VL724 - VLSI Design Automation Project (14EC206) Documentation

Release 3.0

Akshay Revankar

Apr 10, 2018

CHAPTER

ONE

PACKAGE DEPENDENCIES

The following packages are needed for running the project codes:

- Numpy
- Matplotlib
- NetworkX

CHAPTER

TWO

ASSIGNMENT 1 - KERNIGHAN LIN ALGORITHM

Contents

ISCAS‘85 PARSER

The ISCAS ‘85 benchmark circuits are ten combinational networks provided to authors at the 1985 International Symposium on Circuits And Systems. They subsequently have been used by many researchers as a basis for comparing results in the area of test generation.

The ISCAS ‘85 netlists contains information not present in most other netlist formats.For instance the ISCAS ‘85 format lists each network node in levelized order, and this information may be lost when translating to other formats. Also, the ISCAS ‘85 format lists fanout branches separately as distinct nodes (with distinct names) and specifies the connectivity for each fanoutbranch. This is valuable for test generation purposes and must be extracted from other, more generic, netlists.

More information on the ISCAS‘85 netlist format can be found here: `ISCAS Netlist format`

parse_iscas85(*input_filename*)

The function parses the ISCAS‘85 .isc netlist given in the input_filename parameter and the outputs are as follows:

Returns

- node_names: The node list
- matrix adjacency: The Adjacency matrix

print_distance_matrix(*nl*, *nodes*, *adjacency*[, *mat_name="Adjacency"*])

The function prints a matrix in a readable format where the inputs are as follows:

Parameters

- **nl** (*int*) – The number of nodes
- **nodes** (*array*) – A list containing the name of the nodes
- **adjacency** (*matrix*) – The adjacency matrix that needs to be displayed
- **mat_name** (*str*) – (default : “Adjacency”) - The matrix name to be displayed while printing the matrix.

Graph Plotter

I have tried representing the partitions through a plot. The plots will not be displayed if the number of nodes > 200.

This file contains functions to plot different graphs that we will come across in this project:

show_graph_with_labels(*adjacency_matrix*, *mylabels*)

The function plots the graph along with node names given an adjacency matrix

Parameters

- **adjacency_matrix** (*matrix*) – The input graph Adjacency Matrix
- **mylabels** (*array*) – The node labels / names in same order as the adjacency_matrix rows

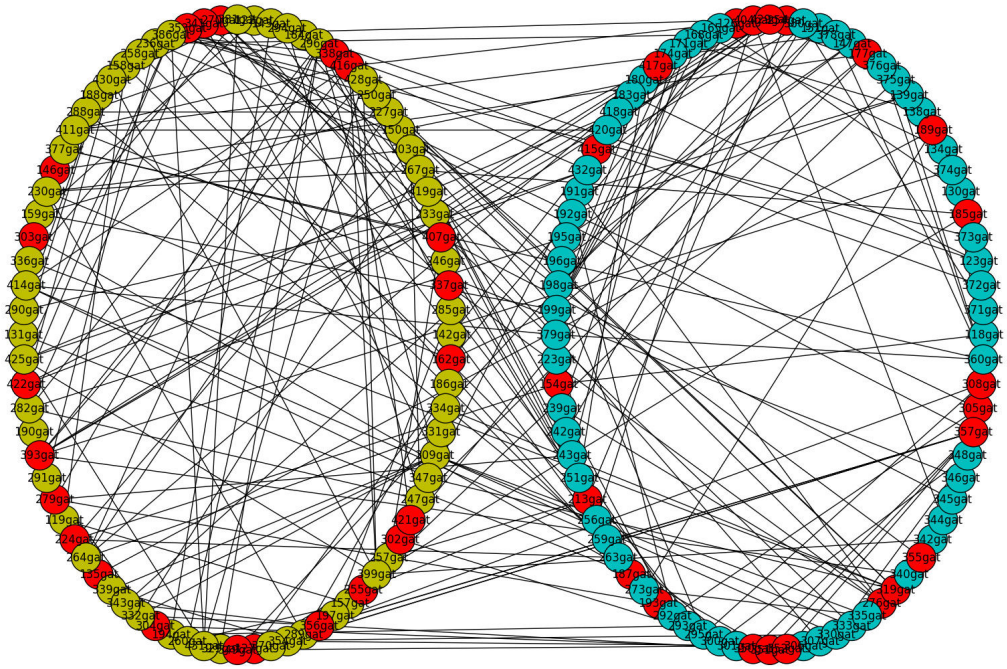
show_partitioned_graph_with_labels (*fig*, *adjacency_matrix*, *mylabels*, *colors*, *positions*)

The function plots the graph alongwith node names given an adjacency matrix. The difference from the previous function is that is allows to specify the figure handle, the color and position of each nodes

Parameters

- **fig** (*figure_handle*) – The matplotlib figure handle
- **adjacency_matrix** (*matrix*) – The input graph Adjacency Matrix
- **mylabels** (*array*) – The node labels / names in same order as the adjacency_matrix rows
- **colors** (*str*) – String containing the colors (as per matplotlib standard) for each node in order as they appear in the node list
- **positions** (*dict*) – a dictionary containing the node index (as it appears in the node list) and a tuple specifying the (x,y) coordinates of the nodes

Example Plot:



The two distinct circles above represent the two partitions and the nodes in red are the nodes currently locked (KL Algorithm)

Kernighan Lin Algorithm

Description

The input to the algorithm is an undirected graph G = (V,E) with vertex set V, edge set E, and (optionally) numerical weights on the edges in E. The goal of the algorithm is to partition V into two disjoint subsets A and B of equal (or nearly equal) size, in a way that minimizes the sum T of the weights of the subset of edges that cross from A to B. If the graph is unweighted, then instead the goal is to minimize the number of crossing edges; this is equivalent to assigning weight one to each edge. The algorithm maintains and improves a partition, in each pass using a greedy algorithm to pair up vertices of A with vertices of B, so that moving the paired vertices from one side of the partition to the other will improve the partition. After matching the vertices, it then performs a subset of the pairs chosen to have the best overall effect on the solution quality T. Given a graph with n vertices, each pass of the algorithm runs in time $O(n^2 logn)$.

