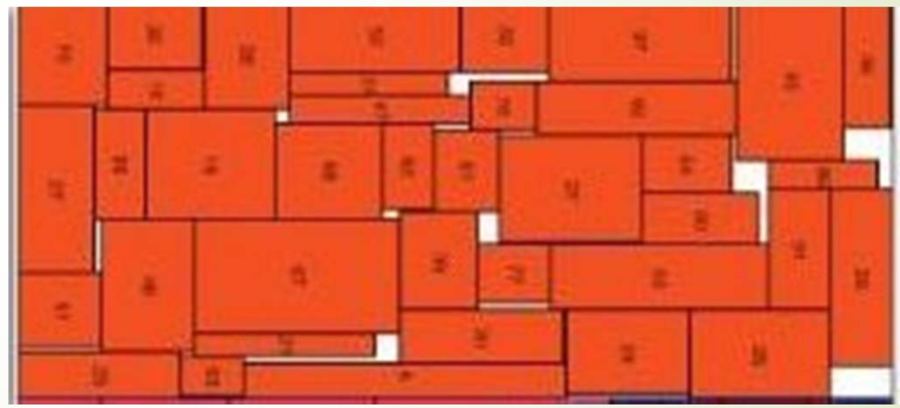
Floorplan using Simulated Annealing



Pramod Nataraja

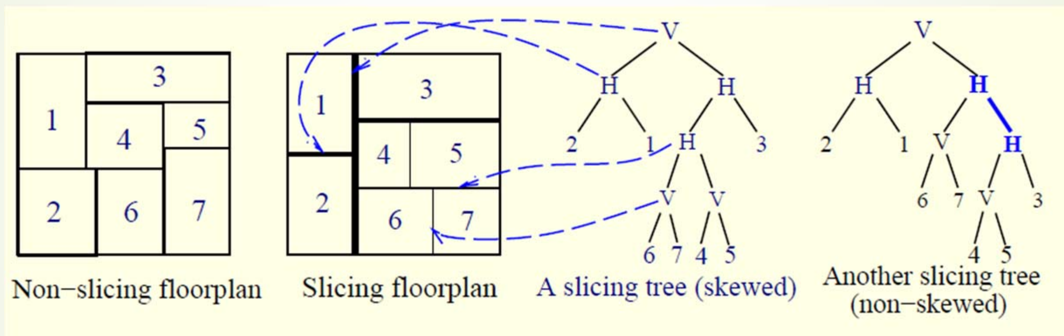
Praveen Kumar C P

Overview

- Normalized Polish Expression
- Problem Formulation
- Algorithm & Implementation
- Results Analysis
- Video
- Conclusion & Possible Improvements

Normalized Polish Expression

- Skewed Slicing Structure
  - There is a 1-1 correspondence between Slicing Floorplan, Skewed Slicing Tree, and Normalized Polish Expression.



- An expression E=e1e2 . . . e2n-1 , where each ei ε {1 , 2, . . . , n , H , V} , 1<=i<=2n-1 , is a Polish Expression of length 2n-1 iff (1) every operand appears exactly once(2) Expression E has balloting property. No of Operands > No of Operators
- A Normalized Polish Expression is one in which there are no consecutive operators of the same type (H or V respectively).

Figure taken from Prof Sung Kyu Lim Lecture

Problem Formulation

- Objective:
  - A feasible Floorplan optimizing the desired cost function.
- Input:
  - n Blocks with areas $A_1, \dots, A_n$ and initial x,y co-ordinates.
  - Initial Polish Expression is also provided
- To Do
  - An iterative process to modify the initial Polish expression by making moves and arriving at a final Polish expression that minimizes the cost function using a process that is analogous to annealing.
- Output:
  - Coordinates (x_i, y_i) for each block.

Problem Formulation II

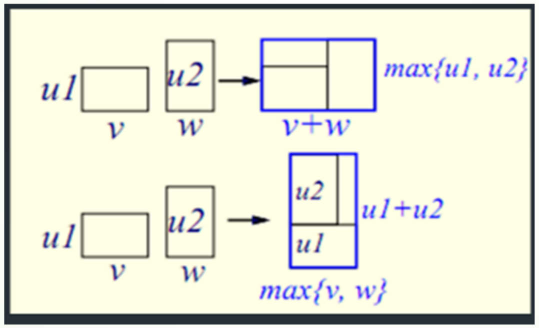
- Cost Function
  - Cost(F) = αA+λW
  - A: area of the smallest rectangle
  - W: overall wiring length
  - λ: user-specified parameter (in our case it's 0)
- Constraints
  - Move(M1, M2, M3) are allowed
  - No Move should violate the balloting property.
- Assumptions/Notes
  - Aspect ratio bound is not considered.
  - Initially reducing the cost function was the main factor and the time taken for annealing algorithm was ignored

Implementation & Area Computation

- Usage of STL::Vector
  - Vector data structure was is used to store the Polish Expression , height and width of individual blocks
  - Reduced the overhead of having to manage data with Structs
  - Reduced the overhead of implementation of methods to insert, delete and modify nodes
- Input Parsing
  - Input files containing Polish Expression are parsed into 3 Vectors
    - One containing the Nodes and Operators
    - 2 containing width and height of each node
    - H and V are represent by -2 and -3 respectively

Area Computation

- Parse the expression to find the first operator (H or V)
- Combine the operator along with the 2 previous nodes in the expression
- Calculate the height and width of the new block obtained as shown below



- Remove the operator and the previous 2 operands from the Polish Expression and replace them with a single new block obtained
- Insert the height and width of the new block at the end of Height and width vectors of the Polish Expression
- Repeat the process until you obtain a single large node. The area of this node gives the overall area of the floorplan.

