HW2

資工19 104062302 洪旻夆

Compile and Execute

I've used C++11, so please enter following command before using Makefile,
$ source /tools/linux/gnu/setup_toolkit.csh

(Below commands are entered under HW2/src directory)

To clean up previous executable file,
$ make clean

To compile C++ file to executable file,
$ make

Usage:
./two-way_min-cut_partition <path/to/testcase/cells_file> <path/to/testcase/nets_file> <path/for/output_file>
e.g.
./two-way_min-cut_partition ../testcase/p2-1.cells ../testcase/p2-1.nets ../output/p2-1.out

Testcases

	p2-1	p2-2	p2-3	p2-4
Cut size	6	193	3490	40254
Runtime (sec)	0.02	0.09	9.01	134.41
I/O time	0 (not accurate)	0.01	0.29	0.55
FM time	0 (not accurate)	0.07	8.8	133.86

Implementations

I. What is the difference your algorithm and FM algorithm described in class? Are they exactly the same?

Yes, my algorithm is exactly the same as the FM algorithm described in class, but there are some implementation details which will de introduced later.

II. Did you implement the bucket list data structure?

Yes, but I used vector of set to implement it.
The vector indices which are ranged from -pmax to pmax is slightly different from the one described in class. I shift all gains to making the indices ranged from 0 to 2pmax by adding pmax which makes vector indexing much easier than having negative gains.
As for the reason of using set to store cells is because it is much easier and efficient to insert, traverse and especially erase by set.

III. How did you find the maximum partial sum and restore the result?

To find maximum partial sum, I traverse through all gains in one pass and accumulate gain in each move while recording until which move has the maximum partial sum.

Before the actual moves with maximum partial sum, I do the shadow moves on copied data structures and record the base cell in each move. After knowing which moves are needed, I directly move the base cell recorded previously without calculating and finding the base cell with maximum gain again.

IV. Please compare your results with the top 5 students' results from last year and show your advantage either in runtime or in solution quality. Are your results better than them?

According to the top 5 students' results from last year, 1st is both good at runtime and small cut size in average, 2nd to 4th are good at cut size but not at runtime, and 5th is good at run time but not the cut size.
Compare to them, my runtime is better than 2nd to 4th but slower than 1st and 5th while my cut sizes are better than all of them in p2-1, p2-2 and p2-4. I have no idea what happen to p2-3 with such a large cut sizeQAQ.

V. What else do you do to enhance your solution quality or to speed up your program?

I have tried to enhance my solution quality by traversing through all the bucket with the same gain and picking the cells which will have the best area balance after the move. I believe that having a better area balance allow later moves to have more choices without violating the area constraint, and may results in a smaller cut size. But it also spends more time on picking base cell and therefore the total runtime also increases.

VI. What have you learned from this homework? What problem(s) have you encountered in this homework?

In this homework, I've learned the FM algorithm thoroughly. After the class of teaching it, I did not understand it at all, but coding it into a program makes me study the algorithm because I need to know the details to implement it.
The problems I've encountered is that it is really hard to debug because the basic testcase p2-1 has 375 cells and 357 nets, and I'm not able to draw all the circuit out to debug. However luckily, I found my bug not in the main algorithm but somewhere else, so I don't need to draw the circuit and count all the gains of each cell.