May 20, 2013

Scientific Research Paper Reading and Implementation

We learn OOP and C++ to meet practical needs. In our Department of Computer Science and Technology, many research projects are implemented in C++. Therefore, as the final project, we will practice implementing scientific research papers so that in the future you will not feel afraid to join our labs. For the following given papers, different papers are assigned different degree of difficulty (难度系数). Just as in the diving competition, higher degree of difficulty corresponds to more efforts as well as more credits (分数). Bottom line (底线) is to correctly implement the paper with the lowest degree of difficulty. For each paper, an extra bonus of up to 10 points will be granted if any interesting extension or improvement is presented with improved solution quality (改进论文中的方法，得到更好的结果).

The papers are given as follows. Please choose one paper to implement. Please note that in research projects, you may use any open source codes you find through Google/Baidu searching. Probably you won't be able to find the source codes of the paper, but you may find the source codes of algorithms used in the paper, which make your implementation easier and faster. Any such open source codes (or libraries) are allowed in implementing the papers.

Requirement for submission:

(1) Include 3 directories: src, testcase, and doc.

(2) All the source codes including makefile need to be put in directory src.

(3) All the testcases need to be put in directory testcase.

(4) The detailed description of the software design (pdf file) needs to be put in directory doc.

Paper No.1: VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair (degree of difficulty: low, full credits: 90)

Requirements and TIPS:

(1) Partial source codes are provided to make the implementation easier. Please read the source codes carefully and find out how to used the code. Modify the provided source codes and add new codes as needed to implement the paper. You may also write your own codes without using the provided source codes if you want to.

(2) Search the Simulated Annealing (SA) method by Google/Baidu and implement this method to optimize the rectangle packing results as described in the paper.

(3) The optimization goal is minimizing the total area of the bounding box of all the placed rectangles. You don't need to implement Section IV.B ("Module Placement with Wires").

(4) Randomly generate 5 different testcases with up to 500 rectangles to test the implemented method.

(5) It is suggested that a validity checking function be implemented to verify the experimental results. For example, there should not be overlaps between the placed rectangles.

Paper No.2: Corner Block List: An Effective and Efficient Topological Representation of Non-Slicing Floorplan (degree of difficulty: medium, total credits: 100)

Requirements and TIPS:

(1) Any source code provided with Paper No.1 can be used here if you think the code helps.

(2) The optimization goal is minimizing the total area of the bounding box of all the placed rectangles. You don't need to consider wire length or soft blocks in the paper.

(3) Search the Simulated Annealing (SA) method by Google/Baidu and implement this method to optimize the rectangle packing results as described in the paper.

(4) Randomly generate different testcases with up to 500 rectangles to test the implemented method.

(5) It is suggested that a validity checking function be implemented to verify the experimental results. For example, there should not be overlaps between the placed rectangles.

Paper No.3: New Algorithms for the Rectilinear Steiner Tree Problem (degree of difficulty: high, total credits: 110)

Requirements and TIPS:

(1) If needed, search using Google/Baidu to learn some basics on MST and RST. Any open source code (e.g., Prim's MST algorithm) you found through Google/Baidu can be used in your implementation.

(2) Randomly generate 5 different testcases with up to 100 points to test the implemented methods (optimal S-RST and optimal L-RST).

(3) It is suggested that a validity checking function be implemented to verify the experimental results. For example, there should not be crossings between the RST edges.

Paper No.4: FOARS: FLUTE Based Obstacle-Avoiding Rectilinear Steiner Tree Construction (degree of difficulty: high, total credits: 120)

Requirements and TIPS:

(1) If needed, search using Google/Baidu to learn some basics on MST and RST. In the paper, you may understand "pin" as points and "wirelength" as rectilinear distance.

(2) The FLUTE source code needs to be downloaded from the following link: http://home.eng.iastate.edu/~cnchu/flute.html. Any open source code (e.g., Dijkstra's and Kruskal's algorithms) you find through Google/Baidu can be used in your implementation.

(3) Please search and download reference papers (参考文献) using Google if needed.

(4) Randomly generate 5 different testcases with up to 5000 points and 5000 obstacles to test the implemented method.

(5) It is suggested that a validity checking function be implemented to verify the experimental results. For example, there should not be crossings between RST edges or between RST edges and obstacles.

Paper No.5: The Polygonal Contraction Heuristic for Rectilinear Steiner Tree Construction (degree of difficulty: high, total credits: 130)

Requirements and TIPS:

(1) If needed, search using Google/Baidu to learn some basics on MST, RST, Delaunay triangulation, etc.

(2) You may use the CGAL source code (http://www.cgal.org) for Delaunay triangulation. Any open source code you found through Google/Baidu can be used in your implementation.

(3) Please search and download reference papers (参考文献) using Google if needed.

(4) Randomly generate 5 different testcases with up to 40000 points a 1000000x1000000 grid to test the implemented method.

(5) It is suggested that a validity checking function be implemented to verify the experimental results. For example, there should not be crossings between the RST edges.