การเปรียบเทียบระหว่าง Tournament Selection และ Combined Rank Method สำหรับ Mimetic Evolvable Hardware

Comparison Between Tournament Selection and Conbined Rank Method for A Mimetic Evolvable Hardware

ประพนธ์ บวรภราดร และ ประภาส จงสถิตย์วัฒนา
Prapon Bavonparadon and Prabhas Chongstitvatana
Department of Computer Engineering
Chulalongkorn University
Bangkok 10330, Thailand
E-Mail: 00279921@student.chula.ac.th, prabhas@chula.ac.th

บทคัดย่อ: ในบทความฉบับนี้ นำเสนอการใช้วิธี tournament selection เพื่อใช้ใน mimetic evolvable hardware ซึ่งสามารถเลียนแบบวงจรเป้าหมายได้ โดยสังเกตลำดับของ อินพุต/เอาต์พุต ก่อนหน้านี้มีผู้ทำการวิจัยโดยใช้ขั้นตอนวิธีพันธุกรรม แบบที่ใช้วิธี combined rank method แต่เนื่องจาก วิธีดังกล่าวค่อนข้างซับซ้อน จึงต้องสร้างไมโครโปรเซสเซอร์ที่ทำงานตามขั้นตอนวิธีพันธุกรรม ในบทความฉบับนี้จึงนำเสนอวิธีการ tournament selection และในบทความนี้ได้ทำการทดลองเปรียบเทียบ ความพยายามเชิงคำนวณ ที่ใช้ของวิธี tournament selection และวิธี combined rank method ผลการทดลองสรุปได้ว่า วิธีการ tournament selection ใช้ ความพยายามเชิงคำนวณไม่มากไปกว่าวิธีการ combined rank method

Abstact: We propose tournament selection method for a mimetic evolvable hardware that can mimic a target circuit by observing partial input/output sequences.The earlier work of a mimetic evolvable hardware used combined rank method. However, this method is rather complex so it was implemented on custom microprocessor. In this paper, we experience for compare computational effort between tournament selection and combined rank method. The result shows that computational effort of tournament selection is not more than combined rank method.

Keyword: Mimetic evolvable hardware, Tournament selection, Combined rank method, Genetic algorithm

1. บทนำ

การออกแบบวงจรตรรกเชิงลำดับตามวิธีการที่ใช้ในปัจจุบัน จำเป็นจะต้องรู้ลักษณะการทำงานของวงจรที่ต้องการก่อนออกแบบ และเมื่อลักษณะการทำงานของวงจรที่ต้องการเปลี่ยนแปลงไป ต้องทำการออกแบบใหม่ จึงเกิดแนวคิดที่จะสร้างวงจรที่สามารถปรับปรุงลักษณะการทำงานแบบอัตโนมัติ ซึ่งทำให้ประหยัดเวลาในการออกแบบวงจรใหม่ และวงจรสามารถปรับเปลี่ยนลักษณะการทำงานตามสภาพแวดล้อมได้ วงจรลักษณะดังกล่าวอาจเรียกอีกชื่อหนึ่งว่า evolvable hardware (EHW)

ในบทความฉบับนี้จะเน้นกล่าวถึงวิธีการสังเคราะห์วงจรตรรกเชิงลำดับ โดยการสังเกตลำดับของ อินพุต/เอาต์พุต ของวงจรเป้าหมายโดยใช้ขั้นตอนวิธีพันธุกรรมเพื่อค้นหาวงจรที่ต้องการ ซึ่งใน [1] ได้ใช้วิธีดังกล่าวเพื่อสังเคราะห์วงจรตรรกเชิงลำดับ สำหรับ programmable array logic (PAL) โดยให้ประชากรแต่ละชุดเป็น bit pattern ของ PAL ผลการทดลองแสดงให้เห็นว่า การสังเคราะห์วงจรตรรกเชิงลำดับ โดยการสังเกตลำดับของอินพุต/เอาต์พุต ของวงจรเป้าหมายโดยใช้ขั้นตอนวิธีพันธุกรรมสามารถสังเคราะห์วงจรที่ทำงานได้ถูกต้องตามต้องการ

[2] ได้ทำการทดลองเปรียบเทียบความพยายามเชิงคำนวณของการสังเคราะห์วงจรตรรกเชิงลำดับ ที่มี การแทนแตกต่างกัน 2 แบบ โดยการสังเคราะห์วงจรใช้ขั้นตอนวิธีพันธุกรรม การแทนที่แตกต่างกัน 2 แบบนั้น คือ แบบ technology-based (ประชากร แต่ละชุดเป็น bit pattern ของ PAL) และแบบ state-based (ประชากรแต่ละ

ชุดเป็น bit string ที่ใช้แทน state transition table) ผลการทดลองแสดงว่า จากลำดับของ อินพุต/เอาต์พุต สามารถสังเคราะห์ state transition table ได้ แต่ใช้ความพยายามมากกว่าแบบ technology-based อย่างไรก็ตาม จากการทดลองเป็นการแสดงให้เห็นว่าสามารถใช้ขั้นตอนวิธีพันธุกรรมสังเคราะห์ state transtition table ได้ ซึ่งข้อดีคือ สามารถนำไปสร้างบนเทคโนโลยีใดๆก็ได้

