테스트 데이터 자동 생성을 위한 적합도 평가 방법의 효율성 향상 기법

(An Improved Technique of Fitness Evaluation for Automated Test Data Generation)

이 선 열 [†] 최 현 재 [†] 정 연 지 [†]
(Sun-Yul Lee) (Hyun Jae Choi) (Yeon Ji Jeong)

배 정 호 [†] 김 태 호 [††] 채 흥 석 [†††]
(Jung Ho Bae) (Taeho Kim) (Heung-Suk Chae)

요 약 테스트 데이터를 자동으로 생성하기 위한 동적 테스트 데이터 생성에 관한 많은 연구가 이루어졌다. 동적 테스트 데이터 생성 방법은 가공 테스트 대상 프로그램(SUT; Software Under Test)을 실행시켜 기존의 테스트 데이터의 적합도를 평가하고, 평가된 적합도 값과 최적화 알고리즘을 이용하여 새로운 테스트 데이터를 생성하는 방법이다. 최근에 전역 최적화 알고리즘을 이용한 동적 테스트 데이터 생성에 관한 많은 연구가 이루어져 왔고, 이 알고리즘을 통해서 테스트 대상 프로그램(SUT)의 커버리지를 높일 수 있는 데이터를 생성할 수 있다는 것이 실험적으로 밝혀졌다. 그러나 최적화 알고리즘은 오랜 연산 시간이 필요하기 때문에, 이를 이용한 방법은 테스트 데이터를 생성하기 위해 많은 시간이 걸린다는 단점이 있다.

본 논문에서는 최적화 알고리즘을 이용한 동적 테스트 데이터 생성의 시간을 줄이기 위하여, 최적화 알고리즘의 절차 중 적합도 평가 시간을 줄이는 방법을 제안한다. 이를 위하여 SUT의 테스트 목표 경로로부터 생성된 적합도 평가 프로그램(FEP)을 정의하고, 가공 SUT 실행하는 대신 소개된 FEP를 이용한 적합도 평가 방법을 제안하고 'ConGA' 라는 도구를 구현한다. 그리고 C언어로 작성된 프로그램을 'ConGA'를 이용하여, 테스트 데이터 생성 효율성을 확인하였다. 이 실험을 통하여 제안된 방법이 기존의 방법보다 테스트 데이터 생성에 걸린 시간을 평균적으로 약 20% 줄인 것을 확인할 수 있었다.

키워드 : 동적 테스트 데이터 생성, 최적화 알고리즘, 적합도 평가 프로그램

Abstract Many automated dynamic test data generation technique have been proposed. The techniques evaluate fitness of test data through executing instrumented Software Under Test (SUT) and then generate new test data based on evaluated fitness values and optimization algorithms. Previous researches and experiments have been showed that these techniques generate effective test data. However, optimization algorithms in these techniques incur much time to generate test data, which results in huge test case generation cost.

In this paper, we propose a technique for reducing the time of evaluating a fitness of test data among steps of dynamic test data generation methods. We introduce the concept of Fitness Evaluation Program (FEP), derived from a path constraint of SUT. We suggest a test data generation method

· 본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다.

† 학생회원 : 부산대학교 컴퓨터공학과
sun302412@pusan.ac.kr
gonoki@pusan.ac.kr
sin24@pusan.ac.kr
jhbae83@pusan.ac.kr
†† 정 회 원 : 한국전자통신연구원 임베디드SW플랫폼연구팀 선임연구원
taehokim@etri.re.kr
††† 정 회 원 : 부산대학교 컴퓨터공학과 교수
hschae@pusan.ac.kr

논문접수 : 2010년 5월 13일
심사완료 : 2010년 10월 25일

정보과학회논문지: 소프트웨어 및 응용 제37권 제12호(2010.12)

based on FEP and implement a test generation tool, named *ConGA*. We also apply *ConGA* to generate test cases for C programs, and evaluate efficiency of the FEP-based test case generation technique. The experiments show that the proposed technique reduces 20% of test data generation time on average

Key words : Dynamic test data generation, optimization algorithms, fitness evaluation

1. 서 론

테스트 데이터 생성은 전체 소프트웨어 테스팅 비용의 약 40%를 차지하는, 많은 비용이 들어가는 작업이다[1,2]. 만약 테스트 데이터 생성 작업을 자동화할 수 있다면, 소프트웨어 테스팅 비용을 획기적으로 절감할 수 있을 것이며, 이를 위한 여러 테스트 데이터 자동 생성 기법들이 제안되어 왔다[3-5].

테스트 데이터를 자동으로 생성하기 위해서 동적 테스트 데이터 생성 방법[3]이 제안되었다. 이 방법은 테스트 대상 프로그램과 최적화 알고리즘을 이용하여 테스트 데이터를 자동으로 생성하는 방법이다. 랜덤이나 언덕 오르기(hill climbing)등의 알고리즘이 최적화 알고리즘으로 사용되었다.

최근 10년 동안, 유전알고리즘[2,4,6,7]과 담금질 기법[5,8] 등의 전역 최적화 알고리즘을 이용한 동적 테스트 데이터 생성 방법에 관한 많은 연구가 이루어져 왔다. 연구 결과를 통해서 전역 최적화 알고리즘을 이용한 방법이 다른 알고리즘을 이용한 방법 보다 SUT의 커버리지가 높은 테스트 데이터를 생성하는 것을 확인할 수 있었다[2,5-7].

그러나 전역 최적화 알고리즘을 적용한 방법은 알고리즘의 오랜 연산 시간으로 인해 테스트 데이터 생성에 많은 시간이 필요하다는 단점이 있다. 이 방법의 테스트 데이터 생성 시간을 줄이기 위하여, Alba and Chicano [7]는 병렬 진화알고리즘(parallel GA)을 테스트 데이터 생성에 적용하였고, Wang[9]은 유전 알고리즘과 지역 최적화 알고리즘을 혼합한 미미틱(memetic) 알고리즘을 적용한 방법을 제시였다. 그리고 Watkins and Hufnagel[10]은 테스트 데이터 생성에 사용된 적합도 평가 함수를 비교하여, 테스트 데이터 생성에 가장 효율적인 적합도 평가함수를 사용할 것을 주장하였다.

