테스트 데이터 생성을 위한 개선된 이웃 선택 방법을 이용한 담금질 기법 기술†

(a improved neighborhood selection of simulated annealing technique for test data generation)

최현재‡ 이선열§ 채흥석¶
(Hyun Jae Choi) (Seon Yeol Lee) (Heung Seok Chae)

요 약 금질 기법을 이용한 테스트 데이터 자동 생성 기법은 오랫동안 연구되어 왔으며, 효율적인 테스트 데이터 생성 방법 중 하나이다. 그러나 인풋 도메인이 넓을 경우, 기존의 담금질 기법은 이웃 선택 기법의 한계 때문에 나쁜 성능을 보였다. 이러한 한계를 극복하기 위해, 우리는 새로운 이웃 선택 기법인 분기 거리 이웃 선택 방법을 제안한다. 제안된 기법의 성능을 검증하기 위해서 우리는 분기 거리 이웃 선택 기법, 기존의 이웃 선택 기법 그리고 랜덤 테스트 데이터 생성 기법을 비교하는 실험을 수행하였다. 실험 결과 제안된 기법이 2가지 기법에 비해 우수한 성능을 보임을 알 수 있었다.

키워드 테스트 데이터 자동 생성, 동적 테스트 데이터 생성, 담금질 기법, 분기 거리 이웃 선택, 패스 테스팅

Abstract Simulated annealing has been studied a long times. And it is one of the effective techniques for test data generation. But basic SA methods showed bad performance because of neighborhood selection strategies in the case of large input domain. To overcome this limitation, we propose new neighborhood selection approach, Branch Distance. We performs case studies based on the proposed approach to evaluate it's performance and to compare it whit basic SA and Random test generation. The results of the case studies appear that proposed approach show better performance than the other approach.

Key words Test data automation, Dynamic test data generation, Simulated annealing, Branch distance neighborhood selection, path testing

1. 서 론

소프트웨어 개발 산업에서 소프트웨어 테스팅은 전체 소프트웨어 개발 비용의 약 50%를 차지하는 많은 비용이 요구되는 작업이다[1]. 소프트웨어 테스팅에 요구되는 높은 비용은 소프트웨어 테스팅이 소프트웨어 개발 프로세스에서 매우 큰 비중을 차지하는 작업인 것을 나타낸다.

소프트웨어 테스팅은 테스트 데이터 생성, 테스트 실행, 테스트 결과 검증의 3단계로 이루어진다. 그 중에서 테스트 데이터 생성은 전체 소프트웨어 테스팅 비용의 약 40%를 차지한다[2][3]. 때문에 소프트웨어 테스팅 비용 절감은 전체 소프트웨어 개발 비용 절감을 위해 중요한 이슈이다.

테스트 데이터 생성 비용 절감을 위해 최근 테스트 데이터 자동 생성 방법의 하나인 탐색 기반

† 본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다.
‡ 학생회원 : 부산대학교 컴퓨터공학부
gonoki@pusan.ac.kr
§ 학생회원 : 부산대학교 컴퓨터공학부
sun302412@pusan.ac.kr
¶ 정 회 원 : 부산대학교 컴퓨터공학부 부교수
hschae@pusan.ac.kr
논문접수 : 2011년 05월 13일
심사완료 : 2011년 06월 10일

소프트웨어 테스팅이 많이 연구되고 있다. 탐색 기반 소프트웨어 테스팅은 메타 휴리스틱 알고리즘을 이용하여 테스트 요구사항을 만족하는 테스트 데이터를 자동적으로 생성하는 기법이다. 탐색 기반 테스트 데이터 생성 기법으로 랜덤, 담금질 기법[3][4][5][6][7], 유전 알고리즘[3][5][7][11][12] [13][14]을 사용하는 연구들이 수행되었다.

그 중 담금질 기법을 이용한 테스트 데이터 자동 생성 기법은 알고리즘이 단순하여 알고리즘 수행 속도가 빠르고 구현이 간단하기 때문에 많은 연구에서 적용되었고 몇몇 연구에서 우수한 성능을 보여주었다[3][4][5][6][7].