In more detail, let I_a be the internal cost of a, that is, the sum of the costs of edges between a and other nodes in A, and let E_a be the external cost of a , that is, the sum of the costs of edges between a and nodes in B. Furthermore, let

$$D_a = E_a - I_a$$

be the difference between the external and internal costs of a . If a and a are interchanged, then the reduction in cost is:

$$T_{old} - T_{new} = D_a + D_b - 2c_{a,b}$$

where $c_{a,b}$ is the cost of the possible edge between a and b.

kl_perform (*filename*, *order="inorder"*, *plot=""*)

Parameters

- **filename** (*str*) – The .isc ISCAS'85 file to be used for KL Partitioning
- **order** (*str*) – inorder | random (default: inorder) If the option is *inorder*, the initial partition in always split in half in the order in which the nodes appear in the .isc file. If the option is *random*, the graph is partitioned into two randomly
- **plot** (*str*) – blank => no animated graph plotting animate => animated graph plotting on each step

•Performs KL Partitioning on a given ISCAS file (excludes input pads).

•Calls *parse_iscas85()* inorder to parse the isc file (provided in *filename*)

•Partitions initially inorder into two or randomly (option provided in *order* parameter)

•Shows animated graph (provided number of nodes < 200) if the parameter *plot* = *animate*

How to execute?

Call

```
python3 kl.py -i <input_isc_file> -o <inorder|random> (-a)
```

Required Arguments:

-i, --input	The input file name for processing the graph
-o, --order	The order to initially partition. inorder | random. For more information refer: *kl_perform()*

Optional Arguments:

-a, --animate	Shows animated graph if argument is mentioned during script call

Example

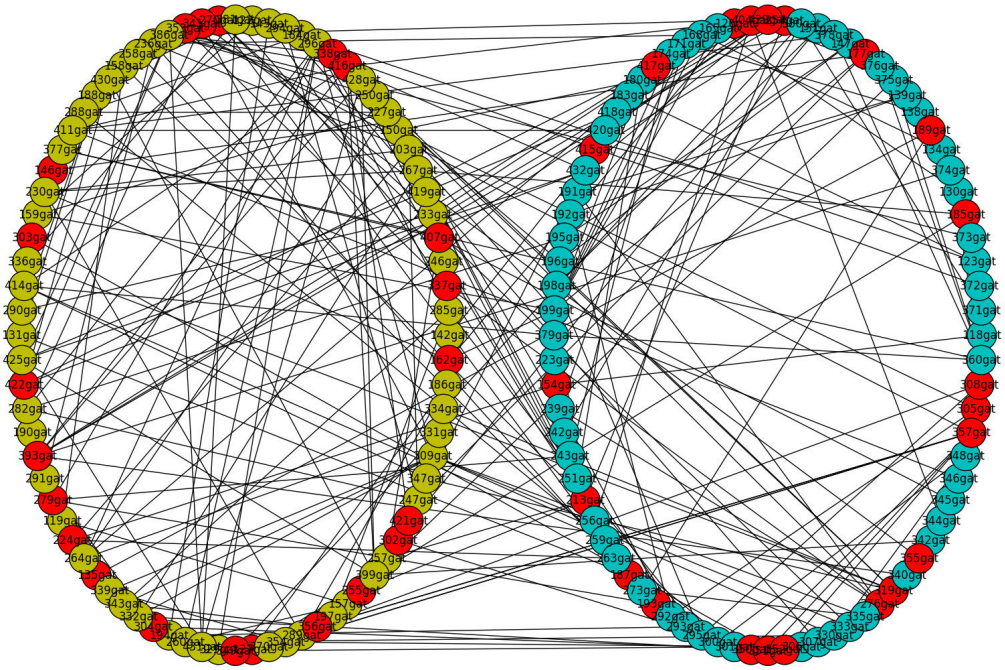
```
python3 kl.py -i c17.isc -o random -a
```