Area Computation

- Example Polish Expression

2-1-0-H-V-3-V-4-V

↓

2-5-V-3-V-4-V

↓

6-3-V-4-V

↓

7-4-V

↓

8

Width and Height Vectors

Initial

W	H
0	0
1	1
2	2
3	3
4	4

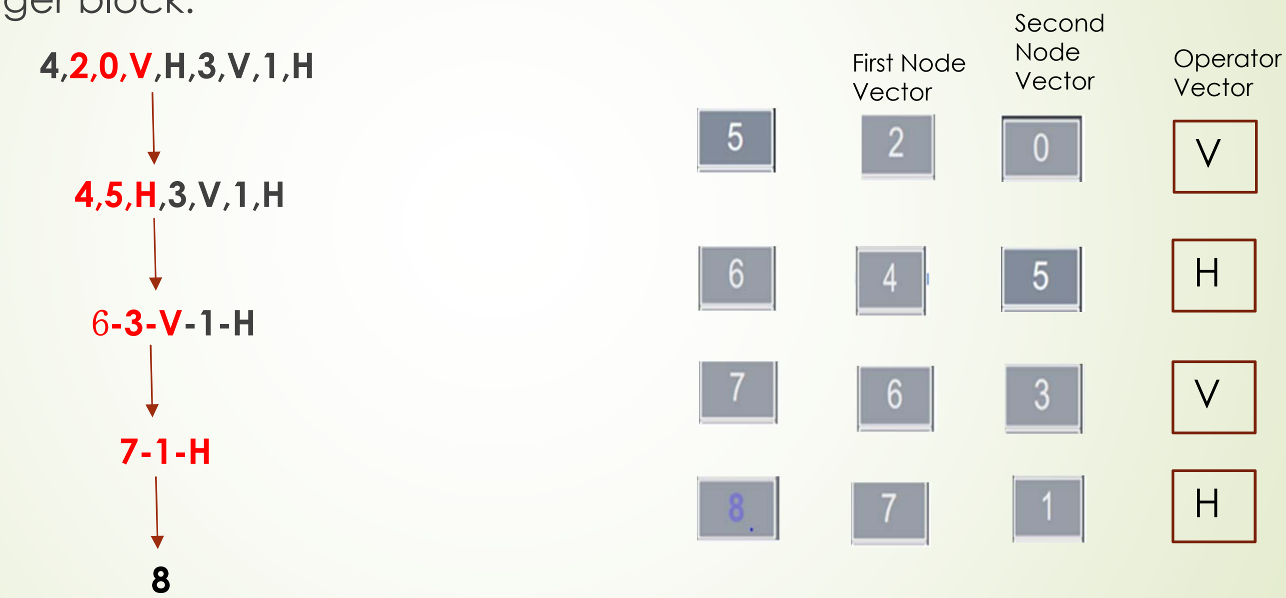
Final

W	H
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8

The total area of the floorplan can be calculated from the dimensions of block 8

X and Y Co-ordinate computation

- During Area Calculation , maintain 3 more Vectors containing left node, right node and the operator each time 2 nodes are combined into a bigger block.



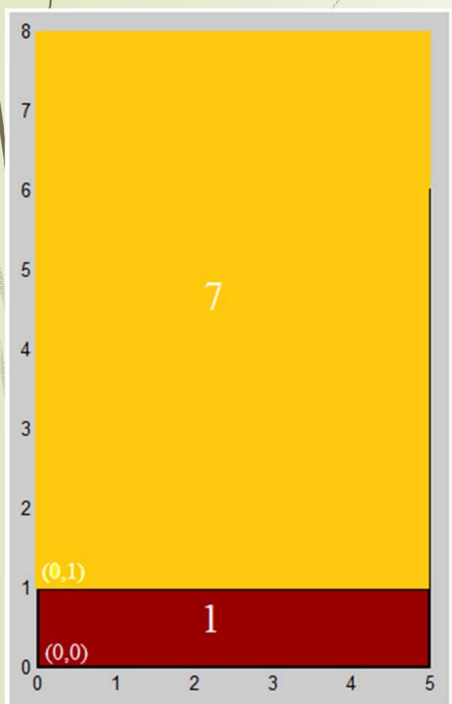
On reaching the final Block , we assign the 0,0 as the x and y co-ordinates of the Final block which is formed by combining all blocks in the floorplan.

X and Y Co-ordinate computation

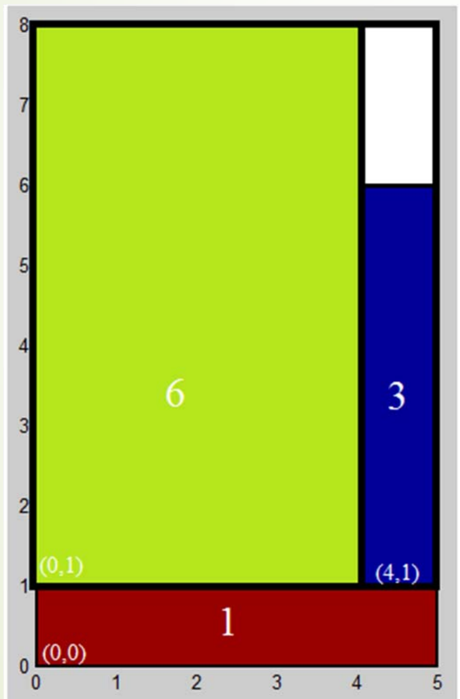
- Iterating backwards, from the Final block , we assign the x and y co-ordinates to each block formed by combining 2 blocks depending on the operator used to combine the 2 blocks to form the bigger block
- IF the operator is H
  - X co-ordinate of the Second Block = X Co –ordinate of the block
  - Y co-ordinate of the Second Block = Y Co-ordinate of the block
  - X co-ordinate of the First Block = X Co –ordinate of the block
  - Y co-ordinate of the First Block = Y Co-ordinate of the block + Height of Second Block
- IF the operator is V
  - X co-ordinate of the Second Block = X Co –ordinate of the block + width of First Block
  - Y co-ordinate of the Second Block = Y Co-ordinate of the block
  - X co-ordinate of the First Block = X Co –ordinate of the block
  - Y co-ordinate of the First Block = Y Co-ordinate of the block

