**Multi-Objective Particle Swarm Optimization-based**

**Test Data Generation for Path Coverage Testing**

**Abstract**

# 1. Introduction

Xem xét một program under test như sau:

int getDayNum(int year, int month)

{

int maxDay=0;

if( month>=1 && month<=12) //bch1: branch 1

{

if(month==2) //bch2: branch 2

{

if(year%400==0||(year%4==0&&year%100==0))

//bch3: branch 3

maxDay=29;

else //bch4: branch 4

maxDay=28;

}

else if(month==4||month==6||month==9||month==11)

//bch5: branch 5

maxDay=30;

else //bch6: branch 6

maxDay=31;

}

else //bch7: branch 7

maxDay=-1;

return maxDay;

}

Với program under test này, Mao [1] đã sử dụng giải thuật PSO với duy nhất 1 hàm fitness function là sự kết hợp của branch functions for branch predicates và branch weight. Phương pháp này đã chứng minh được rằng giải thuật PSO có hiệu quả hơn giải thuật GA trong việc sinh test data, tuy nhiên vẫn có nhược điểm là việc tính toán các branch weight cho một program under test vẫn hoàn toàn là công việc manual.

Cách tiếp cận của chúng tôi là, với mỗi branch predicate, thì tạo ra một fitness function. Sau đó chúng tôi sẽ sử dụng multi-objective particle swarm optimization để tìm nghiệm thỏa mãn cho mỗi fitness function này.

# 2. Background

## 2.1. Fitness function

Table 1. The branch functions for several kinds of branch predicates

|  |  |  |
| --- | --- | --- |
| No | Predicate | Branch distance function *f*(bch*i*) |
| 1 | Boolean | If *true* then 0 else *k* |
| 2 | ¬a | Negation is propagated over *a* |
| 3 | a = b | If abs*(a* − *b)* = 0 then 0 else abs*(a* − *b)* + *k* |
| 4 | a ≠ b | If abs*(a* − *b)* ≠ 0 then 0 else *k* |
| 5 | a < b | If *a* − *b <* 0 then 0 else abs*(a* − *b)* + *k* |
| 6 | a ≤ b | If *a* − *b* ≤ 0 then 0 else abs*(a* − *b)* + *k* |
| 7 | a > b | If *b* − *a <* 0 then 0 else abs*(b* − *a)* + *k* |
| 8 | a ≥ b | If *b* − *a* ≥ 0 then 0 else abs*(b* − *a)* + *k* |
| 9 | a and b | *f (a)* + *f (b)* |
| 10 | a or b | min*( f (a), f (b))* |

Để thuận lợi cho việc trình bày thì có thể gộp các predicate từ 3 đến 8 ở Table 1 thành 1 function được viết như sau:

|  |
| --- |
| **Algorithm 1** Branch distance function (fBchDist) |
| **Input:** double a, operator type, double b |
| **Output:** Branch distance value |
| switch (operator type) {  case “==”: |
| if abs(*a* − *b*) = 0 then retrun 0 else return (abs(*a* − *b*) + *k*) |
| case “!=”: |
| if abs*(a* − *b)* ≠ 0 then return 0 else return *k* |
| case “<”: |
| if *a* − *b <* 0 then return 0 else return (abs*(a* − *b)* + *k*) |
| case “≤”: |
| if *a* − *b* ≤ 0 then return 0 else return (abs*(a* − *b)* + *k*) |
| case “>”: |
| if *b* − *a >* 0 then return 0 else return (abs*(b* − *a)* + *k*) |
| case “≥”: |
| if *b* − *a* ≥ 0 then return 0 else return (abs*(b* − *a)* + *k*) |
| } |

# 3. Related work

# 4. Proprosed approach

## 4.1. Thiết lập fitness function cho mỗi branch

Mỗi branch trong sample getDayNum() bên trên được gán cho mỗi fitness function như sau:

|  |  |  |
| --- | --- | --- |
| No | Branch | Fitness function |
| 1 | bch1 | fBchDist(month, >=, 1) + fBchDist(month, <=, 12) (1) |
| 2 | bch2 | (1) + fBchDist(month, ==, 2) (2) |
| 3 | bch3 | (2) + min(fBchDist(year%400, ==, 0),  fBchDist(year%4, ==, 0) + fBchDist(year%100, ==, 0)) |
| 4 | bch4 | (2) + fBchDist(year%400, !=, 0) +  Min(fBchDist(year%4, !=, 0), fBchDist(year%100, !=, 0)) |
| 5 | bch5 | (1) + min(fBchDist(month, ==, 4), fBchDist(month, ==, 6),  fBchDist(month, ==, 9), fBchDist(month, ==, 11)) |
| 6 | bch6 | (1) + fBchDist(month, !=, 4) + fBchDist(month, !=, 6)  + fBchDist(month, !=, 9) + fBchDist(month, !=, 11)) |
| 7 | bch7 | min(fBchDist(month, <, 1), fBchDist(month, >, 12)) |

## 4.2. Multi-Objective Particle Swarm Optimization (MOPSO)

Sử dụng Multi-Objective trong Particle Swarm Optimization, với mỗi Objective là một fitness function cho mỗi predicate branch của program under test.

# 5. Experimental result

So sánh với kết quả của Mao[1] theo các benchmark và tiêu chí mà Mao [1] đã sử dụng. Hai tiêu chí được mang ra so sánh là:

* Success rate (SR) is the probability of all branches which can be covered by the generated test data.
* Average time (AT) is the average execution time (ms) for realizing all-branch coverage.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Program under test | Success rate (%) | | Average time (ms) | |
| Mao[10]’s PSO | MOPSO | Mao[10]’s PSO | MOPSO |
| triangleType | 99.80 | 100.0 | 0.19 |  |
| calDay | 100.0 | 100.0 | 0.35 |  |
| cal | 100.0 | 100.0 | 0.50 |  |
| remainder | 100.0 | 100.0 | 0.17 |  |
| computeTax | 99.80 | 100.0 | 0.25 |  |
| bessj | 100.0 | 100.0 | 0.66 |  |
| printCalendar | 99.10 | 100.0 | 1.41 |  |
| line | 99.20 | 100.0 | 2.69 |  |

# 6. Conclusion

**References**

1. C. Mao: Generating Test Data for Software Structural Testing Based on Particle Swarm Optimization. Arabian Journal for Science and Engineering, vol 39, issue 6, pp 4593–4607 (June 2014).