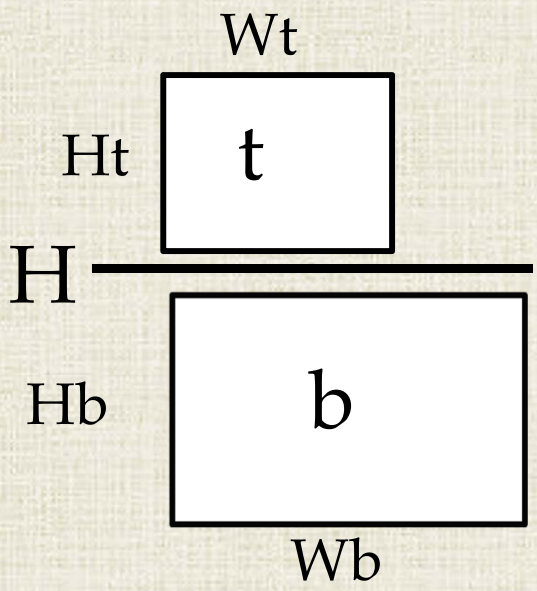
4. The width- height pair are stored in an Array of "pair" .

   wh_pair[0].first = width of module 0.

   wh_pair[0].second = height of the module 0.

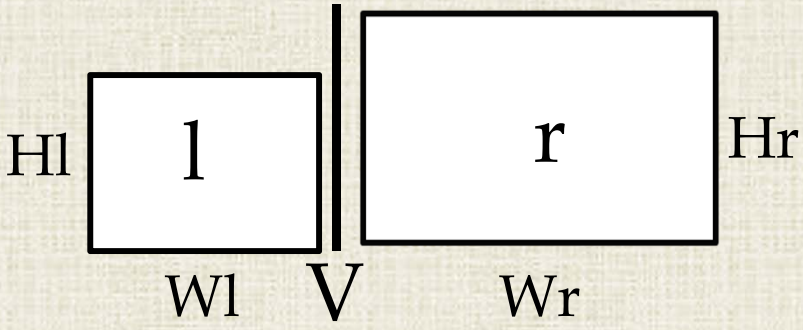
FORMAT OF THE POLISH EXPRESSION

1. Consider a Polish Expression. **2-1-0-H-V-3-V-4-V**
2. Module numbering begins from 0 . Nodes are separated by " **-** "
3. Cut "t-b-H": module t goes to the top and b to the bottom.
4. Cut "l-r-V": module l goes to the left and r to the right



Joining modules t and b leads to the new dimension with

Width = max(Wt, Wb)
Height = Ht + Wt

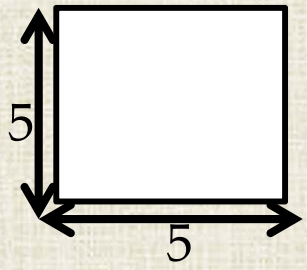


Width = Wl + Wr
Height = Max(Hl, Hr)

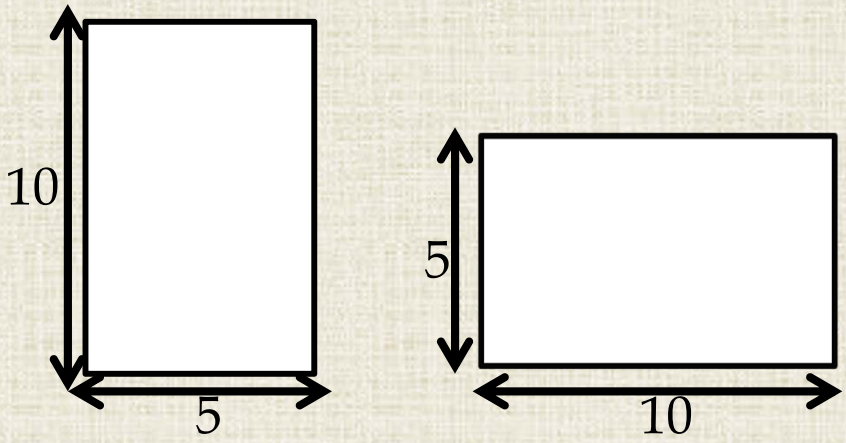
WORKING OF THE ALGORITHM

1. Start with a binary slicing tree representation of the given polish expression.

2. Visiting the leaf nodes (modules) first. Depending on the shape of the module, add the width-height pair for the possible orientations of the module



A square module with a single orientation. (5,5)



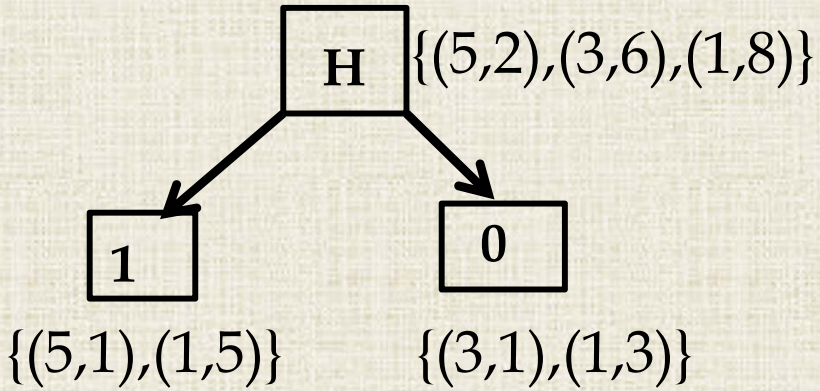
A rectangular module with two orientations. (5,10) & (10,5)

3. Calculate the list of dimension pairs for each non-leaf node all the way up to the root node.

BUILDING THE DIMENSION LIST FOR "H" OR "V" NODE

1. This kind of sorting of the children of an H-node or V-node helps reduce the time and memory requirements.

2. To see how this helps consider the following H-cut with the dimensions of its children sorted into descending widths.



In a Naive Implementation, we would have combined

(5,1) & (3,1) = (5,2)

(5,1) & (1,3) = (5,4) ….unnecessary!!

(1,5) & (3,1) = (3,6)

(1,5) & (1,3) = (1,8)

3. If left_node.width > right_node.width then the rest of the dimensions created using left node will give a higher area

(Due to the decreasing widths/increasing heights relationship.)

GENERATING A PARSE TREE

1. Start processing the Polish Expression from Left to Right.
   If the node is a module, initialize its dimension list depending upon whether it's a square or rectangular module.

