Implementation of Analytical Placer

SME312 Report **Shuo FENG (冯硕)** **11911736**

Introduction

In this report, an analytical (quadratic) placer for standard cell placement has been implemented. This report contains my code structure, usage, and results.

Code Structure

The code structure can be divided into five stages: ***Passer***, ***Matrix Generator***, ***Solver***, ***Printer*** and ***Visualization***. Detailed Explanation will be given next.

Passer

The passer here refers to the code block used for the reading of the input '.txt' files. Since the structure of the input file has been clarified in the Lab requirement file, certain approaches are used. Firstly we should read the input txt using function **fopen**, and dealing with it by function **fgetl** line by line. Then, some loops for the information of gate-to gate and gate-to-pad are saved in the matrix *GateToGate* and *GateToPad*, respectively. The code of ***Passer*** is as follows:

```
% Passer
% ------ read the input txt file
fid = fopen('InputFile01.txt', 'r');
i = 0;
while ~feof(fid)
    linedata = fgetl(fid);
    i = i+1;
    a{i} = linedata;
    if isempty(linedata)
        continue
    end
end
fclose(fid);
% ------ get the size of the entire placement region
sizeP = str2num(a{1});
RawData = [];
for i = 2 : length(a)
    S = regexp(a{i}, '\s+', 'split');
        for j = 1 : length(S)
            num = str2num(S{j});
            RawData = [RawData num];
        end
end
% ------ get gate-to-gate connection
B = find(RawData<0);
GateToGate = [];
for i = 2 : length(B)
    if B(i-1)+1 == B(i)
        Mark = B(i-1);
           GateToGate = RawData(1:B(i-1));
           GateToPadRaw = RawData ((B(i-1)+2):length(RawData)-1);
```

```
            break
    end
end
% ------ get gate-to-pad connection
GateToPad = [GateToPadRaw(1:3)];
row = length(GateToPadRaw)/3;
if i >1
    for i = 2 : row
        GateToPad = [GateToPad; GateToPadRaw(3*i-2:3*i)]
    end
end
```

Matrix Generator

For the Analytical Placement Method, algebraically, we know that for N gates, we get two equivalent NxN matrices (A) for two linear systems:

$$Ax = b_x, Ay = b_y$$

Where b vectors represent Pad coordinates. In this block, three processes should be done:

1. Build connectivity matrix C

   C(i,j)=C(j,i)=w for a net with weight w connected between gates i and j. If no net connects gates i and j, C(i,j)=0. Note that in our cases, w is 1.

2. Build A matrix

   Off diagonal, A(i,j)= −C(i,j). A(i,i) is sum of row i + weight of net from i to pad.

3. Build b vectors

   If gate i connects to a pad at (xi ,yi) with weight wi then bx (i)=wi*xi , by (i)=wi*yi

```
% Mitrix Generator
% ------ get the connectivity matrix C
M = max(max(GateToGate));     % get the dimension of C and A; and the row # of b
C = [zeros(M)];
loc = find (B == Mark);
for i = 1 : loc
    if i > 1
        for j = (B(i-1)+2): B(i)-1
            C(RawData(B(i-1)+1),RawData(j)) = 1;
            C(RawData(j),RawData(B(i-1)+1)) = 1;
        end
    else
        for k = 2 : B(1)-1
            C(RawData(1),RawData(k)) = 1;
            C(RawData(k),RawData(1)) = 1;
        end
    end
end
% ------ generate metrix A
A = [zeros(M)];
sumCol = sum(C);
[m2,n2] = size (GateToPad)
Cpad = zeros(max(M,m2),1);
for i = 1:m2
    Cpad(GateToPad(i)) = Cpad(GateToPad(i)) + 1;
end
```

```
for i = 1:M                     % row
    for j = 1:M                 % column
        if (i ~= j)
            A(i,j) = -C(i,j);
        else
            A(i,j) = sumCol(i) + Cpad(i);
        end
    end
end
% ------ generate matrix bX and bY
bX = zeros(M,1);
bY = zeros(M,1);
for i = 1:m2
    bX(GateToPad(i)) = bX(GateToPad(i)) + GateToPad(i,2);
    bY(GateToPad(i)) = bY(GateToPad(i)) + GateToPad(i,3);
end
```