[3] ได้นำเสนอ mimetic evolvable hardware ซึ่งสามารถเลียนแบบวงจรเป้าหมายได้โดยสังเกตลำดับของอินพุต/เอาต์พุต ในงานฉบับดังกล่าวได้แสดงให้เห็นว่าส่วนที่คำนวณค่าความเหมาะสมของประชากรแต่ละชุดเป็นส่วนที่ใช้เวลาในการคำนวณมากที่สุด ในงานฉบับนั้นจึงสร้าง mimetic evolvable hardware เป็น hardware จริงๆ และสร้างส่วนคำนวณค่าความเหมาะสมเป็นวงจรต่างหากจากส่วนที่ประมวลผลขั้นตอนวิธีพันธุกรรม ผลการทดลองแสดงให้เห็นว่า ส่วนคำนวณค่าความเหมาะสมที่เป็น hardware ทำงานได้เร็วกว่ารุ่นที่เป็น software ถึง 31 เท่า

แต่ในงานฉบับดังกล่าว [3] ขั้นตอนวิธีพันธุกรรมในส่วนที่เลือกประชากรชุดใหม่ ใช้วิธีการ combined rank method ซึ่งจะเลือกชุดที่มีค่าความเหมาะสมสูงสุดและแตกต่างจากประชากรชุดอื่นๆมากที่สุด จึงจำเป็นต้องทำการเรียงข้อมูลใหม่ทุกครั้งก่อนที่จะทำการเลือก ขั้นตอนวิธีที่ใช้จึงค่อนข้างซับซ้อนและใช้เวลาทำงานมาก และเนื่องจากความซับซ้อนของส่วนการเลือกประชากรชุดใหม่จึงไม่สามารถสร้างขึ้นเป็นวงจรโดยตรง ทำให้ต้องสร้างไมโครโปรเซสเซอร์เพื่อใช้ทำงานตามขั้นตอนวิธีพันธุกรรมแทน

จากความซับซ้อนของวิธีการ combined rank method จึงเป็นจุดเริ่มต้นของความคิดที่จะหาวิธีการ selection ที่ไม่ซับซ้อน และสามารถให้คำตอบได้เช่นกัน เพื่อให้สามารถสร้างเป็นวงจรที่สามารถทำงานตามขั้นตอนวิธีพันธุกรรมได้โดยตรง ไม่ต้องสร้างเป็นไมโครโปรเซสเซอร์เพื่อความรวดเร็วในการทำงาน

ในบทความฉบับนี้ได้เลือกวิธีการ tournament selection เพื่อใช้ในขั้นตอนวิธีพันธุกรรม เนื่องจากเป็นวิธีการที่ไม่ซับซ้อน ในบทความฉบับนี้ จะทำการเปรียบเทียบระหว่าง tournament selection และ combined rank method เพื่อทดลองดูว่าสามารถใช้วิธีการ tournament selection กับปัญหานี้ได้หรือไม่

ในส่วนที่เหลือของบทความฉบับนี้ จะกล่าวถึงขั้นตอนวิธีพันธุกรรมที่ใช้ใน [3], ขั้นตอนวิธีพันธุกรรมแบบที่ใช้ tournament selection, การวิเคราะห์เปรียบเทียบขั้นตอนวิธีพันธุกรรมที่ใช้ tournament selection และ combined rank method,วิธีการทดลอง,ผลการทดลอง, สรุป และรายการอ้างอิง

2. ขั้นตอนวิธีพันธุกรรมที่ใช้ใน [3]

ขั้นตอนวิธีพันธุกรรมที่ใช้ใน [3] จะใช้ขนาดประชากรค่อนข้างเล็กเพื่อความเหมาะสมกับการสร้างเป็น hardware ซึ่งมีความจำกัดในเรื่องหน่วยความจำ และเนื่องจากขนาดประชากรที่เล็กจึงต้องใช้จำนวนรุ่นที่มากแทนเพื่อให้สามารถหาคำตอบได้พบ ขั้นตอนวิธีที่ใช้เป็นดังนี้

```
generation = 0;
Initialize P individuals;
While (terminate condition not met) and
(generation < MAXGEN) DO
        Produce Q individual using crossover;
        Produce R individual using mutation;
        Select P individual from (P U Q U R);
        generation = generation +1;
End While
```

จำนวนรุ่นที่มากที่สุดกำหนดไว้เป็น 50,000 ค่าของ P,Q,R ถูกกำหนดเป็น 128,256,128 ตามลำดับ และตัวปฏิบัติการพันธุกรรม crossover,mutation และ selection ถูกกำหนดดังนี้