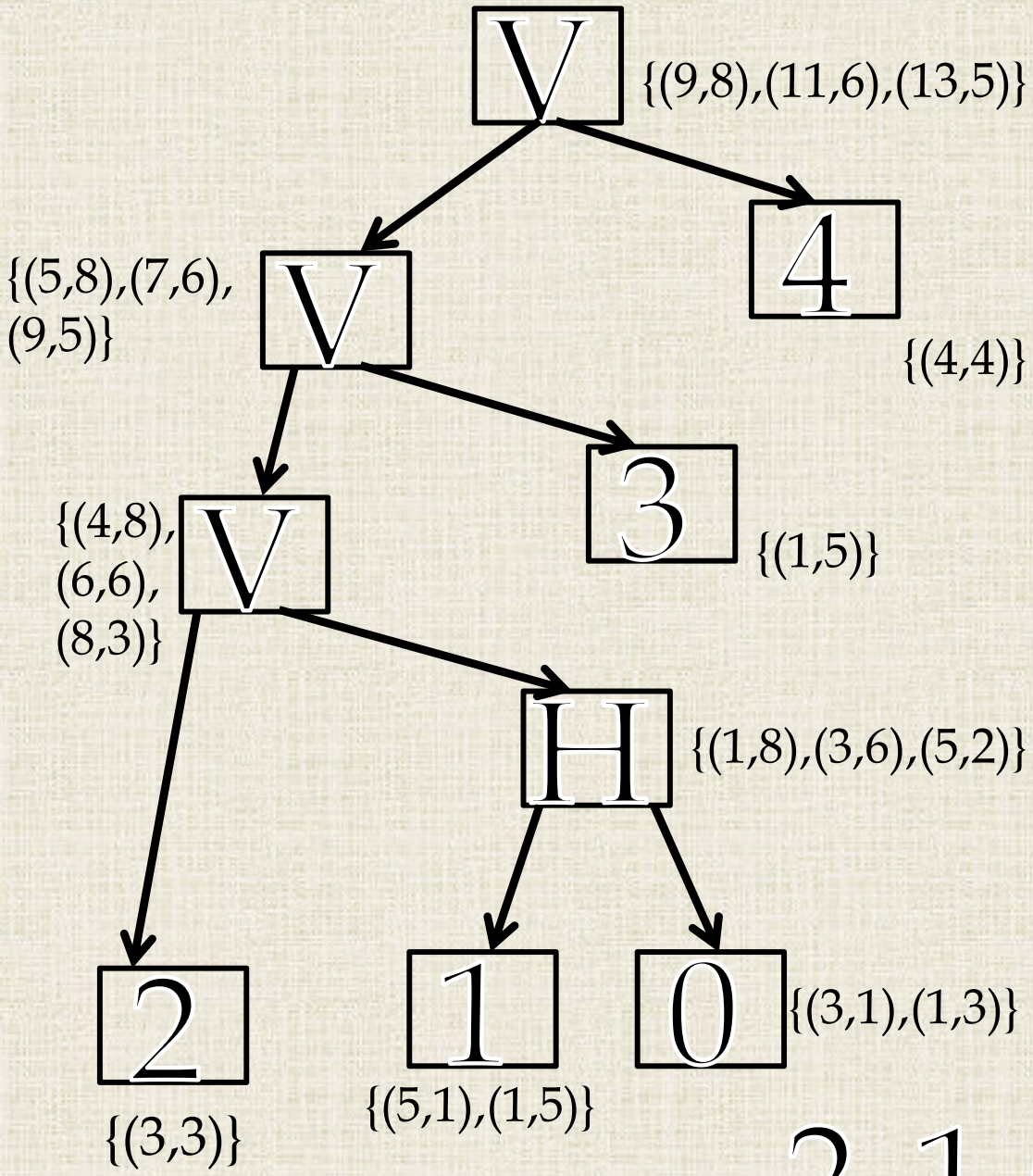
2. The node is then pushed onto a Stack. It has no children nodes.

3. For a V or H node, the top of the Stack is popped and attached as a right child. The one below is attached as the left child.

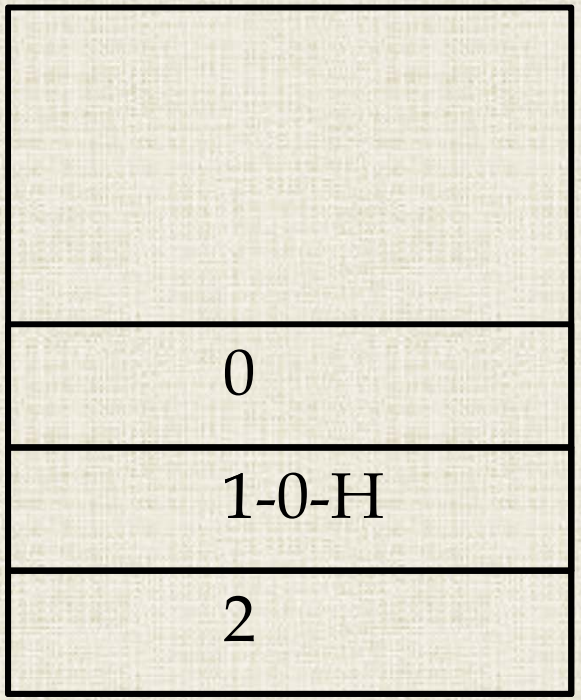
4. Its dimension list is built by considering one element at a time from both its children. Separate calculations need to be done for an H node vs. a V node.

5. For an H-node sort the dimension list of both children into decreasing order of widths & increasing widths for a V-node.

