ALGORITHMS FOR TEST CASE PRIORITIZATION IN REGRESSION TESTING

[1]Anku Sharma, [2]Nancy Sehgal
[1]M.Tech Student, [2]Assistant Professor
[1]Computer Science and Architecture,
[1]Baddi University of Emerging Science and Technologies, Baddi , India

Abstract :Every time designers modify or change their software, even a small twist can have unpredicted consequences. Regression testing is testing current software to make sure that addition or a change hasn't damaged any current functionality. Its main aim is to identify the bugs that may have been accidentally familiarized into a new release candidate, and to guarantee that earlier eliminated bugs last to stay dead. Regression testing is mainly performed in three ways-Retest all, Regression test selection , Prioritization of test cases. Re-test all is a method that checks all the test cases on the present program to check its reliability. Unlike Retest all , Regression test selection executes a portion of the test suite if the cost of choosing the part of the test suite is less than the Retest all technique. Prioritize the test cases so as to increase a test suite's rate of fault detection. Test case prioritization techniques plan test cases so that the test cases that are greater in priority are performed before the test cases that have a lesser priority .Test case prioritization is done by using various algorithms like-Greedy algorithm,Genetic algorithm, Additional greedy algorithm ,Hill climbing,2-Optimal algorithm ,Ant colony algorithm, Particle swarm algorithm ,Bat algorithm.

IndexTerms **– Regression testing,Test case prioritization, Search algorithms,Nature based algorithms.**

1. INTRODUCTION

Regression means retesting the unchanged parts of the application. Regression Testing is the one in which test cases are re-executed in order to check whether the previous functionality of the application is working fine and the new changes have not introduced any new bugs. This test can be performed on a new build when there is a significant change in the original functionality that too even in a single bug fix.This is the method of verification. Verifying that the bugs are fixed and the newly added features have not created any problem in the previous working version of the software.Testers perform functional testing when a new build is available for verification. The intent of this test is to verify the changes made in the existing functionality and the newly added functionality as well. When this test is done, the tester should verify whether the existing functionality is working as expected and the new changes have not introduced any defect in functionality that was working before this change. Regression test should be a part of the release cycle and must be considered in the test estimation. Regression testing is usually performed after verification of changes or a new functionality. But this is not the case always. For the release that is taking months to complete, regression tests must be incorporated in the daily test cycle. For weekly releases, regression tests can be performed when the functional testing is over for the changes.

Regression testing is initiated when a programmer fixes any bug or adds a new code for a new functionality to the system. There can be many dependencies in the newly added and the existing functionality. It is a quality measure to check whether the new code complies with the old code so that the unmodified code is not getting affected. Most of the time testing team has the task to check the last minute changes in the system. In such situation, testing only affected application area is necessary to complete the testing process on time with covering all the major system aspects.This test is very important when there is a continuous change/improvement added in the application. The new functionality should not negatively affect the existing tested code.

1.1. Need of Regression testing

- Alteration in desires and code is changed according to the condition.
- Fault fixing
- Performance issue fix

1.2. Regression testing techniques

- **Corrective Regression Testing:** It is used when there is no alteration in the specifications and test cases can be reprocessed.
- **Progressive Regression Testing:** It is used when the changes are made in the specifications and new test cases are considered.
- **Retest-All Strategy:** This strategy is expensive and time consuming because all tests are reused which results in the execution of unnecessary test cases
- **Selective Strategy:** In this strategy, subset of the existing test cases is used to lower the retesting effort and cost

1.3. Test case prioritization

Rothermel et al. [1] express the test case prioritization problem and define several subjects applicable to its way out.

Definition: The Test Case Prioritization Problem: Given: T, a test suite, PT, the set of permutations of T, and f, a function from PT to the real numbers.
Problem: Find T'ϵ PT such that (T'') (T ∀ PT) (T''≠T') [f(T')≥f(T'')].

Here, P T represents the set of all possible prioritizations (orderings) of T and f is a function that, applied to any such ordering, yields an award value for that ordering. There are many possible goals for prioritization; [30] describes several. Rothermel et al. [1] differentiate two types of test case prioritization: version-specific and general.In general test case prioritization, given program P and test suite T, test cases in T are prioritized with the aim of identifying a test case sequence that will be useful over a sequence of later altered versions of P.The hope is that the resultant prioritized suite will be additional effective than the original suite at meeting the goal of the prioritization, on average over those subsequent releases.In contrast, in version-specific test case prioritization, given program P and test suite T, test cases in T are prioritized with the motive of finding an ordering that will be beneficial on a specific version P0 of P. Version-specific prioritization is achieved after a set of variations have been made to P and prior to regression testing P0.