- Crossover: เลือก binary string โดยการสุ่มมา 1 คู่ จาก P เพื่อสร้าง binary string ใหม่ 1 คู่ โดยการใช้ single point crossover
- Mutation: เลือก binary string โดยการสุ่มมา 1 ชุด จาก P แล้ว mutate เพื่อสร้าง binary string ใหม่โดย $P_m = 0.01$
- Selection: เลือกประชากรที่ดีที่สุด P ชุดจาก (P U Q UR) เพื่อเป็นประชากรในรุ่นถัดไป โดยใช้ combined rank method (fitness rank + diversity rank)

2.1 การเข้ารหัสประชากร

ประชากรแต่ละตัวจะแทน finite-state machine (FSM) ซึ่งประกอบด้วย state transition table และฟังก์ชันที่ทำการ mapping อินพุต และ current state ไปเป็นเอาต์พุต และจากการที่ FSM สามารถสร้างเป็นตารางที่ใช้แสดง state transition และ ฟังก์ชันเอาต์พุต ทำ

ให้สามารถเก็บประชากรแต่ละตัวใน Random Access Memory (RAM)

ประชากรแต่ละตัวถูกแทนโดยการนำ next state และเอาต์พุตมาต่อกันเพื่อสร้างเป็น fixed-length binary string ตาราง 1 แสดงตัวอย่าง FSM ที่ถูกเข้ารหัสโดยการนำ next state และเอาต์พุตของทุกๆแถวมาต่อเข้าด้วยกันได้เป็น “00111100”

State	Input	Next state	Output
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

ตาราง 1: ตัวอย่างของ FSM

สำหรับการประยุกต์ใช้งานจริงนั้น จำนวนของ internal state ที่จำเป็นสำหรับ complete solution เราอาจจะไม่รู้ ดังนั้นจึงจำเป็นจะต้องกำหนดจำนวนของ state ของประชากรแต่ละตัวให้ใหญ่กว่าจำนวน state ที่ใช้ในวงจรเป้าหมาย ดังนั้นจึงอาจเกิด redundant state และ unreachable state ขึ้น

2.2 Selection

วิธีการจำนวนมากถูกเสนอเพื่อเพิ่มประสิทธิภาพของการสร้างใน hardware อย่างไรก็ตาม เราก็ยังบอกไม่ได้ว่าวิธีการเหล่านั้นจะมีประสิทธิภาพสำหรับทุกๆปัญหาหรือไม่ จากการใช้ Simple GA และ steady-state GA กับปัญหานี้พบว่าเกิด pre-mature convergence (การลู่เข้าสู่ local maxima ซึ่งไม่ใช่คำตอบ) ดังนั้นความหลากหลายจึงควรถูกรักษาไว้ในกระบวนการ selection

วิธี Selection ที่เลือกใช้ในงานดังกล่าวคือ combined rank method (fitness rank + diversity rank) ซึ่งมีลักษณะดังนี้

- ขั้นแรก ประชากรตัวที่ดีที่สุดจะถูกเลือก
- ต่อมา นำประชากรที่เหลือมาเรียงโดยใช้ค่าความเหมาะสม และเรียงโดยใช้ความหลากหลาย ซึ่งความหลากหลายวัดโดยนำประชากรตัวนั้นไปเปรียบเทียบกับประชากรตัวอื่นๆทุกตัว และความหลากหลายคือผลรวมจำนวนบิตที่แตกต่างกันกับประชากรตัวอื่นๆ
- ทำการเลือกประชากรที่มีอันดับที่ในการเรียงด้วยค่าความเหมาะสม บวกกับอันดับที่จากการเรียงโดยความหลากหลายน้อยที่สุด ไปเป็นประชากรรุ่นถัดไป
- กระบวนการดังกล่าว จะถูก ทำซ้ำ P–1 ครั้ง เมื่อ P เป็นขนาดประชากร

2.3 ฟังก์ชันความเหมาะสม

ลำดับอินพุต/เอาต์พุต ที่ใช้ประเมินค่าความเหมาะสมถูกสร้างโดยขั้นตอนดังนี้

1. reset วงจรเป้าหมายเพื่อให้อยู่ใน start state
2. สร้างลำดับอินพุตโดยการสุ่ม
3. ป้อนลำดับอินพุตให้กับวงจรและบันทึกลำดับเอาต์พุต
4. ทำซ้ำขั้นตอนที่ 1-4 เพื่อสร้างลำดับอินพุต/เอาต์พุตถัดไป (กรณีที่ต้องการสร้างหลายลำดับ)

ความเหมาะสมของประชากรแต่ละตัวถูกคำนวณจากขั้นตอนต่อไปนี้

1. ให้ fitness,F,=0
2. reset ประชากร(circuit) เพื่อให้อยู่ใน start state
3. ป้อนอินพุตให้กับประชากรและบันทึกเอาต์พุต
4. เปรียบเทียบเอาต์พุตของประชากร กับเอาต์พุตของวงจรเป้าหมายโดย F = F + จำนวนบิตที่เหมือนกัน
5. ทำซ้ำขั้นตอนที่ 2-4 สำหรับลำดับอินพุต/เอาต์พุตที่เหลือ