Example Output

For c432.isc

```
ISCAS'85 Netlist Parse Complete!

Number of nodes are too large. Not printing Adjacency matrix!

INORDER Initial Partition:
Partition A ['118gat', '119gat', '122gat', '123gat', '126gat', '127gat', '130gat',
 ↪'131gat', '134gat', '135gat', '138gat', '139gat', '142gat', '143gat', '146gat',
 ↪'147gat', '150gat', '151gat', '154gat', '157gat', '158gat', '159gat', '162gat',
 ↪'165gat', '168gat', '171gat', '174gat', '177gat', '180gat', '183gat', '184gat',
 ↪'185gat', '186gat', '187gat', '188gat', '189gat', '190gat', '191gat', '192gat',
 ↪'193gat', '194gat', '195gat', '196gat', '197gat', '198gat', '199gat', '203gat',
 ↪'213gat', '223gat', '224gat', '227gat', '230gat', '233gat', '236gat', '239gat',
 ↪'242gat', '243gat', '246gat', '247gat', '250gat', '251gat', '254gat', '255gat',
 ↪'256gat', '257gat', '258gat', '259gat', '260gat', '263gat', '264gat', '267gat',
 ↪'270gat', '273gat', '276gat', '279gat', '282gat', '285gat', '288gat', '289gat',
 ↪'290gat']

Partition B ['291gat', '292gat', '293gat', '294gat', '295gat', '296gat', '300gat',
 ↪'301gat', '302gat', '303gat', '304gat', '305gat', '306gat', '307gat', '308gat',
 ↪'309gat', '319gat', '329gat', '330gat', '331gat', '332gat', '333gat', '334gat',
 ↪'335gat', '336gat', '337gat', '338gat', '339gat', '340gat', '341gat', '342gat',
 ↪'343gat', '344gat', '345gat', '346gat', '347gat', '348gat', '349gat', '350gat',
 ↪'351gat', '352gat', '353gat', '354gat', '355gat', '356gat', '357gat', '360gat',
 ↪'370gat', '371gat', '372gat', '373gat', '374gat', '375gat', '376gat', '377gat',
 ↪'378gat', '379gat', '380gat', '381gat', '386gat', '393gat', '399gat', '404gat',
 ↪'407gat', '411gat', '414gat', '415gat', '416gat', '417gat', '418gat', '419gat',
 ↪'420gat', '421gat', '422gat', '425gat', '428gat', '429gat', '430gat', '431gat',
 ↪'432gat']

Running KL Partitioning algorithm! Please wait...
Iteration 80 / 80. Total Time elapsed: 4154.719591 ms

Final KL Partition:
Partition A ['119gat', '123gat', '127gat', '131gat', '139gat', '143gat', '151gat',
 ↪'157gat', '158gat', '183gat', '184gat', '185gat', '186gat', '187gat', '188gat',
 ↪'191gat', '192gat', '193gat', '194gat', '197gat', '198gat', '203gat', '223gat',
 ↪'224gat', '227gat', '230gat', '233gat', '236gat', '239gat', '243gat', '247gat',
 ↪'251gat', '260gat', '263gat', '264gat', '267gat', '270gat', '276gat', '279gat',
 ↪'285gat', '288gat', '289gat', '290gat', '291gat', '292gat', '293gat', '294gat',
 ↪'295gat', '296gat', '300gat', '301gat', '302gat', '303gat', '304gat', '305gat',
 ↪'306gat', '307gat', '308gat', '309gat', '329gat', '330gat', '331gat', '332gat',
 ↪'333gat', '335gat', '337gat', '339gat', '341gat', '343gat', '348gat', '349gat',
 ↪'350gat', '351gat', '352gat', '353gat', '354gat', '355gat', '356gat', '357gat',
 ↪'370gat']

Partition B ['118gat', '122gat', '126gat', '130gat', '134gat', '135gat', '138gat',
 ↪'142gat', '146gat', '147gat', '150gat', '154gat', '159gat', '162gat', '165gat',
 ↪'168gat', '171gat', '174gat', '177gat', '189gat', '190gat', '195gat', '196gat',
 ↪'199gat', '213gat', '242gat', '246gat', '250gat', '254gat', '255gat', '256gat',
 ↪'257gat', '258gat', '259gat', '273gat', '282gat', '319gat', '334gat', '336gat',
 ↪'338gat', '340gat', '342gat', '344gat', '345gat', '346gat', '347gat', '360gat',
 ↪'371gat', '372gat', '373gat', '374gat', '375gat', '376gat', '377gat', '378gat',
 ↪'379gat', '380gat', '381gat', '386gat', '393gat', '399gat', '404gat', '407gat',
 ↪'411gat', '414gat', '415gat', '416gat', '417gat', '418gat', '419gat', '420gat',
 ↪'421gat', '422gat', '425gat', '428gat', '429gat', '430gat', '431gat', '432gat',
 ↪'180gat']
```

```
Total time taken : 4154.886007 milliseconds

Initial number of cuts = 41
Minimum number of cuts = 21
```



CHAPTER

THREE

ASSIGNMENT 2 - SIMULATED ANNEALING ALGORITHM

Contents

Graph Plotter

I have tried representing the partitions through a plot. The plots will not be displayed if the number of nodes > 200.

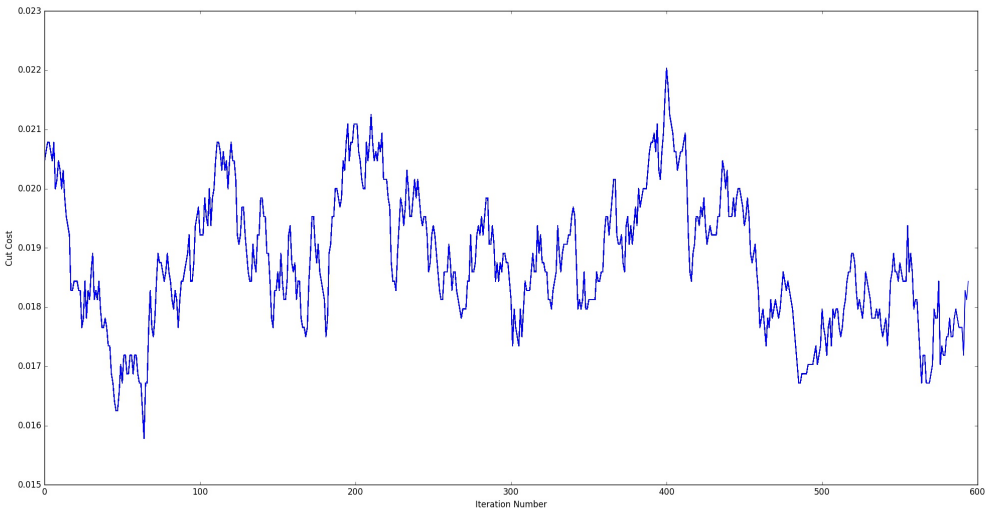
This file contains functions to plot different graphs that we will come across in this project:

cost_plotter(fig,curr_cost,index):

Parameters

- **fig** (*figure_handle*) – The matplotlib figure handle
- **curr_cost** (*array*) – Array of cost for each iteration at current iteration
- **index** (*int*) – The current iteration number

Example Plot:



The cost for SA is taken as the current number of cuts divided by the product of the number of elements in each partition (ie. Cut Ratio) (SA Algorithm).