1. Consider a Polish Expression. **2-1-0-H-V-3-V-4-V**



STACK FOR BUILDING THE PARSE TREE



2 1 0 H V 3 V 4 V

DETERMINIG OPTIMAL ROTATION

1. It is just not sufficient to combine two dimension from children to form a dimension pair in the parent's list.

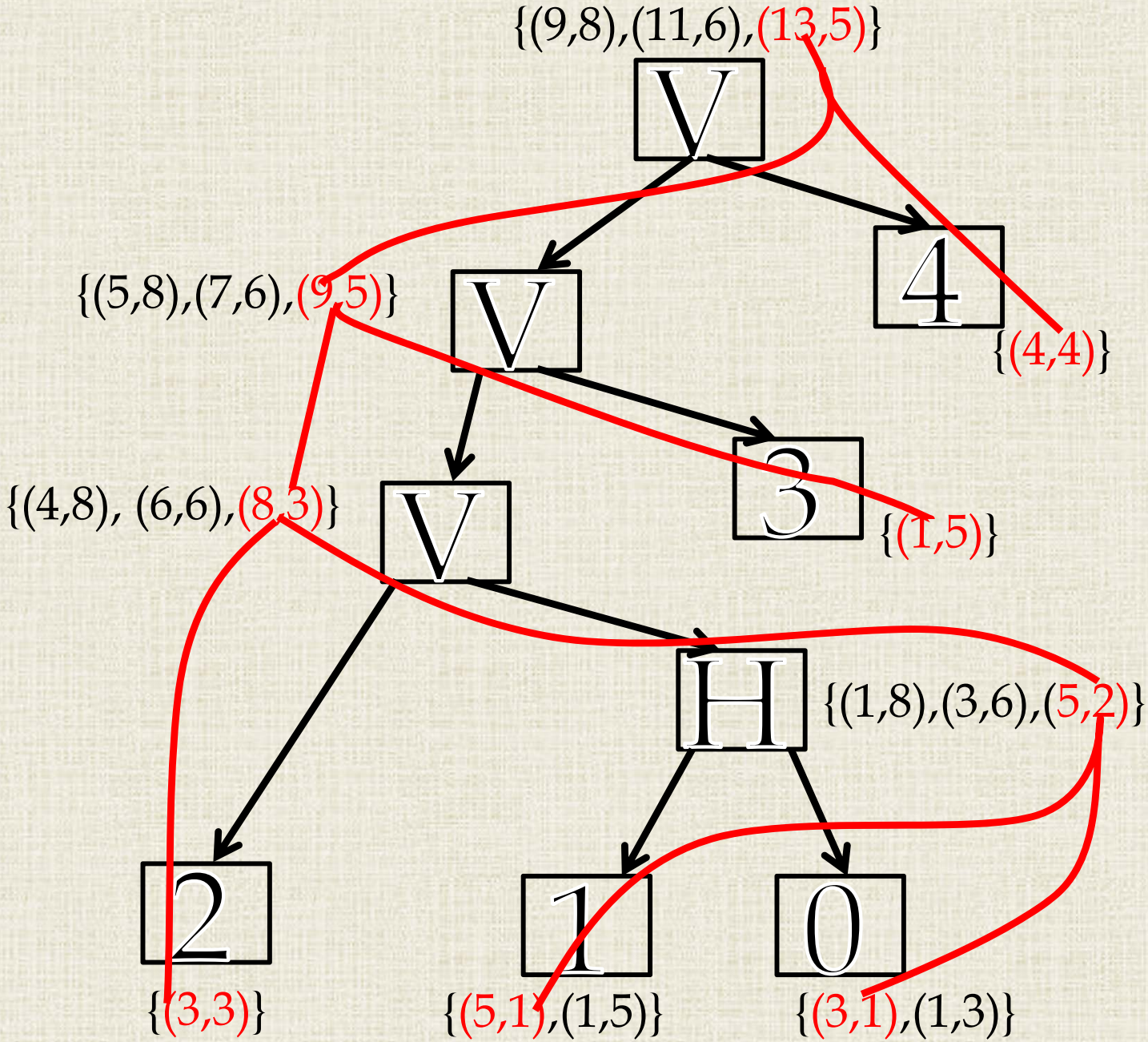
2. It is also essential to maintain the left and right child dimension pointer that led to forming a particular parent list dimension.

3. Once we reach the top node, we have all possible width- height pairs that may give an improvement in the area .

4. Of all the width–height pair at the top node, one with the least area is selected.

5. Using the left and right child dimension pointers traverse down back the tree to change the orientation of the leaves(modules) that result in the minimum area dimensions at top node.

Example: **2-1-0-H-V-3-V-4-V**



DETERMINIG MODULE CO-ORDINATES

1. The co-ordinates are determined during a top down traversal of the parse tree.

2. The top/root node is assigned (x,y) = (0,0)

3. **For a parent H node:**
   1. Right child's coordinates = parent H node's coordinates.
   2. Left child: x- coordinate = parent's X coordinate.
      y- coordinate = parent's Y coordinate+ right child's height

4. **For a parent V node:**
   1. Left child's coordinates = parent V node's coordinates.
   2. Right child: y- coordinate = parent's Y coordinate.
      x- coordinate = parent's X coordinate+ left child's width

GENERATING FLOORPLAN WITH MATLAB

1. Floor plan before and after application of Stockmeyer Algorithm is generated by writing a MATLAB script file.

2. The output from C++ program was stored in a .txt file in a predetermined format as follows.

X co-ordinate	Y co-ordinate	Width	Height	Module No.	1= Rotated 0=Not rotated
0	0	3	3	2	0
3	0	3	1	0	1
3	1	5	1	1	0
8	0	1	5	3	0
9	0	4	4	4	0
….		….		….	
….		….		….	