본 논문에서는 테스트 데이터 생성의 시간을 줄이기 위하여 전역 최적화 알고리즘의 적합도 평가 시간을 줄이는 방법을 제안한다. 이를 위해, 우리는 적합도 평가 함수(FEP)의 개념을 소개한다. FEP는 테스트 데이터의 적합도를 효율적으로 평가하기 위해 테스트 대상 프로그램(SUT)을 추상화한 프로그램이다. 적합도 평가를 위해 SUT를 실행하는 대신 FEP를 실행함으로써, 테스트 데이터를 생성하는 시간을 줄일 수 있다.

제안된 방법의 효율성을 확인하기 위해, 우리는 전역 최적화 알고리즘으로 널리 사용되고 있는 유전알고리즘[11]을 이용한 테스트 데이터 생성 도구인 'ConGA'를 구현하였다. 'ConGA'는 주어진 SUT로부터 자동으로 FEP를 생성하고, 유전알고리즘과 생성된 FEP를 이용하여 테스트 데이터를 자동으로 생성하는 도구이다. 우리는 몇 가지 C언어 프로그램에 'ConGA'를 적용하여 테스트 데이터 생성하는 실험을 수행하였다. 실험을 통하여 제안된 방법이 기존의 방법보다 약 20%의 테스트 데이터 생성 시간을 줄이는 것을 확인하였다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구의 배경으로서, 동적 테스트 데이터 생성과 유전알고리즘에 대하여 소개한다. 3장에서는 생성된 테스트 데이터의 적합도를 평가하는 적합도 평가 프로그램의 생성 방법에 대해서 소개한다. 4장에서는 테스트 데이터 생성 도구 및 구현된 도구를 이용하여 진행된 실험과 그 결과를 소개한다. 5장에서는 유전알고리즘을 이용한 테스트 데이터 생성에 관한 관련 연구를 소개한다. 6장에서는 결론과 향후 연구 방향을 기술한다.

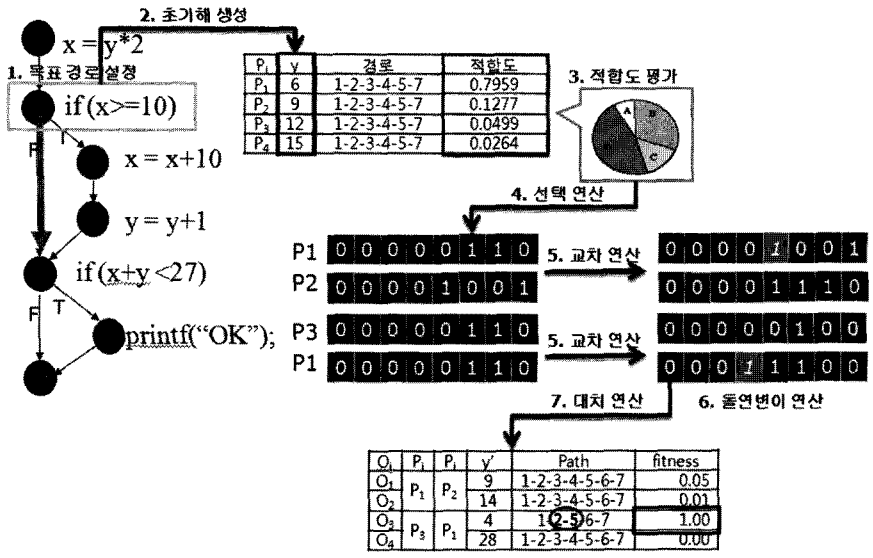
2. 배경 연구

2.1 유전알고리즘을 이용한 동적 테스트 데이터 생성 방법

기존의 동적 테스트 데이터 생성 방법은 테스트 목표 경로를 테스트할 수 있는 테스트 데이터를 생성하기 위해 SUT를 이용하는 방법이다. 만약 목표 테스트 경로를 만족하는 테스트 데이터를 생성하지 못할 경우, SUT 실행 동안 수집된 데이터와 최적화 알고리즘을 이용하여 목표 테스트 경로와 가장 가까운 데이터를 결정하는데 사용될 수 있다. 이러한 피드백을 통하여 원하는 목표 경로를 만족할 때까지 테스트 데이터는 점차적으로 수정된다. 테스트 데이터 생성에 이용되었던 최적화 알고리즘으로는 유전알고리즘, 언덕 오르기, gradient descent, 담금질 기법(SA), 개미 집단 최적화 알고리즘 등이 있다[6,12]. 그림 1은 유전알고리즘을 이용한 동적 테스트 데이터 생성 방법을 간략하게 소개한다.

그림 1의 절차를 간략하게 소개하면 아래와 같다.

1) 목표 경로 설정: SUT에서 테스트 목표 경로를 결정한다. 그림 1에서는 SUT의 if(x >= 10)을 만족하지 않는 경로를 테스트 목표 경로로 설정하였다.



P_i	y	경로	적합도
P_1	6	1-2-3-4-5-7	0.7959
P_2	9	1-2-3-4-5-7	0.1277
P_3	12	1-2-3-4-5-7	0.0499
P_4	15	1-2-3-4-5-7	0.0264

O_i	P_i	P_i	y'	Path	fitness
O_1	P_1	P_2	9	1-2-3-4-5-6-7	0.05
O_2			14	1-2-3-4-5-6-7	0.01
O_3	P_3	P_1	4	1-2-3-4-5-6-7	1.00
O_4			28	1-2-3-4-5-6-7	0.00

그림 1 테스트 데이터 생성 절차

2) 초기해 생성: 목표 경로가 설정되면, 초기 해를 임의로 생성한다. 생성된 임의의 해를 점차적으로 수정하여 목표 경로를 만족하는 테스트 데이터를 생성하게 된다. 그림 1의 예제에서는 6, 9, 12, 15의 테스트 데이터가 임의로 생성되었다.