Simulated Annealing Algorithm

Description

sa_perform(*filename*, *order*, *t*, *r*, *cut_ratio*, *stop_iterations*)

Parameters

- **filename** (*str*) – The .isc ISCAS'85 file to be used for KL Partitioning
- **order** (*str*) – inorder | random (default: inorder) If the option is *inorder*, the initial partition in always split in half in the order in which the nodes appear in the .isc file. If the option is *random*, the graph is partitioned into two randomly
- **t** (*int*) – The initial temperature to start the system at.
- **r** (*float*) – Temperature reduction ratio (0 to 1)
- **cut_ratio** (*str*) – The minimum partition ratio that is ((number of elements in one partition)/(number of elements in other partition))
- **stop_iterations** (*str*) – The maximum number of previous temperature iterations to check for cost reduction before stopping the program

•Performs SA Partitioning on a given ISCAS file (excludes input pads).

•Calls *parse_iscas85()* inorder to parse the isc file (provided in *filename*)

•Partitions initially inorder into two or randomly (option provided in *order* parameter)

•Shows the cost vs time graph

How to execute?

Call

```
python3 sa.py -i <input_isc_file> -o <inorder|random> (-t <initial_
↪temperature> -r <temp_reduction_ratio> -p <partition_ratio> -s <max_
↪stop_iterations>)
```

Use Ctrl+C (SIGNINT) to stop the program and output the best solution

Required Arguments:

-i, --input	The input file name for processing the graph
-o, --order	The order to initially partition. inorder | random. For more information refer: *sa_perform()*

Optional Arguments:

-t, --temperature	The initial temperature to start the system at. (default = 10)
-r, --ratio	Temperature reduction ratio (0 to 1) (default = 0.1)
-p, --partition_ratio	The minimum partition ratio that is ((number of elements in one partition)/(number of elements in other partition)) (default = 1 that is equal sized partitions)
-s, --stop_iterations	The maximum number of previous temperature iterations to check for cost reduction before stopping the program

Example

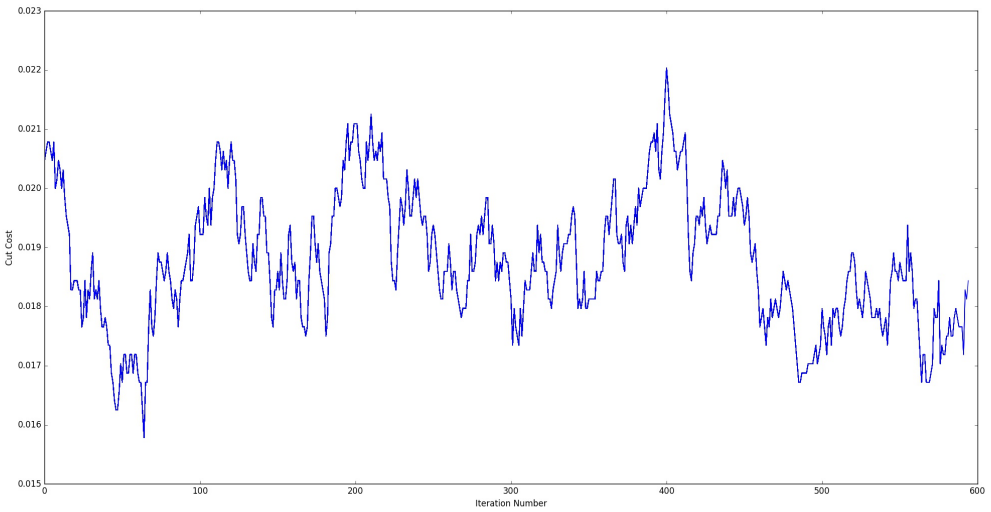
```
python3 sa.py -i c17.isc -o random -t 10 -r 0.1 -p 0.1 -s 3
```

