CS 569 Project Report

Part #3

He Zhang

zhangh7@oregonstate.edu

Description of Algorithm

According to my last version, I make three main changes to improve the efficiency of test algorithm. The changes are described below:

1. Since pure random tests always repeat paths that has been covered, I set a new parameter as "rtTimePara", which determines a test time (time rate) spending on pure random test. I can change this parameter to find out a good time to stop using pure random test when it does not cover any new paths (which basically means it cannot find new bugs). Another way to realize this is not simple set an integer number, but to observe the behavior of random test, when it does not produce useful cases, the algorithm can change it to next phase. So next time it may be a possible way to improve by adding this feedback mechanism.
2. Since traditional breadth first search (BFS) is very slow and can only detect shallow depth, I change to use an improved BFS, which may be called sparse BFS. The point of this change is to limit the number of nodes in a level during BFS. For instance, now we just search ten enabled actions in a state, and just add these ten following states (produced by ten relative actions) into the search queue. Before using, we need to shuffle the new queue. The improved BFS is very fast to reach the maximum depth compared to the traditional BFS. And it may test more combination cases. When the test sequence, which length is proportional to search depth, is longer, it is more likely to find a bug. So the improved BFS has a better performance.
3. When pure random test is finished (controlled by pure random test time parameter), it means normal paths has been covered already. So the left time need to find new branches (or new statements). So after pure random test, I add genetic algorithm in my new algorithm. When random test is running, the coverage information of each test case is recorded. And after random test phase, using this information to find high coverage test cases and mutate them. As it may cover new branches, it may find new bugs.

The algorithm can be described briefly below:

```
MainAlgorithm
{
    While time <= timeout/rtTimePara:
    {
        Do random test
        If find a bug when state is s:
            While BFS layer <= Target and time <= timeout/rtTimePara:
```

```
                Do sparseBFS test start from s (or its neighbor)
    }
    While time <= timeout:
        Do mutate test
}

sparseBFS(queue):
{
    Get s11 from queue
    Search n actions of state s11
    Get new states [s1,…, sn]
    Add [s1,…, sn] to frontier
    Shuffle frontier
    Queue <- frontier
}
```

Test Result Analysis

I using this algorithm to test avlbug1.py from class repo and linkedlist.py. The results is better. For avlbug1.py, this algorithm finds out its two bugs, and for linked list.py this algorithm finds out its 2 bugs out of 3. The setup timeout is 40 seconds and depth is 100. The bug information can be found in faults.test after running test code.
Compared to last version, the coverage has been improved. When running mutate test, it also find some bugs, but not new bugs. I think this is because the SUT I use does not contain so many bugs.

Possible improvement of the Algorithm

As said in the first part, adding feedback mechanism in pure random test phase may be a possible way to improve. Instead of trying proper parameter to control pure random test time, the feedback mechanism can find out the best time by itself.