1.3.1 Objectives of test case prioritization

- Testers can rise the rate of fault detection.
- Testers may demand to rise the rate of detection of high-risk faults, finding those faults prior in the testing process.
- Testers may increase the probability of illuminating regression errors connected to specific code changes prior in the regression testing process.
- Testers may rise their coverage of coverable code under test at a quicker rate.Testers may also increase their self-confidence in the reliability of the system under test at a faster rate.

2. SEARCH ALGORITHMS

2.1 Greedy Algorithm: This Algorithm is an execution of the "next best" search attitude. It works on the principle that the part with the maximum weight is taken first, followed by the element with the second-highest weight, and so on, until a complete, but possibly suboptimal, solution has been constructed. Greedy search pursues to minimize the predictable cost to touch some goal. It is simple, but in circumstances where its results are of high quality, it is smart because it is typically inexpensive both in execution time and implementation.For example- a statement coverage for a program covering m statements and a test suite covering n test cases.

2.2 "Additional" Greedy Algorithm: The "Additional" Greedy Algorithm is also a kind of Greedy Algorithm, but with a altered strategy. It chains feedback from previous selections. It iteratively selects the maximum weight element of the problem from that part that has not already been consumed by previously selected elements.

2.3 Two-Optimal Algorithm: The 2-Optimal (Greedy) Algorithm is an instantiation of the K-Optimal Greedy Approach when K ¼ 2. This approach chooses the next K elements that, taken together, consume the largest part of the problem. In theis case, it is the major remaining part of the problem that is nominated.The K-Optimal approach has been considered in the area of heuristic search to crack the Traveling Salesman Problem (TSP) that is defined as "find the cycle of minimum cost that visits each of the vertices of a weighted graph G at least once" [2].

2.4 Hill Climbing: This is a well-known local search algorithm. There are two primary differences of this strategy: steepest ascent and next best ascent. In our empirical study, steepest ascent is adopted. The steepest ascent Hill Climbing algorithm for test case prioritization is composed of the following steps:

1. Choose a random solution state and make this the current (i.e., initial) state.
2. Calculate all the neighbors of the current state.
3. Move to the state with the major rise in fitness from the current state. If no neighbor has a larger fitness than the current state, then no move is made.
4. Repeat the previous two steps until there is no alteration in the current state.
5. Return the current state as the solution state

2.5 Genetic Algorithms: Genetic Algorithms (GAs) signify a class of adaptive search techniques centred on the processes of natural genetic selection according to Darwinian theory of biological evolution.The search continues through a number of generations until the conclusion condition has been met. The initial population is a set of randomly generated individuals. Each individual is signified by a sequence of variables/parameters (called genes), known as the chromosome. The chromosome encodes a imaginable solution to a given problem. The encoding can take many forms, for example, binary, real-valued, or character-based. A biased selection depending on the fitness value chooses which individuals are to be used as the "parents" for producing the next generation. Crossover is a genetic

operator that syndicates two individuals (the parents) to generate a new individual. A chance of crossover decides whether crossover should be accomplished. The mutation operator alters one or more gene values in the individual, depending on the probability of mutation.

3. NATURE BASED ALGORITHMS

3.1 Ant colony optimization

Some notions such as distance and pheromone are used for finding direction in ant colony optimization (ACO) [3] which is a heuristic method of ants, and it also helps in searching for foods. Ants leave pheromone at decision making points to display way for other ants. As the ants see the pheromone, the ants choose their guidelines. There is an opposite connection between decision making and distance.Distance is a negative factor while creating a route decision. In contrast, pheromones lead to a better choice decision. In a function f(x)=qa+(1/w)bf(x)=qa+(1/w)b where these parameters are employed to input, if *a* and *b* are arbitrary generated values, the route is computed by substituting pheromone and distance for *q* and *w*, respectively. The direction is then obtained according to the obtained results. ACO has been used in following areas before: various subfields of data mining [4], real world problems [5], mathematical modeling [6], network routing problems [7], prediction of energy consumption [7], and software testing [8].

3.2 Particle swarm optimization

PSO was hypothetically presented by Eberhart and Kennedy to solve composite numeric optimization problems. This algorithm is created on the moving behavior of bird and fish swarms. Velocities and positions are modernized for each iteration in PSO. While *pbest* is the best values of each and every particle and it is used for finding optimal solution, *gbest* is the finest solution of the group. Moreover, particles alter their velocities and next locations by concerning previous recorded *gbest*. Note that *gbest* is logged for every step that the best value is predictable to be found at the end of the iteration [9].

