การสร้างกรณีทดสอบโดยใช้ขั้นตอนวิธีเชิงพันธุกรรม
Test Case Generation Using Genetic Algorithms

นันทนี ช่วยชู(Nuntanee Chuaychoo)[1] และ สุภาภรณ์ กานต์สมเกียรติ (Supaporn Kansomkeat)[2]
[1]สาขาวิชาการจัดการเทคโนโลยีสารสนเทศ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์
[2]ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์
[1]nun-ta-nee@hotmail.com, [2]supaporn.k@psu.ac.th

บทคัดย่อ

การทดสอบซอฟต์แวร์ เป็นขั้นตอนหนึ่งที่สำคัญในกระบวนการผลิตซอฟต์แวร์ การทดสอบช่วยให้ซอฟต์แวร์ที่พัฒนามีความถูกต้อง และมีความน่าเชื่อถือเพิ่มมากขึ้น การสร้างกรณีทดสอบเป็นอีกองค์ประกอบหนึ่งที่มีความสำคัญในกระบวนการทดสอบซอฟต์แวร์ คุณภาพของการทดสอบจะขึ้นอยู่กับประสิทธิภาพของกรณีทดสอบ บทความนี้นำเสนอวิธีการสร้างกรณีทดสอบโดยใช้ขั้นตอนวิธีเชิงพันธุกรรม และได้นำขั้นตอนการสร้างกรณีทดสอบที่ได้นำเสนอไปประยุกต์ใช้กับกรณีศึกษา ผลลัพธ์ที่ได้พบว่า กรณีทดสอบที่สร้างขึ้นตามวิธีการที่นำเสนอนี้มีความเหมาะสมสำหรับการทดสอบซอฟต์แวร์

คำสำคัญ : การทดสอบซอฟต์แวร์, ขั้นตอนวิธีเชิงพันธุกรรม, การสร้างกรณีทดสอบ

Abstract

Software Testing is an important step in the software process. It makes the developed software more accurate and reliable. Generating test cases is a key element in the process of software testing. The quality of testing depends on the effectiveness of test. This paper presents test case generation method using genetic algorithms. The proposed method was applied on a case study. The result shows that the generated test cases are appropriate for testing software.

Keywords : Software Testing, Genetic Algorithms, Test Case Generation

1. บทนำ

ในปัจจุบันความต้องการของการใช้คอมพิวเตอร์และเทคโนโลยีสารสนเทศยังคงมีเพิ่มขึ้นเรื่อยๆ ดังนั้นจึงมีความต้องการซอฟต์แวร์เพื่อใช้งานเฉพาะต่าง ๆ อีกมากมาย การทดสอบซอฟต์แวร์ (Software Testing) [1] เป็นขั้นตอนที่สำคัญในกระบวนการพัฒนาซอฟต์แวร์ที่มีบทบาทและมีความสำคัญมาก การทดสอบเป็นกิจกรรมที่จัดทำขึ้นเพื่อปรับปรุงคุณภาพของซอฟต์แวร์ ค้นหาข้อบกพร่องและลดข้อผิดพลาดจากการทำงานของซอฟต์แวร์ให้เหลือน้อยที่สุด

การกำหนดกรณีทดสอบเป็นสิ่งหนึ่งที่สำคัญในกระบวนการทดสอบซอฟต์แวร์ ขั้นตอนการสร้างกรณีทดสอบที่ดี จะมีส่วนทำให้การทดสอบมีประสิทธิภาพ กรณีทดสอบที่มีประสิทธิภาพจะต้องครอบคลุมคุณลักษณะต่างๆ ของระบบและมีจำนวนกรณีทดสอบที่ไม่มากจนเกินไป

ขั้นตอนวิธีเชิงพันธุกรรม (Genetic Algorithms) เป็นวิธีการค้นหาคำตอบที่เหมาะสมโดยใช้หลักการคัดเลือกแบบธรรมชาติจากการจำลองแนวคิดวิวัฒนาการของสิ่งมีชีวิต การนำขั้นตอนวิธีทางพันธุกรรมมาประยุกต์ใช้ในการสร้างกรณีทดสอบสำหรับกระบวนการทดสอบซอฟต์แวร์ จึงเป็นเรื่องที่น่าสนใจ เนื่องจากหลักการทำงานของขั้นตอนวิธีทางพันธุกรรม มีความสอดคล้องกับขั้นตอนการค้นหากรณีทดสอบและมีการปรับเปลี่ยนกรณีทดสอบ จนกระทั่งได้กรณีทดสอบที่เหมาะสมที่สุด สามารถช่วยให้ผู้พัฒนาระบบหรือผู้ทดสอบระบบสามารถทำงานได้ง่ายขึ้น ช่วยลดเวลาและต้นทุนในขั้นตอนการทดสอบซอฟต์แวร์