3) 적합도 평가: 생성된 해가 테스트 목표 경로에 적합한지를 판단하는 과정이다. 적합도 평가 방법에 대해서는 2.2절에서 설명한다. 그림 1의 예제에서는 적합도가 1에 가까우면 가까울수록 목표 경로에 더 적합한 테스트 데이터를 의미하며, 테스트 데이터의 적합도가 1이 되면 그 테스트 데이터는 테스트 목표 경로를 테스트할 수 있는 테스트 데이터가 된다.

4) 선택(selection) 연산: 테스트 목표 경로에 더 적합한 테스트 데이터를 생성하기 위해 현재의 테스트 데이터를 선택하는 연산이다. 유전알고리즘의 대표적인 선택 연산자는 룰렛 선택법(roulette wheel selection)이다.

5) 교차(crossover) 연산: 유전알고리즘의 대표적인 연산으로, 선택된 2개의 테스트 데이터 일부분을 교환하여 새로운 테스트 데이터를 생성하는 연산이다. 1점 교차(one point crossover)가 대표적으로 사용된다.

6) 돌연변이(mutation) 연산: 더 넓은 해공간을 탐색하기 위해 테스트 데이터의 일부분을 변경하는 연산이다. 균등 돌연변이(uniform mutation) 연산자 등이 이용된다.

7) 대치(substitution) 연산: 새로 생성된 테스트 데이터를 기존의 테스트 데이터와 교체하여 새로운 테스트 데이터를 형성하는 연산이다. 새롭게 형성된 테스트 데이터의 적합도를 확인하여 테스트 목표 경로를 만족하는 테스트 데이터가 생성된 것을 확인할 수 있다. 원하는 테스트 데이터가 생성되었다면, 유전알고리즘의 수행을 멈추고 다른 목표 경로를 만족하는 테스트 데이터를 생성하게 된다. 원하는 테스트 데이터가 생성되지 않았다면, 선택 연산부터 반복적으로 유전알고리즘을 수행하게 된다.

2.2 테스트 데이터 적합도 평가

테스트 데이터의 적합도는 테스트 데이터가 테스트 목표 경로를 테스트하기에 적합한지를 나타내는 척도이다. 유전 알고리즘 및 담금질 알고리즘 등의 최적화 알고리즘은 이 적합도를 근거로 하여 다음 세대의 테스트 데이터를 생성한다.

테스트 데이터의 적합도는 다양하게 평가될 수 있는데, 한가지 예를 들면 다음과 같다. SUT에 if (pos >= 21)와 같은 분기문이 있다고 가정하자. 이 분기문의 참 경로를 테스트하고 싶다면, 프로그램이 실행되어 이 분기문에 도달했을 때 pos 값이 21보다 커야 한다. 분기문에 도착한 순간의 pos 값을 pos(x)라는 함수로 정의하고, 이를 이용하여 아래와 같은 테스트 데이터의 적합도 평가 함수를 정의할 수 있다.

$$\text{fitness}(x) = \begin{cases} 21 - \text{pos}(x), & \text{if pos}(x) < 21; \\ 0, & \text{otherwise} \end{cases}$$

이 분기의 참 경로를 테스트하기 위해서는 함수 fitness(x)값이 최소값 0이 되어야 한다. 그러므로 우리가 원하는 테스트 데이터를 찾기 위해서, 우리는 fitness 함수를 최소화시키는 값을 찾아야 한다. 결국 함수 fitness(x)의

표 1 적합도 평가 함수 생성 규칙

범주	연산자	생성코드
대소 연산자	a > b	if(a > b) f_n = 0; else f_n = (b + 1) − a;
	a >= b	if(a >= b) f_n = 0; else f_n = b − a;
	a < b	if(a < b) f_n = 0; else f_n = (a + 1) − b;
	a <= b	if(a < b) f_n = 0; else f_n = a − b;
비교 연산자	a == b	f_n = abs(b − a);
	a != b	if(a != b) f_n = 0; else f_n = K
논리 연산자	a && b	f_n = a + b;
	a || b	f_n = min(a, b);
단항 연산자	a	if(a) f_n = 0; else f_n = K
	!a	if(!a) f_n = 0; else f_n = K

최소값 0에 가까우면 가까울수록 테스트 목표 경로를 테스트하기에 더 적합한 테스트 데이터가 되는 것이다.

이러한 적합도 평가 함수는 분기문에 사용된 연산자에 따라 생성 규칙을 가질 수 있다. 표 1은 여러 연구[3,6]에서 사용된 적합도 평가 함수 생성 규칙을 나타낸다. 만약 분기문에 2가지 이상의 연산자가 사용되었다면, 규칙은 연산자 우선 순위에 따라 적용된다.

표 1의 규칙 이외에도, 다양한 방법으로 테스트 데이터의 적합도가 평가될 수 있다. 그리고 그 방법간의 효율성에 대한 연구[10]도 이루어지고 있다.

2.3 SUT 가공 프로그램

2.2절에서 소개한 테스트 데이터 적합도 평가 방법를 이용하여 적합도를 평가하기 위해서는 적합도 평가 함수가 실행되는 시점이 중요하다. 동적 테스트 데이터 생성 방법에서는 적합도 평가 함수를 SUT 소스 코드에 삽입한 SUT 가공 프로그램을 생성하여 테스트 데이터의 적합도를 평가한다. SUT 가공 프로그램의 생성은 2.2절의 적합도 평가 함수를 SUT에 삽입하는 것이다. 그림 2는 SUT를 표 1의 규칙을 적용하여 SUT 가공 프로그램을 생성한 것을 개념적으로 나타낸다.

그림 2와 같이 SUT 가공 프로그램은 SUT에 비해 더 복잡한 프로그램이 된다. 동적 테스트 데이터 생성 방법에 의해 테스트 데이터에 대한 적합도 평가가 많이 이루어져야 하며, 이것은 SUT 가공 프로그램에 의해 수행되기 때문에 SUT 가공 프로그램의 복잡도는 테스트 데이터 생성 시간에 많은 영향을 주게 된다.