3.3 Local beam search

LBS was first used by Jiang ve Chan in test case prioritization.It is one of the input-based test prioritization techniques and it was established on the basis of adaptive prioritization [10]. Including all input space along with few test cases is the underlying goal of LBS which is important to detect more faults. Jiang and Chan also described that LBS was superior to the other related search-based algorithms such as two-optimal, greedy, GA and hill climbing in expressions of code coverage criteria.

3.4 Bat algorithm

Bats hunt to catch their foods. The shooting method of bats is measured as their echolocation behavior [11]. Bats have following talents for hunting:
1.They can simply catch the position of their prey;
2.Bats differentiate between prey and obstacles;
3.Bats prefer optimal location during finding prey.

Some algorithms are established by taking inspiration from echolocation behavior of bats. For instance, bat algorithm is a nature-inspired optimization method and it was offered by Yang et al. Some parameters, which are continually updated with the moves of bats, are used in bat algorithm [12]. These are vi, fi, xi, ri, and Ai. The character "*i*" of the parameters denotes iteration. Bats travel in a position xi along with a vi. While *v* represents the velocity of a bat, *f* is the frequency value. If velocity is divided with frequency, wavelength is obtained with λ=vfλ=vf. Each bat emits ultrasonic sound, namely pulses, to get signal from a prey. Depending on the proximity of the target, pulse emission ri increases and changes between 0 and 1. Having ri=1ri=1 means that a bat is at the target. Loudness Ai can be determined as the environmental hardship during finding prey. On the contrary, Ai decreases dramatically if a bat is near the prey and Ai becomes "0" at the target. Velocity and frequency are updated as in Eqs. (3–4) and position of a bat changes according to these iterations.
These notions are used for deciding the right path so that bats can find their prey as soon as possible. Moreover, as the distance from a bat to prey increases, the related frequency fi reduces. Pulse emission ri is [0−1][0−1], but it depends on the wavelength emitted by a bat. The steps of the bat algorithm can be summarized as follows:
1.Starting points and velocities are determined,
2.ri and Ai are assigned.
3.local solution is found by checking rand> rir and>ri in a specific iteration such as 20, 50, and 100,
4.new solution is calculated by flying randomly,
5.velocities and frequencies are updated,
6.global best location is denoted by x∗ and it is found by ordering local solutions.

4. REFERENCES

[1] G. Rothermel, R.H. Untch, C. Chu, and M.J. Harrold, "Prioritizing Test Cases for Regression Testing," IEEE Trans. Software Eng., vol. 27, no. 10, pp. 929-948, Oct. 2001.

[2]Wang, J., Osagie, E., Thulasiraman, P., Thulasiram, R.K.: HOPNET: a hybrid ant colony optimization routing algorithm for mobile ad hoc network. Ad Hoc Netw. 7, 690–705 (2009)

[3]Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. Part B 26, 1941

[4] Parpinelli, R.S., Lopes, H.S., Freitas, A.A.: Data mining with an ant colony optimization algorithm. IEEE Trans. Evol. Comput. 6, 321332 (2002).

[5] Bell, J.E., McMullen, P.R.: Ant colony optimization techniques for the vehicle routing problem. Adv. Eng. Inform. 18, 4148 (2004)

[6] Blum, C., Sampels, M.: An ant colony optimization algorithm for shop scheduling problems. J. Math. Model. Algorithms 3, 285–308 (2004)

[7] Wang, J., Osagie, E., Thulasiraman, P., Thulasiram, R.K.: HOPNET: a hybrid ant colony optimization routing algorithm for mobile ad hoc network. Ad Hoc Netw. 7, 690–705 (2009)

[8] Duran Toksar, M.: Ant colony optimization approach to estimate energy demand of Turkey. Energy Policy 35, 3984–3990 (2007).

[9] Yang, Q., Tian, J., Si, W.: An improved particle swarm optimization based on difference equation analysis. J. Differ. Equ. Appl. 23(1-2), 135–152 (2017)

[9] Huaizhong, L., Peng Lam, C.: An ant colony optimization approach to test sequence generation for statebased software testin. In: Fifth International Conference on Quality Software (QSIC05), pp. 255–264. IEEE (2005)

[10] Jiang, B., Zhang, Z., Chan, W.K., Tse, T.H.: Adaptive random test case prioritization. In: 2009 IEEE/ACM International Conference on Automated Software Engineering, pp. 233–244. IEEE (2009)

[11]Yang, X.S.: Bat algorithm for multi-objective optimisation. Int. J. Bio-Inspired Comput. 3(5), 267–274 (2011).

[12]Wang, G.G., Chu, H.E., Mirjalili, S.: Three-dimensional path planning for UCAV using an improved bat algorithm. Aerosp. Sci. Technol. 49, 231–238 (2016)