CHAPTER FOUR
THE PROPOSED ALGORITHM

4.1. Introduction

4.2. Development of the Proposed Algorithm

4.3. Developing the Fitness Function

4.4. Conclusion
Chapter 4:

The Proposed Algorithm

4.1. Introduction:

In software testing, there is a dire need to choose the test cases carefully to achieve the necessary code coverage in a faster rate while avoiding replication. In the area of automatic test data generation using GA, the definition of the objective fitness function is crucial.

For path coverage criteria there is only few GA-based test data generators. Most of the existing studies are experimental to the GA while using very big population size and large number of iterations.

In this chapter, a new algorithm for automatic test data generation is proposed, and the development of the fitness function is explained. An illustration example of the path coverage test is also given. The proposed algorithm in this study defines a new fitness function (enhancement of an existed function) to optimize the effectiveness and
efficiency of the algorithm to achieve full path coverage in less number of iterations using a smaller pop-size i.e. less number of test data.

The development of the proposed algorithm is explained as below; further the fitness function is discussed thoroughly. Thereafter, the preparation of the experiments is described. A brief description of the experimental programs is also expounded.

4.2. Development of the Proposed Algorithm

The test data generation problem for path adequacy criterion can be described as follows: for program $P$ and path $U$, find input $x \in S$, so that $P(x)$ traverse path $U$, where $S$ is the set of all input values.

If all paths through a program have been executed in some test involving a test set $S$, then the path coverage or control flow coverage is 100%.

The proposed algorithm generates test cases with respect to certain path in the control flow graph of the code to be tested. It attempts to identify a test case which is appropriate for the traversal of the path. The algorithm takes the following as inputs: an instrumented version of the program to be tested, the list of paths to be covered (i.e. test requirement to be covered). Also, it accepts the
GA parameters: population size \((\text{pop-size})\), maximum number of generations \((\text{max-gen})\) and probabilities of crossover \((P_c)\) and mutation \((p_m)\).

One way of measuring the quality of the test suite is by performing \textit{code coverage analysis}. This is done by comparing the test suite against the code according to the chosen coverage criteria. The result of the comparison is a quantitative measurement of the coverage, plus an indication of parts of the code that need to be propounded by adding further tests.

The tasks involved in the proposed test data generator are illustrated in fig (4-1) below. The program's source code is first instrumented i.e. modified by adding few lines to identify the goal-paths to be covered and keep track of the path traversed by the input test data. The instrumentation is used to return information that helps to calculate fitness values for the proposed GA.

Path selector selects the goal-path to be covered from the paths-list. The GA generates test data (a generation of individuals). When the program under test is executed by using these test data, the traversed current path is determined and then fitness value is obtained using the proposed SMS fitness function. The fitness value is calculated to help the algorithm search and find the accurate
test cases (fit individuals) that will traverse the edges of the goal-path i.e. cover the path.

Fig (4-1) The Tasks of the Proposed Approach.
Finally, the test data that succeed to cover the selected goal-path is considered fit to be the output of the test data generator. This task is repeated until all goal paths in the paths list are completed. Or otherwise, if the search limits are exceeded and no coverage is achieved then the path is marked as 'uncovered' and that the test data generator has failed to cover that goal-path.

### 4.3. Developing the Fitness Function:

The aim of this work is improving fitness function calculation, which is the costlier component of genetic algorithm, to get performance improvements. Our approach is aimed to develop GA algorithm to generate test data for path coverage. The fitness function proposed to evaluate each test data is a modification to the Hamming Distance, which quantifies the distance between two given paths. The fitness function in this work is named Shifted-Modified-SIMILARITY (SMS). Given a target path and a current one, the similarity is calculated from cascaded edges for each of the paths being compared using hamming distance or symmetric differences (the symmetric difference between set A and set B is \((A \oplus B)\)). Similarities are then normalized and summed associated with a weighting factor which is usually found by experience, to obtain the total similarity. This value is used as the objective function used to evaluate individuals in the population.
For path testing, two different paths may contain the same branches but in different sequences [66]. To increase path coverage within limited time and computational efforts, we proposed the (SMS), where we shift paths being compared to right and to left for calculating that distance.