하지만 담금질 기법은 데이터 범위가 넓어지면 성능이 나빠지는 한계가 있었다[3]. 이러한 담금질 기법의 한계를 극복하기 위해, 본 논문은 담금질 기법의 새로운 이웃 선택 방법인 분기 거리 이웃 선택 방법을 제안한다. 또한 우리는 제안한 방법을 적용하기 위한 도구를 개발하였고 이 도구를 이용해 2개의 프로그램에 대해 실험을 수행하였다. : triangle3, triangle4

본 논문은 탐색 기반 테스트 데이터 생성 기법을 이용하여 패스 테스팅을 수행한다. 패스 테스팅은 생성한 테스트 케이스들이 목표로 하는 패스들을 커버하도록 하는 매우 복잡하고 강력한 기법이다. 탐색 기반 소프트웨어 테스팅의 테스트 적합성 기준을 높은 패스 커버리지로 설정하여 패스 테스팅을 수행한다. 패스 테스팅은 많은 탐색 기반 테스트 데이터 생성 기법에 사용되어 왔다[8][9][10].

이 논문의 기여는 다음과 같다. 첫째, 기존의 담금질 기법을 이용한 테스트 데이터 생성 기법의 한계를 극복하고자 새로운 이웃 선택 방법을 제안한다. 둘째, 제안한 기법의 성능을 평가하기 위해 사례 연구를 수행하였다. 셋째, 사례 연구의 결과를 통해 기존의 담금질 기법과 랜덤 테스트 데이터 생성 기법과 새로이 제안한 분기 거리 이웃 선택을 이용한 담금질 기법을 비교, 분석한다.

본 논문은 다음과 같이 구성된다. 2장에서는 배경 연구를 소개한다. 3장에서는 효과적인 테스트 데이터 생성을 위한 분기 거리 이웃 선택 방법을 제안한다. 4장에서는 제안한 방법을 성능을 알아보기 위해 사례 연구를 수행한다. 5장에서는 결론 및 향후 연구 방향을 기술한다.

2. 배경 연구

배경 연구에서는 탐색 기반 소프트웨어 테스팅, 담금질 기법 그리고 기존의 이웃 선택 방법에 대해서 설명한다.

2.1 탐색 기반 소프트웨어 테스팅

탐색 기반 소프트웨어 테스팅은 테스트 적합성 기준에 따라서 테스트 데이터를 자동적으로 생성하는 프로세스이다. 이 방법은 프로그램 수행을 통해 동적으로 정보를 수집하고 메타 휴리스틱 알고리즘을 사용하여 테스트 적합성 기준을 만족하는 테스트 데이터를 생성한다. 언덕 오르기 알고리즘, 담금질 기법, 탐욕 알고리즘, 유전 알고리즘의 메타 휴리스틱 알고리즘을 탐색에 사용한 탐색 기반 소프트웨어 테스팅 연구가 진행되었다.

테스트 적합성 기준을 만족하는 테스트 데이터를 생성하기 위해서 동적으로 정보를 수집하기 위해 적합도 함수를 사용한다. 적합도 함수는 테스트 데이터가 테스트 목적에 얼마나 적합한지를 표현하는 함수이다. 이것은 테스트 적합성 기준을 만족하는 테스트 데이터 탐색을 인도하는 데 사용된다.

2.2 담금질 기법

담금질 기법은 개체가 국소 최적점에서 벗어나 전역 최적점을 찾을 수 있도록 하는 알고리즘이다. 담금질 기법은 온도에 따른 원자의 에너지 변화를

모델로 한 것이다. 물체의 온도를 높여주면 열에 의해 원자의 위치가 초기 위치로부터 멀어진다. 그 후 물체의 온도를 서서히 내려주면 원자가 안정화되면서 초기 상태보다 내부 에너지가 한층 극소인 상태를 얻을 가능성이 많아진다. 담금질 기법은 이러한 물리 법칙을 모사한 것이다.

표 1. 담금질 기법을 이용한 테스트 데이터 생성 알고리즘

```
초기 재시작 횟수 r = 0
Start:
초기 상태 선택 s
초기 온도 선택 t
초기 상태의 Cost 평가    cost(s)
Loop while (종료 조건 == false)
   If (r ≥ 최대_재시작_횟수) Then
      GoTo Start:
   End If
   새로운 상태 선택 s' ∈ Neighborhood(s)
   새로운 상태의 Cost 평가 cost(s')
   δ = cost(s') - cost(s)
   If (δ ≤ 0) Then
      s = s'
   Else
      [0,1] 범위의 랜덤 값 생성    x
      If (x < exp(-δ/ t)) Then
         현재 상태 갱신    s = s'
      End If
   End If
   온도 냉각    t = coolingScheduling(t)
   재시작 횟수 증가 r = r + 1
End Loop
```