RESULTS

AND

FLOORPLANS

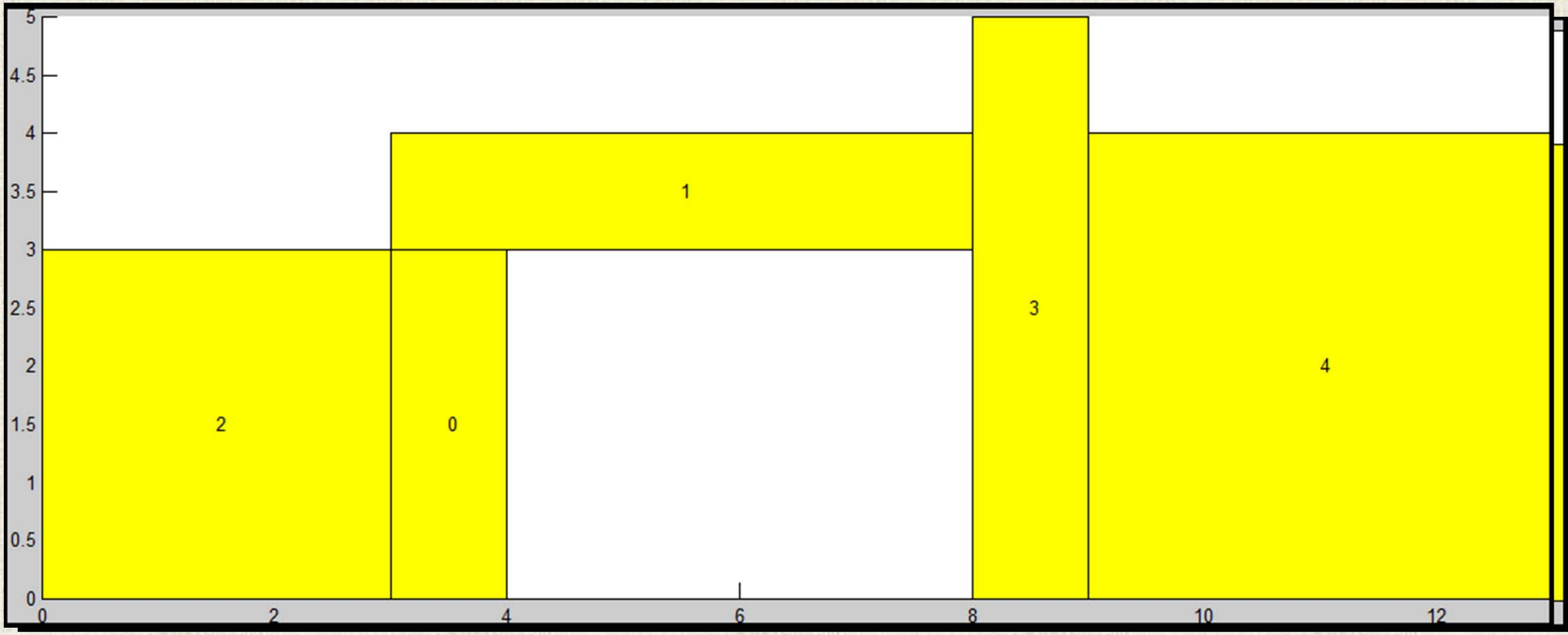
Input filename	Original Area	New Area	% improvement in area	No. of modules Rotated	Modules rotated
5_block.ple	65	65	0 %	1	0
10_block.ple	147	95	35.3741%	4	1,3,5,7
30_block.ple	1075	748	30.4186%	9	2,15,16,18,22,23,24,27,28
100_block.ple	7119	4264	40.1039%	38	1,2,3,5,7,8,10,11,16,18,28,23,26,32,33,41,42,49,50,54,55,62,63,64,66,68,74,75,76,77,78,80,81,82,87,90,91,96
150_block.ple	14104	8316	41.0380%	56	3,7,8,10,18,20,23,25,31,32,34,35,39,44,45,46,47,58,63,64,65,66,70,71,72,73,74,78,79,82,83,85,86,88,91,97,98,99,102,105,113,114,118,119,121,124,134,135,137,139,141,142,143,144,146,147

RUNTIME FOR THE FLOORPLANS

1. The runtime measurements were made on an Intel Core i-5 laptop with 8GB of RAM.
2. In order to have a reliable reading, an average of 10 values was taken for each floorplan.

Input Filename	Runtime in msec
5_block.ple	0.568 ms
10_block.ple	0.7262ms
30_block.ple	1.1202ms
100_block.ple	1.78ms
150_block.ple	2.6928ms