Normally, the absolute measure of distance between a goal path (goal_p) and a current path (current_p) is the number of unmatched edges between the two paths, given in equation (1):

$$D = (\text{current}_p) \oplus (\text{goal}_p)$$ ................................. (1)

The objective is to minimize the distance $D$, and to build a fitness function to minimize distance and increase similarity.

In this approach, the similarity with right-shift and similarity with left-shift ($S_{RS}$) and ($S_{LS}$) respectively, is calculated. The similarity with shift to right ($S_{RS}$) between the goal_p and the current_p is:

$$S_{RS} = n_i(n_i - D_{RS})$$ ................................. (2)

Where $D_{RS}$ is distance with shift to right, and $n_i$ is the number of edges to be compared at step i.

In the same way, the similarity with shift to left ($S_{LS}$) between goal_p and current_p is:
Chapter 4: The Proposed Algorithm

\[ S_{LS} = n_i(n_i - D_{LS}) \] ................................................................. (3)

Where \( D_{LS} \) is distance with shift to left, and \( n_i \) is the number of edges to be compared at step \( i \).

The factor \( n_i \) is used in equations (2) and equation (3) in order to differentiate the good and the bad test cases. As test cases usually converge, the difference between their fitness values may become very small, and best solutions cannot have significant advantage in the selection process (see fitness-scaling in section 3.6.).

Finally, the Shifted-Modified-SIMILARITY function (\( SMS \)) is used to calculate the total similarity between \( \text{goal}_p \) and \( \text{current}_p \) as follows:

\[ SMS = \sum_{i=1}^{lc} (S_{RS} + S_{LS}) \] ................................................................. (4)

The length counter \( (lc) \) counts the minimum number of edges encountered in the \( \text{goal}_p \) and \( \text{current}_p \).

\[
lc = \begin{cases} 
L_{\text{goal}_p}, & \text{if} \ L_{\text{goal}_p} \leq L_{\text{current}_p} \\
L_{\text{current}_p}, & \text{otherwise}
\end{cases}
\] .............................................. (5)

\( L_{\text{goal}_p} \) and \( L_{\text{current}_p} \) are lengths of \( \text{goal}_p \) and \( \text{current}_p \) respectively.
Using SMS as an objective fitness function, the algorithm works as follows: for every test case in the population, a fitness value is calculated (according to its success in covering common edges with the selected goal path) and similarity between selected goal path and current path is obtained.

Then production process is carried out, roulette wheel selection method is used to determine the candidate parents for crossover. When pair is determined, crossover point is randomly selected then crossover itself is carried out.

When crossover is finished, mutation procedure is called out. Mutation operator is applied with given probability over whole population. This encourages the genetic algorithm to discover new test cases. Mutation though, plays secondary role in the genetic algorithm most important factors are still accurateness of fitness function and crossover mechanism.

Next, two new candidates are selected and crossed over and again two new individuals are produced. This takes place N/2 times when population size is N.
Chapter 4: The Proposed Algorithm

Creation of new population is finished at this point. It replaces old population completely.

This process continues until all paths from the list of goal-paths are covered or limits of evolutionary algorithm is exceeded.

Fig (4-2) illustrates the steps of $S_{RS}$. As an example, we assumed that $current\_p=(a,b,c,d,e,f)$ and $goal\_p=(b,d,q,h,m,e)$. As shown in the figure, the $current\_p$ is shifted one edge to the right after each distance calculation.

The first distance is calculated by XORing the encountered edges of $current\_p$ and $goal\_p$ which are $(a-b-c-d-e-f)$ and $(b-d-q-h-m-e)$. The next step is to shift the current path one edge to the right. The second distance is to be calculated by XORing encountered edges of $current\_p$ and $goal\_p$ which are $(b-c-d-e-f)$ and $(b-d-q-h-m)$. The current path is shifted again one edge to the right. The third distance is calculated by XORing encountered edges of $current\_p$ and $goal\_p$ which are $(c-d-e-f)$ and $(b-d-q-h)$. After that we shift the current path one edge to the right. The fourth distance is calculated by XORing encountered edges of $current\_p$ and $goal\_p$ which are $(d-e-f)$ and $(b-d-q)$. And after shifting the current path one edge to the right, fifth distance is calculated by XORing encountered edges of $current\_p$ and $goal\_p$ which are $(e-f)$ and $(b-d)$. Finally,
the current path is shifted one edge to the right. The sixth distance is calculated by XORing encountered edges of current-p and goal-p which are (f) and (d).

To illustrate the proposed algorithm's fitness-function calculation a small example for calculating SMS between the target path and current path is given below. Consider the control flow graph in fig (4-3).

The list of goal paths to cover the control flow graph is as follows:

Path1= b-i

Path2= b-f-h

Path3= a-d-g-h

Path4=a-c-e-g-h
For path coverage, all the 4 paths required to be covered to satisfy the full path coverage criterion. Suppose that path₃ is the goal-path and suppose that path₂ and path₄ executed as current path in different executions. To decide which path is closer to satisfy goal-path, the objective function SMS is used to calculate the distance and similarity between paths.

Case 1: goal-path is path₃ (a - d - g - h)

Current-path is path₂ (b - f - h)

At first, let's calculate the similarity with right shift (S_{RS}) between P₃ and P₂
$S_{RS}$ calculation between (P$_3$-P$_2$):

Obtaining the first distance between P$_3$ and P$_2$ $D_1$. The minimum number of edges encountered between the two paths is now 3 so the number of edges to be compared is 3 which is $(a-d-g)$ and $(b-f-h)$.

$$D_1 = (a - d - g) \oplus (b - f - h) = 3 \Rightarrow S_{RS1} = 3(3 - 3) = 0$$

Next, the second distance $D_2$ is calculated. After shifting the current path one edge to the right, the minimum number of edges encountered between the two paths is now 2 so the number of edges to be compared is 2 which is $(d-g)$ and $(b-f)$.

$$D_2 = (d - g) \oplus (b - f) = 2 \Rightarrow S_{RS2} = 2(2 - 2) = 0$$

Finally, the third distance $D3$ is calculated. After shifting the current path one edge to the right, the minimum number of edges encountered between the two paths is now 1 so the number of edges to be compared is 1 which is edge $(d)$ and edge $(b)$

$$D_3 = (g) \oplus (b) = 1 \Rightarrow S_{RS3} = 1(1 - 1) = 0$$
Now, in the same way the similarity with left shift ($S_{LS}$) between P₃ and P₂ is calculated.

$S_{LS}$ calculation between (P₃-P₂):

$D_1 = (a - d - g) \oplus (b - f - h) = 3 \Rightarrow S_{LS1} = 3(3 - 3) = 0$

$D_2 = (a - d) \oplus (f - h) = 2 \Rightarrow S_{LS2} = 2(2 - 2) = 0$

$D_3 = (a) \oplus (h) = 1 \Rightarrow S_{LS3} = 1(1 - 1) = 0$

The similarity is now found using equation (4) as follows:

$SMS = \sum_{i=1}^{3} (S_{RS} + S_{LS})$

$SMS_{P3-P2} = (0+0) + (0+0) + (0+0) = 0$

Case 2: goal-path is path₃ (a-d-g-h)

Current-path is path₄ (a-c-e-g-h)

$S_{RS}$ calculation between P₃-P₄:

$D_1 = (a - d - g - h) \oplus (a - c - e - g) = 3 \Rightarrow S_{RS1} = 4(4 - 3) = 4$

$D_2 = (d - g - h) \oplus (a - c - e) = 3 \Rightarrow S_{RS2} = 3(3 - 3) = 0$

$D_3 = (g - h) \oplus (a - c) = 2 \Rightarrow S_{RS3} = 2(2 - 2) = 0$

$D_4 = (h) \oplus (a) = 1 \Rightarrow S_{RS4} = 1(1 - 1) = 0$
Chapter 4: The Proposed Algorithm

