

ECEN 654

LAB1 Kernighan-Lin Algorithm

Yinya Lin

UIN: 630009726

Introduction

The most basic approaches to the partitioning problem treat the circuit as a graph. This is true for the first, and most famous partitioning algorithm, called the Kernighan-Lin algorithm. In this lab, we need to perform 2-way-even, and 3-way 3-2-1-weight partitioning of the given net using Kernighan-Lin algorithm. In order to finish the job, we first need to know how the KL algorithm works in a common case, which is 2-way even partitioning. Then, I adjust the original algorithm to satisfy the requirement of this lab. The language I used to program is C programming language.

Method

The common 2-way-even Kernighan-Lin algorithm works as follows:

1. The initial partition is generated "at random." We create two subcircuits S1, and S2. If the circuit has n gates, the first $n/2$ are assigned to S1, and the rest are assigned to S2. Because the gates in a circuit description appear in what is essentially a random order, the initial partition appears to be random.
2. A solution is acceptable only if both subcircuits contain the same number of gates. We will assume that the number of gates is even. The algorithm can be "tweaked" to handle an odd number of gates.
3. The goodness of a solution is equal to the number of graph edges that are cut. Suppose the edge (V,W) exists in the graph derived from the circuit. (Recall that V and W are both gates.) There are two possibilities. V and W can be in different subcircuits, or they can be in the same subcircuit. If V and W are in different subcircuits, we say that the edge (V,W) is cut. Otherwise we say that (V,W) is uncut.
4. The technique for generating new solutions from old solutions is to select a subset of gates from S1, and a subset of gates from S2 and swap them. To maintain acceptability, we always select two subsets of the same size.

Pseudocode:

determine a balanced initial partition of the nodes into sets A1 and B1

do

compute D values for all a in A and b in B for ($n := 1$ to $|V|/2$)

find $a[i]$ from A1 and $b[j]$ from B1, such that $g[n] = D[a[i]] + D[b[j]] - 2*c[a[i]][b[j]]$ is maximal

move $a[i]$ to B1 and $b[j]$ to A1

remove $a[i]$ and $b[j]$ from further consideration in this pass update D values for the elements of $A1 = A1 \setminus a[i]$ and $B1 = B1 \setminus b[j]$

end for

find k which maximizes g_max , the sum of $g[1], \dots, g[k]$ if ($g_max > 0$) then

Exchange $a[1], a[2], \dots, a[k]$ with $b[1], b[2], \dots, b[k]$ until ($g_max \leq 0$)

return $G(V,E)$

3-2-1 weighted unequal sized blocks:

To partition the whole graph $G(V,E)$ into three subgraphs of unequal sizes n_1 , n_2 and n_3 , which V is nodes, divide V into A , B and C , containing n_1 , n_2 and n_3 nodes, that $n_1:n_2:n_3 = 1:2:3$. Begin with random division, apply 2-way partitioning algorithm between each pair of partition, but restrict the maximum number of nodes that can be interchanged to minimum of the pair of subgraphs' nodes.

procedure

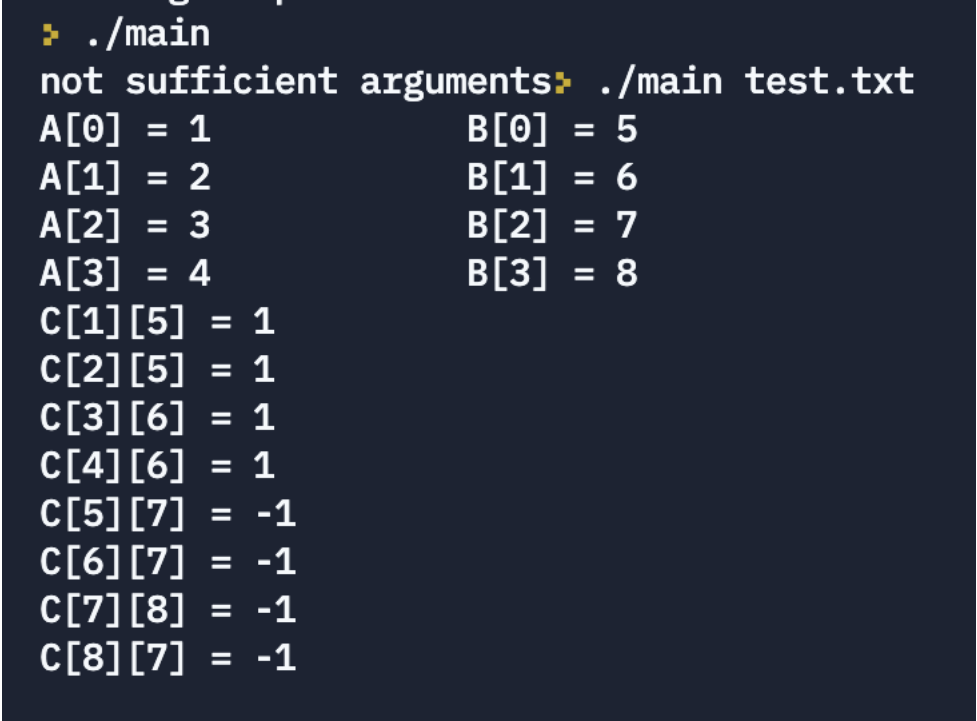
First of all, analyse the net file which is given and transfer it into another txt file which contains nodes and edges. The file format is listed below:

```
8 8
1 5
2 5
.....
```

The second line means connection for nodes 1, the third line means connection for nodes 2 and so on. According to this file, this graph has 8 nodes. It can be divided evenly into 2 subgraphs. Code with comments can be found in related C files.

