Andy, Yu-Guang Chen

EE6094 CAD for VLSI Design
Programming Assignment 3: Floorplan

(Due: 23:59:59, 2023/05/15)

Introduction

You are asked to implement the slicing floorplan design algorithm which is based on SA and normalized polish expression representation for solving rectangle packing problem. The rectangle packing problem is defined as follows: Given many rectangular modules of arbitrary sizes, place them without overlapping on a layer in the smallest bounding rectangle. It can be used to solve VLSI floorplan problem.

Rectangle Packing Problem in this assignment is defined as follows

Input: Given a set of rectangular modules where all of them are soft modules, and the aspect ratio is ranging from **0.5 to 2**

Output: A legal floorplan result

Objective: The bounding box of the packing area is as small as possible

Input file format

The first line gives the number of modules, denoted by *n*. From line 2 through line *n*+1, each line specifies the index and area of a module. All of the area will be integer. The range of the area is [100, 10000). The range of the amount of the module is [1, 100000].

Example:

```
5         //There are 5 modules
0 121   //module 0 with area = 120
1 9300 //module 1 with area = 9300
2 7200
3 1950
4 1200
```

Output file format

For each test case, output the width, height, and area of the best packing found by your program in line 1 with "blank" characters separating them. For each of the next *n* lines (with the increasing order of module indices), output the width and height of a module. Note that width and height could be decimal. Note that in the checker, the width and height will be considered as double format. Moreover, the product of width and height should be equal to original area **within 1% error**. All of the data are separated by a "blank" character. The last line is the normalized polish expression of your floorplan. Every two consecutive elements in the expression are separated by a "blank" character.

Example:

```
1500 1000 1500000    // width, height, area of the packing
12 10                // module 0 width and height [(12, 10)]
93 100               // module 1 width and height [(93, 100)]
…
25 48                // module 4 width and height [(25, 48)]
2 3 V 0 4 H V …      // the normalized polish expression of your floorplan
```

Requirement

1. **Algorithm**: You are asked to realize the slicing floorplan design algorithm which is based on SA and normalized polish expression representation we discussed in the class. You have flexibility to create your own action (movement), probability function, and cooling schedule.

2. You must write this program in C or C++. No open source codes are allowed to use. (i.e., you MUST implement the tool by yourself). You can use any data structure to realize your program. We will verify your program on workstation. Therefore, **you have to make sure your program can be executed correctly and successfully on workstation, any other version of C++ or compilers are not allowed. The run time of your program is limited to at most 2 hours per testcase.** Note that the cout on the screen will slow down the run time. Please use cout carefully.
   The workstation information is shown below:
   - System: CentOS 6.10
   - Compiler: gcc 4.8.2
   - C++ version: C++11

3. We will verify your program on a workstation with a **Makefile**. Therefore, you need to write a **Makefile** which can compile and execute your program directly. Your **Makefile** should at least contain these 3 commands, which are (1) **make all**, (2) **make run**, and (3) **make clean**. The descriptions of each command are shown below.
   (1) **make all**: This command will automatically compile your source codes and generate the corresponding objects and executable file.
   (2) **make run input=testcase1_in.txt output=testcase1_out.txt**: This command will execute your executable file and run your program. Where **testcase1_in.txt** is the input file name, and **testcase1_out.txt** is the output file name.
   (3) **make clean**: This command will automatically remove all the objects and executable

file generated by **make all**.

4. All files should be submitted through ee-class. You have to submit **a source code file** named as *StudID*_PA3.cpp (ex: 9862534_PA3.cpp) and **a report** named *StudID_Name*_PA3_report.pdf (ex: 9862534_陳聿廣_PA3_report.pdf) , and a **Makefile** to compile and execute your program. If your source code contains more than one file, only the "driver" file that contains main function should follow the naming rule mentioned above. Note that the only acceptable report file format is .pdf, no .doc/.docx or other files are acceptable. **BE SURE to follow the naming rule mentioned above. Otherwise, your program will be not graded.**

5. We don't restrict the report format and length. In your report, you have to at least include:
   (1) How to compile and execute your program. (You can use screenshot to explain)
   (2) The completion of the assignment. (If you complete all requirements, just specify all)
   (3) The hardness of this assignment and how you overcome it.
   (4) Any suggestions about this programming assignment?

Grading

The grading is as follows:

(1) Correctness of your code: 30%
(2) The quality of your solution: 30%
(3) Readability of your code: 10%
(4) The report: 10%
(5) Demo session: 20%

Please submit your assignment on time. Otherwise, the penalty rule will apply:

- Within 24hrs delay: 20% off
- Within 48hrs delay: 40% off
- More than 48hrs: 0 point

Contact

For all questions about PA3, please send E-mail to TA 何宜真 (gumi627@gmail.com)

Reference

[1] D. F. Wong and C. L. Liu, "A New Algorithm for Floorplan Design," 23rd ACM/IEEE Design Automation Conference, Las Vegas, NV, USA, 1986, pp. 101-107, doi: 10.1109/DAC.1986.1586075.

[2] Wong, D. F., Hon Wai Leong, and H. W. Liu. Simulated annealing for VLSI design. Vol. 42. Springer Science & Business Media, 2012.