AREA BEFORE = 65



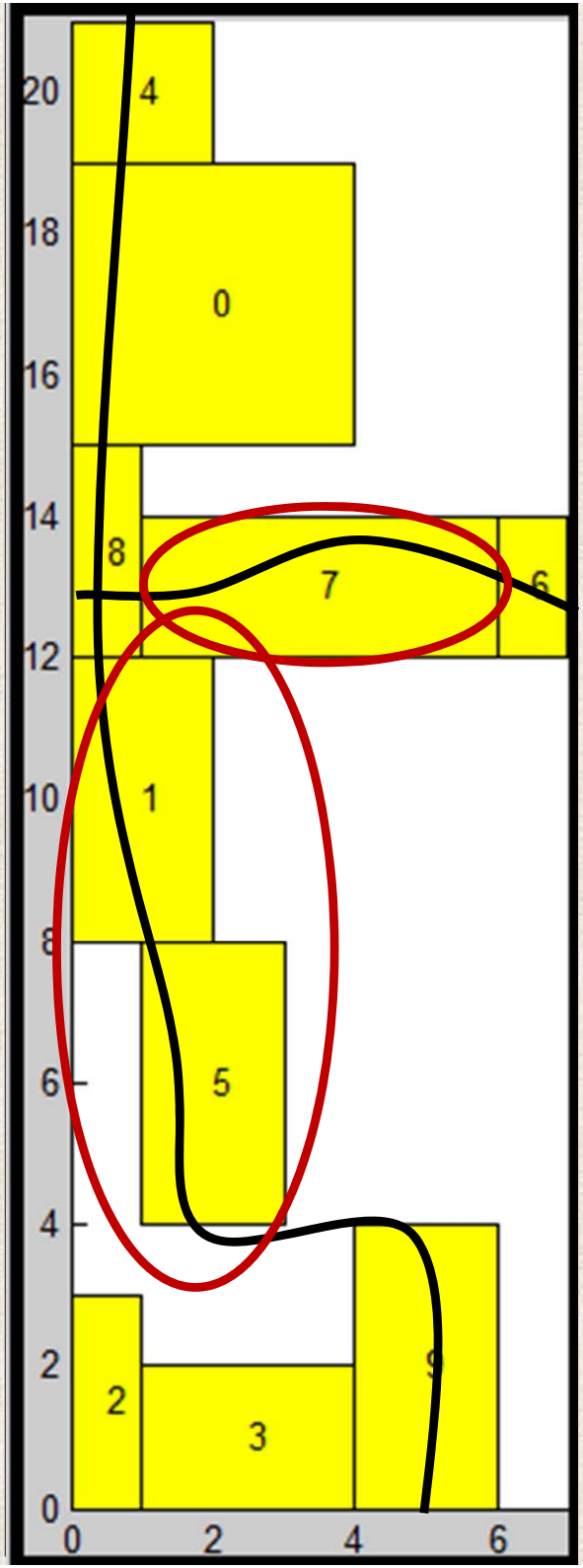
AREA AFTER = 65

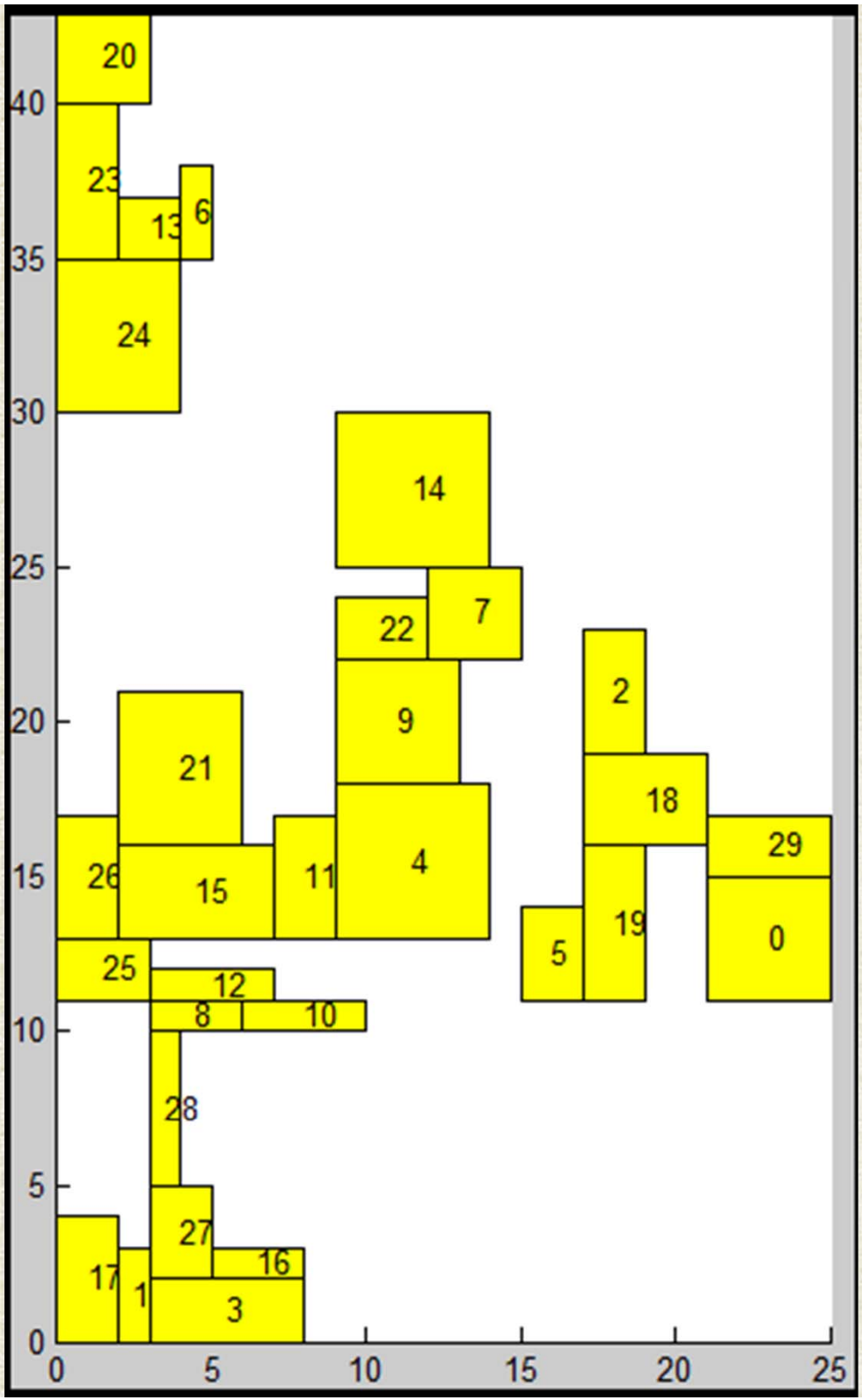
NO IMPROVEMENT!! (5_block.ple)