X and Y Co-ordinate computation

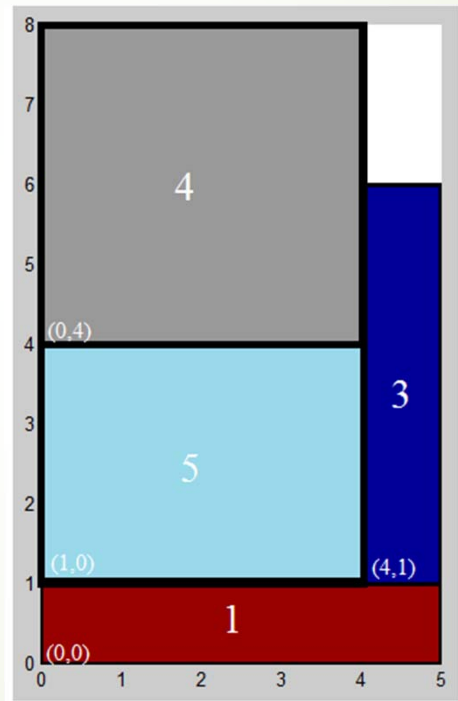
Final Polish Expression : 4,2,0,V,H,3,V,1,H



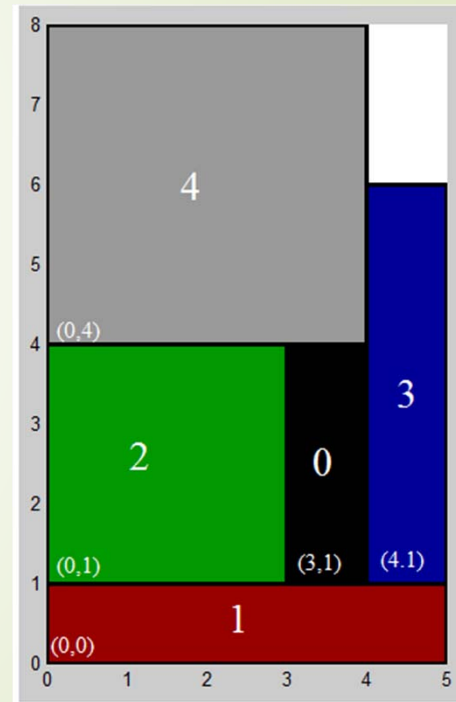
7,1,H



6,3,V,1,H



4,5,H,3,V,1,H



4,2,0,V,H,3,V,1,H

Simulated Annealing

- Locating a good approximation to the global optimum of a given function in a large search space.

```
Algorithm: Simulated_Annealing_Floorplanning(P, ε, r, k)
begin
E ← 12V3V4V...nV; /* initial solution */
Best ← E; T0 ← Δavg/ln(P); M ← MT ← uphill ← 0; N = kn;
repeat
  MT ← uphill ← reject ← 0;
  repeat
     SelectMove(M);
     Case M of
     M1: Select two adjacent operands ei and ej; NE ← Swap(E, ei, ej);
     M2: Select a nonzero length chain C; NE ← Complement(E, C);
     M3: done ← FALSE;
        while not (done) do
           Select two adjacent operand ei and operator ei+1;
           if (ei−1 ≠ ei+1) and (2Ni+1 < i) then done ← TRUE;
        NE ← Swap(E, ei, ei+1);
     MT ← MT + 1; Δcost ← cost(NE) − cost(E);
     if (Δcost ≤ 0) or (Random < e^(−Δcost/T))
     then
        if (Δcost > 0) then uphill ← uphill + 1;
        E ← NE;
        if cost(E) < cost(best) then best ← E;
     else reject ← reject + 1;
  until (uphill > N) or (MT > 2N);
  T = rT; /* reduce temperature */
until (reject/MT > 0.95) or (T < ε) or OutOfTime;
end
```