Solver

Solving the matrix gives the position of the corresponding gates obtained by the ***Analytical Placement*** method. MATLAB can help solve the matrix, the code is shown below:

```
% Solver
x = A\bX
y = A\bY
```

Printer

This is to save the output of the optimized position of gates in txt file.

```
% Printer
fp = fopen('OutputFile01.txt','a');
fprintf(fp,'%d ',x);
fprintf(fp,'\n');
fprintf(fp,'%d ',y);
fclose(fp);
```

Visualization

By using MATLAB, we can visualize the corresponding placement.

```
% Visualize
x1= [x' GateToPad(:,2)']
y1= [y' GateToPad(:,3)']
axis([0 50 0 50])
plot(x1,y1,'--gs',...
    'LineWidth',2,...
    'MarkerSize',10,...
    'MarkerEdgeColor','b',...
    'MarkerFaceColor',[0.5,0.5,0.5])
```

Usage (Take Case 1 as Exp.)

This ***Analytical Placer*** is applicable to any basic layout, as long as the format is the same as the input file required by the topic to achieve a different layout.

Input File Deceleration

In this case, we give the **input txt** as :

```
50 50
blk1    1  2  3  4  -1
blk2    2  5  4  -1
blk3    3  5  6  2  -1
blk4    4  6  3  -1
-1
blk1    1  50   0
blk4    4  0  50
-1
```

Output File Check

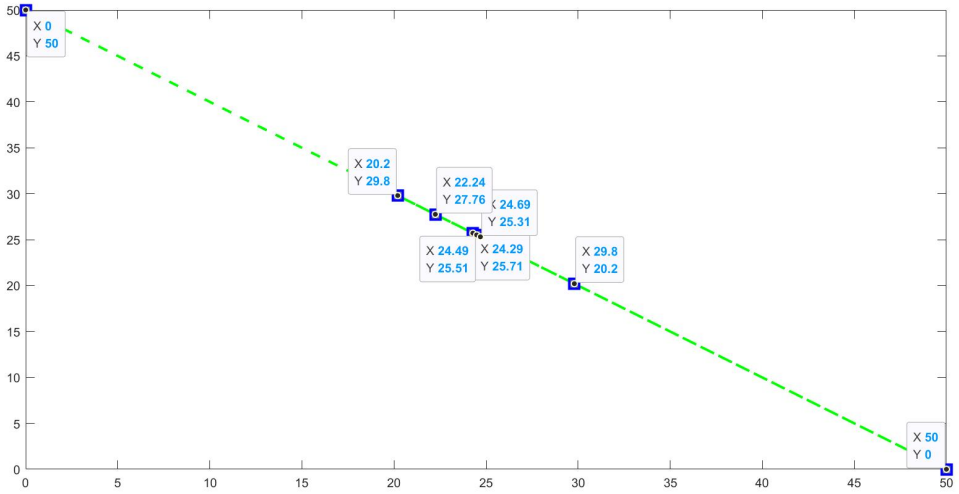
And the **output txt** file is as follows, six gates' locations:

```
2.979592e+01 2.469388e+01 2.428571e+01 2.020408e+01 2.448980e+01 2.224490e+01
2.020408e+01 2.530612e+01 2.571429e+01 2.979592e+01 2.551020e+01 2.775510e+01
```

Where the first line is solved x vector, the second line is the solved y vector.

Visualization Using MATLAB

And the visualization figure is shown below, where the blue box is the pads and gates, the green lines show the connection relationship.



Results (Take Case 2 as Exp.)

The results of the basic test input file given in lab are shown in the "Usage " part. In addition, to test the robustness of the code, another test file is applied. The input file, output file and the visualization will be shown next.

Input File

Different from Case 1, here we give the **input txt** as :

```
1 1
blk1    2 3   5 4  -1
blk2    1  2  3  -1
blk3    3  4  -1
blk4    4  5  -1
-1
blk1    1  0  1
blk3    3   1   0
blk4    4  1  1
blk5    5   0.5 0
-1
```

Output File

In this case, the output file is as follows, five gates in total:

```
4.419643e-01 6.250000e-01 7.008929e-01 7.366071e-01 6.205357e-01
6.428571e-01 5.000000e-01 4.285714e-01 5.714286e-01 3.571429e-01
```

Visualization

The visualization is shown here: