CHAPTER-5

PROBLEM STATEMENT & PROPOSED APPROACH

5.1 INTRODUCTION

Testing is execution of software with the intent of finding errors. It requires searching the inputs’ domains for such values which can invoke different output(s) or execute different component(s) making it necessary for employing an efficient search algorithm for test data generation for fulfilling testing objectives (criteria). The previous chapter covered an extensive literature review of the existing research addressing the software testing problem with respect to techniques and performance measures. Also, Majority of papers on automatically test data generation for software testing problems have concentrated on numerous criteria. The most widely considered coverage measures are statement coverage, branch coverage, path coverage, data flow coverage and number of tardy criteria. In present competitiveness environment, cost and time of software testing must be reduced in order to survive in this dynamic environment which is done by production of unique and effective test cases so that coverage of software code is 100 percent with minimum time. The statement coverage provides 100 percent coverage of software code but due to its time consumption and high cost, it is of no use for an industry. Since it is too much expensive and fails to deliver the product in promised time to the customer. In modern software testing environment, industry has to achieve all the objectives resulting into conflicting objectives. Therefore, there is a need of software testing technique through which industry can achieve most of the objectives simultaneously.

The purpose of the automatic software test case generation is to minimize the test cases. Thus, reduce the efforts and cost involved in the testing. For software testing purpose, solution lies in searching inputs, every possible set of inputs represent the global population in search algorithm and selected inputs from this global set are represented by individuals in the population. Suitability of the individuals can be assessed by following a testing criterion for which a unique fitness function has to be defined.

Thus, there is a strong need of a search algorithm which besides having good search capability should be able to some of these testing problems. Search techniques are applied for generation of test data by transforming testing objective into search
problem. Two components are essential for a problem which is to be modeled as search target. First a mechanism should be derived through which the problem is encoded in search algorithm and second component is assessment of the suitability of solutions produced by search technique to guide the individuals for exploring search space [82].

### 5.2 PROPOSED APPROACH
The proposed approach composed of mainly two components.

a) Proposed Technique - Dominance Tree - Concept for reducing the test cases.
b) Proposed Metaheuristic- Genetic-Particle Swarm Combined Algorithm (GPSCA) - which is used to generate automatic test data.

#### 5.2.1 Proposed Technique
This technique applies the concepts of dominance relations between nodes to reduce the cost of software testing.

##### 5.2.1.1 Dominance Tree
Before discussing the concept of Dominance Tree, the concepts about Control Flow Graph (CFG) is must.

*The Control Flow Graph (CFG)* of a program can be represented by a directed graph \( G = (V,E) \) with a set of nodes \( V \) and a set of edges \( E \). Each node represents a group of consecutive statements, which together constitute a basic block. The edges of the graph are then possible transfers of control flow between the nodes. There are two specially designated blocks, the entry block through which control enters into the flow graph and the exit block through which all control flow leaves. Figure 5.2 shows the control flow graph \( G \) of the example program, which is shown in figure 5.1.

*Dominator Tree:* For \( G = (V,E) \), a directed graph with two distinguished nodes \( n_0 \) and \( n_k \), a node \( n \) dominates a node \( m \), if every path \( P \) from the entry node \( n_0 \) to \( m \) contains \( n \). A dominator tree \( DT(G) = (V,E) \) is a directed graph in which one distinguished node \( n_0 \), called the root, is the head of no edge; every node \( n \) except the root \( n_0 \) is a head of just one edge and there exists a (unique) path (dominance path \( \text{dom}(n) \)) from the root \( n_0 \) to each node \( n \).

The root node dominates all nodes. The dominator tree is an ancillary data structure depicting the dominator relationship. This graph is a tree, since each node has a unique immediate dominator. Figure 5.3 gives the dominance tree of program
5.1. The dominance path of node 5 is \( \text{dom}(5) = 1, 2, 3, 5 \).

<table>
<thead>
<tr>
<th>St. No.</th>
<th>Bl. No.</th>
<th>Program statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>program test;</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>var i, j, k : integer;</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>begin</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>i := 0;</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>j := 0;</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>read( k );</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>while (k &lt;&gt; 0) do</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>begin</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>if ( k mod 2 ) = 0</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>then i := i + 1</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>else</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>j := j + 1;</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>read( k );</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
<td>end;</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
<td>write( i, j );</td>
</tr>
<tr>
<td>15</td>
<td>7</td>
<td>end.</td>
</tr>
</tbody>
</table>

**Figure 5.1 Program 1**

**Figure 5.2 CFG**

**Figure 5.3 Dominance Tree**

Figure 5.4 shows the control flow graph of a program for finding the middle value among the given three values. The dominance tree of this control flow graph is shown in figure 5.5.
**Significance of Dominance Tree**

*Reduce the test cases:* Our proposed approach is based on the concept of dominance relationship between nodes of the program’s control flow graph. The main objective is to cover a subset of statements that guarantees the coverage of all statements of the tested program. The set leaves nodes of the dominance tree are very important as every set of path that covers it also covers all nodes in tree. The set of leaves node of dominance tree shown in figure 5.3 is \( A = \{ 4, 5, 6, 7 \} \). The dominance path of each element of set A is as follow

