Project Progress Report (SimFix++)

Yangruibo Ding (yd2447)
Joe Huang (jch2220)
Columbia University, New York, NY

What we plan to do?

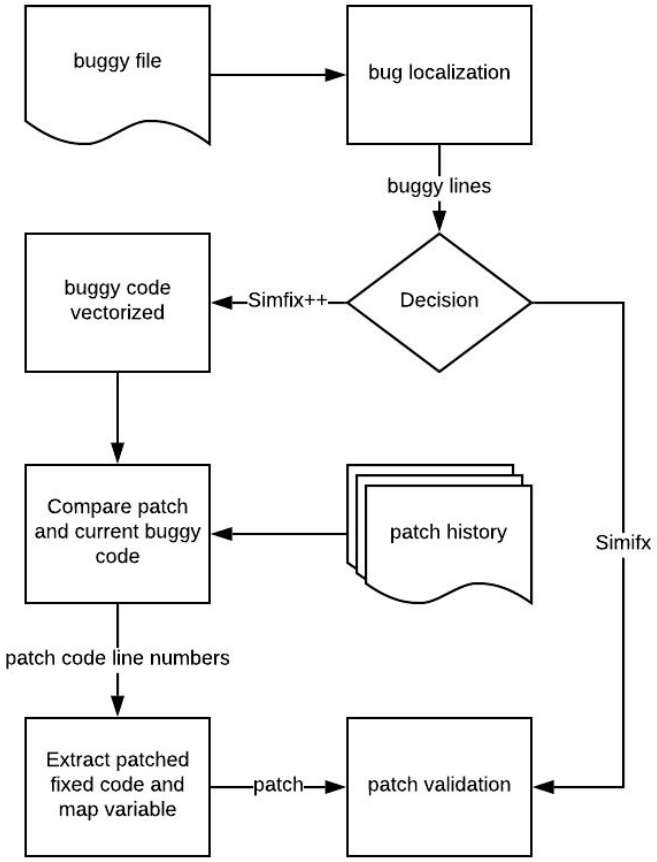

Fig1. SimFix++ Workflow

We give our tool a name of "SimFix++" because the tool will be developed based on the SimFix[1] tool.

We would like to design an automatic bug fixing tool that utilizes a different approach to utilize the existing patch. Previously, SimFix uses the existing patches by extracting the patch that fixes buggy codes and applies those patches to reduce the search space of others produced from similar code. This is beneficial that common bug fixing modification behavior will be prioritized and could prevent the unlikely fix. However, only the

[1] "GitHub - xgdsmileboy/SimFix: Automatically fix programs by ...." https://github.com/xgdsmileboy/SimFix.

fixed patch codes are used to determine if the modification is applicable and any relationship between the patch buggy code and current buggy code is not captured.

To address this issue, we want to take an alternative approach and use the existing patch differently. Instead of finding the modifications in patches, we will compare the current buggy code with the patch buggy code. For every patch buggy code, we will calculate a similarity score with the current buggy code using NLP models. Finally, for top *k* similar patch buggy code, we will utilize its patch to fix the current buggy code.

In Fig1, we show the workflow of our tool. The general input of our tool will be the buggy files, and the bug localizer will give us the buggy lines. Once the buggy lines are obtained, users could decide to use SimFix, SimFix++, or both. For SimFix, nothing is changed from the original code. For SimFix++, we will first convert the buggy to vector and look for similar patch buggy code using NLP models. Finally, both techniques would validate using the produced patches.

What is the research questions we are trying to answer?

How does SimFix++ performance improve compared to that of SimFix?

What is our progress?

To implement SimFix++ on top of SimFix, we need to extract the buggy lines from SimFix, produce patches by applying SimFix++, and perform patch validation using SimFix's strategy. We are currently bridging SimFix++ and SimFix, such that users could use either or both SimFix and Simfix++. We have added some new arguments to separate our new functional branch with the original logic. We have also implemented new functions for similarity calculation between code by using "edit distance" and "Bag-of-words", but which version will be used for the final version of SimFix++ should depend on the experiments.

We have extracted the commit history of the related projects inside Defects4j, which contains more than 22000 pieces of bug-fixing data. We separated the data into the buggy version and the patched version, and placed them into two files. Our newly designed branch will search similar code from the buggy version of commit history and locate the corresponding patched version in the patched version to use it as candidate patch.

How do we evaluate our tool?

Our tool would complement Simfix tool by adding an additional functional branch to improve the performance of original Simfix. Therefore, the evaluation will be based on the same bug database with Simfix used, "Defects4J". In the previous work of Simfix, it can fix 34 bugs in this bug database. In our opinions, our SimFix++ should at least not crash the previous version of SimFix. The overall performance could be both better or worse than before, but applying the patch history will be a good start to integrate the OpenNMT model for future research. However, out of the same intuition but based on different material, the newly designed branch to search similar code in commit history is expected to have similar performance with original version.

How we plan to deliver the code, documentation and other software artifacts for your project

We will post the codes, documentation, and any relevant files to Github repository, which probably is a private one, and we will add the reviewers of our project as collaborators.

How we plan to demo

For our demo, we are planning to demonstrate SimFix++ by briefly explaining the tool and the buggy program for testing, showing a walkthrough of the tool, and finally discussing the evaluation of its performance. Specifically, we will focus on showing the code and data we added to SimFix, and run the tool to show its performance