3. ขั้นตอนวิธีพันธุกรรมแบบที่ใช้ tournament selection

ขั้นตอนวิธีพันธุกรรมแบบที่ใช้ tournament selection ทำการปรับปรุงจากขั้นตอนวิธีพันธุกรรมแบบที่ใช้ combined rank method โดยขั้นตอนวิธีที่ใช้เป็นดังนี้

```
generation = 0;
Initial P individuals;
While (terminate condition not met) and (generation < MAXGEN) Do
    Produce Q individuals from P individuals using crossover
    Mutation every individuals in Q
    Copy every individuals from Q to P
    Evaluate fitness value for every individuals in P
    generation =generation + 1;
End While
```

จำนวนรุ่นที่มากที่สุดกำหนดไว้เป็น 50,000 เท่ากับในแบบที่ใช้ combined rank method และค่าของ P,Q เป็น 512 เท่ากัน (เท่ากับ P+Q+R ที่ใช้ในวิธีการ

combined rank method) ตัวปฏิบัติการพันธุกรรม crossover, mutation ถูกกำหนดดังนี้

- Crossover : ทำการเลือก binary string 1 คู่จาก P โดยวิธี tournament selection เพื่อสร้าง binary string ใหม่ 1 คู่ โดยการใช้วิธี single point crossover
- Mutation จะนำ binary string แต่ละชุดใน Q มาทำให้กลายพันธุ์ โดยความน่าจะเป็นที่จะ toggle ในแต่ละ bit จาก 0 เป็น 1 หรือ 1 เป็น 0, P_m, =0.01

ประชากรแต่ละตัวจะใช้แทน FSM โดยวิธีการเข้ารหัสเหมือนกับในแบบที่ใช้ combined rank method และใช้ฟังก์ชันความเหมาะสมแบบเดียวกันด้วย

tournament selection ที่ใช้มี tournament size = 3 โดยมีวิธีการดังนี้

- สุ่มประชากรจาก P เป็นจำนวนชุดเท่ากับ tournament size
- จากประชากรที่สุ่มขึ้นมาเลือกตัวที่มีค่าความเหมาะสมมากที่สุดเป็นตัวแทนเพื่อนำไปใช้ crossover ต่อไป

4.วิเคราะห์เปรียบเทียบขั้นตอนวิธีพันธุกรรมที่ใช้ tournament selection และ combined rank method

ในส่วนนี้จะทำการเปรียบเทียบความพยายามเชิงคำนวณ ที่ใช้ในขั้นตอนวิธีพันธุกรรมแต่ละรุ่น ของขั้นตอนวิธีพันธุกรรมที่ใช้ tournament selection และ combined rank method โดยนิยามให้

n คือ ขนาดประชากร

m คือ ความยาวของ binary string ที่ใช้แทน FSM ซึ่งคำนวณได้จากสูตร $m=2^{(i+a)}*(o+a)$

โดย i แทน จำนวนบิตของอินพุต,o แทน จำนวนบิตของเอาต์พุต,a แทน จำนวนบิตของ available internal state

l คือ ความยาวของลำดับอินพุต/เอาต์พุต

s คือ จำนวนลำดับอินพุต/เอาต์พุต

4.1 ขั้นตอนวิธีพันธุกรรมที่ใช้ combined rank method

จากขั้นตอนวิธีที่แสดงไว้ ในขั้นตอนวิธีพันธุกรรมแต่ละรุ่นจะแบ่งตัวปฏิบัติการได้เป็น 2 กลุ่ม คือ reproduction และ selection

Reproduction ซึ่งประกอบด้วย

- Crossover เพื่อสร้างประชากรจำนวน Q ตัว โดยการ crossover แต่ละครั้งใช้ single point crossover ซึ่งต้องทำการ copy binary string ทีละบิตซึ่งยาว m ดังนั้นความพยายามที่ใช้จึงเป็น *O(Q*m)*
- Mutation เพื่อสร้างประชากรจำนวน R ตัว โดยการกลายพันธุ์ binary string ทีละบิตซึ่งยาว m ดังนั้นความพยายามที่ใช้จึงเป็น *O(R*m)*
- เมื่อทำการสร้างประชากรชุดใหม่แล้ว จะต้องทำการคำนวณค่าความเหมาะสมซึ่งจะป้อนอินพุต เข้าไปทีละบิตแล้วดูผลลัพธ์ โดยลำดับของอินพุต/เอาต์พุตมีความยาว *l* และใช้จำนวนลำดับของอินพุต/เอาต์พุตเท่ากับ s ดังนั้นความพยายามที่ใช้จึงเป็น *O((Q+R) * (l*s))*