표 1은 담금질 기법을 이용한 테스트 데이터 생성 알고리즘을 의사 코드로 설명한 것이다. 초기 매개변수 설정을 하고 알고리즘을 시작한다. 현재 상태의 이웃을 선택하여 새로운 상태를 선택하고 적합도 함수를 이용하여 새롭게 선택된 상태의 cost를 평가한다. cost 값은 매개변수로 전달되는 상태가 얼마나 테스트 적합성 기준에 가까운가를 나타낸다. 이 값은 0에 가까울수록 적합성 기준에 더 가깝고 크면 클수록 적합성 기준에서 더 먼 것을 표현한다. 새로운 상태가 현재 상태보다 테스트 적합성 기준에 가까워 cost 값이 더 작을 경우 현재 상태를 갱신한다. 만약 새로운 상태가 현재 상태보다 테스트 적합성 기준에 더 멀어서 cost 값이 더 클 경우에도 온도를 통해 구해지는 확률(exp(-δ/t))로 현재 상태를 갱신한다. 그 후, 냉각 스케쥴링을 이용하여 온도를 감소시킨다. 종료 조건이 만족될 때까지 이러한 과정을 반복한다.

담금질 기법의 특징으로는 상태 전이 결정을 할 때 탐색 조건에 가까운, 좋은 상태로만 전이하는 것이 아니라 확률적으로 나쁜 상태로 전이할 수 있도록 하는 것이다. 확률은 일반적으로 exp(-δ/t) 라는 수식으로 결정된다. 확률에 영향을 주는 매개변수는 온도이다. 만약 온도가 높으면 확률이 높고 만약 온도가 낮으면 확률이 낮게 결정된다. 이와 같은 수행을 통해 현재의 상태가 국소 최적에 빠져 있더라도 이를 벗어나 전역 최적점을 찾을 수 있도록 한다. 알고리즘의 수행이 진행될수록 온도가 점차 낮아져서 시간이 지날수록 좋은 상태로만 전이할 확률이 점차 높아진다.

온도 냉각 알고리즘에 의해 나쁜 상태로 전이할 확률이 반복 횟수가 증가함에 따라 낮아진다. 이 때문에 반복 횟수가 늘어나면 상태가 지역 최적에서 벗어나지 못할 수 있다. 때문에 일정 횟수 이상 반복 수행하였을 경우 알고리즘을 재시작하는 것이 성능이 더 좋을 수 있다.

담금질 기법을 이용한 테스트 데이터 생성 기법의 주요 매개변수는 초기 상태 생성 방법, 데이터 범위, 적합성 평가 함수, 이웃 선택 방법, 이웃 크기, 최대 재시작 횟수, 초기 온도 설정, 냉각 알고리즘이 있다.

2.3 기존의 이웃 선택 방법

기존의 이웃 선택 방법은 기존의 테스트 데이터 자동 생성 연구들에서 이용한 기본적인 담금질 기법의 이웃 선택 알고리즘을 말한다.

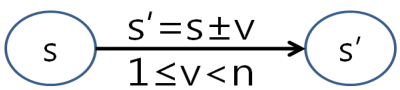

그림 1. 기존의 이웃 선택 방법

그림 1은 기존의 이웃 선택 방법을 그림으로 표현한 것이다. s' 는 새로운 상태, s는 현재 상태, v는 변화 값, n은 이웃 크기를 나타낸다. 이웃 크기는 현재 상태의 이웃을 결정하는 범위 값이다. 기존의 이웃 선택 방법은 현재 상태에서 이웃 크기 이내의 범위를 이웃이라고 판단한다. 따라서 새로운 상태는 이웃 크기 이내의 범위의 값이 선택된다.

표 2. 기존의 이웃 선택 알고리즘

```
현재 상태 s
현재 이웃 크기 n
변화 값 선택 v = {v| 1≤v<n}
새로운 상태 선택 s' = s ± v
If (isOutOfDomainRange(s')) Then
    다시 새로운 상태 선택 s' =
BasicNeighbor- hoodSelection (s)
End If
```

표 2는 기존의 이웃 선택 알고리즘을 의사 코드로 표현한 것이다. 만약 이웃 선택 알고리즘으로 생성된 새로운 상태가 데이터 범위를 초과한다면 다시 이웃 선택 알고리즘을 수행하여 새로운 상태를 구한다.

3. 분기 거리 이웃 선택

분기 거리 이웃 선택에서는 분기 거리에 대해서 설명하고 분기 거리 이웃 선택 방법에 대해 소개한다.

3.1 분기 거리

분기 거리는 테스트 데이터가 얼마나 목표에 가까운지를 평가하는 값이다. 이것은 탐색 기반 테스트 데이터 자동 생성기법에서 적합도 함수로 흔히 사용되는 Korel 의 방법[8]을 조금 수정한 것이다. 분기 거리의 값은 목표에 가까울수록 값이 작아진다.

