Team Project: Projects and Scientific Research Paper Implementation

One of the following projects needs to be finished in team. As different projects have different degrees of difficulty (难度系数). Just like in the diving competition (跳水比赛), higher degree of difficulty corresponds to more efforts as well as more credits (分数). Bottom line (底线) is to correctly finish one project with the lowest degree of difficulty. For each project, an extra bonus of up to 10% will be granted considering the degree of difficulty, as well as any interesting extension and/or improvement in solution quality.

Please choose only ONE from the following projects to implement in team. When implementing the research papers, you may use any open source codes you find through Google/Baidu searching engine. You may not be able to find the exact source codes of the project, but you may find source codes of algorithms used in the paper, which make your implementation easier and faster. Any of such open source codes (or libraries) are allowed in implementing the research papers. Please use OOP design principles and design patterns as much as possible.

Requirement for submission:

(1) Please submit a zip/rar ball including 3 directories: **src**, **testcase**, and **doc**.

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

(3) All the testcases (测试用例) need to be put in directory **testcase**.

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

(5) Each team may have 2-3 students, and each student needs to submit a separate document (the pdf file) describing his own work in the project.

Choose one from the following projects for implementation:

(I) Project: Speech Input Tool: (degree of difficulty: 1.0)

Requirements and TIPS:

(1) Implement a software using OOP design principles, which integrates an Automatic Speech Recognition (ASR) engine (自动语音识别引擎) into a text editor (e.g., notepad, MS word, vi, emacs, Sublime Text, etc) either as a plugin (文本编辑器的插件) or as a stand-alone tool (独立运行的工具，此时可能需要自己实现文本编辑工具). The objective is to take speech input, translate it into written text, and then insert the translated text into current document opened by the text editor in real-time (实时将语音转换为文本，插入到文本编辑器打开的当前文档中).

(2) Please use Google/Baidu to search for possible solutions to the requirements, and to find by yourself some open-source ASR engines to integrate into a text editor. The following link may be helpful: http://cmusphinx.sourceforge.net/

(3) The Speech Input Tool needs to support both English and Chinese speech. You may think of how to design the user interface for better user experience.

(II) Project: Software Plagiarism Detection (软件剽窃检测, degree of difficulty: 1.0)

Requirements and TIPS:

(1) Implement the Software Plagiarism Detection project using OOP design. Please read the attached exercise, one of our future homeworks, to understand the background. Read the related references as needed for how to realize software plagiarism detection.

(2) Please note that you may learn from existing open-source software plagiarism detection systems (search for them through Baidu/Google by yourself). But you should not copy from their codes.

(3) Manually design at least 5 single-file or multiple-file projects (you may even use homeworks of your team members) to test your software plagiarism detector.

(III) Project: Large-Scale Multiple-Terminal Path Finding (degree of difficulty: 1.1)

Requirements and TIPS:

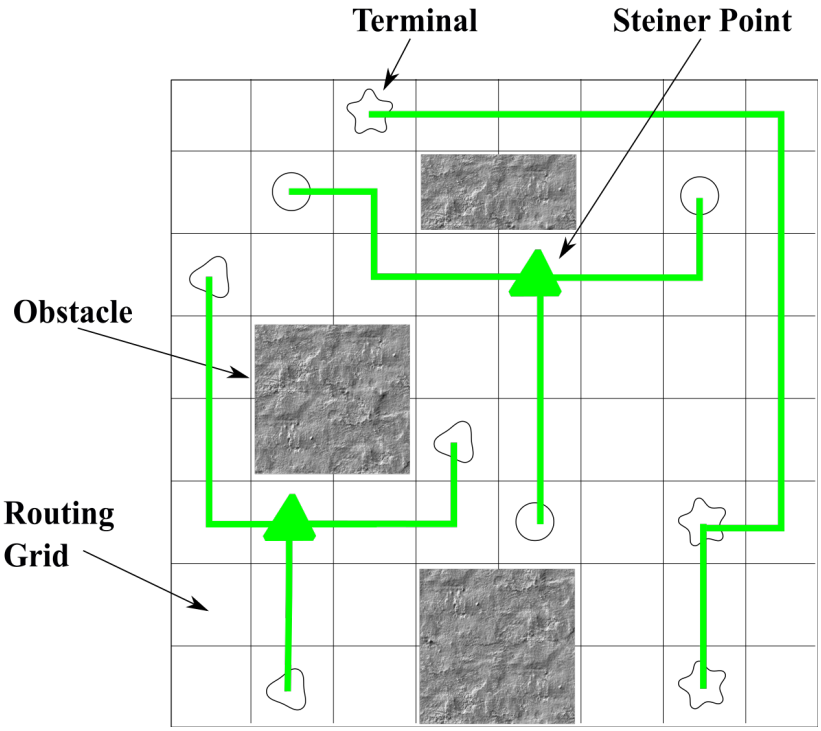


Figure 1. Example of the path-finding problem.