For the part 1:

The following picture shows the first step: divide the node into 2 partition randomly, and calculate the cost for each pair.



```
➤ ./main
not sufficient arguments➤ ./main test.txt
A[0] = 1          B[0] = 5
A[1] = 2          B[1] = 6
A[2] = 3          B[2] = 7
A[3] = 4          B[3] = 8
C[1][5] = 1
C[2][5] = 1
C[3][6] = 1
C[4][6] = 1
C[5][7] = -1
C[6][7] = -1
C[7][8] = -1
C[8][7] = -1
```

Then, we calculate each gain of all nodes, and get the maximum gain, at the same time, we swap these pair.

```
maxgain = 2
swap node 6 and 1

maxgain = 0
swap node 7 and 2

maxgain = 0
swap node 8 and 3

maxgain = -2
swap node 5 and 4

maxgain = 0
swap node 1 and 2

maxgain = -2
swap node 5 and 6

maxgain = 0
swap node 7 and 3

maxgain = 2
swap node 8 and 4
```

When the maximum positive gain G_m of a pass is smaller or equal to 0, we will get the final partition A and B, and the cut set.

```
> clang-7 -pthread -lm -o main main.c
> ./main
not sufficient arguments>
> ./main test.txt
Partition A:
6 2 3 4
Partition B:
5 1 7 8
Cut Set:
6 7
2 5
> █
```

For the part 2, similarly, we divided all nodes randomly into three partitions and the number of nodes in the three partitions is 3:2:1.

A[0] = 1 B[0] = 7 C[0] = 11

A[1] = 2 B[1] = 8 C[1] = 12

A[2] = 3 B[2] = 9

A[3] = 4 B[3] = 10

A[4] = 5

A[5] = 6

Then, we calculate the gain of all effective pairs (effective pair means two nodes at two different partitions), and find the maximum gain. We locked this pair of nodes, and swap this pair of nodes. Repeat this step, until all the pairs are locked.

```

D[5] = 1, D[11] = 4      cost[5][11] = 0  G[5][11] = 5
D[10] = 3, D[11] = 4     cost[10][11] = 0   G[10][11] = 7
D[5] = 1, D[10] = 3      cost[5][10] = 0  G[5][10] = 4
D[5] = 1, D[12] = 4      cost[5][12] = 1  G[5][12] = 3
D[10] = 3, D[12] = 4     cost[10][12] = 1   G[10][12] = 5
D[6] = 0, D[7] = 2      cost[6][7] = 1   G[6][7] = 0
D[6] = 0, D[11] = 4      cost[6][11] = 0  G[6][11] = 4
D[7] = 2, D[11] = 4      cost[7][11] = 1  G[7][11] = 4
D[6] = 0, D[7] = 2      cost[6][7] = 1   G[6][7] = 0
D[6] = 0, D[12] = 4      cost[6][12] = 0  G[6][12] = 4
D[7] = 2, D[12] = 4      cost[7][12] = 1  G[7][12] = 4
D[6] = 0, D[8] = -3      cost[6][8] = 0   G[6][8] = -3
D[6] = 0, D[11] = 4      cost[6][11] = 0  G[6][11] = 4
D[8] = -3, D[11] = 4     cost[8][11] = 0  G[8][11] = 1
D[6] = 0, D[8] = -3      cost[6][8] = 0   G[6][8] = -3
D[6] = 0, D[12] = 4      cost[6][12] = 0  G[6][12] = 4
D[8] = -3, D[12] = 4     cost[8][12] = 0  G[8][12] = 1
D[6] = 0, D[9] = 1      cost[6][9] = 0   G[6][9] = 1
D[6] = 0, D[11] = 4      cost[6][11] = 0  G[6][11] = 4
D[9] = 1, D[11] = 4      cost[9][11] = 1  G[9][11] = 3
D[6] = 0, D[9] = 1      cost[6][9] = 0   G[6][9] = 1
D[6] = 0, D[12] = 4      cost[6][12] = 0  G[6][12] = 4
D[9] = 1, D[12] = 4      cost[9][12] = 0  G[9][12] = 5
D[6] = 0, D[10] = 3      cost[6][10] = 1  G[6][10] = 1
D[6] = 0, D[11] = 4      cost[6][11] = 0  G[6][11] = 4
D[10] = 3, D[11] = 4     cost[10][11] = 0   G[10][11] = 7
D[6] = 0, D[10] = 3      cost[6][10] = 1  G[6][10] = 1
D[6] = 0, D[12] = 4      cost[6][12] = 0  G[6][12] = 4
D[10] = 3, D[12] = 4     cost[10][12] = 1   G[10][12] = 5
fix[11][10] = 1 fix[10][11] = 1

```

maxgain =7 , swap the node 10 and 11

result

For the part one, when the node 6, 2, 3, 4 in the partition A , and node 5, 1, 7, 8 in the partition B, we will get the minimum cut. and the cut set is (6,7) and (2,5)

```
❏ clang-7 -pthread -lm -o main main.c
❏ ./main
not sufficient arguments❏
❏ ./main test.txt
Partition A:
6 2 3 4
Partition B:
5 1 7 8
Cut Set:
6 7
2 5
❏ █
```

part2 result

```
A[0] = 1    B[0] = 12    C[0] = 7
A[1] = 2    B[1] = 8     C[1] = 11
A[2] = 3    B[2] = 9
A[3] = 4    B[3] = 10
A[4] = 5
A[5] = 6
```