표 3은 연산자 유형별 분기 거리 계산의 방법을 표로 나타낸 것이다.

표 3. 분기 거리

연산자 유형	조건 예	분기 거리 (K = 1000)
Equal	if (a == 1)	if (a == 1) 0 else abs(a - 1)
NotEqual	if (a != 1)	if (a != 1) 0 else K
And	if ((a==1) && (b==1))	sum((a == 1) ? 0 : abs(a - 1), ((b == 1) ? 0 : abs(b - 1))
Or	if ((a==1) ¦¦ (b==1))	min((a == 1) ? 0 : abs(a - 1), ((b == 1) ? 0 : abs(b - 1))
Grater	if (a > 1)	if (a > 1) 0 else 1 - a + 1
GraterEqual	if (a >= 1)	if (a >= 1) 0 else 1 - a
Less	if (a < 1)	if (a < 1) 0 else a - 1 + 1
LessEqual	if (a <= 1)	if (a <= 1) 0 else a - 1
Not	if (!(a == 1))	if (!(a == 1)) 0 else abs(a - 1)

(a == 1) 분기의 경우 만약 a와 1이 같다면 목표 조건을 달성하였기 때문에 분기 거리를 0으로 계산한다. 하지만 만약 a와 1이 다르다면 a와 1 사이의 차이 값이 (a == 1) 조건을 만족하기 위한 분기 거리가 된다. 예를 들어, a가 2일 경우 분기 거리는 abs(2-1) = 1이 된다.

(a != 1) 분기의 경우 목표 조건을 달성하였을 경우 분기 거리를 0으로 계산한다. 만약 a와 1이 같아 목표 조건을 달성하지 못하였을 경우 임의의 큰 값 K를 분기 거리로 나타낸다. 이것은 a가 어느 값으로 이동하여도 목표 분기의 조건을 달성할 수 있기 때문이다. 실험에서는 K 값을 1000으로 설정하였다.

And 조건의 경우 좌변과 우변의 두 조건을 모두를 만족해야 하기 때문에 좌변과 우변의 분기 거리를 더하여 조건의 분기 거리로 표현한다.

Or 조건의 경우 좌변과 우변의 두 조건 중 하나만을 만족하면 되기 때문에 좌변과 우변의 분기 거리 중 작은 값을 선택하여 조건의 분기 거리로 표현한다.

크거나 작은 비교 조건을 표현하는 분기문의 경우 비교 조건의 좌변과 우변 중 작은 값에서 큰 값을 뺀 후 1을 더하여 분기 거리를 표현한다.

크거나 같은 혹은 작거나 같은 비교 조건을 표현하는 분기문은 비교 조건의 좌변과 우변 중 작은 값에서 큰 값을 뺀 값을 분기 거리로 표현한다. 이 경우 양 변의 값이 같아지기만 해도 목표 조건을 만족하기 때문에 다른 값을 더하지 않는다.

3.2 분기 거리 이웃 선택 방법

분기 거리 이웃 선택 방법은 데이터 범위가 넓을 경우 성능이 나쁘다는 기존의 이웃 선택 방법의 한계를 극복하기 위한 것이다. 만약 분기 거리가 멀 경우, 즉 목표에서 먼 테스트 데이터는 이웃을 분기 거리만큼 떨어진 값으로 선택하여 다음 테스트 데이터를 분기 거리만큼 먼 거리에 위치한 값으로 선택한다. 분기 거리가 가까울 경우에는 기존의 이웃 선택 방법을 이용하여 기존의 담금질 기법의 장점을 그대로 활용하고자 하였다.

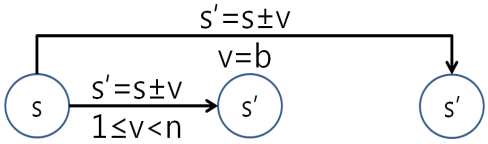

그림 2. 분기 거리 이웃 선택

분기 거리 이웃 선택 방법은 데이터 범위가 넓을 경우 성능이 나쁘다는 기존의 이웃 선택 방법의 한계를 극복하기 그림 2는 분기 거리 이웃 선택 기법을 그림으로 표현한 것이다. s’ 는 새로운 상태, s는 현재 상태, v는 변화 값, n은 이웃 크기, b는 분기 거리를 나타낸다.