บทความนี้นำเสนอการสร้างกรณีทดสอบที่สามารถครอบคลุมการทำงานและเงื่อนไขการทำงานของโปรแกรมโดยประยุกต์ใช้ขั้นตอนวิธีเชิงพันธุกรรม

บทความนี้ประกอบด้วยหัวข้อต่างๆ ดังต่อไปนี้คือ หัวข้อที่ 2 เสนอทฤษฎีที่เกี่ยวข้อง หัวข้อที่ 3 เสนอวิธีการสร้างกรณีทดสอบโดยใช้ขั้นตอนวิธีเชิงพันธุกรรม หัวข้อที่ 4 เสนอกรณีศึกษา และหัวข้อที่ 5 สรุปผลและงานในอนาคต

2. ทฤษฎีที่เกี่ยวข้อง

ในส่วนนี้นำเสนอเกี่ยวกับทฤษฎีที่เกี่ยวข้อง ประกอบไปด้วยการทดสอบซอฟต์แวร์และขั้นตอนวิธีเชิงพันธุกรรม

2.1 การทดสอบซอฟต์แวร์ (Software Testing)

การทดสอบเป็นปัจจัยสำคัญปัจจัยหนึ่งของการประกันคุณภาพซอฟต์แวร์ดังนั้นในกระบวนการพัฒนาซอฟต์แวร์จึงต้องมีขั้นตอนการทดสอบซอฟต์แวร์ สำหรับเทคนิคในการทดสอบซอฟต์แวร์ (Software Testing Technique) ที่นิยมใช้กันแพร่หลายได้แก่

1. Black-Box Testing คือการทดสอบการทำงานของซอฟต์แวร์ที่ไม่สนใจกลไกภายในของระบบ การทำการทดสอบเน้นเพื่อการตรวจสอบผลการทำงานของระบบในแต่ละหน้าที่ตามข้อกำหนดความต้องการ (Requirement Specification) ว่าถูกต้องตามความต้องการหรือไม่

2. White-Box Testing คือการทดสอบการทำงานของซอฟต์แวร์ ซึ่งพิจารณากลไกภายในของระบบโดยจะมุ่งเน้นพิจารณาโครงสร้างภายใน

2.2 ขั้นตอนวิธีเชิงพันธุกรรม (Genetic Algorithms)

ขั้นตอนวิธีเชิงพันธุกรรม เป็นกระบวนการหนึ่งที่อาศัยทฤษฎีการวิวัฒนาการของสิ่งมีชีวิต นำเสนอโดย John Holland [2] สามารถนำมาใช้เพื่อค้นหาคำตอบที่ดีขึ้นและเหมาะสมมากขึ้นโดยผ่านการเรียนรู้ด้วยตัวเอง เปรียบเสมือนการวิวัฒนาการของสิ่งมีชีวิตที่สามารถวิวัฒนาการเพื่อการดำรงชีวิตอยู่ กระบวนการทำงานของขั้นตอนวิธีเชิงพันธุกรรมจะเริ่มจากการสุ่มโครโมโซมขึ้นมาจำนวนหนึ่งเพื่อสร้างเป็นกลุ่มประชากรต้นกำเนิด (Initial Population) และทำการประเมินค่าความเหมาะสม (Fitness Evaluation) จากนั้นจะคัดเลือกประชากร (Selection) มาจำนวนหนึ่งเพื่อใช้เป็นต้นกำเนิดทางสายพันธุ์หรือเป็นกลุ่มประชากรรุ่นพ่อแม่ จากนั้นจะนำประชากรเหล่านี้เข้าสู่กระบวนการปฏิบัติการทางสายพันธุ์ (Genetic Operation) ได้แก่ การไขว้เปลี่ยน (Crossover) และการกลายพันธุ์ (Mutation) ซึ่งจะทำให้เกิดกลุ่มประชากรที่เป็นรุ่นลูก และเมื่อนำประชากรกลุ่มนี้ไปประเมินค่าความเหมาะสมแล้วพบคำตอบที่เหมาะสมกับปัญหา กระบวนการทำงานก็จะจบลง แต่หากคำตอบที่ได้ยังไม่เหมาะสม ก็จะดำเนินต่อไปโดยการนำประชากรรุ่นลูกที่ได้ไปแทนที่ (Replacement) ประชากรรุ่นพ่อแม่ เมื่อทำการแทนที่เสร็จแล้วก็จะนำประชากรกลุ่มใหม่ที่ได้กลับเข้าสู่กระบวนการคัดเลือกใหม่อีกครั้ง ซึ่งกระบวนการเหล่านี้จะดำเนินการซ้ำไปเรื่อยๆจนกระทั่งได้ประชากรกลุ่มที่เหมาะสม