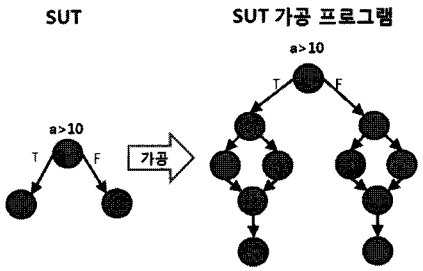


그림 2 SUT 가공 프로그램의 생성

3. 적합도 평가 프로그램을 이용한 테스트 데이터 자동 생성

본 논문에서는 테스트 데이터의 적합도를 평가하기 위해서 '적합도 평가 프로그램'을 이용한다. 적합도 평가 프로그램을 이용하는 목적은 적합도를 평가하기 위한 프로그램의 복잡도를 줄여 테스트 데이터 생성의 효율성을 얻기 위함이다. 그림 3은 기존 방법과 제안 방법의 적합도 평가에 이용하는 프로그램 생성에 관한 차이를 나타낸다.

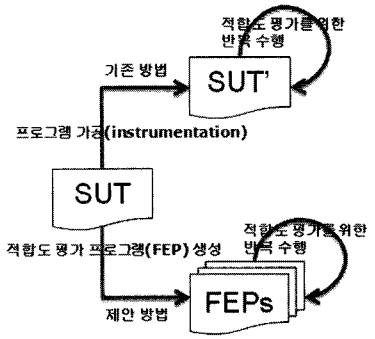


그림 3 기존 방법과 제안 방법의 적합도 평가 방법의 차이

그림 3에 보이는 것처럼 기존 방법은 SUT를 테스트 데이터의 적합도를 평가할 수 있도록 가공하고, 이를 반복 실행하여 테스트 데이터를 생성한다. 본 논문에서 제안하는 방법은 SUT의 테스트 목표 경로를 추상화하고 이를 이용하여 적합도 평가 프로그램을 생성한다.

본 절에서는 테스트 데이터의 적합도 평가 및 본 논문에서 제안하는 적합도 평가 프로그램의 생성에 대해서 알아보겠다.

3.1 적합도 평가 프로그램

적합도 평가 프로그램은 SUT의 테스트 목표 경로로부터 생성된다. 그림 4는 적합도 평가 프로그램을 설명하기 위한 CFG이다.

그림 4의 CFG는 6개의 경로를 가지고 있다. 그리고

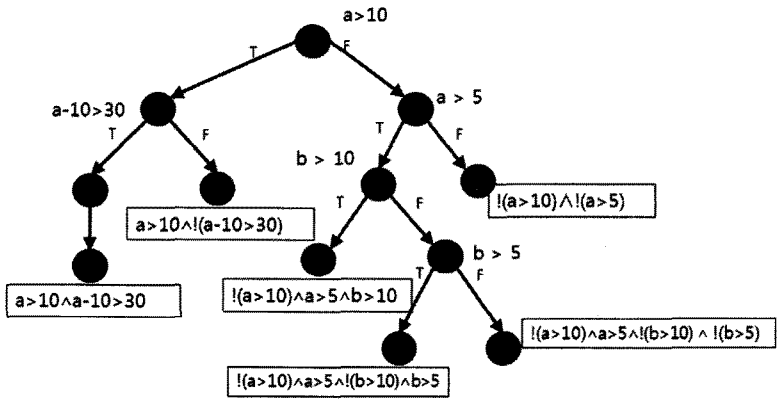


그림 4 SUT 예제 프로그램

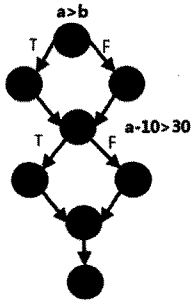


그림 5 적합도 평가 프로그램의 CFG

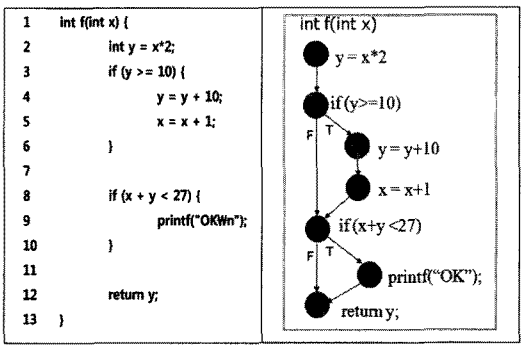


그림 6 예제 프로그램과 그 CFG

그림 4의 텍스트 박스는 각 경로의 제약 조건이며, 이는 경로 제약 조건은 테스트 목표 경로가 된다. 만약, 이를 이용하여 SUT 가공 프로그램을 생성한다면 상당히 복잡한 프로그램이 될 것이다. 이에 반해, 적합도 평가 프로그램은 전체 SUT가 아닌 개별적인 테스트 목표 경로로부터 생성되기 때문에 SUT 가공 프로그램보다 복잡하지 않은 프로그램이 될 수 있다. 그림 5는 그림 4의 첫 번째 경로 제약 조건 a>10∧a-10>30 으로부터 생성한 적합도 평가 프로그램의 CFG를 나타낸다.

본 논문에서는 그림 5와 같은 적합도 평가 프로그램을 이용하여 테스트 데이터의 적합도 평가방법을 제안한다. 그리고 4장에서 제안 방법과 기존 방법을 비교하는 실험을 수행하여 테스트 데이터 생성의 효율성을 비교한다.

3.2 적합도 평가 프로그램 생성 예제

동적 테스트 데이터 생성에 대한 많은 연구가 수행되었기 때문에 여러 가지 적합도 평가 방법이 존재한다. 본 논문에서는 2.2절에서 소개되었던 표 1의 적합도 평가 함수 생성 규칙을 이용한 적합도 평가 프로그램 생성에 대해 설명한다. 그림 6은 적합도 평가 프로그램 생성을 위한 예제 프로그램과 그 CFG이다.

표 2 그림 6 프로그램의 경로와 그 제약 조건

경로	기본 경로	경로 제약 조건
1	1-2-5-7	(2*x<10) && (x+2*x>=27)
2	1-2-5-6-7	(2*x<10) && (x+2*x<27)
3	1-2-3-4-5-6-7	(2*x>=10) && (x+1+2*x+10<27)
4	1-2-3-4-5-7	(2*x>=10) && (x+1+2*x+10>=27)

그리고 표 2는 그림 6 프로그램이 가지는 4가지 경로와 제약 조건을 나타낸다.