(1) Write the program using OOP design. In our individual projects, the path finding problems are solved using different formulations, e.g., SAT, column generation, etc. In this project, our objective is to solve the multiple-terminal path finding problem (please read our individual project No. 5 for details) on much larger number of routing grids. Here, for the N×N routing grids and M sets of terminals (each set has at least 3 terminals), we require that N >= 300, and M >= 10.

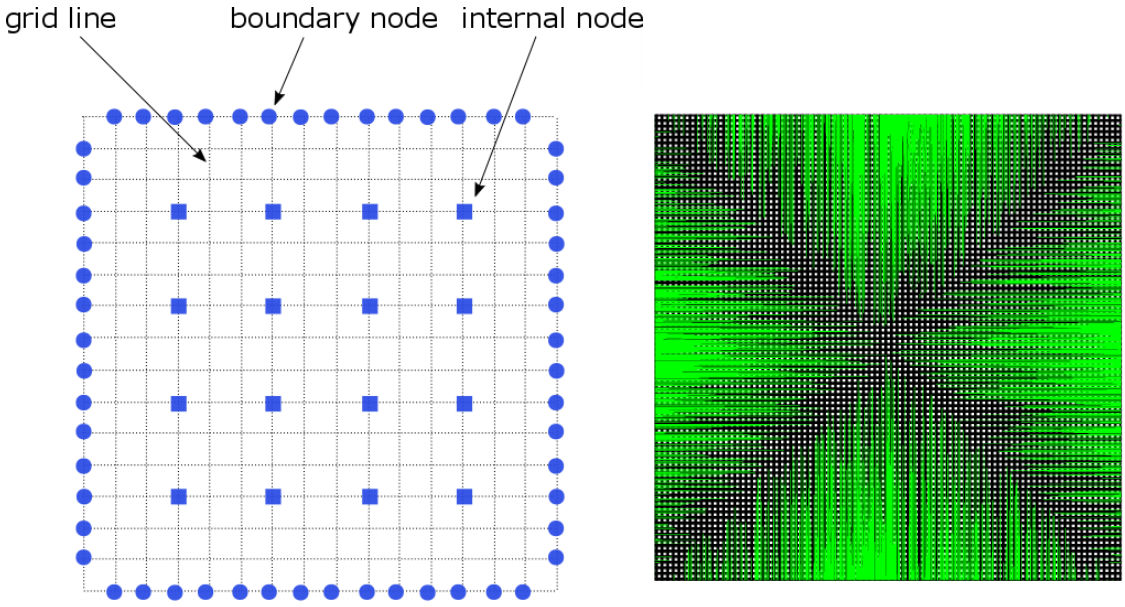
(2) The SAT and column generation methods in the individual projects can also be used in solving this project. However, due to the large problem size (问题规模大) in this project, you probably need to think of some divide-and-conquer (分而治之) strategy. For example, is it possible to merge the adjacent 2×2 or 10×10 grids into a larger grid to reduce the problem size (将相邻的 2×2 或 10×10 个网格合并为一个大网格，从而降低问题规模), and then map back the solutions from the reduced problem to the original problem (将缩小后问题的解映射到原问题)? Besides, any other methods can be used provided that it solves the problem with good solution quality and runtime (可接受的运行时间). Due to the difficulty in solving this problem, it is not required to obtain the optimal solution.

(3) Randomly generate 10 different testcases with different sizes of routing grids (e.g., 300×300, 350×350 …), different sets of terminals, (e.g., 10 sets, 11 sets …), and different number of obstacles (e.g., 5, 6 …). The coordinates of the terminals and obstacles need to be randomly selected without duplicates (输入点不重合).

(4) Report the statistics of the experimental results (实验结果), e.g., total runtime, number of sets of terminals successfully connected, total path length, etc. Figures and tables on the experimental results are welcome.

