VLSI Design Automation (EECE 6086C)

Ranga Vemuri

Home Work - 1

In this assignment you will implement a partitioning algorithm to solve the balanced bi-partitioning (BB) problem. Among the two students in the same group, the one with the smaller M number should implement SA and the other student should implement KL. The two students can discuss and share any data structures and functions between their two programs. You are not allowed to collaborate or share any code with anyone else.

A net-list containing a number of cells connected via two-terminal nets is to be partitioned into two equal size segments. Assume that all cells are of the same unit size. The goal is to partition the net-list into two segments such that 1) the two segments have equal number of cells and 2) the cutset (the number of nets from one segment to another) is minimized. Your program should output the two partition segments and the size of the cutset for the best solution found.

A set of development benchmark netlists will also be provided via canvas. These are for your own use during development and testing. We will use our own benchmarks for evaluation of your submissions. Our netlists will contain no more than 1,000,000 cells and 10,000,000 nets. We will allow a maximum of eight hours execution time for any benchmark on the VLSI lab computers.

The net-lists will be specified in the following format:

First Line: Number of Cells (c)
Second Line: Number of nets (n)
Next n lines specify the nets.

All nets are assumed to be two terminal nets. There may be parallel nets (you can collapse them into a single edge and represent the number nets collapsed as edge weight). Cells are assumed to be numbered from 1 to c. c is assumed to be even. Each net is specified in a line in the following format:

CellNumber-1 CellNumber-2

The nets are implicitly numbered from 1 to n in the order in which they are specified.

Here is an example net-list.

```
6
8
1 5
1 4
1 3
5 2
4 5
3 4
```

3 2
2 3

The output of your program should have the following format:

First Line: Cutset Size
Second Line: List of Cells in the first segment (separated by spaces)
Third Line: List of Cells in the second segment (separated by spaces)

Following is an example output file:

3
1 5 4 3
2 6 7 8

Benchmark files will be named B1, B2, B3 etc. The corresponding result files should be named R1, R2, R3 etc.

You are ***not*** allowed to use an existing SA or KL library/package. You are allowed and encouraged to use existing and well-tested random number generators and data structure libraries such as the Standard Template Library and the C++ Standard Library.

Please submit the following:

1. A report including a section on SA and a section on KL written individually by the respective students and a section on comparative assessment of the two programs written jointly by both students.

2. In the SA and KL sections, include a clear description of the exact algorithm you have implemented (if they deviate from those in the text book) and all the main data structures. Explain why you think your program is good and include any discussion and data you choose. (For example, you should include the cutsets, execution times and memory requirements for the development benchmarks. You should include the execution trajectories in case of SA or KL. You should include a discussion about how you tuned your program.)

   We will evaluate your program based on how it handles our benchmarks with respect to the size of the netlists handled, cutset sizes, execution times and memory requirements, the approach used, quality of the report, quality of software and its documentation.

   For SA, include the tuning method and final SA parameters used. Include convergence plots (cutset vs. temperature, temperature vs iteration, boltz vs iteration, and number of accepted moves vs temperature).

   For KL, include plots indicating cutset vs pass, K (subqueue length) vs pass, pairwise gains vs queue position for each pass, and any other interesting data.

3. In the SA-KL comparison section, you can use the data you have collected as above (and any additional data such as the time you took to develop the programs) to provide your comparative assessment of the two programs. Which one is better on each of the metrics you have used to

compare them and why? If you had more time, could you have improved either program so that it can do better? How?

4. Your programs. Only C or C++ will be accepted and C++ is preferred. If you wish to use any other language you need prior permission. The programs should be modular, well documented, indented and/or pretty-printed, easy to understand and should readily compile on the VLSI lab workstations.

5. Include a single makefile named "Makefile" to compile your program or provide clear compile instructions in your report. The executable must be named "saprog" or "klprog" and must take input from stdin and produce output to stdout. We will run it, for example, as "saprog < B1 > R1". Note that we should NOT have to recompile your program for each netlist. We should be able to compile and execute your programs on the VLSI Lab machines.

6. Note that all your results should be reproducible so that we can reproduce your runs and evaluate your submissions. If this requires a specific seed to the random number generator, please make sure to include the seed(s) for each run in your reports.

7. Your entire submission must be uploaded as a single zip file via canvas. Your submission should include the program source files including the Makefile if needed, executables, results files named R1, R2,...and the report in pdf. You may receive additional submission instructions via email/canvas.

We assume that all the work you submit was done by yourself. If we discover suspected plagiarism of any kind, it will be reported and may have serious consequences. Please familiarize yourself with the University policies on plagiarism.

As usual, don't delete your programs and your output files until the end of the quarter. You may be asked for further information during grading. Please do ask any questions or clarifications.

Don't be discouraged if you cannot get through all the benchmarks provided. Start early so that you have plenty of time for experimenting with the larger benchmarks.

You may be interviewed individually or as a team as part of the evaluation process.