Test Case Prioritization

Mihail Milchev Mihaylov

Reg. No: 201888925

1. Introduction

This report is used to describe issues regarding regression testing. More specifically, how to ensure the best selection of tests is run within a limited resource environment. This is done with the help of 3 algorithms: Genetic Algorithm, Hill Climb Algorithm and Random Search Algorithm. Those are later used to compare the results and check the efficiency of each of them. The programming language used is Java. The input file that is given is a text file containing all of the tests and the results from these tests. In this case there are two text files: a small one with 216 tests and a big one with 1000 tests. Each line in the files consists of a test number followed by 9 (in the small data set) and 38 (in the big data set) faults with their results indicated by a 1 or a 0 depending on the faults the test finds.

2. Key Aspects

2.1 Representation

The representation is one of the key elements when working with the 3 aforementioned algorithms. In this assignment, there is a text reader implemented which reads the lines from the files and stores them in a list of objects called “Test”. These object stores the name of the test and the result from the test. The result is stored in a string. Another object implemented is called “Solution”. This object is used to store the list of tests which a solution holds and evaluate their fitness function. All solutions are stored in lists with objects “Solution”.

2.2 Fitness function

The fitness function is used to determine the best solution of the problem. There are 2 aspects considered when evaluating the fitness function. The first one is coverage which is necessary to ensure that the highest number of faults have been uncovered in the given solution. The second one is order of the tests in a solution. It is crucial that the maximum number of faults are uncovered as early in the testing as possible. Those 2 aspects can be covered using a formula called APFD (Average Percentage of Faults Detected). The formula used goes:

$$\mathrm{APFD}(\sigma_j, P, \Phi) = 1 - \frac{\sum_{\phi_f \in \Phi} reveal(\phi_f, \sigma_j)}{|\sigma_j||\Phi|} + \frac{1}{2|\sigma_j|}. \quad (1)$$

This can be calculated by subtracting the sum of the index of the test where the faults first appear divided by the multiplication of the number of tests in a solution and the number of maximum possible faults in a test from 1 and then adding one over 2 times the number of tests in a solution. The highest possible evaluation would be 1, which can only be accomplished if only 1 test is ran and it uncovers all of the faults in the test.

3. Algorithms

3.1 Genetic Algorithm

Genetic Algorithm is an algorithm which starts with an initial population of randomly generated solutions, which is then ordered using the fitness function and using crossover and mutation new generations of solutions are created and evaluated until a termination event is reached. The crossover is between 2 parent solutions from the previous generation. A random point is selected and a new solution is created by taking the first half of the first parent solution and then going through the second solution from the start adding tests which are not already in the child solution until the child solution is the same size as the parent solution. Then another child solution is created using the first part of the second parent solution and then the same way going through the first parent and adding tests which are not already in the child solution. The first 10% of a new generation are selected using parents from the first 10% of the previous generation. The rest of the new generation is created using a crossover between the entire previous generation. The crossover chance selected for this project is 80% which means that in 80% of the cases a crossover between 2 parents occurs. In the remaining 20% the 2 parents proceed to the new generation unchanged. Another way to modify the solutions is using mutation. Mutation is when in a small amount of cases a single test in a solution is randomly changed with another test. In this assignment the mutation chance is set to 5%. Which means around 1 in 20 solutions would have a mutated test. The final key component in the genetic algorithm implemented is the termination function. This function is used to determine when the best possible solution is reached. For this Genetic Algorithm the termination event is if a better solution is not reached in the last 100 generations, consider this the best solution. The size of the initial population and the number of tests for the solutions are discussed in the results section of this report.

3.2 Hill Climbing Algorithm

Hill Climbing algorithm is an algorithm which creates a random solution and then slightly modifies it until the best modification of the solution is reached. Hill Climbing is a local search algorithm, but if more than 1 random solution is picked and it's neighbours evaluated it can be used in a more global search of all possible solution. The number of random solutions inspected is depending on the number of tests in a solution. The number of checkpoints and the number of tests in a solution are discussed in the results section of this report. The neighbours of a solution for this algorithm are solutions with 1 test moved

around in the order of the solution and also solutions with 1 test changed with the next or the previous test in the full data set of tests.

3.3 Random Search Algorithm

The Random Search Algorithm creates a certain number of randomly generated solutions and returns the best of them as a final solution. In this assignment random search is used to compare the results from this algorithm against the results from the Genetic Algorithm and the Hill Climbing Algorithm. The number of randomly generated solutions is discussed in the results section.

4. Results

4.1 Small data matrix

For the small data matrix each of the solutions consists of 5 tests. After running the Genetic Algorithm a fair amount of times, the size of the population was determined to be 500. With a population of 500 this algorithm reaches the best possible solution with 5 tests in 9 out of 10 runs. The size of the population is inversely proportional to the termination event and the number of generations required to reach the best solution. For example, with the termination event being to stop after 100 generations if no better solutions are found, and the size of the population is 700, the best possible solution would be reached every time the algorithm is run. The highest evaluation for the small data matrix with solutions consisting of 5 tests is: 0.76666667/1;

After running the Hill Climbing Algorithm the optimal number of checkpoints in order to reach the best solution is considered to be 10,000. The results from considering 10,000 checkpoints give an evaluation of around 0.7/1 in 9 out of 10 runs.

In order to achieve similar results using the Random Search Algorithm it is required to create 50,000 random solutions.

4.2 Big data matrix

For the big data matrix the number of tests in a solution depend on the resources available to run those tests. For this matrix solutions with 25, 40 and 100 tests are considered. The hill climbing algorithm uses a population of 5,000 solutions. The number of checkpoints in the Hill Climbing algorithm are set to be 10,000 and the number of randomly generated solutions for the Random Search Algorithm is 100,000. Another thing to consider about the big data matrix is the time it takes to run the algorithm. Since the number of tests in a solution greatly increases the number of neighbours in the Hill Climbing Algorithm, the time it takes to run this algorithm is a lot higher than the time for the Genetic Algorithm or the Random Search. Due to that fact the number of checkpoints considered is limited to only 10,000, which in turn does not yield as high results as the genetic algorithm. Even though this algorithm is a lot slower and limited than the Random Search algorithm it still gives better results than it because of its more targeted approach. The time it takes to run the Hill Climbing algorithm with 40 tests in a solution and 10,000 checkpoints is around 3 times more than the time to run the Genetic Algorithm with a population of 10,000. However,

lowering the number of checkpoints as the number of tests in a solution increase would greatly lower the run time and would still provide optimal results in some runs.

5. Conclusion

To sum up, after running the three algorithms numerous times with different values it has been determined that when the small data matrix is considered with 5 tests in a solution, the results and the run-time of the 3 algorithms are very similar, with genetic algorithm reaching the best possible solution a greater number of times than the other 2 algorithm. The same can not be said about the big data matrix. Due to the increased total number of tests and the number of tests in every solution the genetic algorithm proves to be the most effective of the 3 algorithms and would always reach a better solution and is also faster than the hill climbing algorithm. As for the random search algorithm, is always at a disadvantage because it lacks the targeted approach of the genetic algorithm.

References:

1. “Time-Aware Test Suite Prioritization” by Walcott el al.