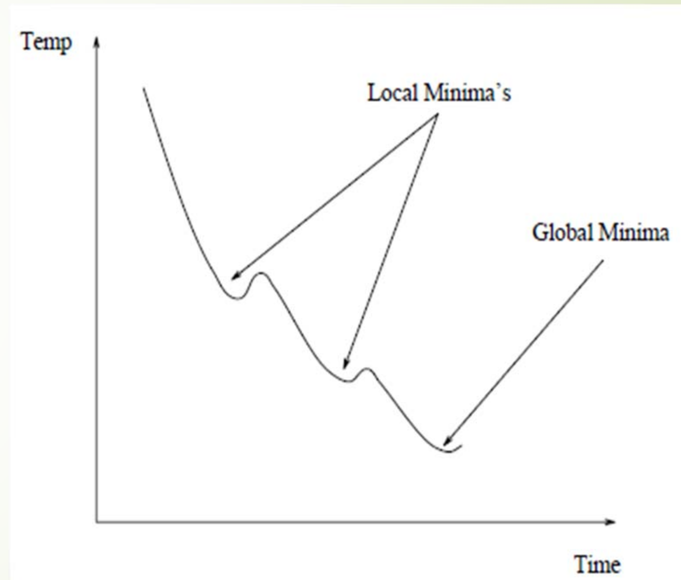
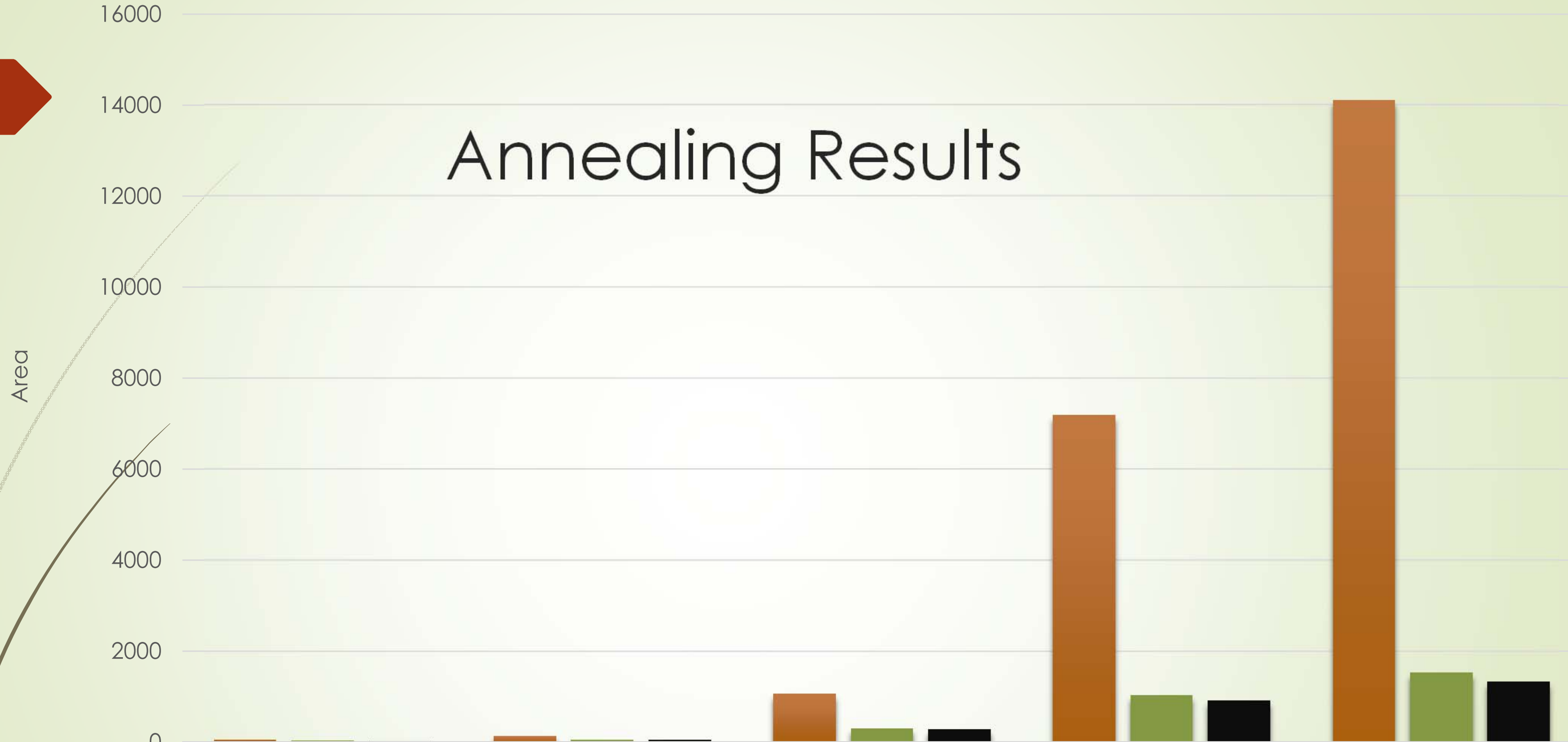


Figure taken from Prof Sung Kyu Lim Lecture

Annealing Results



	5_block.ple	10_block.ple	30_block.ple	100_block.ple	150_block.ple
Initial Area	65	147	1075	7199	14104
Final Area	40	70	315	1044	1540
Total Block Area	38	68	290	920	1342
% White Space	5	2.8	7.9	11.8	12.8