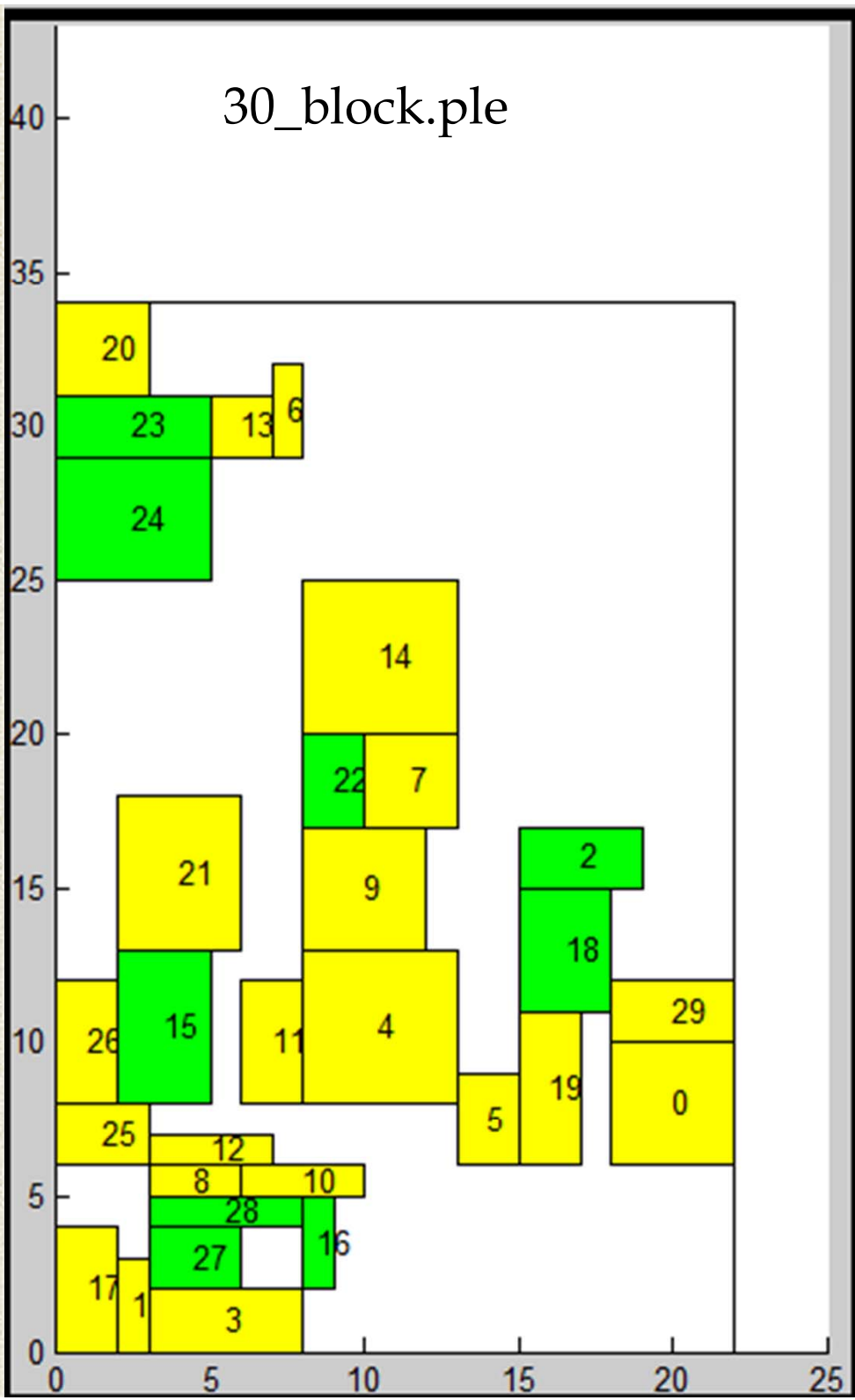
Example: 10_block.ple

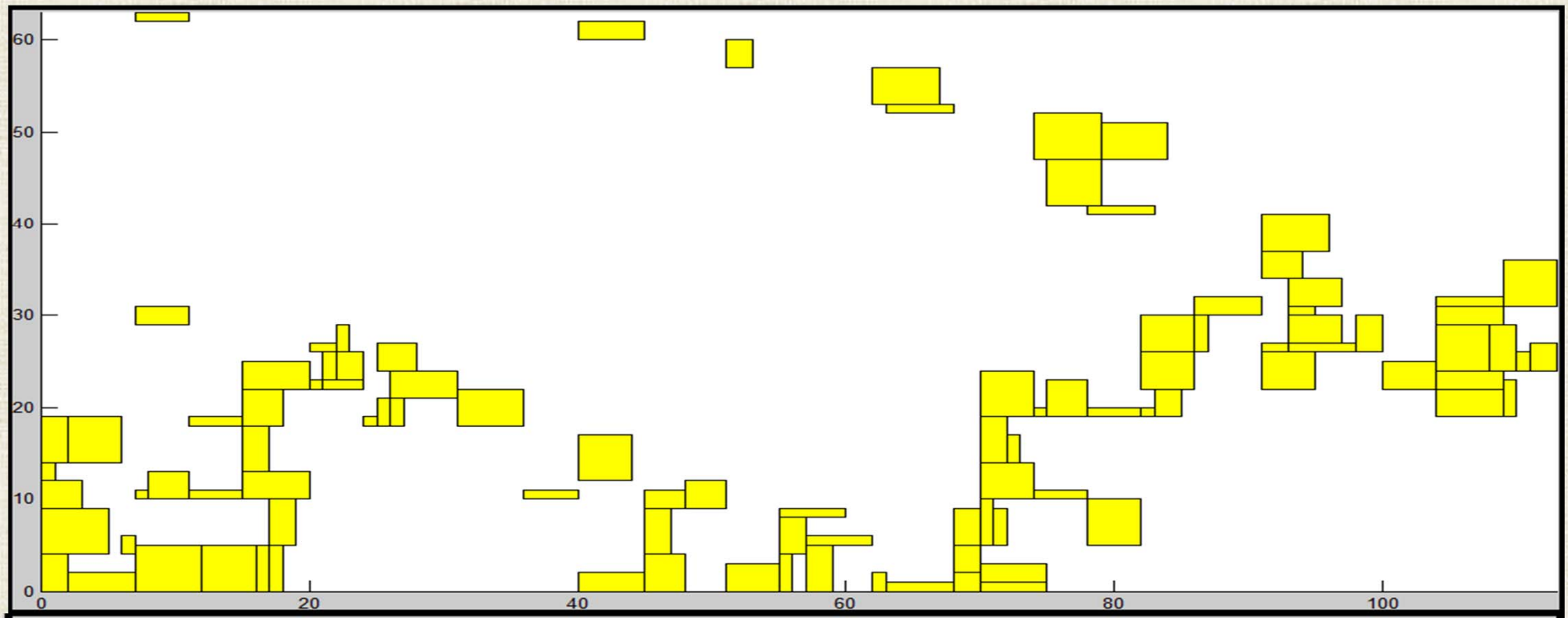
Module 7 that lies on the horizontal critical path has rotated

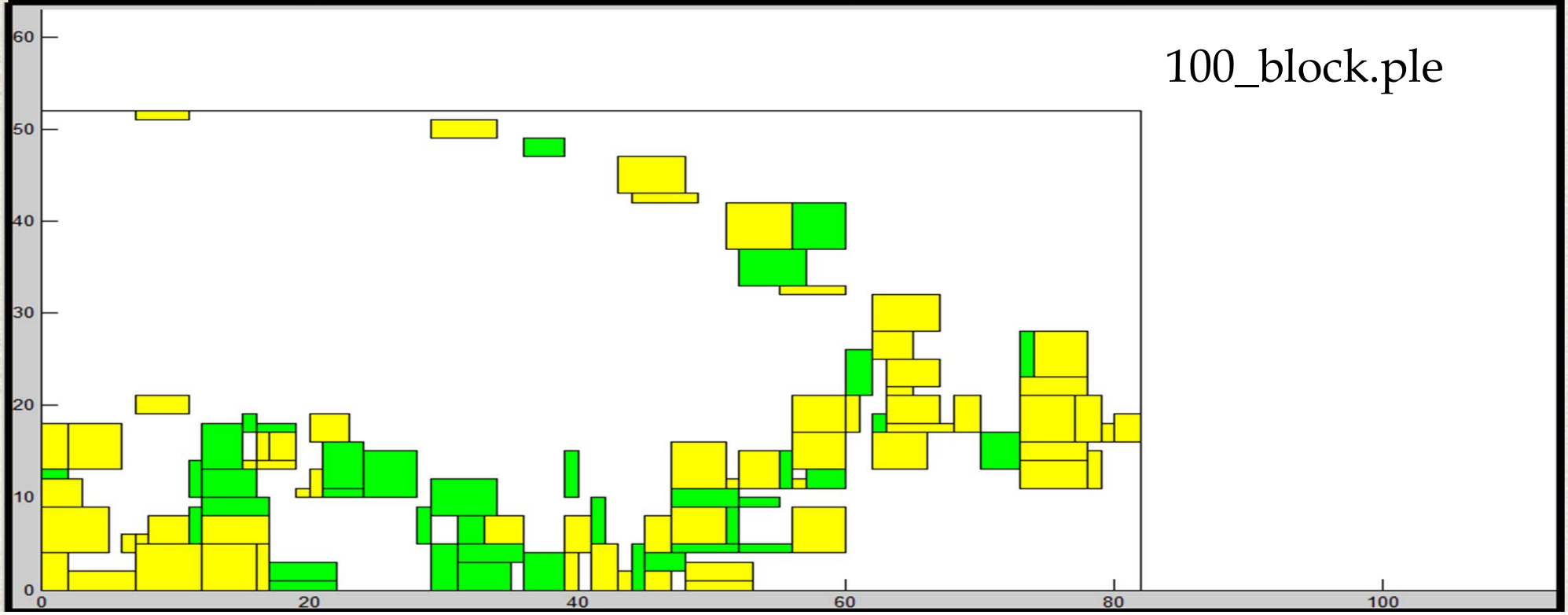
Modules 1 and 5 that lie on the vertical critical path have rotated.