- \( \text{dom} (4) = 1,2,3,4 \)
- \( \text{dom} (5) = 1,2,3,5 \)
- \( \text{dom} (6) = 1,2,3,6 \)
- \( \text{dom} (7) = 1,2,7 \)

Covering an element from the set A means covering its dominance path. Covering all elements of set of A means covering all dominance paths which results in covering all nodes of CFG that results in the coverage of all statements

**5.2.2 Proposed Fitness Function**

In dynamic analysis, a program is actually executed with values of the inputs and then fitness function determines the extent up to which it has satisfied the testing criterion, which becomes the fitness of the individual/particle. The search algorithm uses a new evaluation (fitness) function to evaluate the generated test data based on the concepts of the dominance relations between nodes of the program’s control flow graph. The fitness function used to evaluate each test case by executing the program
with it as input and recording the traversed nodes in the program that are covered by this test case. The set of traversed nodes is denoted by exePath. Also, it finds the dominance path dom(n) of the target node n. The fitness function is the ratio of the number of covered nodes of the dominance path of the target node to the total number of nodes of the dominance path of the target node. The fitness value \( \text{fit}(v_i) \) for each chromosome/particle \( v_i \) \((i = 1, \ldots, \text{pop}_\text{size})\) is calculated as follows:

1. Find exePath: the set of the traversed nodes in the program that are covered by a test case.
2. Find dom(n): dominance path of the target node n (the set of dominator nodes from the entry of the dominator tree to n).
3. Determine \( \text{dom}(n) - \text{exePath} \): uncovered nodes of the dominance path (the difference between the dominance path and the traversed nodes).
4. Determine \( \text{dom}(n) - \text{exePath} \) : covered nodes of the dominance path (the complement set of the difference set between the dominance path and the traversed nodes).
5. Calculate \(|\text{dom}(n) - \text{exePath}'|\): number of covered nodes of the dominance path (cardinality of the complement set).
6. Calculate \( \text{dom}(n) \): number of nodes of the dominance path of the target node n (cardinality of the dominance set).

Then, \( \text{fit}(v_i) = \frac{|(\text{dom}(n) - \text{exePath})'|}{|\text{dom}(n)|} \)

The fitness value is the only feedback from the problem for the GA, PSO and other optimization problem. A test case that is represented by the chromosome/individual/particle \( v_i \) is optimal if its fitness value \( \text{fit}(v_i) = 1 \)

### 5.2.3 Proposed Metaheuristics

It has been commonly accepted that finding optimality to NP hard problems is not a viable option since large amount of computational time is needed for judgment of such solutions. In reality, a good initial solution can be obtained by a heuristic/metaheuristic in a reasonable computational time with less number of test cases.
5.2.3.1 Genetic-Particle Swarm Combined Algorithm (GPSCA)

The proposed GPSCA combines the power of both GA with PSO to form a hybrid algorithm. The combination of GA and PSO always performs better than GA or PSO alone. The proposed GPSCA consists of three major operators: Enhancement, Crossover and Mutation which is shown in figure 5.6.

**Enhancement:** In each generation, after the fitness values of all the individuals in the same population are calculated, the top-half best-performing ones are marked. These individuals are regarded as elites. Instead of reproducing the elites directly to the next generation as elite GAs do, first enhance the elites. The enhancement operation tries to mimic the maturing phenomenon in nature, where individuals will become more suitable to the environment after acquiring knowledge from the society. Furthermore, by using these enhanced elites as parents, the generated off-springs will achieve better performance than those bred by original elites. PSO is used to enhance individuals of the same generation. Here, the group constituted by the elites may be regarded as a swarm, and each elite corresponds to a particle in it. In PSO, individuals of the same generation enhance themselves based on their own private cognition and social interactions with each other. In GPSCA, adopt and regard this technique as the maturing phenomenon. Based on PSO, equation 1 may be applied to the elites. These enhanced individual by PSO contributes towards half population of next generation and other half comes through by further enhancement by GA. The enhanced members

---

**Figure 5.6 Proposed GPSCA Technique**
of PSO are fed to GAs for further enhancement. GAs use roulette wheel selection method for selection of individual that is provided by PSO. These selected members go under crossover and mutation to produce more enhanced individual that contributes towards other half population of next generation. Again this process is repeated till satisfied the objective.

**Crossover:** To produce well-performing individuals, in the crossover operation parents are selected from the enhanced elites only. To select parents for the crossover operation, the roulette wheel selection scheme is used. Two off-springs are created by performing crossover on the selected parents. In this study, we used single point crossover. Crossover probability is represented by $P_c$.

**Mutation:** Mutation is an operator whereby the allele of a gene is altered randomly so that new genetic materials can be introduced into the population. Mutation probability is represented by $P_m$. 