Example Output

For c432.isc

```
ISCAS'85 Netlist Parse Complete!

Number of nodes are too large. Not printing Adjacency matrix!

RANDOM Initial Partition:
Partition A ['296gat', '414gat', '416gat', '195gat', '142gat', '341gat', '425gat',
 ↪'233gat', '258gat', '346gat', '193gat', '347gat', '151gat', '351gat', '355gat',
 ↪'203gat', '306gat', '431gat', '184gat', '260gat', '329gat', '343gat', '357gat',
 ↪'198gat', '292gat', '243gat', '348gat', '304gat', '334gat', '332gat', '419gat',
 ↪'146gat', '119gat', '333gat', '349gat', '300gat', '250gat', '174gat', '289gat',
 ↪'127gat', '302gat', '126gat', '290gat', '371gat', '150gat', '415gat', '330gat',
 ↪'376gat', '337gat', '190gat', '180gat', '407gat', '264gat', '192gat', '418gat',
 ↪'303gat', '340gat', '344gat', '381gat', '118gat', '372gat', '335gat', '432gat',
 ↪'157gat', '375gat', '165gat', '188gat', '183gat', '123gat', '421gat', '147gat',
 ↪'282gat', '294gat', '339gat', '417gat', '158gat', '308gat', '379gat', '380gat',
 ↪'242gat']

Partition B ['122gat', '373gat', '374gat', '130gat', '131gat', '134gat', '135gat',
 ↪'138gat', '139gat', '377gat', '143gat', '378gat', '386gat', '393gat', '399gat',
 ↪'154gat', '404gat', '411gat', '159gat', '162gat', '420gat', '168gat', '171gat',
 ↪'422gat', '177gat', '428gat', '429gat', '430gat', '185gat', '186gat', '187gat',
 ↪'189gat', '191gat', '194gat', '196gat', '197gat', '199gat', '213gat', '223gat',
 ↪'224gat', '227gat', '230gat', '236gat', '239gat', '246gat', '247gat', '251gat',
 ↪'254gat', '255gat', '256gat', '257gat', '259gat', '263gat', '267gat', '270gat',
 ↪'273gat', '276gat', '279gat', '285gat', '288gat', '291gat', '293gat', '295gat',
 ↪'301gat', '305gat', '307gat', '309gat', '319gat', '331gat', '336gat', '338gat',
 ↪'342gat', '345gat', '350gat', '352gat', '353gat', '354gat', '356gat', '360gat',
 ↪'370gat']

Running SA Partitioning algorithm! Please wait...
^C

Final SA Partition:
Partition A ['296gat', '416gat', '195gat', '142gat', '258gat', '346gat', '347gat',
 ↪'151gat', '203gat', '260gat', '329gat', '198gat', '243gat', '348gat', '334gat',
 ↪'146gat', '333gat', '349gat', '289gat', '127gat', '126gat', '290gat', '415gat',
 ↪'264gat', '418gat', '303gat', '344gat', '381gat', '118gat', '335gat', '432gat',
 ↪'157gat', '375gat', '188gat', '183gat', '421gat', '147gat', '282gat', '379gat',
 ↪'380gat', '242gat', '197gat', '374gat', '353gat', '309gat', '319gat', '130gat',
 ↪'159gat', '177gat', '162gat', '338gat', '337gat', '407gat', '259gat', '295gat',
 ↪'193gat', '186gat', '185gat', '273gat', '378gat', '138gat', '233gat', '404gat',
 ↪'199gat', '223gat', '336gat', '339gat', '414gat', '251gat', '158gat', '196gat',
 ↪'227gat', '276gat', '165gat', '174gat', '135gat', '293gat', '345gat', '131gat',
 ↪'341gat']

Partition B ['122gat', '373gat', '134gat', '139gat', '377gat', '386gat', '393gat',
 ↪'399gat', '154gat', '168gat', '422gat', '187gat', '189gat', '191gat', '194gat',
 ↪'213gat', '224gat', '230gat', '239gat', '246gat', '247gat', '254gat', '256gat',
 ↪'257gat', '263gat', '267gat', '270gat', '279gat', '285gat', '288gat', '291gat',
 ↪'301gat', '305gat', '331gat', '342gat', '350gat', '352gat', '356gat', '360gat',
 ↪'370gat', '308gat', '294gat', '417gat', '255gat', '431gat', '332gat', '372gat',
 ↪'190gat', '330gat', '306gat', '425gat', '357gat', '184gat', '192gat', '430gat',
 ↪'354gat', '302gat', '304gat', '307gat', '123gat', '236gat', '376gat', '355gat',
 ↪'420gat', '143gat', '300gat', '119gat', '429gat', '419gat', '428gat', '250gat',
 ↪'180gat', '371gat', '411gat', '340gat', '343gat', '351gat', '292gat', '171gat',
 ↪'150gat']
```

```
Total time taken : 272935.472965 milliseconds

Initial number of cuts = 132
Minimum number of cuts = 102. Partition size A:B = 80:80
```



CHAPTER

FOUR

ASSIGNMENT 3 - PLACEMENT AND ROUTING

Contents

Simulated Annealing Algorithm for Placement

Description

placement_perform(*filename*, *peorder*, *t*, *r*, *stop_iterations*, *animate*, *padding*)

Parameters

- **filename** (*str*) – The .isc ISCAS'85 file to be used for KL Partitioning
- **peorder** (*str*) – H | V (default: V) Initial order for the Polish Expression
- **t** (*int*) – The initial temperature to start the system at.
- **r** (*float*) – Temperature reduction ratio (0 to 1)
- **stop_iterations** (*str*) – The maximum number of previous temperature iterations to check for cost reduction before stopping the program
- **animate** – Animates the simulated annealing placement progress. Avoid for large netlist!!

•Performs SA Placement on a given ISCAS file (excludes input pads).

•Calls *parse_iscas85()* inorder to parse the isc file (provided in *filename*)

•Partitions initially in the format or 01V2V3V4V.... or 01H2H3H4H.... depending on the value of peorder

•Shows the final placed floorplan

•Allows to select old runs to directly plot the floorplan

Maze routing implementation using BFS/DFS

Description

routing_perform(*filename*, *data_file*, *bend_cost*, *via_cost*, *animate*, *layers*)

Parameters

- **filename** (*str*) – The .isc ISCAS'85 file to be used for KL Partitioning
- **data_file** (*str*) – The previously saved placement floorplan data.
- **bend_cost** (*float*) – Specify the cost for bends in routing (default : 0)
- **via_cost** (*float*) – Specify the cost for vias in routing (default : 0)

- **layers** (*int*) – Number of layers to use (default : 2)
- **animate** – Animates the simulated annealing placement progress. Avoid for large netlist!!

•Performs Maze routing on a given ISCAS file (excludes input pads) and the previous placed floorplan from the placement run.

•Calls *parse_iscas85()* inorder to parse the isc file (provided in *filename*)

•Routes the previously placed floorplan using two layers, via and bend costs

•Shows the final placed floorplan and the routes in two layer (Layer 1 - Blue, Layer 2 - Red)

Plotter for Placement and Routing

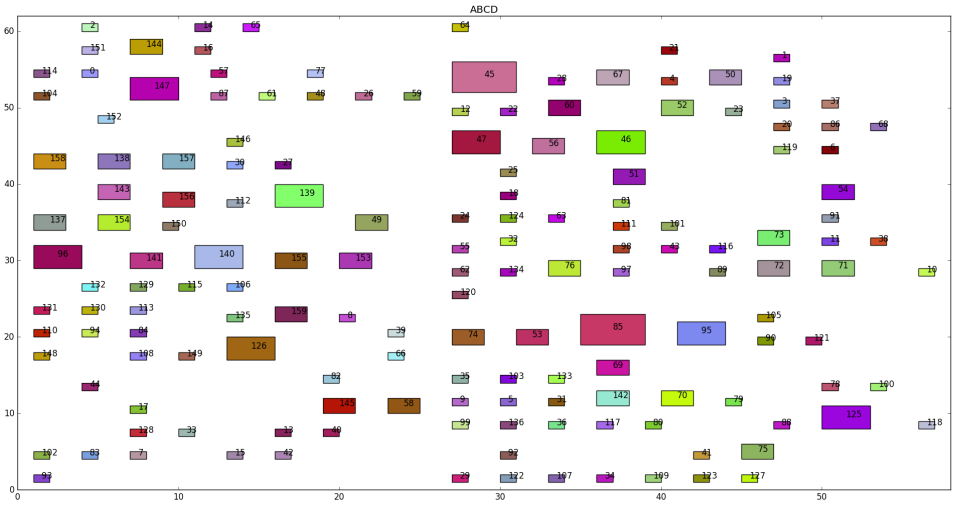
I have tried representing the entire routed floorplan through a plot (Layer 1 - Blue, Layer 2 - Red, Layer 3 - Yellow, Layer 4 - Green , Other layers - random). The vias are marked with yellow circles.