Tuning Parameters

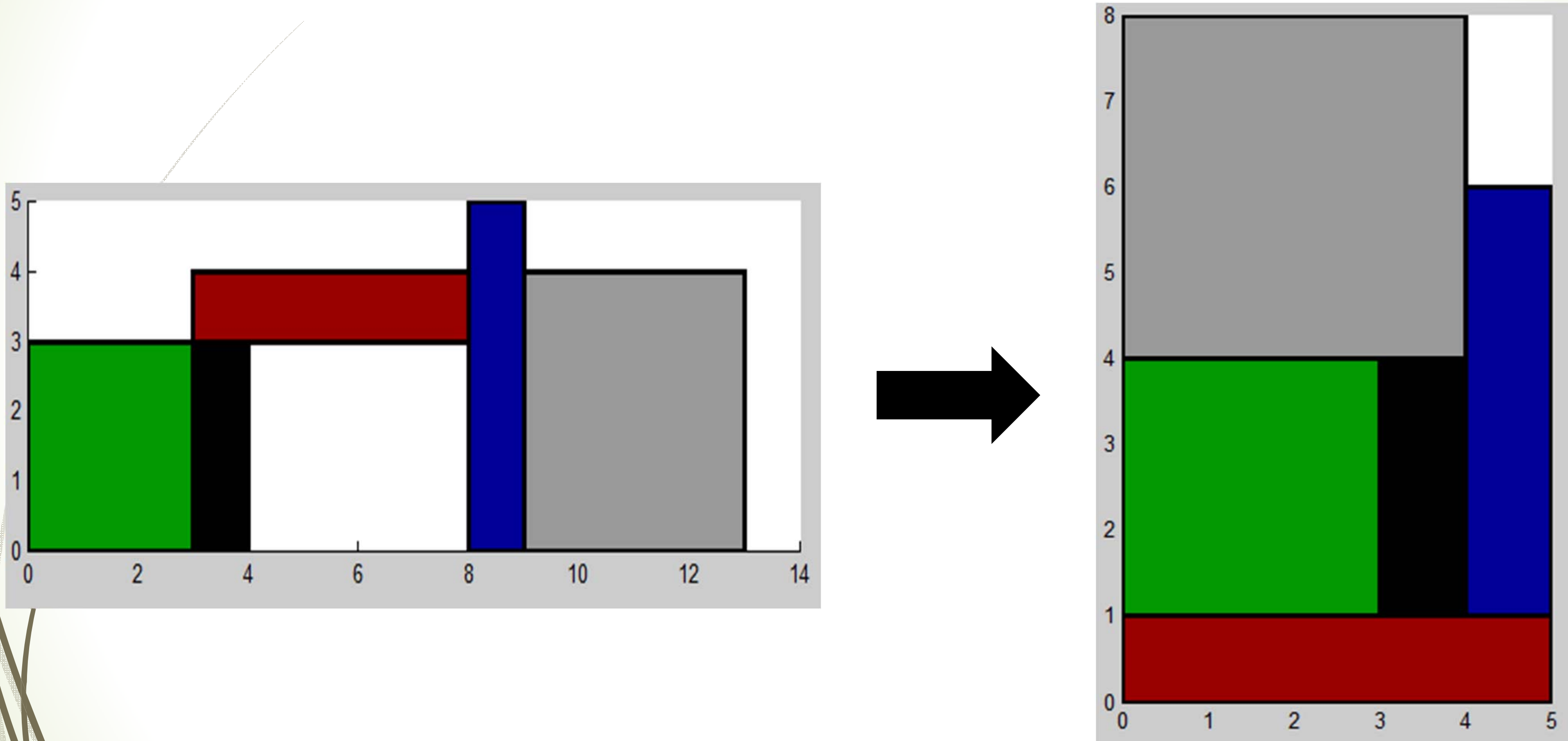
- The runtime and quality of solution depends on various parameters..
- k : Number of moves allowed at each temperature level.
- P : Initial Probability for deciding the starting temperature.
- ε : Lowest Temperature until which annealing is performed.
- r : Temperature reducing factor.

Nodes	k	P	r	ε	Avg Time
5	10	0.85(T0= 144)	0.85	0.1	0.4 sec
10	15	0.95(T0=539)	0.95	0.01	4.2 sec
30	50	0.95(T0=1764)	0.95	0.00001	153 sec
100	100	0.95(T0=2000)	0.85	0.0000001	37.8 min
150	50	0.9(T0=4187)	0.95	0.000000001	61.6 min

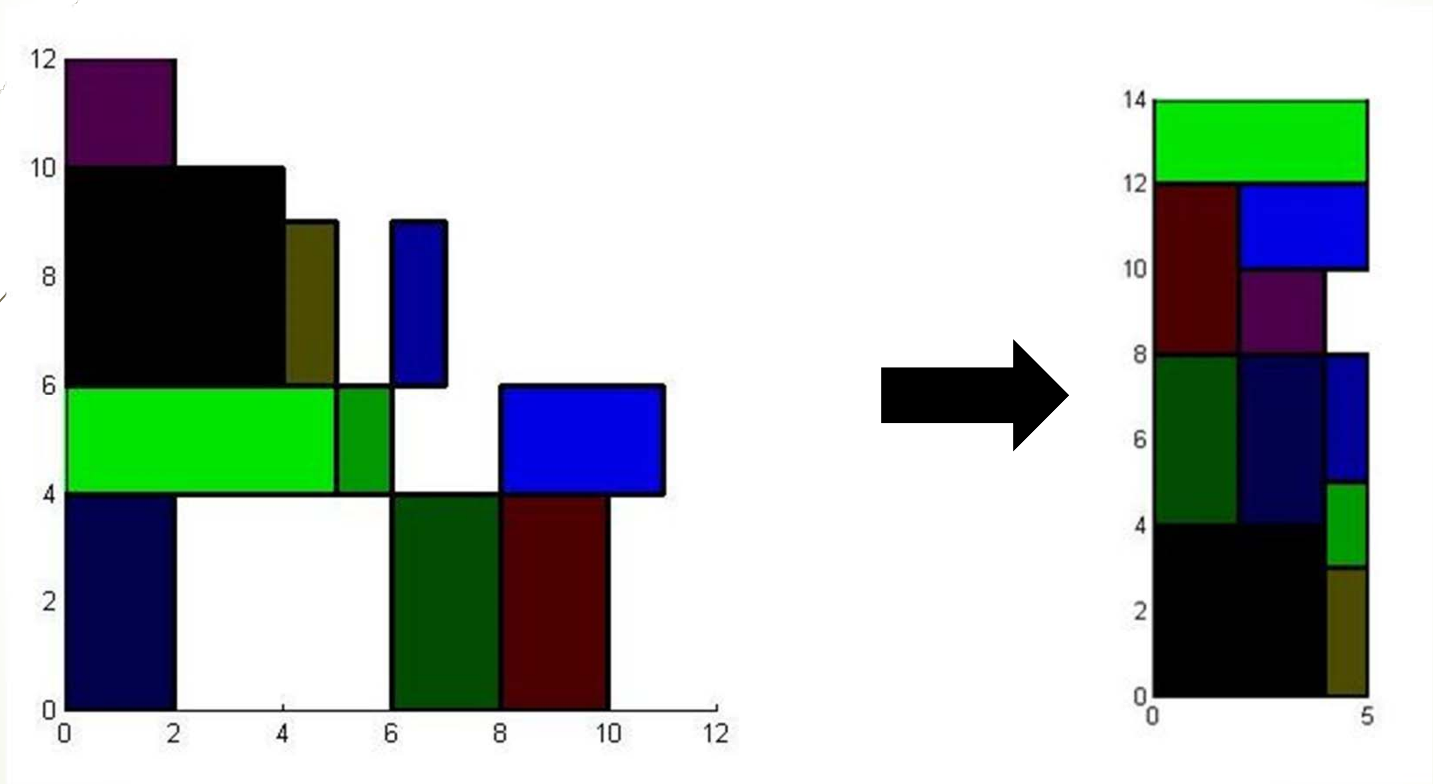
Tradeoff between quality and runtime(100 Nodes)

