

#141

BICA AI
2020

Algorithm for Optimal and Complete Testing of Software and Hardware Data Security Tools.

Tatiana M. Kanner¹, Andrei M. Kanner^{*2}, Anna V. Epishkina³.

National Research Nuclear University MEPhI (Moscow Engineering Physics Institute),

¹sheikot@mail.ru ²kanner@mail.ru ³avepishkina@mephi.ru



Andrei M. Kanner

SUMMARY

- The existing approaches to testing software and hardware Data Security Tools (DST) or other tools don't ensure **completeness** and **optimality of testing**.
- We can ensure **completeness** of testing with mathematical models using the **automata theory**.
- We can ensure **both completeness and optimality of testing** using the **graph theory**.

INTRODUCTION

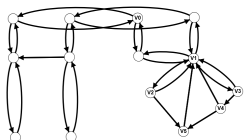
It's problematic to justify in practice that some tool has been tested:

- in all possible states, with all possible combinations of parameters and settings (**the completeness of testing**);
- in an optimal way with the minimum steps (**the optimality of testing**).

Some functionality remains untested under certain conditions, although it has taken a rather long time to conduct such a defective testing.

- Let's represent some DST as a directed graph:
- the vertices will be the states of the software or hardware components;
 - the directed edges will be the transitions of the tool when performing some functions.

Fig.1 Initial graph for some DST



Then the **testing problem** consists in traversing from some initial vertex over all the edges (with the min. number of steps) directed to the vertices, where any security function of the DST can be implemented. Hmm... Wait, it looks like some sort of the well-known **Route Inspection Problem!**

METHODS

First let's simplify our graph, because we only need to test the security functions of the DST. The complete algorithm constructing the corresponding graph by removing some vertices and edges not used in solving the testing problem is given in [1]. We will get the graph as in Fig. 2. If this graph has isolated vertices then it is obviously that the problem has no solution (there is some error in the DST).



Fig.2 The derived graph for some DST

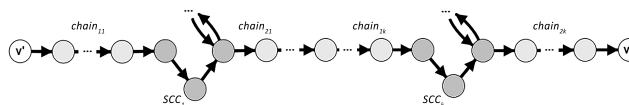


Fig.3 General form of the graph

The problem has no solution either if we can't use the algorithm for the Route Inspection Problem, so the general form of the graph should be as in Fig. 3. We can check it using the **Kosaraju-Sharir Algorithm** in two depth-first searches of the graph as in Fig. 4.

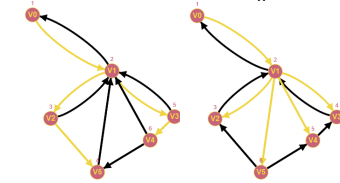


Fig.4 The result of DFS from v_0 for the derived graph and the inverted graph – there's only one SCC

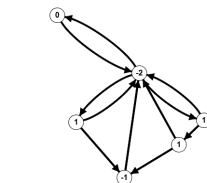


Fig.5 The corresponding graph with the difference of out- and in-degrees

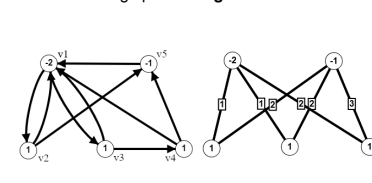


Fig.6 The digraph of unbalanced vertices and shortest paths from the upper to lower vertices

Then we should check if there are any unbalanced vertices in the graph, see Fig. 5. If there is no unbalanced vertices – then there is the **Eulerian cycle** in the graph. If we have only 2 unbalanced vertices with the difference of -1 and +1 – then there is the **Eulerian path** in the graph. In both cases we can skip and go to the **last step**.

Otherwise, let's consider unbalanced vertices separately as in Fig. 6. We can use the **Floyd-Warshall algorithm** to calculate the shortest paths between the negative-balanced and the positive-balanced vertices of the bigraph in Fig. 6.

Now using **Hungarian algorithm**, the **Kuhn-Munkres algorithm** or the **Ford-Fulkerson algorithm** we can find the combination of paths that can be reused, to become all the vertices balanced as in Fig. 7.

We can add this reused paths to the derived graph, which will become a multigraph with all the vertices balanced as in Fig. 8.

Now **the last step** is to use a cycle-based algorithm known as the **Hierholzer's algorithm** to construct the Eulerian cycle in the resulting graph.

[1] Kanner, A. M., Kanner, T. M.: *Testing Software and Hardware Data Security Tools Using the Automata Theory and the Graph Theory*. Proceedings of USBREIT, IEEE, pp. 615-618 (2020). doi:10.1109/usbreit48449.2020.9117757



Fig.7 A combination of reused paths for all vertices to become balanced

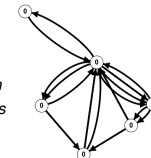


Fig.8 Multigraph with reused edges

RESULTS

Execution of the last step of the algorithm allows us to construct one of the optimal solution to the testing problem. The traversing of the graph as in Fig. 9 will guarantee the completeness and optimality of testing the DST. But keep in mind that there could be more than one optimal solution to the testing problem.

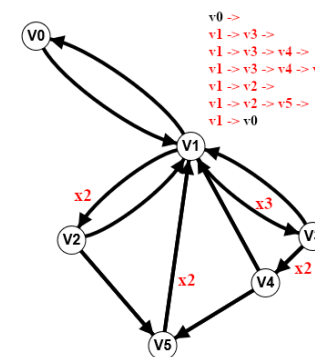


Fig.9: One of the optimal graph traversal, for edges labeled with the number of repeated passes virtual duplicate-edges were constructed

CONCLUSIONS

- The solution to the problem of ensuring completeness and optimality of testing exists if and only if the derived graph is in the form shown in Fig. 3.
- If the original graph satisfies this condition, a solution to the problem of ensuring completeness and optimality exists as well. Otherwise, the testing problem **has no solution**.
- All the steps of the proposed algorithm for solving the testing problem use only well-known algorithms on graphs, they are sequential without nesting. The algorithm's complexity is $O(|V|^3)$.