ดังนั้น ในส่วน reproduction สำหรับขั้นตอนวิธีพันธุกรรมแต่ละรุ่นความพยายามจึงเป็น

$$O\left((Q+R) * Max\{m, l*s\} \right)$$

Selection

ทำการเลือกประชากรที่เหมาะสม จำนวน P ชุด จาก (P U Q U R) เพื่อเป็นประชากรรุ่นต่อไป โดยจากขั้นตอนวิธีพันธุกรรมที่แสดงไว้แล้ว ในการเลือกประชากรแต่ละครั้ง จะต้องทำการเรียงข้อมูล 2 ครั้ง ซึ่งการเรียงข้อมูลใช้วิธี quick sort ดังนั้น ในการเลือกประชากรแต่ละครั้งความพยายามเป็น $O(n \log n)$ โดย n = P+Q+R

ดังนั้น ในการเลือกประชากร P ชุด ความพยายามที่ใช้จึงเป็น $O(P*n \log n)$ หรือ $O(n^2 \log n)$

สรุป จากส่วน reproduction ความพยายามเป็น $O((Q+R)*Max\{m,l*s\})$ หรือ $O(n * Max\{m,l*s\})$ และส่วน selection ความพยายามเป็น $O(n^2 \log n)$ ดังนั้นขั้นตอนวิธีพันธุกรรมแต่ละรุ่นมีความพยายามเชิงคำนวณเป็น $O(n * Max\{m,l*s\}) + O(n^2 \log n)$

4.2 ขั้นตอนวิธีพันธุกรรมที่ใช้ tournament selection

จากขั้นตอนวิธีที่ได้แสดงไว้ ในขั้นตอนวิธีพันธุกรรมแต่ละรุ่นจะมี operation ดังนี้

- crossover เพื่อสร้างประชากรใหม่ Q ชุด โดยเลือกประชากร 1 คู่ จาก P โดยวิธี tournament selection ซึ่งมีความพยายามเป็น *O(1)* และในการ crossover แต่ละครั้งใช้ single point crossover ซึ่งมีความพยายามเป็น *O(m)* ดังนั้น ความพยายามที่ใช้ในการสร้างประชากรใหม่ Q ตัวจึงเป็น *O (Q*m)*
- mutation โดยจะนำประชากร Q ตัวมาทำการกลายพันธุ์ซึ่งในการ mutation แต่ละครั้ง ความพยายามเป็น *O(m)* ดังนั้นในการ mutation ประชากรจำนวน Q ตัว ความพยายามที่ใช้จึงเป็น *O(Q*m)*

- การ copy ประชากรจาก Q ไป P ใช้ความพยายามเป็น $O(Q*m)$
- ส่วนคำนวณค่าความเหมาะสม ใช้วิธีการแบบเดียวกับ ขั้นตอนวิธีพันธุกรรมแบบที่ใช้ combined rank method ดังนั้น ความพยายามที่ใช้จึงเป็น $O(Q*(l*s))$

<u>สรุป</u> ในขั้นตอนวิธีพันธุกรรมแต่ละรุ่นใช้ความพยายามเชิงคำนวณเป็น $O(Q * Max\{m,l*s\})$ และในที่นี้ P และ Q เป็นขนาดของประชากรจึงแทน Q ด้วย n ดังนั้นความพยายามเชิงคำนวณเป็น $O(n * Max\{m,l*s\})$

4.3 เปรียบเทียบความพยายามเชิงคำนวณของทั้ง 2 วิธี

จากความพยายามเชิงคำนวณของขั้นตอนวิธีพันธุกรรมที่ใช้ combined rank method เป็น

$$O(n * Max\{m,l*s\}) + O(n^2 \log n)$$

และความพยายามเชิงคำนวณของขั้นตอนวิธีพันธุกรรมที่ใช้ tournament selection เป็น $O(n * Max\{m,l*s\})$

ในกรณีที่ค่า n มากพอที่จะละเลย m และ l*s ความพยายามเชิงคำนวณของวิธี combined rank method เป็น $O(n^2 \log n)$ ซึ่งวิธี tournament selection จะใช้ความพยายามเชิงคำนวณน้อยกว่า

แต่ในกรณีที่ค่า n น้อยกว่า m และ l*s ความพยายามเชิงคำนวณของวิธี combined rank method เป็น $O(n * Max\{m,l*s\})$ ซึ่งเท่ากับวิธี tournament selection

ดังนั้นจึงสรุปว่า ในแต่ละรุ่นของขั้นตอนวิธีพันธุกรรมแบบที่ใช้ tournament selection ใช้ความพยายามเชิงคำนวณไม่มากกว่าแบบ combined rank method

5. การทดลอง

ในบทความฉบับนี้ ต้องการจะทดสอบว่า ขั้นตอนวิธีพันธุกรรมแบบที่ใช้ tournament selection เมื่อเปรียบเทียบกับแบบที่ใช้ combined rank method จะมีความพยายามเชิงคำนวณแตกต่างกันอย่างไร