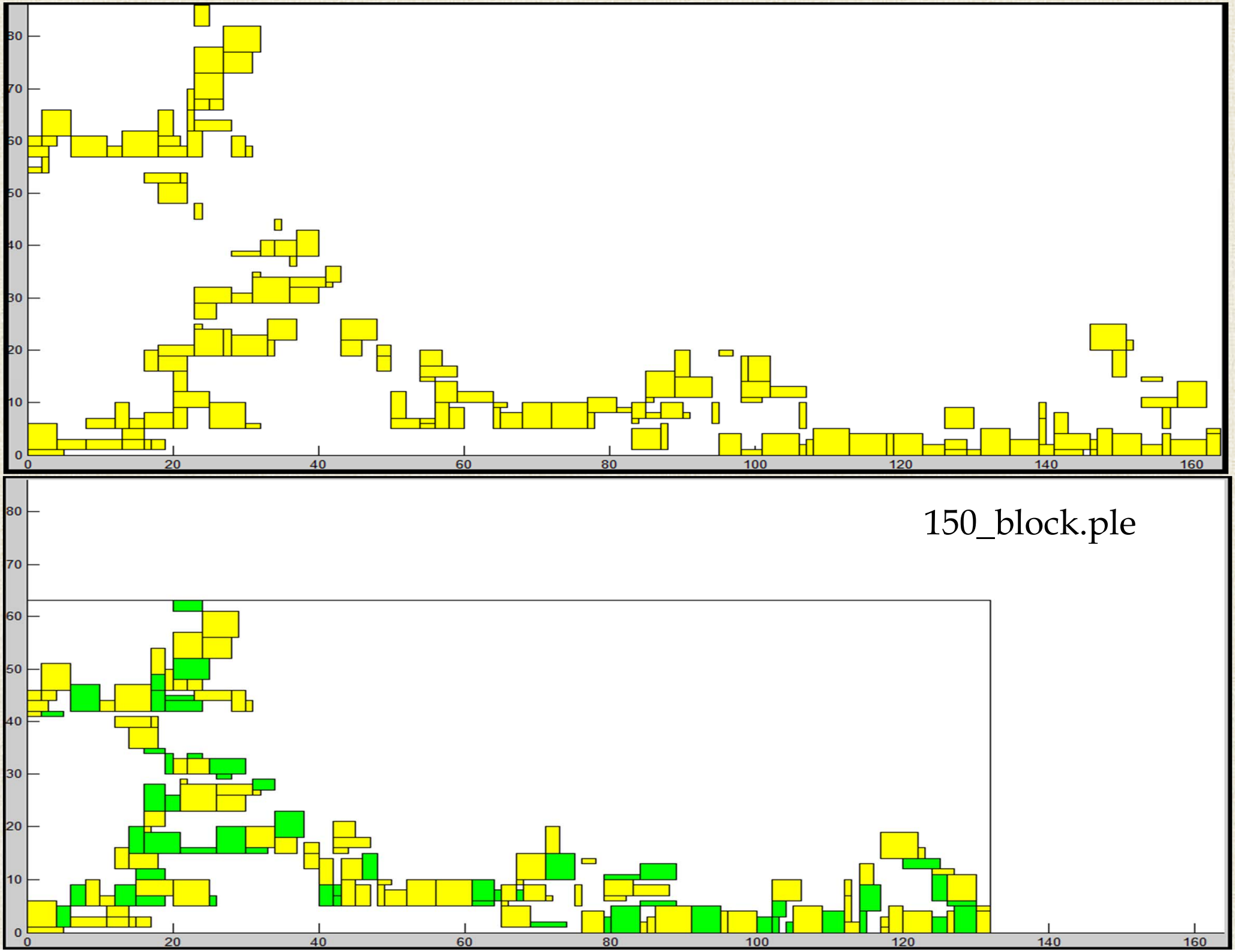

20
23
13
6
24
14
22
7
9
2
21
18
29
26
15
11
4
19
5
0
25
12
8
10
28
27
16
17
1
3
0
5
10
15
20
25
0
5
10
15
20
25
30
35
40


30_block.ple
20
23
13
6
24
14
22
7
9
2
21
18
29
26
15
11
4
19
5
0
25
12
8
10
28
27
16
17
1
3
0
5
10
15
20
25
0
5
10
15
20
25
30
35
40