(5) It is suggested that a validity checking function (验证正确性) be implemented to verify the experimental results are correct. For example, you may check the connectivity between the sets of terminals, whether different paths cross each other, whether there are loops in the computed paths, etc.

(IV) Project: Rule-Based Regular Routing Method (degree of difficulty: 1.1)



(a) Escape routing requirement (b) Escape routing example

Figure 2. Example of the escape routing problem.

Requirements and TIPS:

(1) Write the program using OOP design. The escape routing problem is as follows: given the equally spaced internal node array and the number of grid lines between internal nodes, find routing paths along the grid lines to

connect each internal node out to one boundary node. Routing paths should not cross each other.

(2) The target problem size is around 5000 internal nodes (e.g., 70 x 70 = 4900, 120 x 40 = 4800, etc.). For each given problem size, the objective is to find the required number of grid lines between adjacent internal nodes, and compute the routing paths for all the internal nodes with minimized total path length.

(3) You may think of network flow formulation to solve this escape routing problem. However, the problem size may be too large. Then you may think of solving the whole problem by divide-and-conquer strategy (分治策略). But the problem size may still be too large. Based on the observation on the optimal routing solutions, it seems that there are some regular routing rules (有规律的布线规则) for such problems. Please try to think of some regular routing rules, and then present the rule-based regular routing method, where the routing paths are computed by equations (通过公式计算布线路径) rather than searched by solvers. Any other fast routing methods are also welcome.

(4) Implement the network flow formulation, and your proposed rule-based routing method (or any other fast routing methods that you may think of), and then test the different routing methods on 5 randomly generated testcases (from 50x50 to 80x80 internal nodes). It is allowed that the rule-based routing method may compute sub-optimal routing solutions, provided that the speed is really fast (如果运行时间足够快，允许次优解).

(5) Report the statistics of the experimental results (实验结果), e.g., total runtime, total path length, etc. Figures and tables on the experimental results are welcome.

(6) It is suggested that a validity checking function (验证正确性) be implemented to verify the experimental results are correct. For example, you may check the connectivity between the internal node and the boundary node, whether different paths cross each other, whether there are loops in the computed paths, etc.

(V) Project: Camelyon Challenge (Difficulty: 1.15)

Requirements and TIPS:

(1) Use machine learning for metastatic region detection (采用机器学习方法检测癌症转移区域). The official website of the Camelyon challenge 2016 is as follows: https://camelyon16.grand-challenge.org/

(2) Write the program using OOP design. Only those who are familiar with machine learning (especially deep learning) methods and Python programming language are suggested to try this project. Only this project may be finished using Python programming language.

(3) Any open-source algorithms and/or libraries can be used in your implementation. For example, you may use TensorFlow or Caffe you have learned during our code reading homework, or any other deep learning frameworks. The following links may be helpful:

(a) https://www.kaggle.com/c/diabetic-retinopathy-detection,

(b) https://github.com/sveitser/kaggle_diabetic/,

(c) https://github.com/JeffreyDF/kaggle_diabetic_retinopathy/,

(d) https://github.com/linhj184169280/CAMELYON16-Challenge/

(4) Objective: design your own deep learning neural network, and test your solution on common benchmarks (在公共竞赛测试用例上测试). Your solution should be comparative or even better than any existing open-source methods.
(5) If you cannot find good computing resource by yourself, we can provide one GPU server with one account for each team. Please send me an email and ask for the account. Please note that it is not guaranteed that the GPU server always works without crashes. And please make backups (备份) of your source codes and try your algorithms as soon as possible.

NOTE: For some projects, plotting the results could be of great help in checking the correctness. For those who want to visually check the results, here are some suggestions:
(1) Write out PostScript files from the program and then open it with PS viewers. Here is a link for the tutorial on writing PS files: http://www.tailrecursive.org/postscript/drawing.html
(2) Write out BMP files from the program. Search the BMP file format using Google/Baidu to see how to write a BMP file. For example:

(a) http://stackoverflow.com/questions/2654480/writing-bmp-image-in-pure-c-c-without-other-libraries

(b) http://blog.art21.org/2011/09/13/how-to-create-a-bitmap-image-file-by-hand-without-stencils/#.U0z8uF6aIaA

(3) Learn and try to integrate existing graph drawing tools (e.g., GraphViz: http://www.graphviz.org/Gallery.php) for plotting the results. Or write your own viewer using any GUI programming language (e.g., Qt, Java, OpenGL, etc.) to open text input file and plot it.