จากที่ได้แสดงให้เห็นแล้วว่า ในแต่ละรุ่นของขั้นตอนวิธีพันธุกรรม แบบที่ใช้ tournament selection ใช้ ความพยายามเชิงคำนวณไม่มากกว่าแบบที่ใช้วิธี combined rank method แต่เนื่องจากความพยายามเชิงคำนวณที่ใช้ทั้งหมดจะขึ้นกับจำนวนรุ่นก่อนพบคำตอบด้วย และในบทความนี้ยังไม่สามารถพิสูจน์เพื่อเปรียบเทียบได้ว่า ทั้ง 2 วิธีจะมีจำนวนรุ่นก่อนพบคำตอบแตกต่างกันอย่างไร ดังนั้นจึงใช้เวลาในการประมวลผลโปรแกรมซึ่งใช้ขั้นตอนวิธีพันธุกรรมที่แตกต่างกันทั้ง 2 แบบ เป็นเครื่องมือวัดความพยายามเชิงคำนวณแทน

ในการทดสอบเพื่อวัดเวลาที่ใช้ประมวลผลโปรแกรมขั้นตอนวิธีพันธุกรรมทั้ง 2 แบบ จะต้องควบคุมสภาพแวดล้อมต่าง ๆ ให้ ใกล้เคียงกันมากที่สุด โดยวิธีการดังนี้

1. ควบคุม overhead ต่าง ๆ ของโปรแกรมทั้ง 2 แบบให้ใกล้เคียงกันมากที่สุด โดยในขั้นต้นจะพัฒนาโปรแกรมแบบที่ใช้วิธี combined rank method ด้วยภาษา C ให้เสร็จก่อน แล้วจึงปรับปรุงไปเป็นโปรแกรมแบบที่ใช้ tournament selection โดยพยายามให้โปรแกรมทั้ง 2 คล้ายกันมากที่สุดด้วยการนำโปรแกรมต้นฉบับส่วนที่ใช้กันได้จากโปรแกรมแบบ combined rank method มาใช้กับโปรแกรมแบบ tournament selection ด้วยวิธีการนี้จะสามารถควบคุม overhead ต่างๆของโปรแกรมทั้ง 2 แบบให้ใกล้เคียงกันได้
2. โปรแกรมทั้ง 2 แบบถูกนำมาทดสอบบนเครื่องคอมพิวเตอร์เดียวกันซึ่งใช้ระบบปฏิบัติการ LINUX ในการวัดเวลา ใช้คำสั่ง gprof บน LINUX ซึ่งสามารถวัดเวลาของแต่ละโปรแกรมได้ โดยในการทดลองไม่มีโปรแกรมอื่นๆทำงานอยู่ในขณะเดียวกัน
3. ในการทดลองจะกำหนด parameter ต่างๆของโปรแกรมทั้ง 2 แบบให้ใกล้เคียงกันมากที่สุด โดย parameter เหล่านั้นได้แก่

- จำนวนรุ่นที่มากที่สุดเป็น 50,000 รุ่น เหมือนกันสำหรับโปรแกรมทั้ง 2 แบบ
- ความยาวของลำดับอินพุต/เอาต์พุต (l) กำหนดเป็น 100 และใช้จำนวนลำดับของอินพุต/เอาต์พุต (s) เท่ากับ 10 ลำดับ เหมือนกันสำหรับโปรแกรมทั้ง 2 แบบ
- จำนวนประชากร สำหรับโปรแกรมแบบ combined rank method กำหนดให้ P = 128, Q = 256, R = 128 ดังนั้นจำนวนประชากร (n) เท่ากับ 128 + 256 + 128 = 512 ชุด และสำหรับโปรแกรมแบบ tournament selection กำหนดจำนวนประชากร (n) เท่ากับ 512 ชุด
- ความยาวของ binary string ที่ใช้แทน FSM (m) ขึ้นกับวงจรที่ใช้โดยแสดงในตาราง 2

วงจรที่ใช้ทดสอบเพื่อสังเคราะห์ โดยใช้ขั้นตอนวิธีพันธุกรรมแสดงไว้ในตาราง 2 ซึ่งแต่ละวงจรมีลักษณะการทำงานดังนี้

Serial Adder :- ทำการบวกเลข 2 จำนวนที่เข้ามาทางอินพุต โดยการป้อนอินพุต จะป้อนจาก LSB ไป MSB และเอาต์พุตของวงจรคือ ค่าผลรวมของเลข 2 จำนวนนั้น

Reversible - 4 Counter :- เป็นวงจรนับที่สามารถจะควบคุมให้นับตามปกติ (0-1-2-3) หรือนับย้อน (3-2-1-0) โดยใช้อินพุตเป็นตัวควบคุม

0101 Detector :- เมื่ออินพุตของวงจรเป็นลำดับ 0-1-0-1 วงจรจะให้เอาต์พุตเป็น 1

1010 Detector :- เมื่ออินพุตของวงจรเป็นลำดับ 1-0-1-0 วงจรจะให้เอาต์พุตเป็น 1

1011 Detector :- เมื่ออินพุตของวงจรเป็นลำดับ 1-0-1-1 วงจรจะให้เอาต์พุตเป็น 1

ในการทดลอง สำหรับวงจรที่ใช้ทดสอบแต่ละแบบจะทำการทดลอง 20 ครั้ง โดยจะบันทึกรุ่นที่พบคำตอบและเวลาที่ใช้สำหรับการทดลองแต่ละครั้ง

