Automated Evaluation of Programming Assignments

Rishabh Manoj

International Institute of Information Technology - Bangalore

October 24, 2017

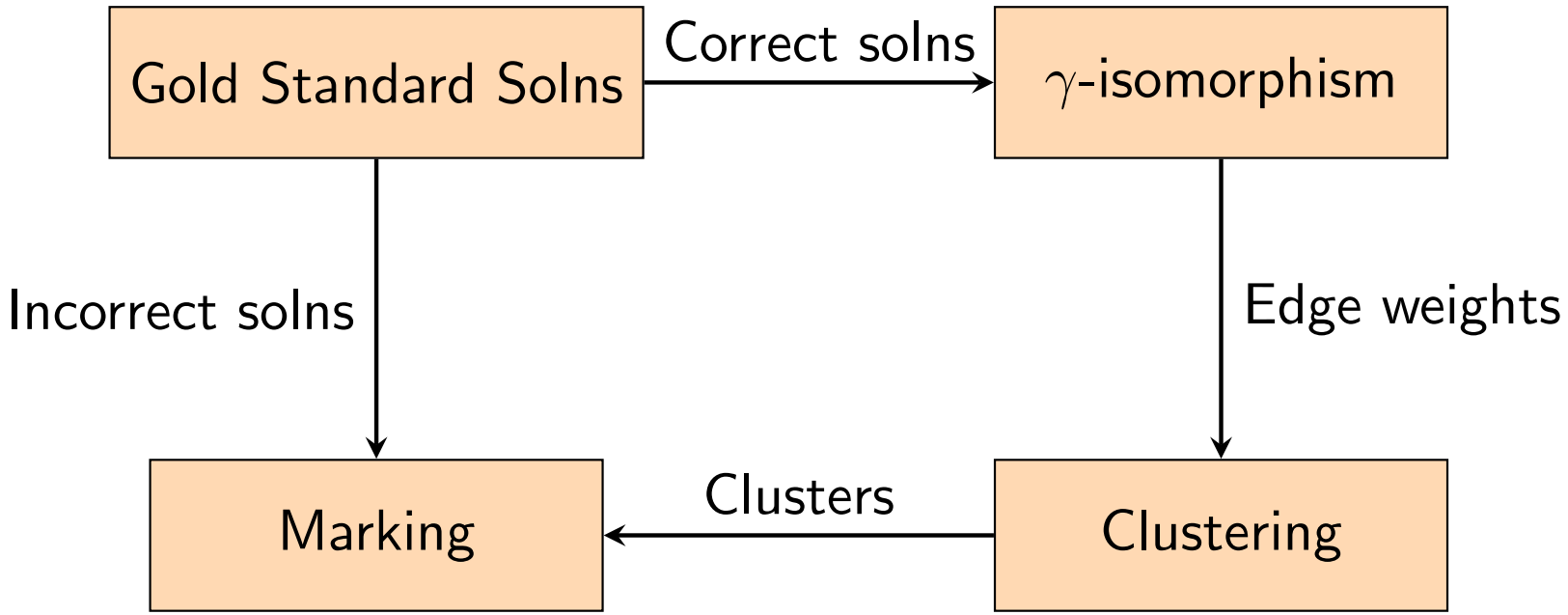
Problem Statement

- Automate the grading of programming solutions given a reference solution to a programming problem.
- Grades based on "correctness" measures of incorrect solutions.

Proposed Methods

- **Gold Standard Solutions**: Separate given solns into *correct* and *incorrect* groups based on test-cases generated from reference solns.
- **Similarity Measure**: Calculate the similarity measure of the correct solns with each other.
- **Clustering**: A fully connected graph can be constructed if each soln is considered as a node and the similarity measure value it's edge weight. Cluster this graph. Expected output is clusters of solutions with similar approaches.
- **Marking**: Calculate similarity measures of every incorrect solns to a representative solns of each of the clusters. Assign grades based on the "closeness" to cluster.

Model Architecture



Similarity Measure

Definitions

1. **Data dependency:** A data dependency is a situation in which a program statement refers to the data of a preceding statement.
2. **Control dependency:** Control dependency is a situation in which a program instruction executes if the previous instruction evaluates in a way that allows its execution. (*if* statements, *while* $\cdots$)
3. **Isomorphism:** A graph can exist in different forms having the same number of vertices,edges and also having same edge connectivity. Such graphs are called isomorphic graphs.

Program Dependence Graph:
Program dependence graph (PDG) is a graphical representation of the source code of a program. Basic statements, like variable declarations, assignments, and procedure calls, are represented by program vertices in PDGs. The data and control dependencies between statements are represented by edges between program vertices in PDGs.

Similarity Measure

γ-isomorphism:[1]
The central idea is to generate *Program Dependence Graphs* for the solutions and check for sub-graph isomorphism between any two PDG's.
If it is sub-graph isomorphic then based on the number of vertices mapped we calculate similarity.

Algorithm:

- Generate PDG's of the solution.
- Use γ-isomorphism method to calculate similarity between PDG's of all correct solutions.
    - This γ-isomorphism method checks for sub-graph isomorphism between the PDG's and calculates similarity measure.

References:

Chao Liu et al. "GPLAG: Detection of Software Plagiarism by Program Dependence Graph Analysis". In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '06. Philadelphia, PA, USA: ACM, 2006, pp. 872–881. ISBN: 1-59593-339-5. DOI: 10.1145/1150402.1150522. URL: http://doi.acm.org/10.1145/1150402.1150522.