3. การสร้างกรณีทดสอบโดยใช้ขั้นตอนวิธีเชิงพันธุกรรม

ขั้นตอนการสร้างกรณีทดสอบโดยใช้ขั้นตอนวิธีเชิงพันธุกรรม แสดงดังภาพที่ 1 ประกอบด้วย 2 ขั้นตอนหลัก คือ

1) การสร้างกราฟควบคุมการไหลและระบุเส้นทางเป้าหมาย (Generating Control Flow Graph and Specifying Target Paths)

2) การสร้างกรณีทดสอบ (Generating Test Cases)

3.1 การสร้างกราฟควบคุมการไหลและระบุเส้นทางเป้าหมาย

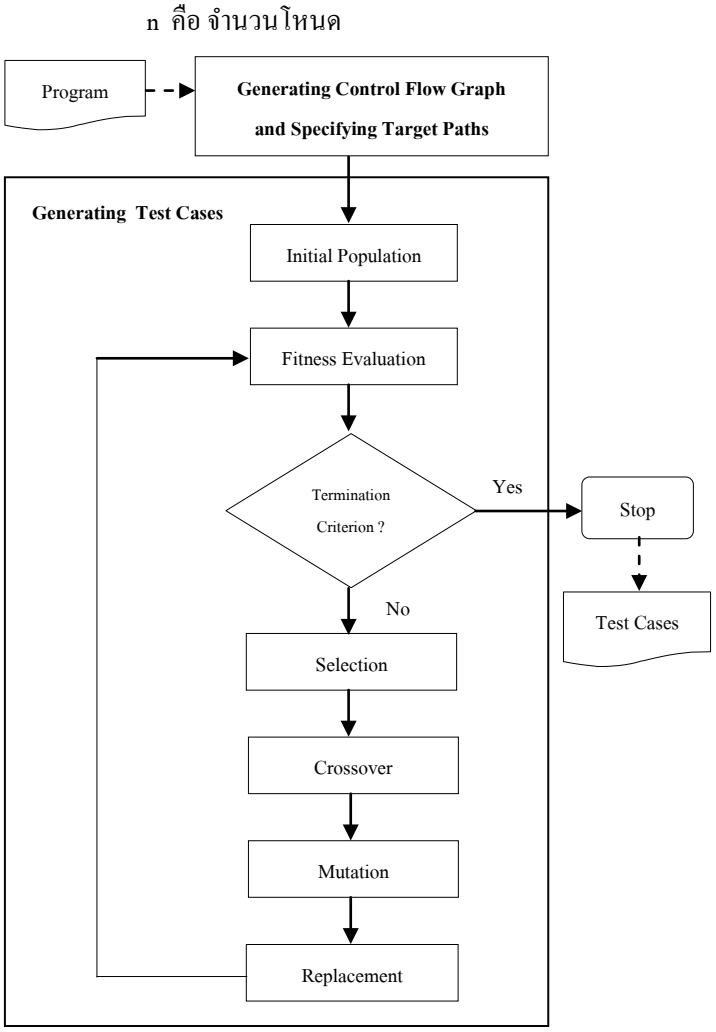
การสร้างกราฟควบคุมการไหล พิจารณาจากรหัสต้นทาง (Source Code) ของโปรแกรมที่ต้องการทดสอบ กราฟควบคุมการไหลสร้างขึ้นตามหลักการที่ได้นำเสนอโดย Amman และ Offutt [3] ภาพที่ 2 แสดงตัวอย่างรหัสต้นทางของโปรแกรม Triangle ที่ต้องการทดสอบ ภาพที่ 3 แสดงกราฟควบคุมการไหลของโปรแกรม Triangle