6. ผลการทดลอง

จากการทดลองได้ผลดังแสดงในตารางที่ 3

วงจรที่ใช้ทดสอบ	อินพุต (bits)	เอาต์พุต (bits)	จำนวนของ internal state	จำนวนของ available internal state	ความยาวของ binary string ที่ใช้แทน FSM ,m (bits)
Serial Adder	2	1	2	4	48
Revesible-4 Counter	1	2	4	8	64
0101 Detector	1	1	4	8	80
1010 Detector	1	1	4	8	80
1011 Detector	1	1	4	8	80

ตาราง 2: วงจรที่ใช้ทดสอบ

วงจรที่ใช้ทดสอบ	จำนวนครั้งที่พบคำตอบใน 50,000 ร่น (จากการทดลอง 20 ครั้ง)		% ที่พบคำตอบใน 50,000 รุ่น		จำนวนรุ่นที่พบคำตอบโดยเฉลี่ย		เวลาโดยเฉลี่ยสำหรับกรณีที่พบคำตอบ (วินาที)	
	Com.	Tour.	Com.	Tour.	Com.	Tour.	Com.	Tour.
Serial Adder	20	20	100%	100%	16	15	4.37	0.90
Reversible-4 Counter	20	15	100%	75%	287	1377	98.79	81.74
0101 Detector	19	10	95%	50%	107	826	31.54	49.00
1010 Detector	20	16	100%	80%	87	72	27.59	4.30
1011 Detector	20	10	100%	50%	148	730	45.31	43.07

ตาราง 3: ผลการทดลอง

* Com. แทน โปรแกรมขั้นตอนวิธีพันธุกรรมแบบที่ใช้ combined rank method

Tour. แทน โปรแกรมขั้นตอนวิธีพันธุกรรมแบบที่ใช้ tournament selection

จากผลการทดลองในตารางที่ 3 ทำการสร้างเป็นตารางที่ 4 เพิ่ม ซึ่งแสดงเวลาเฉลี่ย กรณีที่พบคำตอบ 20 รุ่น เพราะจากผลการทดลอง จำนวนครั้งที่พบคำตอบในวิธี combined rank method และวิธี tournament selection แตกต่างกัน จึงไม่สามารถนำเวลามาเปรียบเทียบกันโดยตรง จึงต้องนำข้อมูลไปแปลงเป็นเวลาเฉลี่ยในกรณีที่พบคำตอบ 20 รุ่น โดยใช้สูตร

เวลาเฉลี่ยกรณีที่พบคำตอบ 20 รุ่น = เวลาเฉลี่ย x 20 / จำนวนครั้งที่พบคำตอบโดยเฉลี่ย

ตัวอย่างเช่น กรณีวงจร 0101 Detector ที่ทดลองโดยโปรแกรมขั้นตอนวิธีพันธุกรรมแบบที่ใช้ tournament selection จากการทดลองพบคำตอบ 10 ครั้ง และเวลาเฉลี่ยเป็น 49.00 วินาที ดังนั้น

เวลาเฉลี่ยกรณีที่พบคำตอบ 20 รุ่น = 49.00 x 20 / 10 = 98.00 วินาที

ซึ่งหมายถึงว่า จะต้องทดลอง 40 ครั้ง จึงจะได้พบคำตอบ 20 ครั้ง (พบคำตอบ 50%) และจากการที่ต้องทดลองเพิ่ม 2 เท่า เวลาเฉลี่ยที่ใช้ จึงควรเพิ่มขึ้น 2 เท่า ในลักษณะที่เป็นอัตราส่วนกัน

จากข้อมูลในตารางที่ 4 วิธี tournament selection ใช้เวลาโดยเฉลี่ยน้อยว่าวิธี combined rank method สำหรับวงจร Serial Adder และ1010 Detector และใช้เวลามากกว่าสำหรับวงจร 0101 Detector และ 1011 Detector และใช้เวลาใกล้เคียงกันสำหรับวงจร Reversible-4 Counter ซึ่งเป็นวงจรที่ใช้เวลามากที่สุด จึงอาจสรุปได้ว่า วิธี tournament selection ใช้เวลาในการประมวลผลไม่มากไปกว่าวิธี combined rank method

วงจรที่ใช้ทดสอบ	Combined rank method	Tournament selection
Serial Adder	4.37	0.90
Reversible-4 Counter	98.79	108.99
0101 Detector	33.20	98.00
1010 Detector	27.59	5.38
1011 Detector	45.31	86.14