표 4. 분기 거리 이웃 선택 알고리즘

```
분기 거리 b
현재 상태 s
이웃 크기 n
새로운 상태 선택 s’ = s ± b
If (b < n || isOutOfDomainRange(s’)) Then
    새로운 상태 선택 s’ = BasicNeighborhood
Selection(s)
End If
```

표 4는 분기 거리 이웃 선택 알고리즘을 의사 코드로 표현한 것이다. 만약 분기 거리가 이웃 크기보다 작거나 분기 거리 이웃 선택 알고리즘으로 생성된 새로운 상태가 데이터 범위를 초과한다면 기존의 이웃 선택 알고리즘을 이용하여 새로운 상태를 구한다.

4. 사례 연구

사례 연구에서는 테스트 대상 프로그램에 대해서 소개하고 실험에서 적용할 담금질 기법의 세부 매개변수에 대해서 설명한다. 그리고 실험 평가기준을 소개하고 실험 결과를 보여주고 이를 분석한다.

4.1 테스트 대상 프로그램

2가지의 C 프로그램에 대해서 실험을 수행하였다. triangle3과 triangle4는 [15][16] 에서 사례 연구로 사용되었던 프로그램이다. 두 프로그램 모두 3개의 매개변수를 받아 삼각형이 될 수 있는지, 삼각형이 될옷 수 있다면 어떤 삼각형이 될 수 있는지 유형을 분석하는 기능을 수행한다. triangle4가 triangle3 프로그램보다 분기문이 더 복잡하게 구성되어 있다는 차이가 있다.

표 5. 테스트 대상 프로그램

프로그램 이름	LOC	패스 개수	매개변수 개수
triangle3	69	9	3
triangle4	97	14	3

표 5는 테스트 대상 프로그램을 나타낸 것이다. 두 프로그램 모두 3개의 int 타입 매개변수를 입력으로 받는다. 실험은 Intel Core 2 Duo E6600, 400MHz 1G RAM의 성능을 가진 컴퓨터로 수행하였다. triangle3의 코드 라인 수는 69이고 도달할 수 있는 패스는 9개, 도달할 수 없는 패스는 0개

이다. triangle4의 코드 라인 수는 97이고 도달할 수 있는 패스는 14개, 도달할 수 없는 패스는 0개이다.

4.2 담금질 기법의 세부 매개변수

담금질 기법을 이용한 테스트 데이터 생성 기법에는 성능에 영향을 줄 수 있는 여러 가지 매개변수들이 존재한다. 이러한 매개변수들의 값 설정에 대해서 설명한다.

표 6. 세부 매개변수

매개변수 종류	세부 매개변수
냉각 알고리즘(b=0.001)	$t_{i+1}=t_i/(1+b*t_i)$
데이터 범위	[0..1000], [0..10000]
이웃 선택 방법	기존 이웃 선택, 분기 거리 이웃 선택
이웃 크기	50
적합성 평가 함수	분기 거리
초기 상태 생성 방법	랜덤 생성
초기 온도 설정	50
최대 재시작 횟수	200

표 6은 실험에서 사용한 담금질 기법의 세부 매개변수를 나타낸 것이다. 매개변수 종류로는 냉각 알고리즘, 데이터 범위, 이웃 선택 방법, 이웃 크기, 적합성 평가 함수, 초기 상태 생성 방법, 초기 온도 설정, 최대 재시작 횟수가 있다.

냉각 알고리즘은 [17] 에서 제안한 방법으로 반복 수행됨에 따라 온도가 서서히 낮아진다. 논문에서는 이 식의 b를 0에 가까운 값으로 설정하라고 언급하였고 0.001을 b 값으로 설정하여 실험에서는 수행하였다.

데이터 범위는 생성하는 테스트 데이터 값의 범위를 말한다. 담금질 기법이 데이터 범위가 넓어질 경우 성능이 떨어지는 한계를 극복하고자 제안한 방법이기 때문에 이를 실험하기 위해 0 이상 1000이하의 상대적으로 좁은 범위와 0이상 10000이하의 상대적으로 넓은 범위를 데이터 범위로 설정하였다.

이웃 선택 방법은 기존의 이웃 선택 방법과 새로이 제안한 분기 거리 이웃 선택방법 2가지를 사용하여 실험을 수행하였다

이웃 크기는 새로운 상태를 생성하기 위해 이웃 선택을 수행할 때 선택 기준이 되는 값이다. 이 값은 [3] 에서 사용된 바 있는 50을 적용하였다.

적합성 평가 함수는 테스트 데이터가 얼마나 목표에 가까운지를 나타내는 함수이다. 이것은 3.1. 에서 설명한 분기 거리 방법을 이용하였다.