K	P	R	ε	Final Area	Avg Time
20	0.9	0.85	0.0001	1462	322sec
50	0.9	0.95	0.00000001	1100	7998sec

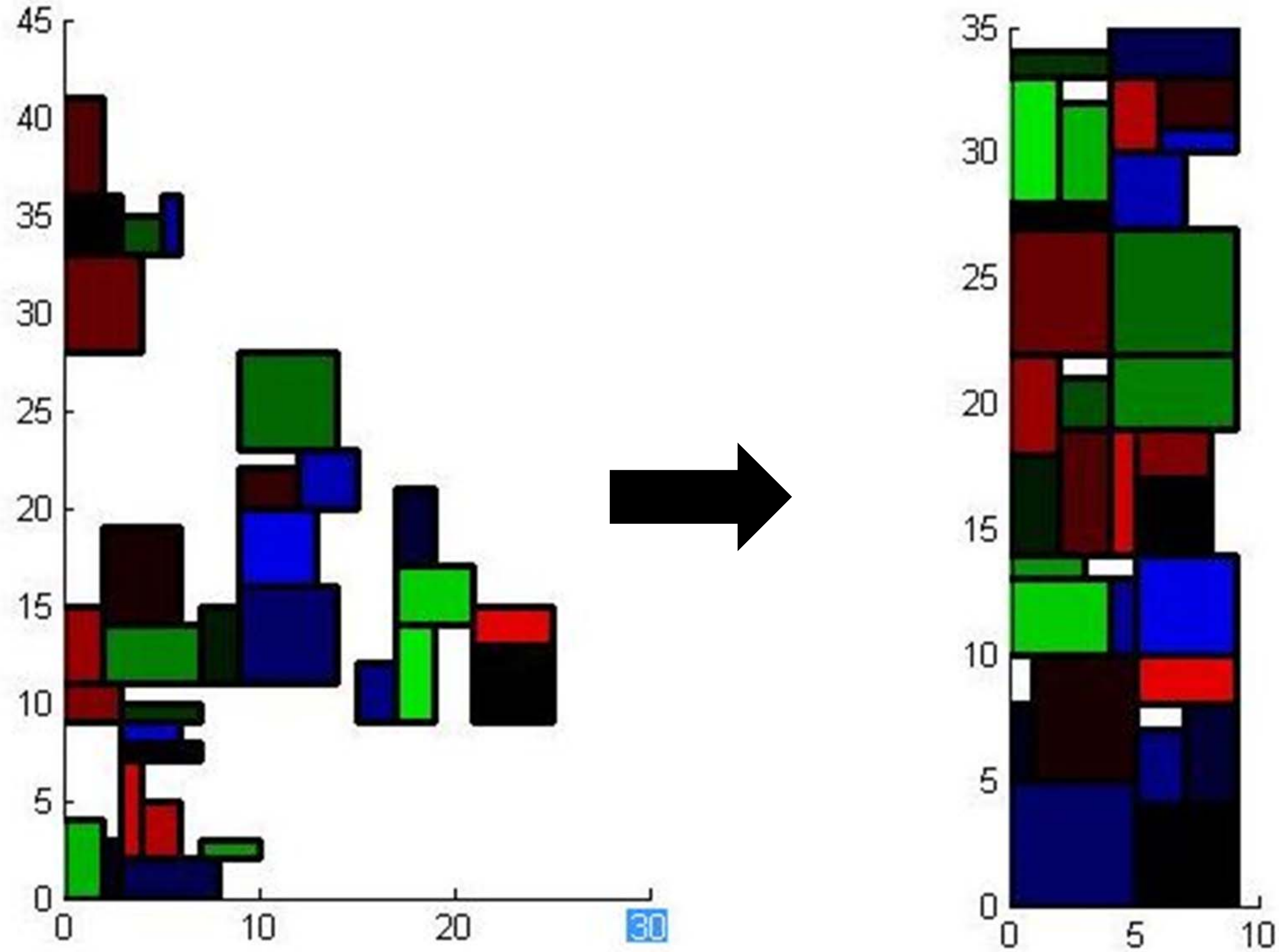
Transformation of 5 Blocks



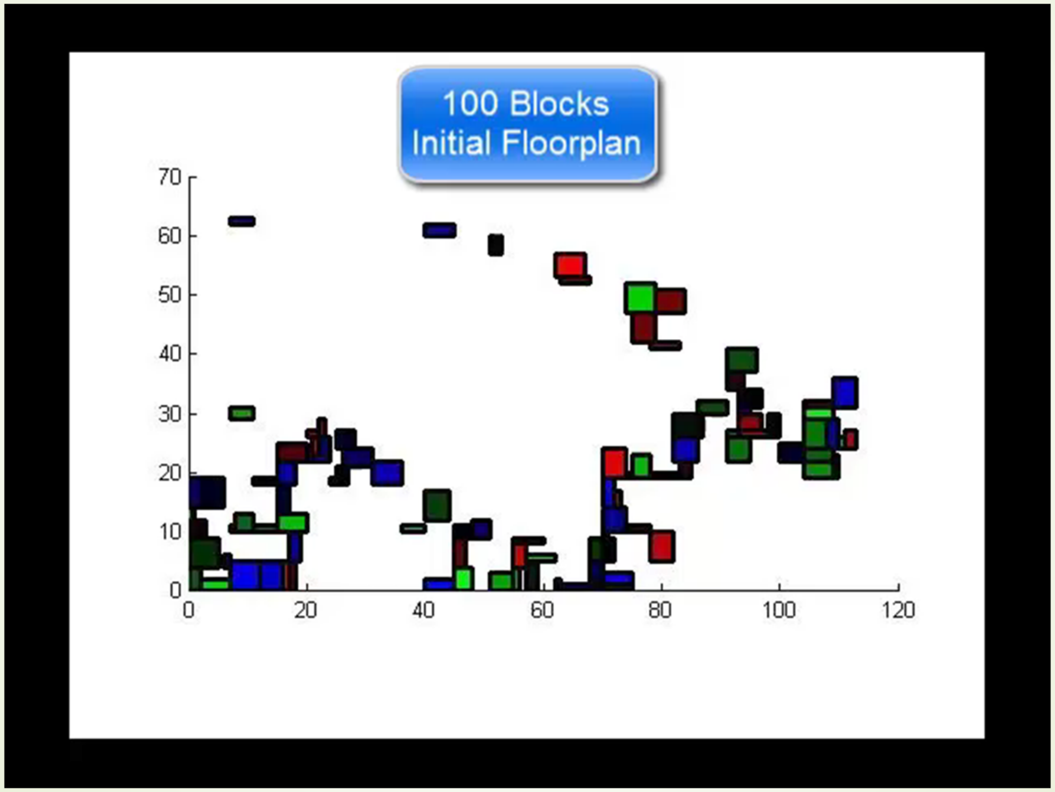
Transformation of 10 Blocks



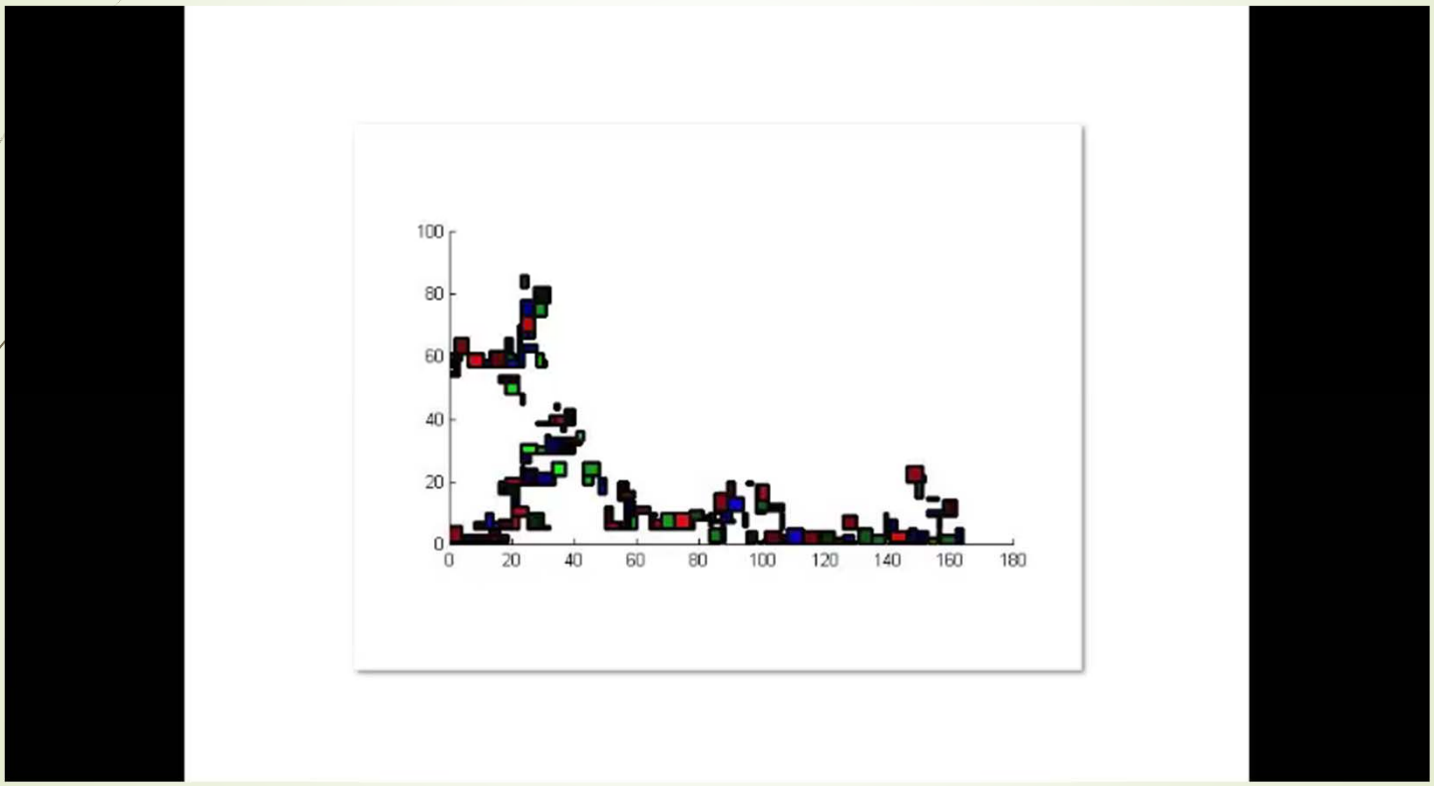
Transformation of 30 Blocks



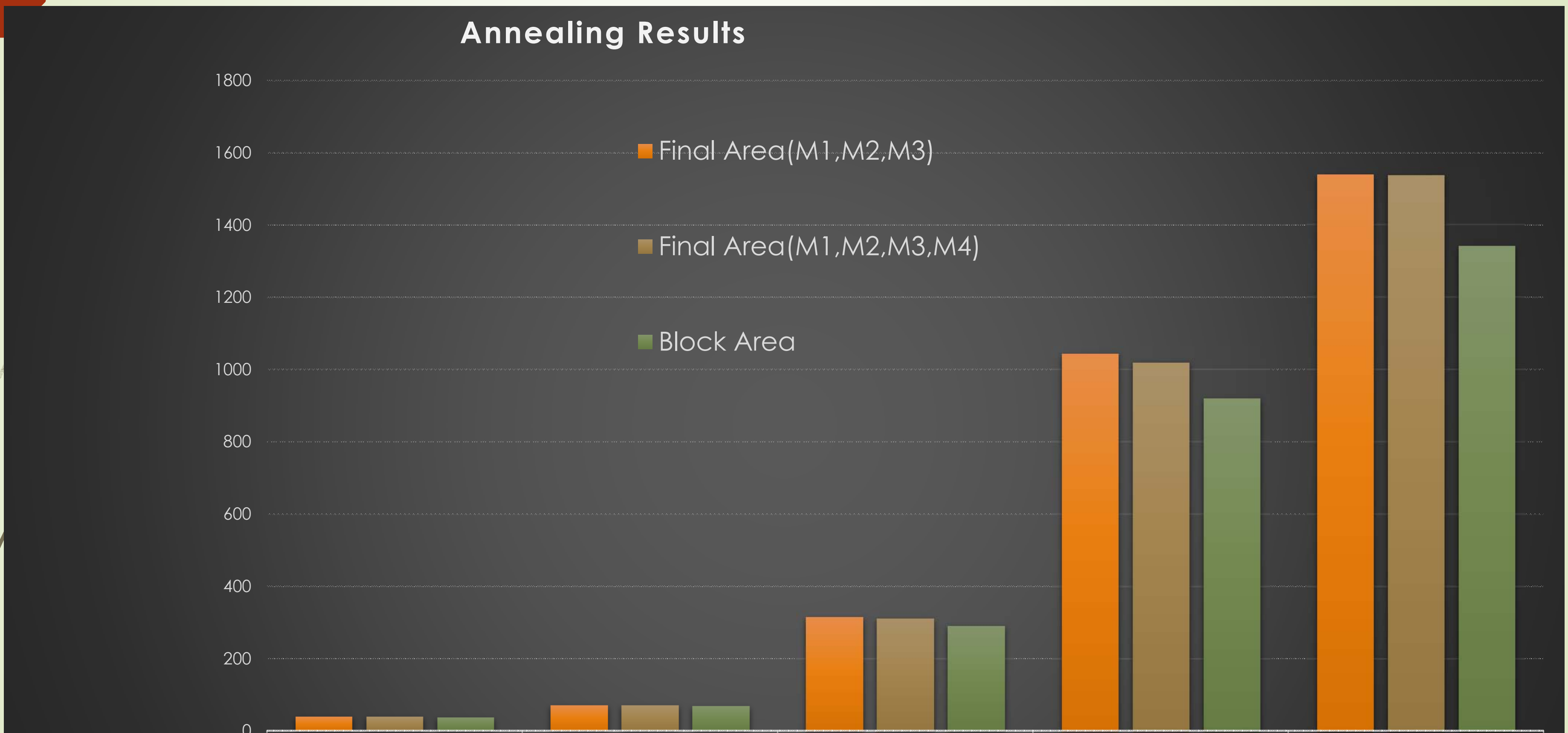
100 Block Video



150 Block Video



Allowing rotation of Block(M4 move)



	5_block	10_block	30_block	100_block	150_block
Final Area(M1,M2,M3)	40	70	315	1044	1540
Final Area(M1,M2,M3,M4)	40	70	312	1020	1537
Block Area	38	68	290	920	1342

Conclusion

- Simulated Annealing is an effective method for floorplanning as this methods helps us explore more solution space and increase the chance of finding good quality solution
- There is a tradeoff between quality of solution and execution time.
- The execution time depends on the tuning parameters , so arriving at optimal set of values these parameters is time consuming.

Possible Improvements

- Aspect Ratio may also be considered as another factor for the cost function.
- Wirelength also can be added to the cost function to minimize the wirelength as well
- An efficient method to compute the tuning parameters needs to be devised.