ตาราง 4: เวลาเฉลี่ยกรณีที่พบคำตอบ 20 รุ่น

เมื่อพิจารณาผลการทดลองในแง่ของจำนวนรุ่นที่พบคำตอบโดยเฉลี่ย วิธีการ tournament selection จะใช้จำนวนรุ่นมากกว่าหรือเท่ากับวิธี combined rank method ที่เป็นเช่นนี้ เพราะขนาดของประชากรค่อนข้างเล็ก ซึ่งในปัญหานี้ การรักษาความหลากหลายเป็นเรื่องสำคัญ โดยวิธี combined rank method ถูกสร้างเพื่อรักษาความหลากหลายไว้โดยเฉพาะ ถึงแม้ขนาดประชากรเล็ก แต่ก็ยังสามารถลู่เข้าสู่คำตอบได้เร็ว ในขณะที่วิธี tournament selection ไม่มีกลไกในการรักษาความหลากหลายด้วยตัวเอง จำเป็นต้องอาศัยขนาดประชากรที่ค่อนข้างใหญ่ เพื่อให้มีความหลากหลายมากพอที่จะลู่เข้าสู่คำตอบได้เร็ว ดังนั้นจากขนาดของประชากรที่เล็ก ทำให้จำนวนรุ่นที่พบคำตอบโดยเฉลี่ยสำหรับวิธี tournament selection ค่อนข้างมากเมื่อเทียบกับวิธี combined rank method

และเมื่อพิจารณาผลการทดลองในแง่ของจำนวนครั้งที่พบคำตอบใน 50,000 รุ่น จะเห็นได้ว่าวิธี combined rank method ให้คำตอบเกือบ 100% ของจำนวนครั้งที่ทำการทดลอง แต่วิธี tournament selection ในกรณีที่พบคำตอบน้อยสุด พบคำตอบเป็น 50% ของจำนวนครั้งที่ทำการทดลอง ที่เป็นเช่นนี้ อาจอธิบายในลักษณะคล้ายกับที่อธิบายในเรื่องจำนวนรุ่นที่พบคำตอบ คือเนื่องจากขนาดประชากรที่ค่อนข้างเล็กและวิธี tournament selection ไม่มีกลไกในการรักษาความหลากหลายทำให้ในบางกรณีเกิดการลู่เข้าสู่ local maxima ซึ่งไม่ใช่คำตอบ ดังนั้นถ้าหากเพิ่มจำนวนประชากรให้มากกว่านี้หลายเท่า คาดว่าจำนวนครั้งที่พบคำตอบควรจะเข้าใกล้ 100% ของจำนวนครั้งที่ทดลองมากขึ้น

7. สรุป

จากการวิเคราะห์ได้แสดงให้เห็นว่า วิธีการ tournament selection ใช้ความพยายามเชิงคำนวณในขั้นตอนวิธีพันธุกรรมแต่ละรุ่นไม่มากไปกว่าวิธีการ combined rank method และจากการทดลองวัดเวลาที่ใช้ในการประมวลผลโปรแกรมขั้นตอนวิธีพันธุกรรมทั้ง 2 วิธี แสดงว่าวิธี tournament selection ใช้เวลาทั้งหมดไม่มากไปกว่าวิธี combined rank method ซึ่งเวลาชี้ให้เห็นความพยายามเชิงคำนวณ ดังนั้นจึงสรุปได้ว่าวิธีการ tournament selection ใช้ความพยายามเชิงคำนวณทั้งหมดไม่มากไปกว่าวิธี combined rank method

จากผลการทดลอง อาจเห็นข้อด้อยของวิธี tournament selection ที่ใช้ คือ จำนวนครั้งที่พบคำตอบใน 50,000 รุ่น ไม่เป็น 100% ของจำนวนครั้งที่ทดลอง ที่เป็นเช่นนี้เนื่องจากขนาดของประชากรที่ค่อนข้างเล็ก จึงอาจลู่เข้าสู่ local maxima ได้ ดังนั้นถ้าหากปรับเพิ่มจำนวนประชากรที่ใช้ให้มากขึ้น จำนวนครั้งที่พบคำตอบของวิธี tournament selection ควรจะเพิ่มมากกว่าในการทดลองนี้

จากการที่วิธี tournament selection เป็นวิธีการที่ไม่ซับซ้อน และเป็นวิธีการที่ใช้ความพยายามเชิงคำนวณไม่มากไปกว่าวิธี combined rank method ดังนั้นวิธี tournament selection จึงเป็นทางเลือกใหม่ที่เหมาะสม ที่จะนำมาสร้างเป็น mimetic evolvable hardware

8. รายการอ้างอิง

[1] C.Manovit, C. Aporntewan, and P. Chongstitvatana. Synthesis of synchronous sequential logic circuit from partial input/output sequence. In Proc. of Int. Conf. on Evolvable Systems (ICES'98), pages 98-105,1998.

[2] C.Manovit, C. Aporntewan, and P. Chongstitvatana. Comparison of Technology-Based and State-Based

Representations for the Synthesis of Synchronous Sequential Logic from Partial Input/Output Sequence. In Proc. of Conf. of Electrical Engineering, Bangkok, pages 210-213,1998.

[3] C.Aporntewan. A Mimetic Evolvable Hardware for Sequential Circuits. Master's thesis, Chulalongkorn University,1999.

[4] D.E. Goldberg. Genetic Algorithm in search, optimization and machine learning. Addison-Wesley,1989.