placement_plotter(fig,mylabels,nodes,root_node,adjacency_matrix,params,padding):

Parameters

- **fig** (*figure_handle*) – The matplotlib figure handle
- **mylabels** (*array*) – The node labels / names in same order as the adjacency_matrix rows
- **root_node** (*dict*) – Dictionary containing the entire tree of the polish expression wiith root node as the starting index.
- **adjacency_matrix** (*matrix*) – The input graph Adjacency Matrix
- **params** (*tuple*) – Tuple containing best result information like area, cost
- **padding** (*int*) – Padding used in the placement of nodes

Example Plot:



routing_plotter(fig,mylabels,nodes,root_node,grid,padding,bend_cost=0,via_cost=0):

Parameters

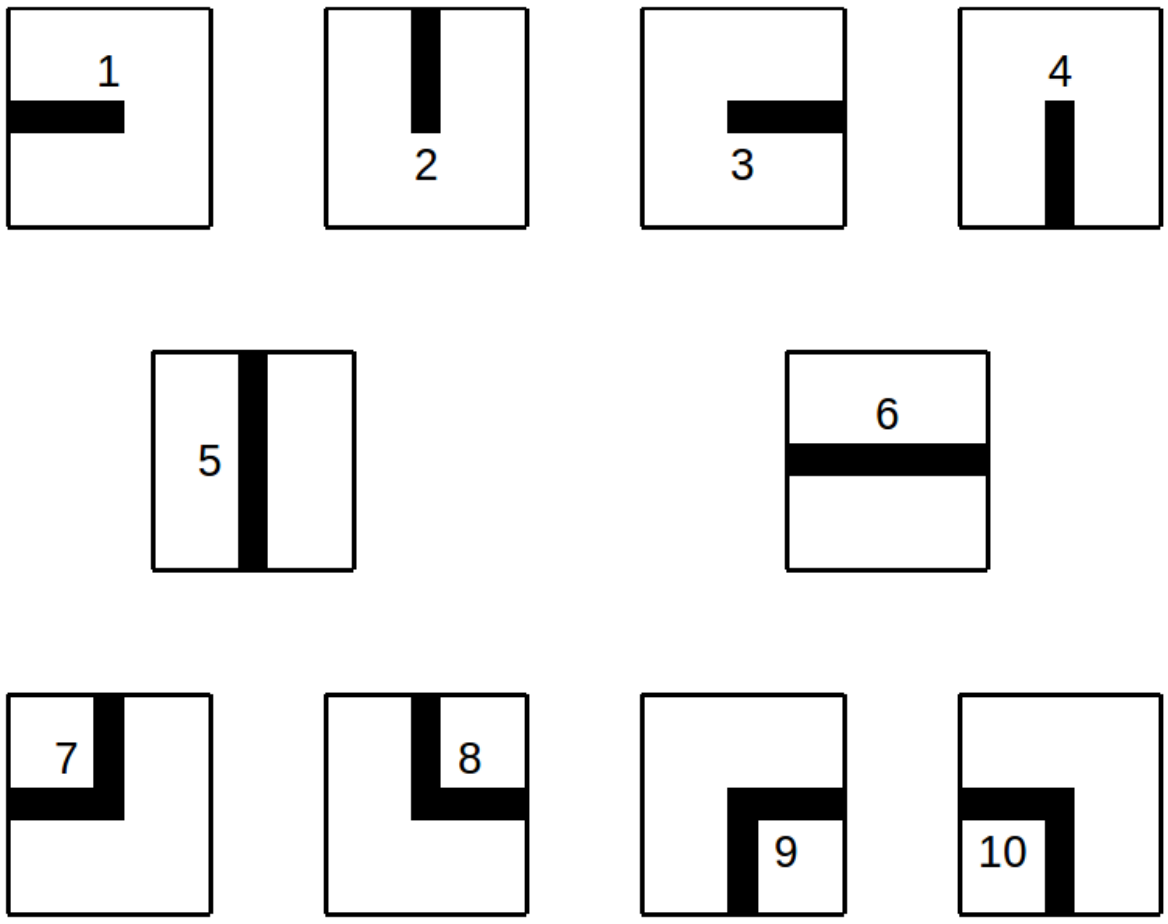
- **fig** (*figure_handle*) – The matplotlib figure handle
- **mylabels** (*array*) – The node labels / names in same order as the adjacency_matrix rows
- **nodes** (*array*) – The node list with a dictionary containing original node information such as width,height
- **root_node** (*dict*) – Dictionary containing the entire tree of the polish expression wiith root node as the starting index.
- **grid** (*matrix*) – The grid is a matrix with dictionary containing the routing and block information.
- **padding** (*int*) – Padding used in the placement of nodes
- **bend_cost** (*int*) – Cost used for bends during routing
- **via_cost** (*int*) – Cost used for vias during routing