표 2에서 3번째 경로 2*x >= 10 && (x + 1 + 2*x + 10 < 27) 의 경로 제약에 대한 경로 적합도 평가 프로그램을 생성한다면 연산자 우선순위에 따라 아래와 같은 절차로 생성된다.

1) >= 연산에 대한 규칙 적용

연산자 우선 순위에 따라 x*2 >= 10 && (x + 1 + x*2 + 10 < 27) 중 x*2 >= 10이 적합도 평가 함수로 먼저 생성된다. >= 연산자의 규칙 if(a >= b) fn = 0; else fn = b - a;를 적용하면 그림 5와 같은 코드가 생성된다.

```
if((x*2) >= 10) fitness1 = 0;
else            fitness1 = 10 - (x*2);
```

그림 7 x*2 >= 10로부터 생성된 코드

2) < 연산에 대한 규칙 적용

두 번째 연산자 우선순위를 가지는 (x + 1 + x*2 + 10 < 27) 식에 대한 적합도 평가 함수 생성 규칙을 적용하면 그림 6과 같은 코드가 생성된다.

```
if((((x+1)+(x*2))+10) < 27) fitness2 = 0;
else fitness2 = ((((x+1)+(x*2))+10) + 1) - 27;
```

그림 8 (x + 1 + x*2 + 10 < 27)로부터 생성된 코드

3) && 연산 규칙 적용

경로 제약 x*2 >= 10 && (x + 1 + x*2 + 10 < 27) 중, 최하위 연산자 우선순위를 가지는 && 연산자의 피연산자는 x*2 >= 10와 (x + 1 + x*2 + 10 < 27)의 결과 값이다. x*2 >= 10의 결과 result1과 (x + 1 + x*2 + 10 < 27)의 결과 result2에 대해서 규칙 1을 적용하면 그림 7과 같은 코드가 생성된다.

```
fitness3 = fitness2 + fitness1;
```

그림 9 result1과 result2에 &&연산을 적용한 코드

위의 3단계 절차를 거쳐 생성된 코드는 그림 8과 같다.

```
double path_constraint(int x) {
        double fitness1;
        double fitness2;
        double fitness3;
        if((x*2) >= 10)     fitness1 = 0;
        else                fitness1 = 10 - (x*2);

        if((((x+1)+(x*2))+10) < 27)  fitness2 = 0;
        else                         fitness2 = ((((x+1)+(x*2))+10) + 1) - 27;

        fitness3 = fitness2 + fitness1;

        return fitness3;
}
```

그림 10 생성된 적합도 평가 프로그램

그림 8의 FEP는 목표 경로 x*2 >= 10 && (x + 1 + x*2 + 10 < 27)에 대해서 생성된 테스트 데이터의 적합도를 평가할 수 있다. FEP가 반환하는 값은 [0, ∞)의 값을 가질 수 있으며, 값이 작을수록 목표경로에 더 적합한 테스트 데이터임을 나타낸다. 예제 프로그램의 다른 경로에 대해서도 같은 방법으로 적합도 평가 프로그램을 생성할 수 있다.

4. 사례 연구

본 논문에서는 제안된 테스트 데이터 생성 방법과 기존의 테스트 데이터 생성 방법의 효율성을 비교하기 위해 아래의 2가지 측면에서 실험한다.

(1) 시간에 따라 SUT의 전체 경로의 커버리지 달성율 비교 실험: 빠른 시간에 더 높은 경로 커버리지를 달성하는 방법이 더 효율적인 테스트 데이터 생성 방법이다.

(2) 시간에 따른 유전알고리즘 반복 횟수를 비교 실험: 본 논문에서 제안한 적합도 평가 함수가 유전알고리즘의 효율성을 향상 시킬 수 있는지를 판단한다.

본 절에서는 위와 같은 실험을 수행하기 위해 구현한 테스트 데이터 생성 도구를 소개한다. 그리고 이를 이용하여 제안된 방법과 기존의 방법간의 실험 결과를 비교한다.

4.1 테스트 데이터 생성 도구

본 논문에서 제안한 방법의 테스트 데이터 생성의 효율성을 확인하기 위해서 'ConGA'라는 도구를 제작하였다. 그림 11은 'ConGA'의 논리적 아키텍처를 나타낸다.

'ConGA'는 원시 소스 코드를 분석하여 제어 흐름 그래프를 생성한다. 그리고 제어 흐름 그래프로부터 테스트 대상 프로그램의 경로들로 이루어진 경로 트리를 생성하고, 생성된 경로 트리에 심볼릭 실행을 적용하여 경로 제약 조건을 생성한다. 각각의 경로 제약 조건은 3장에서 소개한 방법을 이용하여 적합도 평가 프로그램을 생성하는데 사용된다.

테스트 데이터 생성기는 2장에서 소개된 절차에 의해서 테스트 데이터를 생성하게 된다. 임의로 초기 테스트 데이터를 생성하고, 이를 적합도 평가 프로그램을 이용

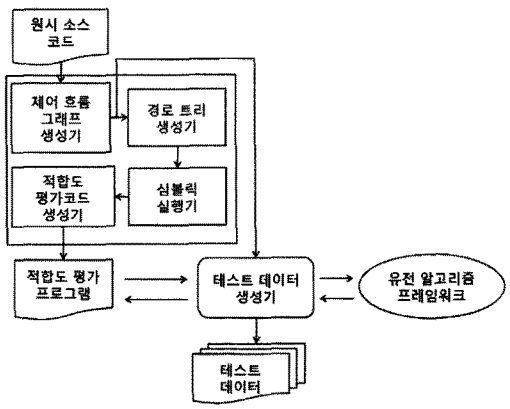


그림 11 'ConGA'의 논리적 아키텍처

하여 적합도 평가를 한다. 적합도 평가가 완료되면, 유전알고리즘의 반복을 통하여 테스트 목표 경로에 적합한 테스트 데이터를 생성하게 된다.