초기 상태 생성 방법은 초기 테스트 데이터 생성 방법을 말한다. 랜덤 방법을 이용하여 초기 상태를 생성한다.

초기 온도 설정은 담금질 기법에서 목표와 가깝지 않은, 좋지 않은 상태로 전이할 확률을 결정하는 매개변수인 온도의 초기값을 설정하는 것이다. 이 값은 50으로 설정하였다.

최대 재시작 횟수는 담금질 기법이 지역 최적점에 빠지는 것을 방지하기 위해 알고리즘을 재시작하게 하는 수행 횟수를 말한다. 최대 재시작 횟수만큼 반복한 후 담금질 알고리즘을 재시작하게 된다. 이 값은 200으로 설정하였다.

4.3 평가기준

실험의 성능이 우수하다는 것을 보이기 위한 평가기준이 필요하다. 본 연구에서는 생성한 테스트 데이터의 평가를 위해 패스 커버리지를 이용한다.

본 논문은 기존의 담금질 기법, 새로이 제안한 분기 거리 이웃 선택을 이용한 담금질 기법, 랜덤을 이용하여 테스트 데이터를 생성하는 실험을 수행하였다. 알고리즘이 서로 차이가 있기 때문에 알고리즘의 차이 때문에 각 알고리즘의 반복 수행 횟수당 수행 시간이 차이가 날 수가 있다. 따라서 보다 공정한 실험을 위해 커버리지/반복횟수가 아닌 커버리지/시간(초) 차트를 이용해 실험 결과를 표현한다.

또한 탐색 기반 소프트웨어 테스팅 방법은 메타 휴리스틱 알고리즘을 사용하기 때문에 커버리지 달성 시간이 수행에 따라 다를 수 있다. 따라서 탐색 기반 소프트웨어 테스팅 방법의 우연성을 줄이기 위하여 각 알고리즘을 10번 반복 수행하여 커버리지 최대, 최소, 평균 값을 구하여 이를 실험 결과로 분석한다.

4.4 실험 결과

실험 결과는 분기 거리를 이용한 담금질 기법(BD), 기존의 담금질 기법(Basic), 랜덤(Random) 알고리즘을 이용하여 300초 동안 수행하였다. 탐색 기반 소프트웨어 테스팅 기법의 임의성을 줄이기 위해 실험을 10번 반복 수행한다. 반복 수행한 결과의 평균 패스 커버리지를 커버리지/시간(초) 차트로 표현하여 결과를 분석한다.

1) triangle3

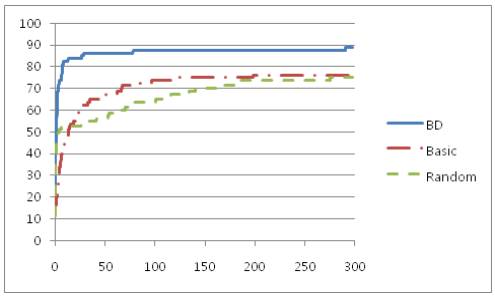

그림 3. triangle3[0..1000]

그림 3은 [0..1000] 데이터 범위를 사용하여 triangle3 프로그램의 테스트 데이터를 자동 생성한 결과를 커버리지/시간 차트로 표현한 것이다. 모든 구간에서 BD의 성능이 우수하다. 그리고 Basic과 랜덤의 성능 차이가 크기 않은 것을 볼 수 있다.

표 7. triangle3[0..1000]의 최종 커버리지

알고리즘 종류	최종 패스 커버리지
BD	88.75
Basic	76.25
Random	75

표 7은 [0..1000] 데이터 범위를 사용하여 triangle3 프로그램의 테스트 데이터를 자동 생성한 최종 패스 커버리지 결과를 표로 나타낸 것이다. BD가 Basic보다 16.39% 더 높은 최종 커버리지 결과를 보인다. 그리고 BD가 랜덤보다 18.33% 더 높은 최종 커버리지 결과를 보인다.

