**Automatic test data generation for path testing**

**using adjustment**

**1. Tóm tắt nội dung bài báo**

**2. Tổng quan**

+ Kiểm thử phần mềm và sinh dữ liệu kiểm thử phần mềm

Kiểm thử phần mềm là quá trình thực hiện một chương trình hoặc ứng dụng nào đó với mục đích là tìm kiếm các lỗi phần mềm. Nó cũng có thể được coi như là một quá trình xác nhận và chứng minh rằng một chương trình phần mềm, ứng dụng hoặc sản phẩm đáp ứng đầy đủ các yêu cầu nghiệp vụ và kỹ thuật của thiết kế và phát triển.Kiểm thử phần mềm là một phần rất quan trọng nhưng tốn nhiều công sức trong chu trình phát triển phần mềm [1–3]. Do đó đã có nhiều nghiên cứu về automate testing, để giảm chi phí và đạt được sự tin tưởng hơn trong kết quả thu được [4].

Công việc quan trọng trong kiểm thử phần mềm là tạo ra input test data [3]. Search-based test data generation mục đích là để tự động hóa công việc này, bằng cách tìm kiếm các test case (là các input data, hoặc là cả input-output) mà thỏa mãn một tiêu chuẩn kiểm thử đã được lựa chọn.

+ Phương pháp kiểm thử phần mềm sử dụng các kỹ thuật metaheuristic (Search Based Software Testing)

+ Phương pháp sinh dữ liệu kiểm thử phần mềm sử dụng metaheuristic

**3. Bài toán sinh dữ liệu kiểm thử đường dẫn**

+ Bài toán sinh dữ liệu kiểm thử đường dẫn

Kiểm thử đường dẫn là tìm kiếm trong toàn bộ chương trình các test case phù hợp để có thể phủ được toàn bộ các đường dẫn khả thi trong chương trình được thực hiện kiểm thử (software under test (SUT)). Tuy nhiên nhìn chung là không thể thực hiện được mục tiêu này vì một số lý do. Đầu tiên là chương trình có thể có một số lượng vô hạn các đường dẫn nếu trong chương trình có các vòng lặp. Thứ hai, số lượng đường dẫn trong một chương trình là hàm mũ của số lượng các câu lệnh rẽ nhánh và nhiều đường dẫn trong số đó là không thể thực thi được. Thứ ba, số lượng test case sẽ là rất lớn vì mỗi đường dẫn có thể được phủ được bởi vài test case. Vì những lý do này, bài toán sinh dữ liệu kiểm thử đường dẫn trở thành bài toán NP-khó để có thể tạo test data phủ được toàn bộ các đường dẫn.

Since it is impossible to cover all paths in a software, the problem of path testing

selects a subset of paths to execute and find test data to cover it. Various test

data generation methods have been proposed in the literature; they can be classified in

[Hajnal and Forg´acs 1998]: random test data generators, symbolic evaluators and function

minimization methods. Random test data generators select test data randomly from

the input variables domain. The symbolic execution methods are static, in the sense that

they analyse a program to obtain a set of symbolic representations of each condition predicate

along a selected path. The expressions are obtained by attributing symbolic values

to the input variables. If the predicates are linear, then the solution can be obtained by

using linear programming [McMinn 2004]. Function minimization methods, on the other

hand, are dynamic, since they are based on program execution. They perform an exploratory

search [Korel 1990], in which the selected input variables are modified by an

small amount and submitted to the program.

Vì không thể phủ toàn bộ các đường dẫn trong một phần mềm, thế nên bài toán sinh dữ liệu kiểm thử đường dẫn sẽ chọn một tập các đường dẫn để thực thi và tìm các test data để phủ chúng. Có nhiều phương pháp sinh test data được đề xuất cho đến thời điểm hiện tại, và có thể được phân chia như là: sinh test data ngẫu nhiên, symbolic execution hay là các phương pháp tối thiểu hóa hàm. Sinh data ngẫu nhiên là lựa chọn các test data ngẫu nhiên từ tập các biến input. Symbolic execution là phương pháp phân tích tĩnh, từ mỗi đường dẫn của chương trình sẽ có được một tập các biểu thức logic biểu diễn cho các điều kiện vị từ của đường dẫn đó, rồi sau đó dùng các constraint solver để tìm được nghiệm cho các biểu thức logic này, cũng chính là test data để phủ đường dẫn này.

+ Giả thuyết về hạn chế của phương pháp sinh dữ liệu kiểm thử đường dẫn sử dụng giải thuật di truyền. Giả thuyết này sẽ được kiểm chứng bằng thực nghiệm.

Mô tả sơ lược giải thuật di truyền ở đây.

Với bài toán phân loại tam giác thì cũng chưa phủ được các đường dẫn nào.

+ Giả thuyết về hiệu quả của việc kết hợp giải thuật di truyền (GA) với phân tích tĩnh. Giả thuyết này cũng sẽ được kiểm chứng bằng thực nghiệm.

**4. Thiết kế các giải thuật**

GA và GA + Static Analysis cho bài toán sinh dữ liệu kiểm thử đường dẫn.

**5. Thí nghiệm và phân tích kết quả**

+ Các test programs + target paths

+ Các thông số cần phân tích: Em cần chạy mỗi test program với 2 giải thuật: GA (Không điều chỉnh), GA+ static Analysis (có điều chỉnh). Chạy 30 lần cho mỗi cặp giải thuật + test program. Ghi lại số thế hệ và lần gọi objective function của một đường dẫn mục tiêu khi nó được thực thi.

+ Phân tích kết quả: Phân tích chỉ ra GA không có khả năng phủ được các đường dẫn khó (có điều kiện bằng), trong khi GA + Static Analysis có thể. So sánh 2 giải thuật về số lượng đường dẫn trung bình phủ được trong 30 lần với mỗi test program. Phân tích chi tiết về các đường dẫn phủ được và không phủ được của cả 2 phương pháp, và thời điểm mà chúng được phủ...

**6. Kết luận và hướng phát triển**

**7. Tài liệu tham khảo**

[1] J.A. Whittaker, What is software testing? and why is it so hard?, IEEE Software 17 (1), 02/2000, 70–79.

[2] N. Mansour, M. Salame, Data generation for path testing, Software Quality Control 12(2), 06/2004, 121–136.

[3] P. McMinn, Search-based software test data generation: a survey, Software Testing, Verification & Reliability14 (2), 06/2004, 105–156.

[4] P.M.S. Bueno, M. Jino, Identification of potentially infeasible program paths by　monitoring the search for test data, Proceedings of the 15th IEEE　International Conference on Automated Software Engineering 2000 (ASE2000), IEEE Computer Society, Grenoble, France, 09/2000, 209–218.

[x] I. Hermadi, C. Lokan, R. Sarker, Dynamic stopping criteria for search-based test data generation for path testing, Information and Software Technology, 56 (4), 04/2014,395–407.