유전알고리즘에는 성능에 영향을 미칠 수 있는 매개변수와 유전연산자가 존재하기 때문에, 구현된 도구에서는 매개변수와 유전연산자를 유연하게 적용할 수 있도록 유전알고리즘 프레임워크를 구현하였다. 그리고 현재 구현된 유전알고리즘 프레임워크에서 선택 가능한 유전연산자와 매개변수는 표 3과 같다.

표 3 유전알고리즘 프레임워크에서 사용 가능한 유전알고리즘 매개변수

범주	선택 기능
초기 해 생성방법	임의 생성
종료 조건	유전알고리즘의 생성(generation) 횟수 테스트 목표를 만족하는 테스트 데이터 생성
해의 표현방법	실수 표현
부모 선택 연산자	토너먼트 선택, 확률 바퀴(roulette wheel) 선택
교차 연산자	일점 교차(one point crossover)
돌연변이 연산잔	동적 돌연변이, 가우시안 돌연변이

- 초기 해 생성방법: 초기의 테스트 데이터를 생성에 관련된 매개변수이다. 프레임워크에서 구현된 초기 해 생성 방법은 임의의 테스트 데이터를 생성이다.
- 종료 조건: 테스트 데이터 생성을 종료하는 조건을 나타내는 매개변수이다. 본 논문에서는 테스트 목표를 만족하는 테스트 데이터가 생성되거나, 목표를 만족하는 테스트 데이터를 생성하지 못한 채 지정된 횟수만큼 프로그램(FEP, SUT)이 수행되었을 때 테스트 데이터 생성을 종료한다.
- 해의 표현 방법: 유전알고리즘은 문제의 해를 구하기 위해서 생성된 각 개체를 특정한 형태로 변형하여 표현해야 한다. 구현된 프레임워크에서는 실수 표현법 또는 이진 표현법으로 테스트 데이터를 표현할 수 있다.
- 부모 선택 연산자: 다음 세대를 구성할 테스트 데이터를 선택하는 연산자이다. 토너먼트 선택 연산자와 확률 바퀴(roulette wheel) 선택 연산자가 구현되어 있다.
- 교차 연산자: 두 개체의 일부를 임의로 선택된 교차점을 중심으로 교환하는 연산자이다. 일점 교차(one point crossover)이 구현되어 있다.
- 돌연변이 연산자: 한 개체내의 임의의 유전자를 변경시켜서 새로운 개체를 만들어내는 연산자이다. 구현된 돌연변이 연산자는 동적 돌연변이 연산자이다.

4.2 실험 결과

SUT는 [13,14]에서 테스트 데이터 생성 실험에 사용되었던 triangle3.c와 triangle4.c 이며, 각각은 C언어로 구현되었다. triangle3.c는 30줄 정도의 소스 길이를 가지며 14개의 분기와 8개의 경로를 가진 소스 코드이다. triangle4.c는 35줄 정도의 소스 길이를 가지며 26개의 분기와 14개의 경로를 가진 소스 코드이다.

실험에 사용된 유전알고리즘의 연산자는 토너먼트 선택, 일점 교배, 동적 돌연변이가 사용되었으며, 유전알고리즘의 개체 수는 50개를 선택하였다. 실험 결과는 각 테스트 데이터 생성 기법을 10번씩 반복하여 얻은 결과를 평균하여 나타내었다.

그림 12와 13은 triagle3.c와 triangle4.c의 시간에 따른 소스 코드의 경로 커버리지(path coverage) 달성율을 나타내는 그래프이다.

두 그래프 모두 매 10초 마다 본 논문에서 제안된 테스트 데이터 생성 방법과 기존의 테스트 데이터 생성 방법의 경로 커버리지 기준을 나타내고 있다. triangle3.c 소스 코드의 경우, 90%의 경로 커버리지 기준을 만족시키기 위해 제안된 방법이 80초, 기존의 방법은 110초가 소요되었다. triangle4.c 소스 코드의 경우, 90%의 경로