\(S_{LS}\) calculation between (P₃-P₄):

\[D_1 = (a - d - g - h) \oplus (a - c - e - g) = 3 \Rightarrow S_{LS1} = 4(4 - 3) = 4\]

\[D_2 = (a - d - g) \oplus (c - e - g) = 2 \Rightarrow S_{LS2} = 3(3 - 2) = 3\]

\[D_3 = (a - d) \oplus (e - g) = 2 \Rightarrow S_{LS3} = 2(2 - 2) = 0\]

\[D_4 = (a) \oplus (g) = 1 \Rightarrow S_{LS4} = l(1 - 1) = 0\]

The total similarity will be:

\[SMS = \sum_{i=1}^{4} (S_{RS} + S_{LS})\]

\[SMS_{P3,P4} = (4+4) + (0+3) + (0+0) + (0+0) = 11\]

Case (1) and case (2) show that path4 (P₄) is closer to the goal-path P₃ than P₂ which means that similarity between P₃ and P₄ is More than similarity between P₃ and P₂.

Using \(SMS\), an input test data that executes Path₄ as current-path is considered more fit than a test data that executes path₂.

As a result of this comparison, a test case that covers P₄ is more fit to be selected for reproduction of next generation.
There is a need to make special preparation for testing, including the modification of the code to be tested. Such preparation can assist in automating the testing process, and in keeping track of the testing results and of statistics about the testing process. Such modification of the code is called *instrumentation*. Instrumentation is used in order to return information regarding values of the fitness.

One would be able to know the path covered by each test data candidate by each test data generated. The path traversed can uniquely be determined by observing the edge selected by the conditional statement generally depends on Boolean expression and edge selected can be identified by observing the values of these expressions.

For the purpose of this study of path coverage criterion, instrumenting a program means recording the edges traversed when branch condition is encountered. The edge traversed by an input test data are fundamental information required to form and calculate the proposed SMS fitness function.

Figure (4-4) below, shows a pseudo code for the proposed algorithm.
Chapter 4: The Proposed Algorithm

Input:
- P*, instrumented copy of the program P.
- List of goal-paths to be covered.
- Population size, pop-size.
- Maximum number of generation, max-gen.
- Probability of crossover, $P_c$.
- Probability of mutation, $P_m$.

Start:
For each goal-p, in the goal-paths list do
Begin:
  goal-p-covered=false
  create first generation randomly
  while (max-gen not exceeded) and (goal-p-covered=false)
  {
    Call GENERATE to generate new population
    If (goal-p-covered=true)
      Report good test data
      Update coverage of paths
  } //end while
End for

The GENERATE procedure
Begin:
  For each chromosome in the current-population do
  {
    Convert chromosomes into decimal values
    Execute P* with the data set as input.
    Evaluate fitness of current test cases using SMS as the objective function
  } end for
  
  Select parents from current-population using roulette wheel method or random method
  create new-population using crossover then mutation operator and $P_c$, $P_m$ parameters
  current-population=new-population
End

Output:
Return set of good-test data.
Return set of goal paths covered.
Report uncover goal-paths if any.

Fig (4-4) The proposed algorithm
The test data generation process will be stopped in two different cases as shown in Fig (4-5):

1. All goal paths have been successfully covered by data generated by the algorithm. And the test cases can form a fit test suite that satisfies the testing objective or adequacy criteria.

2. The objectives of the test data generator have been violated and limits (max no of generation or time given for the task) have been exceeded and yet some test objectives are not satisfied.

Fig (4-5) Termination cases of the test generation process
4.4. Conclusion

This chapter, so far has discussed the details of the proposed algorithm. As a summary, the proposed algorithm generates test data that satisfy the coverage criteria of a selected goal path using GA parameters and operators. The algorithm then measures the quality of the generated test case by comparing it against the code and the paths to be covered. The calculated fitness value of the generated test case helps the proposed algorithm to quantitatively measure the code coverage and indicate the parts of the code that need to be covered by adding further tests. This algorithm is now ready and will be used in this study for experimentation and validation.