กราฟควบคุมการไหลจากขั้นตอนข้างต้น จะถูกนำมาใช้เพื่อระบุเส้นทางเป้าหมาย โดยจะคำนวณจำนวนเส้นทางจากการคำนวณค่าความซับซ้อนไซโคลมาติก (Cyclomatic Complexity) [4] ซึ่งใช้ในการวัดความซับซ้อนของโปรแกรมเป็นตัวบ่งชี้ว่าโปรแกรมนั้นมีความยากต่อการบำรุงรักษา และการทดสอบมากน้อยเพียงใด โดยสามารถคำนวณได้ดังนี้

$$V(G) = e-n+2 \quad (1)$$

โดยที่ V(G) คือ ค่าความซับซ้อนไซโคลมาติก

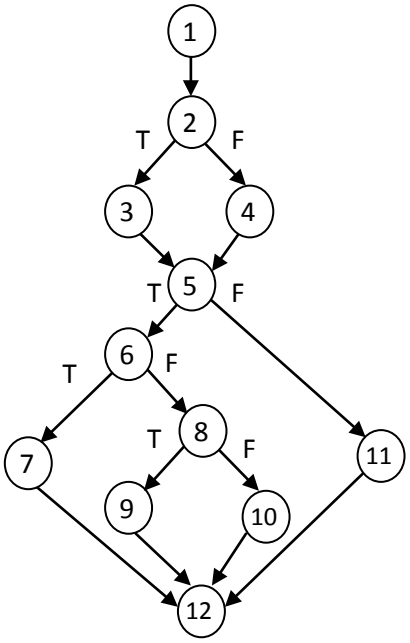
e คือ จำนวนเส้นเชื่อมต่อระหว่างโหนด



ภาพที่ 1 : แผนภาพขั้นตอนการสร้างกรณีทดสอบโดยใช้ขั้นตอนวิธีเชิงพันธุกรรม

```
1    Input (a,b,c) as Integer
2    If (a<b + c) And (b<a + c) And (c<b + a) Then
3       Is_Triangle = True
     Else
4       Is_Triangle = False
     End If
5    If Is_Triangle Then
6       If (a = b) And (b = c) Then
7            output = "Equilateral"
8    ElseIf (a<>b) And (b<>c) And (a<>c) Then
9            output = "Scalene"
        Else
10           output = "Isosceles"
        End If
     Else
11      output = "Not a Triangle"
12   End If
```

ภาพที่ 2 : แสดงตัวอย่างรหัสต้นทางของโปรแกรม Triangle



ภาพที่ 3 : แสดงกราฟควบคุมการไหลของโปรแกรม Triangle

3.2 การสร้างกรณีทดสอบ

การสร้างกรณีทดสอบถูกดำเนินการโดยใช้กระบวนการขั้นตอนวิธีเชิงพันธุกรรม ประกอบด้วยขั้นตอนดังนี้

ขั้นตอนที่ 1 การสร้างกลุ่มประชากรต้นกำเนิด (Initial Population)

กลุ่มประชากรต้นกำเนิดจะถูกสร้างขึ้นโดยใช้วิธีการแบ่งส่วนที่สมดุล (Equivalence Partitioning) ข้อมูลนำเข้าของโปรแกรมที่ต้องการทดสอบแต่ละตัวจะถูกแบ่งส่วนออกเป็นกลุ่มๆ ประชากรต้นกำเนิดหนึ่งประชากร ได้จากการนำส่วนใดส่วนหนึ่งของข้อมูลนำเข้าทุกตัวมารวมกัน ดังนั้นหนึ่งประชากรต้นกำเนิดหมายถึง หนึ่งชุดข้อมูลทดสอบ ซึ่งจำนวนโครโมโซมของหนึ่งประชากรจะเท่ากับจำนวนตัวแปรของหนึ่งชุดข้อมูลทดสอบ

ขั้นตอนที่ 2 การประเมินค่าความเหมาะสม (Fitness Evaluation)

การประเมินค่าความเหมาะสมจะพิจารณาจากการประเมินค่าความครอบคลุมของชุดข้อมูลที่ใช้ในการทดสอบ สำหรับบทความนี้ การประเมินค่าความครอบคลุมใช้วิธีการคำนวณหาค่า distance ของคำสั่งที่เป็นเงื่อนไขในการทำงานของโปรแกรม ตามทฤษฎี Korel's distance function [5] ดังแสดงในตารางที่ 1