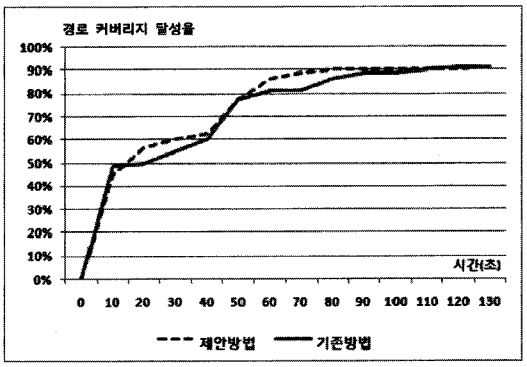


그림 12 triangle3.c 소스코드의 시간-커버리지 그래프

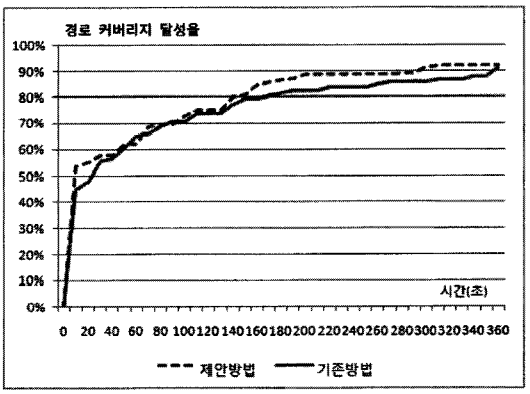


그림 13 triangle4.c 소스코드의 시간-커버리지 그래프

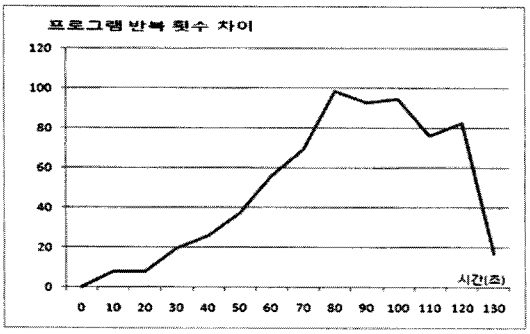


그림 14 triangle3.c 소스코드의 시간-프로그램 반복 횟수 차이 그래프

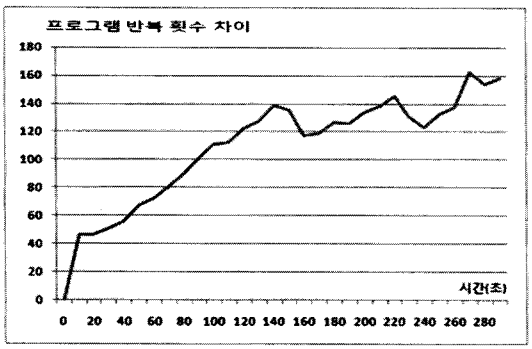


그림 15 triangle4.c 소스코드의 시간-프로그램 반복 횟수 차이 그래프

커버리지 기준을 만족시키기 위해 제안된 방법이 290초, 기존의 방법은 360초가 소요되었다. triangle3.c의 경우 27%, triangle4.c의 경우 19%의 테스트 데이터 생성의 효율성이 향상된 것을 확인할 수 있다.

그림 14, 15는 제안된 방법과 기존의 방법의 시간에 프로그램의 반복 횟수 차이를 나타내는 그래프이다. 제안된 방법은 적합도 평가 프로그램을 반복하고, 기존의 방법은 테스트 대상 프로그램의 가공 소스 코드를 반복한다. 시간이 지남에 따라 triangle3.c의 경우 1초당 1번, triangle4.c의 경우 2초당 1번 제안된 방법이 기존의 방법보다 프로그램을 더 수행하는 것을 확인할 수 있다. 이런 차이가 나는 이유는 제안된 방법이 SUT 대신 FEP를 이용하여 테스트 데이터의 적합도를 평가하여 테스트 데이터의 적합도 평가 시간을 줄였기 때문이다. 적합도 평가 시간을 줄임으로써 유전알고리즘의 수행 횟수의 차이를 만들고, 적합도가 더 뛰어난 테스트 데이터가 생성될 확률을 증가시킨다. 결국, 그림 11, 12와 같이 제안된 방법의 경로 커버리지 달성율이 기존 방법의 경로 커버리지 달성율보다 평균적으로 높게 나타나는 원인이 되는 것이다.

5. 관련 연구

Xiao 등[2]은 테스트 데이터 생성의 효율성을 향상시키기 위해서 가장 효율적인 최적화 알고리즘을 이용할 것을 주장하고 있다. 그들은 유전알고리즘, 담금질 기법, genetic simulated annealing, simulated annealing with advanced adaptive neighborhood, random의 알고리즘을 이용하여 테스트 데이터를 생성하는 실험을 수행하였다. 실험을 통하여 유전알고리즘이 테스트 데이터를 생성하기 위한 가장 우수한 최적화 알고리즘으로 결론을 내리고 테스트 데이터 생성을 위한 최적화 알고리즘으로 유전알고리즘을 이용할 것을 주장한다.

Watkins와 Hufnagel[10]은 기존의 동적 테스트 데이터 자동 생성 기법에 사용되었던 적합도 평가 방법들의 효율성을 비교, 분석하기 위한 실험을 수행하였다. 실험에 사용된 적합도 평가 함수는 BP1, BP2, NEHD, IPP, DO, DO-NEHD, Static, Random이다. 실험을 통하여 BP1, BP2, IPP의 적합도 평가 방법을 이용한 방법이 다른 적합도 평가 방법을 이용한 방법보다 테스트 데이터 생성의 효율성 측면에서 더 우수한 것을 알 수 있었다.

최적화 알고리즘을 개선하여 테스트 데이터 생성의 효율성을 높이기 위한 연구도 존재하였다. Alba와 Chicano[7]의 논문에서는 진화알고리즘을 병렬적으로 수행하여 테스트 데이터 생성의 효율성을 향상시키려 하였으며, Wang[9]은 기존의 유전알고리즘을 개선한 미미틱(memetic) 알고리즘을 이용하여 테스트 데이터를 생성하려 하였다.

기존 연구들이 적합도 평가를 위해 SUT 가공 프로그램을 이용한 것과 달리, 본 논문에서는 SUT 가공 프로그램의 복잡도를 낮춘 적합도 평가 프로그램을 이용하여 테스트 데이터 생성의 효율성을 향상시키는 연구를 진행하였다. 실험 결과, BP1[10]의 적합도 평가 방법에 제안된 방법을 적용하여 기존의 방법보다 약 20%의 테스트 데이터 생성 시간을 줄일 수 있었다.

6. 결론 및 향후 연구

전역 최적화 알고리즘을 이용한 테스트 데이터 생성의 단점은 테스트 데이터 생성에 많은 시간이 필요하다는 것이다[2,7,9]. 본 논문에서는 테스트 데이터의 적합도를 평가하기 위해 테스트 대상 소프트웨어(SUT) 가공 프로그램 대신 적합도 평가 프로그램(FEP)을 이용한 테스트 데이터 생성 방법을 제안하였다. 그리고 제안된 방법을 구현한 도구인 'ConGA'를 제작하여, C언어로 작성된 프로그램을 대상으로 테스트 데이터 생성 실험을 수행하였다.

실험결과를 통해 본 논문에서 제안된 방법이 기존의

방법[6]에 비해 실험 대상 소스 코드의 경로 적합성 기준 90%를 만족하는데 걸리는 시간을 약 20%정도 단축시키는 것을 확인할 수 있었다. 테스트 데이터 생성 시간이 단축된 이유는 제안된 방법이 SUT 가공 프로그램보다 복잡하지 않은 FEP를 이용하여 테스트 데이터의 적합도를 평가 시간을 줄였기 때문이다. 이로 인하여 유전알고리즘의 수행 횟수의 차이를 만들 수 있고, 유전알고리즘의 더 많은 반복 수행 때문에 적합도가 더 뛰어난 테스트 데이터가 생성될 확률을 증가시킬 수 있었다.