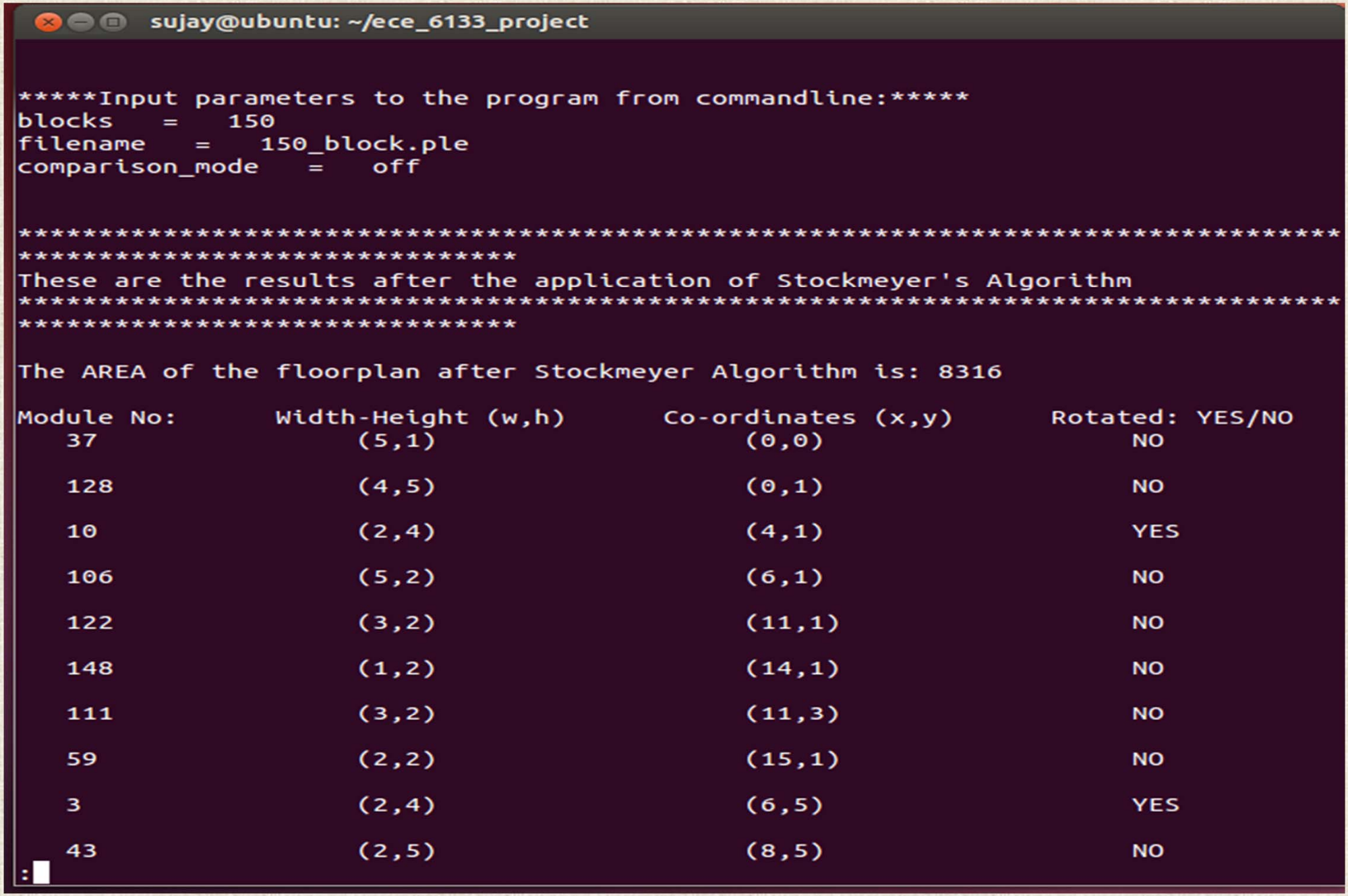




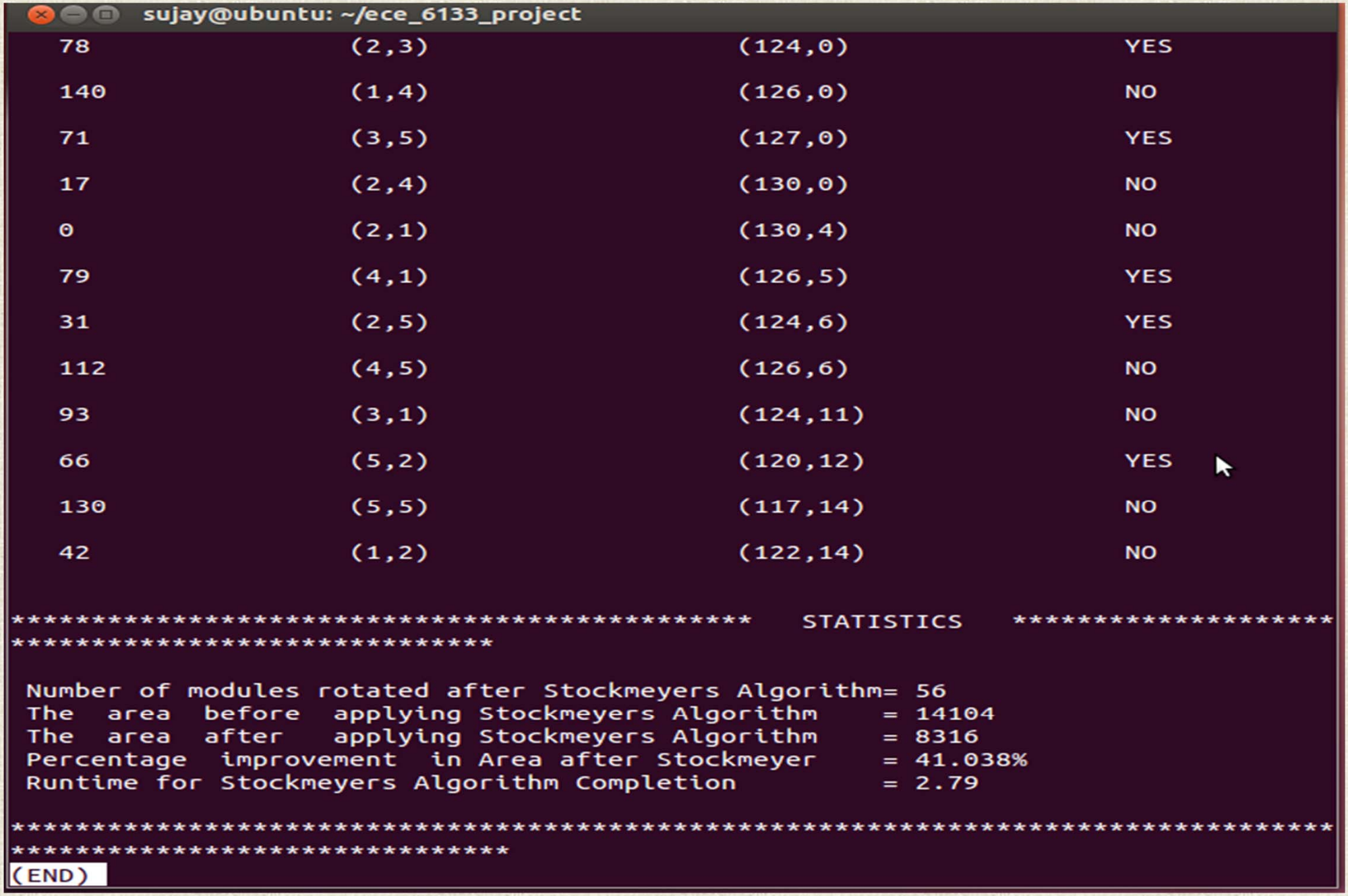
100_block.ple


150_block.ple
0
10
20
30
40
50
60
70
80
0
20
40
60
80
100
120
140
160

SCREENSHOTS OF PROGRAM EXECUTION(1)



SCREENSHOTS OF PROGRAM EXECUTION(2)



CONCLUSIONS

1. Stockmeyer Algorithm provides an optimum Solution if it exists.

2. It provides area improvement if the modules that lie on the critical path rotate.

3. The algorithm may provide improvement when combined with other heuristic floorplanners (Simulated Annealing on the Polish Expression)

QUESTIONS?

REFERENCES

1. Larry Stockmeyer, "Optimal orientation of cells in slicing floorplan designs".

2. Lim, Sung Kyu, " Practical Problems in VLSI Physical Design Automation"

3. Class notes for ECE 6133 Spring 2013, Professor Lim, Sung Kyu.

4. Sample Project Slides and Reports, http://users.ece.gatech.edu/limsk/course/ece6133/