ตารางที่ 1 : Korel's distance function

No	Predicate	Distance if path taken
1	A=B	ABS(A-B)
2	A≠B	K
3	A<B	(A-B)+K
4	A≤B	(A-B)
5	A>B	(B-A)+K
6	A≥B	(B-A)
7	X OR Y	MIN(Distance(X),Distance(Y))
8	X AND Y	(Distance(X) + Distance(Y)

ค่าความเหมาะสม สามารถคำนวณจากค่าผลต่างระหว่าง distance ดังนี้

$$\text{Fitness function} = \{ \text{dist(TG)} \} - \{ \text{dist(TR)} \} \quad (2)$$

โดยที่ dist(TG) คือ distance ของเส้นทางเป้าหมาย (Target Path)

dist(TR) คือ distance ของเส้นทางการเดินทางที่เกิดจากชุดข้อมูลทดสอบ (Traversal Path)

ค่าความเหมาะสมที่ได้ สามารถประเมินค่าความครอบคลุมได้ ดังนี้

$$\text{Fitness function} = \begin{cases} 0 & ;\text{covered} \\ \text{otherwise} & ;\text{not covered} \end{cases}$$

กรณีที่ค่าความเหมาะสม มีค่าเท่ากับศูนย์ แสดงว่าชุดข้อมูลทดสอบนั้นสามารถใช้เป็นกรณีทดสอบสำหรับเส้นทางเป้าหมายที่กำลังพิจารณา ในกรณีที่ค่าความเหมาะสม มีค่าไม่เท่ากับศูนย์ แสดงว่าชุดข้อมูลทดสอบนั้นยังไม่ครอบคลุมเส้นทางเป้าหมาย

เส้นทางเป้าหมายต่างๆ จะถูกตรวจสอบ โดยจะทำการตรวจสอบว่าชุดข้อมูลต่างๆ ครอบคลุมทุกเส้นทางเป้าหมายหรือไม่ กรณีที่ครอบคลุมทุกเส้นทางจะสิ้นสุดการทดสอบ

ขั้นตอนที่ 3 การคัดเลือก (Selection)

การคัดเลือกจะกระทำการคัดเลือกชุดข้อมูลเพื่อนำมาเป็นประชากรรุ่นพ่อแม่ โดยเลือกชุดข้อมูลที่ให้ค่าความเหมาะสมน้อย ซึ่งหมายถึงชุดข้อมูลที่ให้ค่าผลต่างระหว่าง distance น้อยที่สุด

ขั้นตอนที่ 4 การไขว้เปลี่ยน (Crossover)

การไขว้เปลี่ยนจะดำเนินการโดยเลือกชุดข้อมูลที่ให้ค่าผลต่างระหว่าง distance น้อยที่สุดมา 2 ชุดข้อมูล และทำการไขว้เปลี่ยน โดยใช้วิธีการไขว้เปลี่ยนแบบจุดเดียว (Single-Point Crossover) ซึ่งจะทำให้ได้ชุดข้อมูลใหม่ 2 ชุดข้อมูล ตัวอย่างเช่น ชุดข้อมูลที่ถูกเลือกมาคือ $\{x_1, x_2, x_3, x_4\}$ และ $\{y_1, y_2, y_3, y_4\}$ จะทำการไขว้เปลี่ยนชุดข้อมูลทั้งสอง ณ ตำแหน่งที่ 4 ดังนั้นจะได้ชุดข้อมูลใหม่ คือ $\{x_1, x_2, x_3, \mathbf{y_4}\}$ และ $\{y_1, y_2, y_3, \mathbf{x_4}\}$

ขั้นตอนที่ 5 การกลายพันธุ์ (Mutation)

การกลายพันธุ์จะกระทำโดยเลือกชุดข้อมูลที่ให้ค่าผลต่างระหว่าง distance น้อยที่สุด เพื่อทำการกลายพันธุ์แบบจุดเดียว (Single-Point Mutation) ซึ่งจะทำให้ได้ชุดข้อมูลใหม่ 1 ชุดข้อมูล ตัวอย่างเช่น ชุดข้อมูลที่ถูกเลือกมาคือ $\{x_1, x_2, x_3, x_4\}$ จะทำการกลายพันธุ์ ณ ตำแหน่งที่ 2 ด้วยค่า x_m ดังนั้นจะได้ชุดข้อมูลใหม่ คือ $\{x_1, \mathbf{x_m}, x_3, x_4\}$