향후 연구에서는 테스트 자동 데이터 생성기(ConGA)를 보완하여 다양한 SUT에 적용하고, 이를 통해 테스트 데이터 생성의 효율성 향상의 여부를 확인해 보겠다. 제안된 방법에서 심볼릭 실행(symbolic execution)은 테스트 데이터를 생성하기 위한 절차 중 하나이다. 심볼릭 실행을 이용한 방법은 포인터와 함수 호출등의 문제가 제기되어 왔기 때문에, 도구를 보완하여 문제가 되는 SUT에도 제안 방법을 적용하여 제안 방법의 효율성을 확인해 보겠다. 그리고 Watkins and Hufnagel[10]에 소개된 다양한 적합도 평가 방법에 제안된 방법을 적용해 보겠다. 본 논문에서는 제안된 방법은 다양한 적합도 평가 방법에 적용될 수 있기 때문에, 제안된 방법이 적용된 적합도 평가 방법의 효율성 비교는 의미있는 작업이 될 것으로 생각한다. 마지막으로, 유전 알고리즘뿐만 아니라 다양한 전역 최적화 알고리즘에 제안된 방법을 적용하여 각 알고리즘에 따른 테스트 데이터 생성의 효율성을 비교해보겠다. 유전알고리즘과 같이 전역 최적화 알고리즘에는 성능에 영향을 미칠 수 있는 매개변수가 존재하므로, 이를 손쉽게 적용할 수 있는 프레임워크를 개발하여, 각 최적화 알고리즘의 매개변수에 따른 테스트 데이터 생성의 효율성에 대해 조사해 보겠다.

참 고 문 헌

[1] B. Beizer. Software Testing Techniques. International Thomson Computer Press, 1990.

[2] Man Xiao, Mohamed El-Attar, Marek Reformat, "Empirical evaluation of optimization algorithms when used in goal-oriented automated test data generation techniques," *Empir Software Eng.*, vol.12, pp.183-239, 2007.

[3] KOREL, B, "Automated software test generation," *IEEE Trans. Softw. Eng.*, vol.16, no.8, pp.870-879, 1990.

[4] P. M. S. Bueno and M. Jino, "Identification of potentially infeasible program paths by monitoring the search for test data," in *Proceedings of the fifteenth IEEE international conference on Automated Software Engineering (ASE '00)*, pp.209-218, 2000.

[5] Y. Zhan and J. A. Clark, "Search-based mutation testing for Simulink models," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, Washington DC, USA: ACM, 2005.

[6] Michael CC, McGraw G, Schatz MA., "Generating software test data by evolution," *IEEE Transactions On Software Engineering*, vol.27, no.12, pp.1085-1110, 2001.

[7] E. Alba and F. Chicano, "Observations in using parallel and sequential evolutionary algorithms for automatic software testing," *Computers and Operations Research*, vol.35, no.10, pp.3161-3183, 2008.

[8] Y. Zhan and J. A. Clark, "The state problem for test generation in Simulink," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '06)*, Seattle, Washington, USA: ACM, 2006.

[9] Wang, H.-C, "A Hybrid Genetic Algorithm for Automatic Test Data Generation," Master's thesis, National Sun Yat-sen University, Department of Information Management, Taiwan, China, 2006.

[10] E. M. H. Alison Watkins, "Evolutionary test data generation: a comparison of fitness functions," *Software: Practice and Experience*, vol.36, pp.95-116, 2006.

[11] S. Ali, L. C. Briand, H. Hemmati, and K. R. Panesar-Walawege, "A systematic review of the application and empirical investigation of search-based test-case generation," *IEEE Transactions on Software Engineering, Accepted for future publication.*

[12] T. Shiba, T. Tsuchiya, and T. Kikuno, "Using artificial life techniques to generate test cases for combinatorial testing," in *Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC '04)*, pp.72-77, 2004.

[13] R. Sagarna and J. A. Lozano, "Scatter search in software testing, comparison and collaboration with estimation of distribution algorithms," *European Journal of Operational Research*, vol.169, no.2, pp.392-412, 2006.

[14] Sagarna R., and Lozano J. A., "On the performance of Estimation of Distribution Algorithms applied to software testing," *Applied Artificial Intelligence*, vol.19 no.5, pp.457-489, 2005.

이 선 열

2008년 부산대학교 전자전기정보컴퓨터공학과 졸업(학사). 2010년 부산대학교 대학원 컴퓨터공학과(공학석사). 2010년~현재 부산대학교 대학원 컴퓨터공학과 박사과정. 관심분야는 소프트웨어공학, 소프트웨어 테스팅 자동화, 테스트 데이터 자동 생성 등

최 현 재

2009년 부산대학교 전자전기정보컴퓨터공학과 졸업(학사). 2010년 현재 부산대학교 대학원 컴퓨터공학과 석사과정. 관심분야는 소프트웨어공학, 테스팅 자동화 등

정 연 지

2009년 부산대학교 정보컴퓨터공학과 졸업(학사). 2010년~현재 부산대학교 대학원 컴퓨터공학과 석사과정. 관심분야는 소프트웨어공학, 메타몰픽 테스팅 등

배 정 호

2007년 부산대학교 전자전기정보컴퓨터공학과 졸업(학사). 2009년 부산대학교 대학원 컴퓨터공학과(공학석사). 2010년~현재 부산대학교 대학원 컴퓨터공학과 박사과정. 관심분야는 소프트웨어공학, 테스팅 자동화 등

김 태 호

1991년 3월~1995년 2월 성균관대학교 정보공학과 학사. 1995년 3월~2005년 2월 KAIST 전산학 석사. 박사. 2001년 6월~2002년 6월 SRI International 방문연구원. 2005년 3월~현재 ETRI 소프트웨어연구부문 선임연구원. 관심분야는 정형 명세 및 검증, 프로그램 정적 분석, 임베디드 소프트웨어

채 흥 석

1994년 서울대 원자핵공학 학사. 1996년 한국과학기술원 전산학 석사. 2000년 한국과학기술원 전산학 박사. 2000년~2003년 (주)동양시스템즈 기술연구소 선임연구원. 2003년~2004년 한국과학기술원 전산학과 초빙교수. 2004년~현재 부산대학교 컴퓨터 공학과 조교수. 관심분야는 객체지향 방법론, 소프트웨어 테스팅, 소프트웨어 메트릭, 소프트웨어 유지보수, 미들웨어 설계, 프로덕트라인 공학