รายงานวิจัยฉบับสมบูรณ์

โครงการ การวิเคราะห์ผลกระทบของการเรียนรู้
จากความสัมพันธ์เชิงลบในขั้นตอนวิธีประมาณการแจกแจง

โดย นางสาวสุนิสา ริมเจริญ

2 มิถุนายน 2558

สัญญาเลขที่ TRG5680073

รายงานวิจัยฉบับสมบูรณ์

โครงการ การวิเคราะห์ผลกระทบของการเรียนรู้
จากความสัมพันธ์เชิงลบในขั้นตอนวิธีประมาณการแจกแจง

นางสาวสุนิสา ริมเจริญ

คณะวิทยาการสารสนเทศ มหาวิทยาลัยบูรพา

สนับสนุนโดยสำนักงานกองทุนสนับสนุนการวิจัย

และ คณะวิทยาการสารสนเทศ มหาวิทยาลัยบูรพา

(ความเห็นในรายงานนี้เป็นของผู้วิจัย สกว.ไม่จำเป็นต้องเห็นด้วยเสมอไป)

กิตติกรรมประกาศ

งานวิจัยนี้สำเร็จลุล่วงไปได้จากการสนับสนุนเงินทุนวิจัยของสำนักงานกองทุนสนับสนุนการวิจัย สำนักงานคณะกรรมการการอุดมศึกษา และร่วมกับคณะวิทยาการสารสนเทศ มหาวิทยาลัยบูรพา ผู้วิจัยขอขอบคุณหน่วยงานทั้งสามที่ให้โอกาสกับผู้วิจัยมา ณ โอกาสนี้

ขอขอบคุณ ศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา นักวิจัยที่ปรึกษา ที่คอยช่วยเหลือให้คำแนะนำ ให้กำลังใจกับผู้วิจัยเสมอมา

ขอขอบคุณ ทุก ๆ ท่านที่ไม่ได้เอ่ยนามมาในที่นี้ ที่มีส่วนเกี่ยวข้องในการเตรียมข้อมูล ช่วยรวบรวม ช่วยอ่าน ช่วยให้ข้อเสนอแนะ ติชม จนผู้วิจัยสามารถทำงานวิจัยชิ้นนี้ได้สำเร็จลุล่วง

Abstract

Project Code: TRG5680073

Project Title: Analyzing the Effect of Negative Correlation Learning on Estimation of Distribution Algorithms

Investigator: Miss Sunisa Rimcharoen
Faculty of Informatics, Burapha University

E-mail Address: rsunisa@buu.ac.th, palm.sunisa@gmail.com

Project Period: June 2013 – May 2015

This research proposes a behavior analysis of the univariate estimation of distribution algorithms including the population-based incremental learning (PBIL) and the compact genetic algorithm (cGA). We also propose the frequency-based compact genetic algorithm (fb-cGA) which is a version of the cGA enhanced by the use of a new updating strategy. The algorithm counts the numbers of probability updates and the continuities of probability-update directions, and uses them to adaptively update the algorithm's step sizes. This method requires fewer function evaluations, and achieves solutions that are more accurate than the ones of the conventional cGA. It has been shown that the fb-cGA can reduce the number of function evaluations to only one ninth, as compared with the one of the cGA on ten copies of a 3-bit trap function using a tournament size of 2. The behavior of the fb-cGA on various problems is also examined. The results of the analysis show that information from the algorithm's past experience (i.e., the numbers of probability updates and the continuities) can help the fb-cGA update the probability vector towards a more promising direction, requiring fewer function evaluations.