โดยที่ตำแหน่งหรือโครโมโซมที่จะกระทำการไขว้เปลี่ยนในขั้นตอนที่ 4 และการกลายพันธุ์ในขั้นตอนที่ 5 จะได้จากการคัดเลือกโดยวิธีการสุ่ม เนื่องจากจำนวนตัวแปรหรือโครโมโซมของชุดข้อมูลมีจำนวนไม่มาก ดังนั้นการดำเนินการไขว้เปลี่ยนและการกลายพันธุ์จึงมีการกระทำแบบจุดเดียว

ขั้นตอนที่ 6 การแทนที่ (Replacement)

ชุดข้อมูลชุดใหม่ที่ได้จากขั้นตอนที่ 4 และ 5 จะถูกนำไปแทนที่ข้อมูลที่เป็นประชากรรุ่นพ่อแม่ และนำข้อมูลชุดใหม่ไปประเมินค่าความเหมาะสมในขั้นตอนที่ 2 อีกครั้ง

การกระทำขั้นตอนที่ 1 – 6 จะถูกกระทำซ้ำๆ โดยจำนวนรุ่นของประชากรจะเท่ากับจำนวนรอบการทำซ้ำ โดยกำหนดให้มีการทำซ้ำสูงสุด 20 รอบ

4. กรณีศึกษา

บทความนี้ เลือกใช้โปรแกรม Triangle เพื่อแสดงการประยุกต์ใช้ขั้นตอนการหากรณีทดสอบที่ได้นำเสนอ

จากกราฟควบคุมการไหลของโปรแกรม Triangle ดังแสดงในภาพที่ 3 สามารถคำนวณหาค่าความซับซ้อนไซโคลมาติกได้เท่ากับ 5 ดังนั้นเส้นทางในการทดสอบโปรแกรมมี 5 เส้นทาง ซึ่งสามารถระบุเส้นทางเป้าหมายทั้ง 5 เส้นทางได้ดังตารางที่ 2

ตารางที่ 2 : เส้นทางสำหรับทดสอบโปรแกรม Triangle

Path	The description of target path
P1	1-2T-3-5T-6T-7-12
P2	1-2T-3-5T-6F-8T-9-12
P3	1-2T-3-5T-6F-8F-10-12
P4	1-2F-4-5F-11-12
P5	1-2T-3-5F-11-12

กลุ่มประชากรต้นกำเนิด ซึ่งเป็นกลุ่มของชุดข้อมูลเริ่มต้นจะถูกสร้างขึ้นโดยใช้วิธีแยกส่วนที่สมดุล (Equivalence Partitioning) สำหรับโปรแกรม Triangle มีข้อมูลนำเข้าเป็นจำนวนเต็ม 3 จำนวน (a , b , c) โดยจำนวนเต็มสามารถแบ่งส่วน (Partition) ของข้อมูลออกเป็น 3 กลุ่ม คือ มากกว่า , น้อยกว่า และเท่ากับศูนย์

การกำหนดกลุ่มของชุดข้อมูลเริ่มต้น จะกำหนดตามหลักการ Each Choice (EC) ที่ถูกนำเสนอโดย Amman และ Offutt [3] โดยหลักการนี้กำหนดให้แต่ละส่วนที่ถูกแบ่ง จะต้องถูกใช้อย่างน้อย 1 ครั้งในการทดสอบ ดังนั้น ชุดข้อมูลที่ใช้สำหรับโปรแกรม Triangle จึงประกอบด้วย 3 ชุดข้อมูล คือ {a<0,b<0,c<0} , {a=0,b=0,c=0} และ {a>0,b>0,c>0} จากนั้นทำการสุ่มค่าของข้อมูลที่ใช้สำหรับทดสอบแสดงดังตารางที่ 3

ตารางที่ 3 : ชุดข้อมูลที่เบื้องต้น สำหรับการทดสอบโปรแกรม Triangle

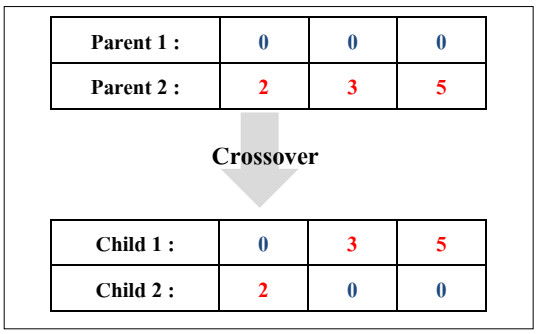
ชุดข้อมูล / ตัวแปร	a	b	c
1	-3	-2	-5
2	0	0	0
3	2	3	5

