**Introduction**

Existing tools and approaches for detecting the code similarity are based on parsing the program and tokenizing them into strings. These tools fail to detect the similar code when the suspect program involves changing the inconsequential details that do not alter the meaning of the original program.

We propose an approach which represents both original and suspect program as a graph called Program Dependence graph and uses Graph Aligner (GRAAL): network alignment algorithm to find the duplicate code.

**Program Dependence Graph**

- Each node in the graph represents a program statement.
- Every node is assigned a label and sub-labels that best describes the statement

Labels: DECL, ASSIGN, CTRL, CALL, RETURN

Sub-Labels: LT, GT, LEQ, GEQ, EQ, INEQ, MOD, AND, OR, INC, DEC, PRINT, ADD, SUB, MUL

**Goals**

- Develop an efficient tool to identify code clones
- Translate Orbit Counting algorithm from C++ to Java.
- Modify and implement Graph Aligner algorithm for aligning Program Dependence graphs.

**System Design**

Both original program and suspect program are converted to Program Dependence graph (PDG) using ANTLR4 java parser.

**Code Similarity Detection: Using Program Dependence Graphs**

Kanth Kumar Dayanand (kkd6428@rit.edu) | Advisor: Dr. Carlos R. Rivero

B. Thomas Golisano College of Computing & Information Sciences

- Original program and suspect program are converted to Program Dependence Graph (PDG) using ANTLR4 java parser.
- For all 2-5 node Graphlets, the algorithm computes a signature vector for each node in the program dependence graph.
- Choose a pair of nodes with minimal cost as an initial seed.
- Builds a sphere of all possible radii around the seed and start greedily aligning the pair of nodes that are not aligned.

**Results**

- Edge correctness percentage is better than similarity percentage reported by JPlag for programs that involves changing only programming constructs/expressions.
- Application is tested with more than 250 programs generated by source code and bytecode obfuscation.

**Future Work**

- PDGs needs to be improved to handle more programming constructs.
- Application currently works only on Java programs, can be extended to other languages.
- PDGs needs to be updated to connect the vertices that represents function call in the caller and first statement in callee function.

**Modified Cost Function**

```
private static int partition(Comparable[] a, int lo, int hi) {  
    int i = lo; // Partition point
    for (int j = lo; j < hi; j++) {  
        if (a[j].compareTo(a[hi]) < 0) {  
            swap(a, i, j);  
            i++;  
        }  
    }  
    return i;  
}
```

```
private static void swap(Comparable[] a, int i, int j) {  
    Comparable tmp = a[i];  
    a[i] = a[j];  
    a[j] = tmp;  
}
```

**Example Alignment**

```
public static float value(String s) {  
    return Float.parseFloat(s);  
}
```

```
public static int value(String s) {  
    return (int) Float.valueOf(s);  
}
```

**Graph Aligner (GRAAL) Algorithm**

The cost of aligning every node in original graph with each node in suspect graph is computed using their signature similarity.

Choose a pair of nodes with minimal cost as an initial seed.

Builds a sphere of all possible radii around the seed and start greedily aligning the pair of nodes that are not aligned.

**Introduction**

```
public static float value(String s) {  
    return Float.parseFloat(s);  
}
```

```
public static int value(String s) {  
    return (int) Float.valueOf(s);  
}
```

**Results**

- Edge correctness percentage is better than similarity percentage reported by JPlag for programs that involves changing only programming constructs/expressions.
- Application is tested with more than 250 programs generated by source code and bytecode obfuscation.

**Future Work**

- PDGs needs to be improved to handle more programming constructs.
- Application currently works only on Java programs, can be extended to other languages.
- PDGs needs to be updated to connect the vertices that represents function call in the caller and first statement in callee function.