Grid

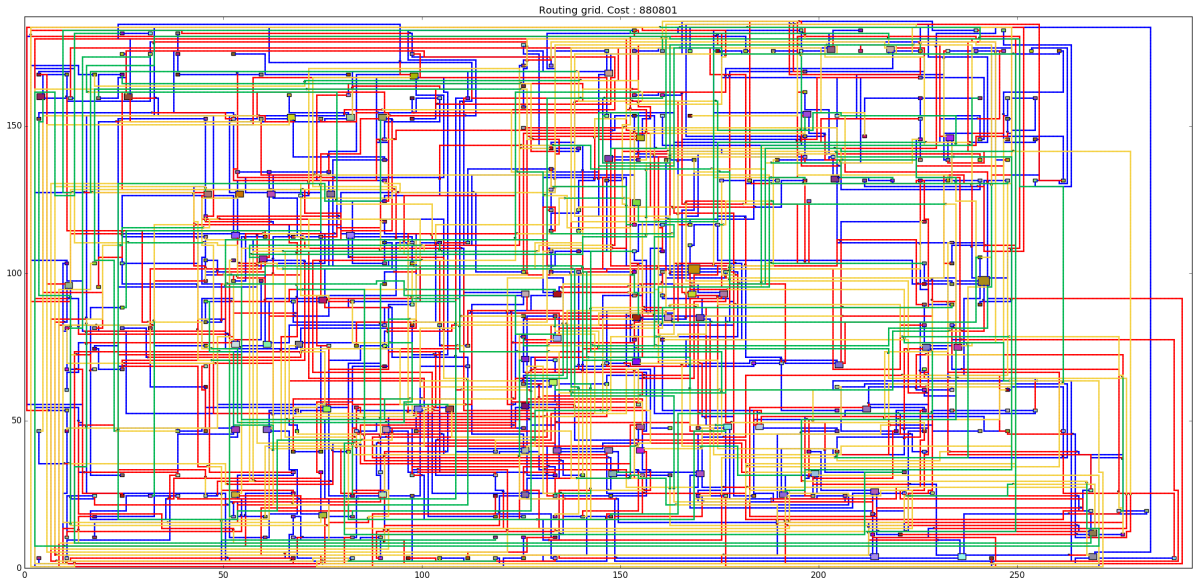
How is the grid encoded?: The grid is a dictionary with elements: * dir - Direction in which movement is taking place * via - True or False whether via is present * via_joint - Type of joint when via is being placed (5-10) * net - The type of net in that block (1-10)

The net values 1-10 are as follows:

Example Plot:



Example Plot:



Features

- The application provides you the facilty to run placement and multi layer routing for ISCAS'85 format netlists.
- Includes option to animate graph (not recommended at all on a large netlist)
- You can select the old placement run to continue routing later.
- You can select the old placement and corresponding old routing run to directly plot the routed floorplan.

How to execute?

Call

```
python3 place_n_route.py -i <input_isc_file> -m <placement|routing|both>␣
↪(-t <initial_temperature> -e  <polishexp order> -p <padding> -l <layers>
↪ -b <bend_cost> -v <via_cost> -r <temp_reduction_ratio> -s <max_stop_
↪iterations> -a animate)
```

Use Ctrl+C (SIGNINT) to stop the program and output the best solution

Required Arguments:

-i, --input The input file name for processing the graph

Optional Arguments:

-m ROUTING_MODE, --mode ROUTING_MODE mode - Either run placement or routing or both (default = placement)

-l LAYERS, --layers LAYERS Number of layers to use.

-e POLISHEXP_ORDER, --peorder POLISHEXP_ORDER peorder - Polish Expression - V (for vertical) | H (for horizontal)

-p PADDING, --padding PADDING Extra passing to be added to each block (default = 1)

-t TEMPERATURE, --temperature TEMPERATURE The initial temperature to start the system at (default = 1000)

-b BEND_COST, --bend-cost BEND_COST Bend cost (default = 0)

-v VIA_COST, --via-cost VIA_COST Via cost (default = 0)

-r RATIO, --ratio RATIO Temperature reduction ratio (0 to 1) (default = 0.1)

-s STOP_ITERATIONS, --stop_iterations STOP_ITERATIONS The maximum number of previous temperature iterations to check for cost reduction before stopping the program

-a, --animate Animate the Graph during processing (output will be slower)

Example

Placement

Example

```
python3 place_n_route.py -i c432.isc -t 10000
```

Example Output

For c432.isc

```
ISCAS'85 Netlist Parse Complete!

Running Simulated Annealing for Placement! Please wait...

Temperature = 10000.000000, Best Cost = 3200.000000
Temperature = 1000.000000, Best Cost = 3150.000000
Temperature = 100.000000, Best Cost = 3150.000000
Temperature = 10.000000, Best Cost = 3150.000000
Temperature = 1.000000, Best Cost = 3150.000000

Total time taken : 0.052071 minutes
File saved at 09-04-2018 22:40:35.
Plotting.... Please Wait...
Press to continue
```

Routing Example

```
python3 place_n_route.py -i ../ISCAS85/DATA/c432.isc -m routing
```

Example Output

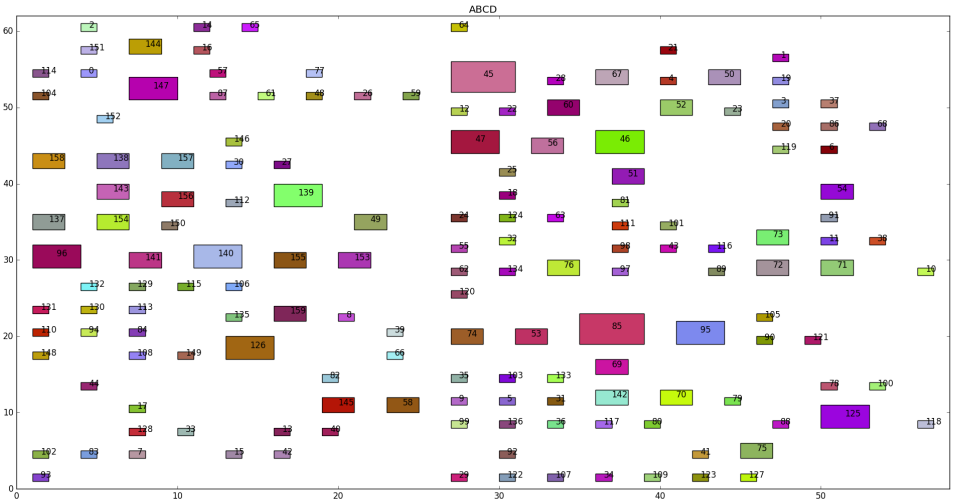
For c432.isc

```
Choose from one of the previous placement runs:
        1 : 24-03-2018 23:21:02 @ 7913
        2 : 25-03-2018 00:10:28 @ 8022
        3 : 25-03-2018 19:29:36 @ 7596
        4 : 27-03-2018 02:11:56 @ 26444
        5 : 28-03-2018 00:54:00 @ 626855
Select file number (1-5): 4
Choose from one of the previous routing runs:
        0 : New run
```