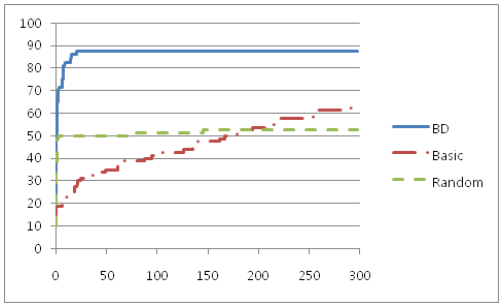

그림 4. triangle3[0..10000]

그림 4는 [0..10000] 데이터 범위를 사용하여 triangle3 프로그램의 테스트 데이터를 자동 생성한 결과를 커버리지/시간 차트로 표현한 것이다. 모든 구간에서 BD의 성능이 우수하다. 그리고 Basic의 경우 시간의 경과에 따라 커버리지가 지속적으로 증가하여 193초 이후로는 랜덤보다 높은 커버리지를 보여준다. 랜덤은 50%의 커버리지는 매우 빠른 시간에 달성하지만 그 이후로 커버리지 증가 폭이 작아진다.

표 8. triangle3[0..10000]의 최종 커버리지

알고리즘 종류	최종 패스 커버리지
BD	87.5
Basic	62.5
Random	52.5

표 8은 [0..10000] 데이터 범위를 사용하여 triangle3 프로그램의 테스트 데이터를 자동 생성한 최종 패스 커버리지 결과를 표로 나타낸 것이다. 분기 거리를 이용한 담금질 기법(BD)이 기존의 담금질 기법(Basic)보다 40% 더 높은 최종 커버리지 결과를 보인다. 그리고 분기 거리를 이용한 담금질 기법(BD)이 랜덤(Random) 알고리즘보다 66.67% 더 높은 최종 커버리지 결과를 보인다.

2) triangle4

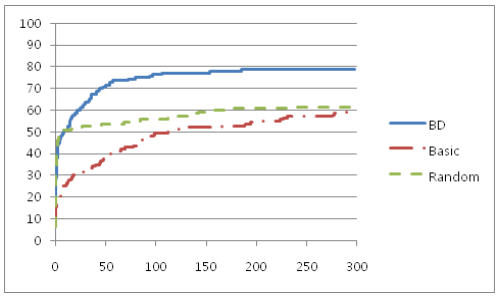

그림 5. triangle4[0..1000]

그림 5는 [0..1000] 데이터 범위를 사용하여 triangle4 프로그램의 테스트 데이터를 자동 생성한 결과를 커버리지/시간 차트로 표현한 것이다. 대부분의 구간에서 BD의 성능이 우수하다. 그리고 Basic과 랜덤의 성능 차이가 크기 않은 것을 볼 수 있다.

표 9. triangle4[0..1000]의 최종 커버리지

알고리즘 종류	최종 패스 커버리지
BD	78.57
Basic	59.29
Random	61.43

표 9는 [0..1000] 데이터 범위를 사용하여 triangle4 프로그램의 테스트 데이터를 자동 생성한 최종 패스 커버리지 결과를 표로 나타낸 것이다. BD가 Basic보다 32.53% 더 높은 최종 커버리지 결과를 보인다. 그리고 BD가 랜덤보다 27.91% 더 높은 최종 커버리지 결과를 보인다.

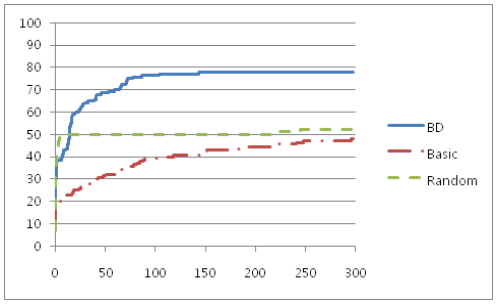

그림 6. triangle4[0..10000]

그림 6은 [0..10000] 데이터 범위를 사용하여 triangle4 프로그램의 테스트 데이터를 자동 생성한 결과를 커버리지/시간 차트로 표현한 것이다. 대부분의 구간에서 BD의 성능이 우수하다. 그리고 Basic의 경우 시간의 경과에 따라 커버리지가 지속적으로 증가하지만 랜덤보다 나쁜 성능을 보여준다. 랜덤은 50%의 커버리지는 매우 빠른 시간에 달성하지만 그 이후로 커버리지 증가폭이 작아진다.

표 10. triangle4[0..10000]의 최종 커버리지

알고리즘 종류	최종 패스 커버리지
BD	77.86
Basic	47.86
Random	52.14

표 10은 [0..10000] 데이터 범위를 사용하여 triangle4 프로그램의 테스트 데이터를 자동 생성한 최종 패스 커버리지 결과를 표로 나타낸 것이다. 분기 거리를 이용한 담금질 기법(BD)이 기존의 담금질 기법(Basic)보다 62.69% 더 높은 최종 커버리지 결과를 보인다. 그리고 분기 거리를 이용한 담금질 기법(BD)이 랜덤(Random) 알고리즘보다 49.32% 더 높은 최종 커버리지 결과를 보인다.