การประเมินค่าความครอบคลุมของชุดข้อมูลที่ใช้ในการทดสอบถูกกระทำโดยวิธีการคำนวณหาค่า distance ของคำสั่งที่เป็นเงื่อนไขตามทฤษฎี Korel's distance function โดยเส้นทางเป้าหมายแต่ละเส้นทางจะถูกนำมาพิจารณาเพื่อหาชุดข้อมูลที่เหมาะสมสำหรับการทดสอบเส้นทางนั้นๆ ในกรณีที่ยังไม่มีชุดข้อมูลที่เหมาะสมสำหรับเส้นทางใด เส้นทางนั้นจะถูกดำเนินการพิจารณาต่อไป

ตัวอย่างเช่น จากตารางที่ 2 กำหนดให้ P4 เป็นเส้นทางเป้าหมายสำหรับการหากรณีทดสอบ ดังนั้นคำสั่งที่เป็นเงื่อนไขในเส้นทาง P4 ประกอบด้วย {2F ,5F} พิจารณาชุดข้อมูลทั้ง 3 ในตารางที่ 3 นำแต่ละชุดข้อมูลมาคำนวณหาผลต่างระหว่าง distance ของเส้นทางการเดินทางของข้อมูลชุดนั้น กับเส้นทางเป้าหมาย P4 จะพบว่า ผลต่างระหว่าง distance ของเส้นทางการเดินทางของข้อมูลชุดที่ 1 กับเส้นทางเป้าหมาย P4 มีค่าเท่ากับศูนย์ แสดงว่าข้อมูลชุดที่ 1 มีความครอบคลุมเส้นทางเป้าหมาย ดังนั้นกระบวนการหากรณีทดสอบสำหรับเส้นทางเป้าหมาย P4 ก็จะจบลง

ตัวอย่างกำหนดให้ P1 เป็นเส้นทางเป้าหมายสำหรับการหากรณีทดสอบคำสั่งที่เป็นเงื่อนไขในเส้นทาง P1ประกอบด้วย {2T,5T,6T} เมื่อคำนวณหาผลต่างระหว่าง distance โดยใช้ชุดข้อมูลทั้ง 3 ในตารางที่ 3 จะพบว่าไม่มีข้อมูลชุดใดให้ค่าผลต่างระหว่าง distance เท่ากับศูนย์ ดังนั้นแสดงว่ายังไม่มีข้อมูลชุดใดครอบคลุมเส้นทางเป้าหมาย จึงต้องมีการดำเนินการต่อไปโดยการไขว้เปลี่ยนและการกลายพันธุ์

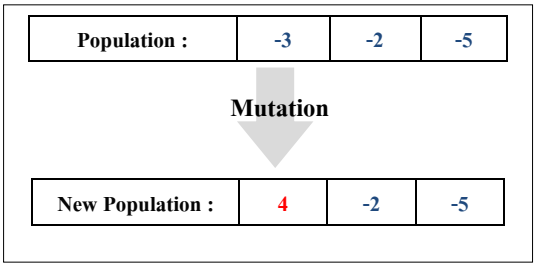
ในการไขว้เปลี่ยน จะเลือกชุดข้อมูลที่ทำให้ค่าผลต่างระหว่าง distance น้อยที่สุด 2 อันดับ และสุ่มเลือกตำแหน่งในการไขว้เปลี่ยนเพื่อให้ได้ข้อมูลชุดใหม่สำหรับการทดสอบในรอบต่อไป ตัวอย่างการไขว้เปลี่ยน แสดงดังภาพที่ 4



ภาพที่ 4 : การไขว้เปลี่ยนชุดข้อมูลแบบจุดเดียว (Single-Point

Crossover)

กรณีที่ทำการไขว้เปลี่ยนแล้ว แต่กรณีทดสอบยังไม่ครอบคลุม จะกระทำการกลายพันธุ์ โดยเลือกชุดข้อมูลที่ให้ค่าผลต่างระหว่าง distance น้อยที่สุด และสุ่มเลือกตำแหน่งในการกลายพันธุ์ ตัวอย่างการกลายพันธุ์แสดงดังภาพที่ 5



ภาพที่ 5 : การกลายพันธุ์ชุดข้อมูลแบบจุดเดียว (Single-Point Mutation)

นำชุดข้อมูลใหม่ที่ได้จากขั้นตอนที่ 4 และ 5 มาเป็นชุดข้อมูลใหม่สำหรับการหากรณีทดสอบในรอบถัดไป