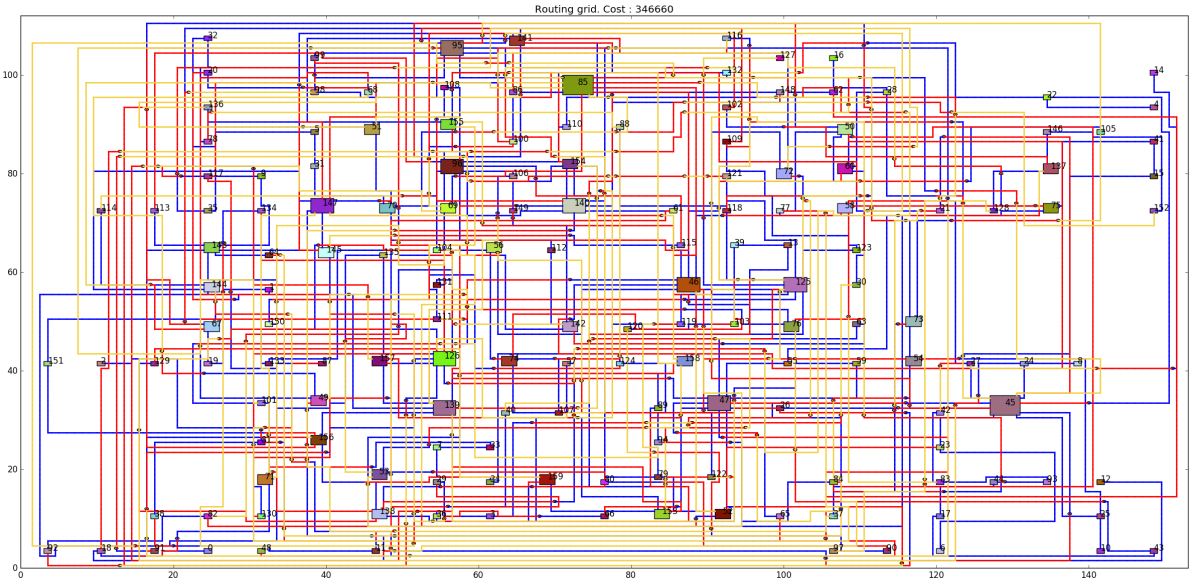
```
        1 : 27-03-2018 14:33:43
        2 : 10-04-2018 10:13:27
        3 : 10-04-2018 10:21:02
Select file number (0-3): 0

Running Route Design! Please wait...
Routed : 255/255 : 100.00%
Total time taken : 2.5151 minutes
File saved at 10-04-2018 11:04:15.
Plotting.... Please Wait...
```

Example Plots

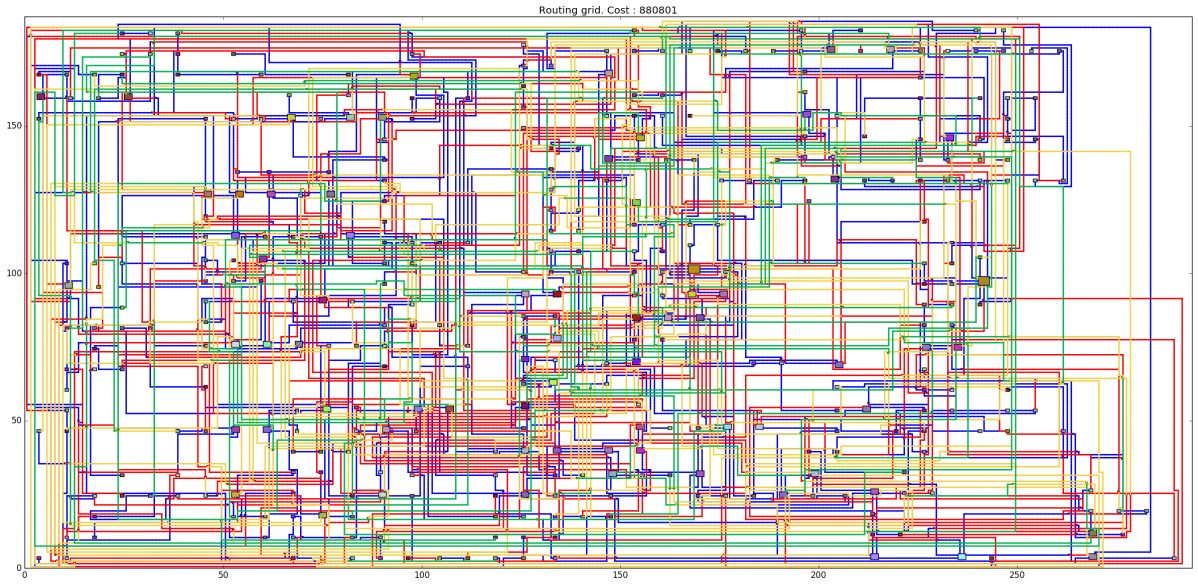
- Placement run for c432:



- Routing run for c432:



- Routing run for c880:



- genindex
- search

K

P

R

S