University of Toronto, Faculty of Applied Science and Engineering
Department of Electrical and Computer Engineering

ECE 1387 - CAD for Digital Circuit Synthesis and Layout

Assignment #3 - Branch and Bound Partitioning / Monte Carlo Sampling

November 2013 J. Anderson

Assignment Date: November 3, 2013
Due Date: November 22, 2013, beginning of class
Late Penalty: **-2 marks per day late, with total marks available = 20**

You are to write an implementation of the branch-and-bound partitioning algorithm described in class. It is to take a netlist of (equal-sized) blocks and divide it **into two pieces** with the objective being to minimize the number of **nets** that connect blocks in the two partitions; that is, you are to minimize the "cut set" (crossing count).

You are to implement **exact** branch and bound – that is, your lower bound function must represent a true lower bound. Aim to design a clever bounding function to reduce the size of the explored search space.

In class, we discussed several potential decision trees for this problem. *You may use any decision tree design you wish.*

As in the previous assignments, your program should display its progress and results using graphics. Illustrate, as best you can, the present state of the decision tree, and the pruning as it progresses.

The netlist format is similar to that used in A2 (analytical placement) and it specifies the blocks (cells) and the connectivity between them. Each line has the following form:

blocknum netnum$_1$ netnum$_2$ netnum$_3$ ... netnum$_n$ -1

where blocknum is a positive integer giving the number of the block, and the netnum$_i$ are the numbers of the nets that are attached to that block. Every block that has the same netnum$_i$ on its description line is attached. Note that each block may have a different number of nets attached to it. Each line is terminated by a –1. A –1 appearing by itself on a line terminates the netlist. Example input file:

```
1 2 3 4 -1
2 5 4 -1
3 5 6 2 -1
4 6 3 -1
 -1
```

In this example, block 1 is connected to nets 2, 3 and 4. Note that a net may connect more than two blocks (i.e. there may be multi-fanout nets). Also note that net numbers are not related to block numbers.

You are to minimize the number of **nets** that connect blocks in the two partitions. This means that each net contributes either 0 or 1 to the total crossing count, and never contributes more than this. A net contributes 1 to the crossing count if it connects cells that lie in both partitions.

Test your program on the **five** test circuits provided on the course web page.

Run your partitioner 3 times on each test circuit as follows: 1) with equal partition sizes, 2) where partition sizes may differ by up to two blocks, and 3) where partition sizes may differ by up to four blocks. Your program may not complete for the 5th circuit, which is large – that is fine and will incur no penalty.

BONUS: Implement Monte Carlo sampling of the decision tree and run this on the 4 test circuits and also the large 5th circuit. The idea is that, at lower levels of the tree, your partitioner should probabilistically "ignore" chunks of the tree. For example, one "crude" implementation is to only explore 10% of the branches beyond a certain level L, where the specific 10% to explore is determined by rolling a weighted coin (generating a random #). In essence, this approach "samples" portions of the decision tree, allowing the algorithm to traverse deeper in a reasonable amount of time. The approach carries the name "Monte Carlo" because of the gambling casinos in that city. Design your Monte Carlo-style partitioner to accept different random seeds. Explore the run-time vs. quality trade-offs in the Monte Carlo-style B&B partitioning, and the improvements offered by multiple runs with different seeds. Use equal-sized partitions for this step.

What to hand in:

1. The location of the executable and code on either EECG or ECF**. Please check file permissions!**

2. A short 2-3 page description of the flow of your program. You should describe:

   - The branching structure (that is, the design of your decision tree). What do the levels of the tree represent? How did you "order" the levels?

   - How you determined the initial "best" solution.

   - The bounding function that you used (and any others you investigated). Describe its effectiveness on reducing the exploration space.

   - How you traverse the tree and prune it with the bounding function.

   - An overview of your approach to Monte Carlo sampling (if you implemented this).

3. Also provide the following:

   - A paper plot of the results from the test circuits.

   - A count of the number of tree nodes that are visited by B&B each test circuit, for each of the 3 different balance constraints. There will be a **prize** for the smallest number of nodes traversed in partitioning cct4 (with equal partition sizes). **CONTEST RULES:** Tuning to the algorithm to the exact problem is not allowed. Regarding the definition of "visited", if you call your bounding function for a node, that counts as visiting the node. Likewise, if a node is disqualified owing to balance constraints, the node still counts as visited.

   - The run-time for each of the test circuits.

   - The value of the net **crossing count** that you achieved on each test circuit.

   - If you did the Monte Carlo sampling, report the crossing count, tree nodes visited, and run-time for the approaches you studied, across different random seeds. Comment on how the results compare with those achieved for the "normal" (non-Monte Carlo) B&B partitioning.