กระบวนการค้นหาชุดข้อมูลสำหรับเส้นทางเป้าหมายจะถูกกระทำซ้ำๆ การทำงานจะจบลงเมื่อสามารถหาชุดข้อมูลสำหรับใช้เป็นกรณีทดสอบสำหรับทุกๆเส้นทางเป้าหมาย ตารางที่ 4 แสดงกรณีทดสอบที่ครอบคลุมเส้นทางเป้าหมายทั้งหมดสำหรับการทดสอบโปรแกรม Triangle

ตารางที่ 4 : กรณีทดสอบที่ครอบคลุมเส้นทางเป้าหมายสำหรับการทดสอบโปรแกรม Triangle

Target Path	Test Case	Input values			No. of GA
		a	b	c	
P1	1	2	2	2	5
P2	2	2	4	5	6
P3	3	4	4	2	7
P4	4	-3	-2	-5	7

ตารางที่ 4 แสดงกรณีทดสอบที่ครอบคลุมเส้นทางทั้งหมดสำหรับการทดสอบโปรแกรม Triangle มีทั้งหมด 4 กรณีทดสอบ จำนวนรอบการดำเนินการขั้นตอนวิธีเชิงพันธุกรรม (No. of GA) เพื่อค้นหากรณีทดสอบที่ครอบคลุมเส้นทางเป้าหมาย P1 ถึง P4 มีจำนวน 5, 6, 7 และ 7 รอบตามลำดับ สำหรับเส้นทางเป้าหมาย P5 ไม่สามารถค้นหากรณีทดสอบที่ครอบคลุมได้ โดยใช้จำนวนรอบในการค้นหา 20 รอบซึ่งเป็นจำนวนรอบสูงสุดที่กำหนดไว้ จากการวิเคราะห์เส้นทาง P5 พบว่าเป็นเส้นทางที่เป็นไปไม่ได้ (infeasible path) ซึ่งเป็นเส้นทางที่ไม่มีกรณีทดสอบใดๆที่สามารถผ่านเส้นทางนี้ได้ ดังนั้นจึงสอดคล้องกับผลที่ได้รับจากขั้นตอนที่ได้นำเสนอ

5. สรุปผลและงานในอนาคต

บทความนี้เสนอการสร้างกรณีทดสอบโดยใช้ขั้นตอนวิธีเชิงพันธุกรรมเพื่อคัดเลือกกรณีทดสอบที่เหมาะสมสำหรับการทดสอบซอฟต์แวร์ การดำเนินการเริ่มจากนำรหัสต้นทางของโปรแกรมที่ต้องการทดสอบมาแปลงเป็นกราฟควบคุมการไหล ต่อจากนั้นจะค้นหากรณีทดสอบที่ครอบคลุมทุกเส้นทางการทำงานของโปรแกรมโดยใช้ขั้นตอนวิธีเชิงพันธุกรรม เมื่อนำวิธีการที่ได้นำเสนอไปประยุกต์ใช้กับกรณีศึกษาพบว่า ขั้นตอนวิธีที่นำเสนอสามารถสร้างกรณีทดสอบสำหรับการทดสอบซอฟต์แวร์ได้อย่างมีประสิทธิภาพ

ในอนาคตได้วางแผนที่จะพัฒนาเครื่องมือเพื่อสร้างกรณีทดสอบจากวิธีการที่นำเสนอและนำไปประยุกต์ใช้กับกรณีศึกษาอื่นๆ เพื่อประเมินค่าความถูกต้องของกรณีทดสอบและพัฒนาวิธีการจัดการกับการหากรณีทดสอบสำหรับเส้นทางที่เป็นไปไม่ได้

6. เอกสารอ้างอิง

[1] C. Ghezzi, M. Jazayeri, and D. Mandrioli. “Fundamentals of Software Engineering 2nd,” *Prentice Hall PTR Upper Saddle River, NJ, USA*, 2002.

[2] G. Winter, J. Periaux, M.Galan and P.Cuesta. “Genetic Algorithms in Engineering and Computer Science,” *Wiley Publisher*, 1996.

[3] P. Ammann and J. Offutt. “Introduction to Software Testing,” *Cambridge University Press*, 2008.

[4] T. J. McCabe. “A Complexity Measure,” *IEEE Transactions on Software Engineering*, pp. 308-320, 1976.

[5] B. Korel. “Automated Test Data Generation,” *IEEE Transactions on Software Engineering*, pp.870-879, 1990.