3) 실험 결과 분석

위의 실험들을 통해 데이터 범위가 좁을 경우 랜덤의 성능이 우수하고 Basic과 랜덤의 성능이

큰 차이가 나지 않는 것을 볼 수 있다. 그리고 데이터 범위가 넓어질 경우 Basic과 랜덤은 성능이 나빠지지만 BD 는 성능이 크게 변하지 않는 것을 볼 수 있다.

5. 결론 및 향후 연구

기존의 담금질 기법의 한계를 극복하기 위해 새로운 이웃 선택 방법을 제안하고 2가지 프로그램에 대해서 사례 연구를 수행하였다. 새로이 제안한 기법과 기존의 담금질 기법, 랜덤 알고리즘에 대해서 사례 연구를 수행하여 새로이 제안한 기법이 성능이 우수하며 기존의 담금질 기법의 한계로 지적되었던 데이터 범위가 넓어지는 경우에 대해서도 성능이 우수함을 보였다.

향후 연구에서는 좀더 다양한 프로그램에 대해 제안한 기법을 적용해보고 이를 다른 탐색 기반 소프트웨어 테스팅 기법과 비교해볼 계획이다. 또한 담금질 기법의 매개변수의 변화에 따른 성능의 비교를 통해 우수한 성능을 보이는 매개변수를 찾아낼 계획이다.

참 고 문 헌

[1] B. Beizer, Software testing techniques, Van Nostrand Reinhold Co. New York, NY, USA, 1990, .

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

[3] 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 (2007) 12:183-239

[4] N. Tracey, J. Clark, and K. Mander. Automated program flaw finding using 담금질 기법. In Software Engineering Notes, Issue 23, No.2, Proceedings of the International Symposium on Software Testing and Analysis (ISSTA 1998), pages 73-81, 1998.

[5] C. Michael and G. McGraw. Automated software test data generation for complex programs. In 13th IEEE International Conferance on Automated Software Engineering, pages 136-146, October 1998.

[6] 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.

[7] N. Tracey. A Search-Based Automated Test-Data Generation Framework for Safety Critical Software. PhD thesis, University of York, 2000.

[8] Korel, B. 1990. Automated software test generation, IEEE Trans. on Software Engineering 16(8): 870-879.

[9] Lin, J-C. and Yeh, P-U. "Automatic Test Data Generation for Path Testing using GAs," Information Sciences, 131: 47-64, 2001.

[10] N. Mansour and M. Salame. Data generation for path testing. Software Quality Journal, 12(2):121-134, 2004.

[11] Michael CC, McGraw G, Schatz MA. "Generating software test data by evolution," IEEE Transactions On Software Engineering 2001; 27(12):1085-1110.

[12] B. F. Jones, H. H. Sthamer, and D. E. Eyres, "Automatic structural testing using 유전 알고리즘s," Software Engineering Journal, vol. 11, pp. 299-306, 1996

[13] 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) 2000, pp. 209-218.

[14] N. Tracey, J. Clark, K. Mander, and J. McDermid. Automated test data generation for exception conditions. Software - Practice and Experience, 30(1):61-79, 2000.

[15] 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.

[16] 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.

[17] Lundy M, Mees AI (1986) Convergence of an annealing algorithm. Math Program 34(1):111-124

저 자 소 개

최 현 재

2009년 부산대학교 전자전기정보컴퓨터공학과 졸업(학사)
2010년 현재 부산대학교 대학원 컴퓨터공학과 석사 과정.

<관심분야> 소프트웨어공학, 테스팅 자동화

이 선 열

2008년 부산대학교 전자전기정보컴퓨터공학과 졸업(학사)
2010년 부산대학교 대학원 컴퓨터공학과(공학석사)
2010년~현재 부산대학교 대학원 컴퓨터공학과 박사과정.

<관심분야> 소프트웨어공학, 소프트웨어 테스팅 자동화, 테스트 데이터 자동 생성

채 흥 석

1994년 서울대 원자핵공학 학사
1996년 한국과학기술원 전산학 석사
2000년 한국과학기술원 전산학 박사
2000년~2003년 ㈜ 동양시스템즈 기술연구소 선임연구원

2003년~2004년 한국과학기술원 전산학과 초빙교수
2004년~현재 부산대학교 컴퓨터 공학과 부교수.

〈관심분야〉 객체지향 방법론, 소프트웨어 테스팅, 소프트웨어 메트릭, 소프트웨어 유지보수, 미들웨어 설계, 프로덕트라인 공학