Keywords: compact genetic algorithm, behavior analysis, probability vector

บทคัดย่อ

รหัสโครงการ: TRG5680073

ชื่อโครงการ: การวิเคราะห์ผลกระทบของการเรียนรู้จากความสัมพันธ์เชิงลบในขั้นตอนวิธีประมาณการแจกแจง

ชื่อนักวิจัย: สุนิสา ริมเจริญ
คณะวิทยาการสารสนเทศ มหาวิทยาลัยบูรพา

E-mail Address: rsunisa@buu.ac.th, palm.sunisa@gmail.com

ระยะเวลาโครงการ: มิถุนายน 2556 – พฤษภาคม 2558

งานวิจัยนี้นำเสนอการวิเคราะห์พฤติกรรม ของขั้นตอนวิธีประมาณการแจกแจงแบบที่ตัวแปรไม่ขึ้นต่อกัน ซึ่งในที่นี้ได้ทำการศึกษาขั้นตอนวิธีการเรียนรู้เพิ่มขึ้นแบบอาศัยประชากร (PBIL) และขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับ (cGA) นอกจากนี้ยังได้นำเสนอการปรับปรุงขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับด้วยค่าความถี่ (fb-cGA) ซึ่งเป็นการปรับขึ้นมาจากขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับเดิม โดยขั้นตอนวิธีที่นำเสนอนี้จะใช้วิธีการนับจำนวนครั้งของการปรับปรุงความน่าจะเป็น และนับต่อเนื่องของความน่าจะเป็นที่มีทิศทางการปรับปรุงไปในทางเดียวกัน ค่าความถี่กับค่าความต่อเนื่องนี้จะถูกในในกระบวนการปรับค่าของเวกเตอร์ความน่าจะเป็น วิธีการที่นำเสนอนี้ส่งผลให้ขั้นตอนวิธีใช้จำนวนครั้งในการประเมินคุณภาพคำตอบน้อยลง และยังให้ผลความสำเร็จในการแก้ปัญหาที่มีความถูกต้องมากขึ้นกว่าขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับเดิม ผลการทดลองแสดงให้เห็นว่า fb-cGA สามารถลดจำนวนครั้งของการประเมินคุณภาพคำตอบลงได้ถึง 1 ใน 9 สำหรับการทดลองกับปัญหากับดัก ผลการวิเคราะห์พฤติกรรมการทำงานของขั้นตอนวิธีนี้ แสดงให้เห็นว่าข้อมูลจากประสบการณ์ที่ผ่านมาของอัลกอริทึม (กล่าวคือจำนวนครั้งของการปรับปรุงความน่าจะเป็นและค่าความต่อเนื่อง) สามารถช่วยให้ fb-cGA มีข้อมูลประกอบการตัดสินใจได้มากขึ้น นำไปสู่การปรับปรุงค่าความน่าจะเป็นในเวกเตอร์ความน่าจะเป็นไปในทิศทางที่นำไปสู่คำตอบ โดยที่ใช้จำนวนครั้งในการประเมินค่าความเหมาะสมน้อยลง

คำหลัก : ขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับ, การวิเคราะห์พฤติกรรม, เวกเตอร์ความน่าจะเป็น

สารบัญ

เรื่อง **หน้า**

บทที่ 1

บทนำ

ขั้นตอนวิธีเชิงวิวัฒนาการ (Evolutionary Algorithm) เป็นขั้นตอนวิธีทางคอมพิวเตอร์ที่ถูกคิดค้นขึ้นโดยอาศัยแรงบันดาลใจจากกระบวนการวิวัฒนาการทางธรรมชาติ ซึ่งเป็นวิธีที่สิ่งมีชีวิตต่าง ๆ ใช้ปรับปรุงสายพันธุ์เพื่อการอยู่รอดและทำให้เกิดลูกหลานรุ่นถัด ๆ ไปที่มีความสามารถมากขึ้น มีการปรับตัวที่เหมาะสมต่อสภาพแวดล้อมมากยิ่งขึ้น กระบวนการวิวัฒนาการนี้เป็นกลไกสำคัญในการดำรงเผ่าพันธุ์ของสิ่งมีชีวิตต่าง ๆ มาช้านาน หัวใจหลักที่ทำให้กระบวนการวิวัฒนาการทางธรรมชาตินี้ประสบความสำเร็จก็คือ หลักการคัดสรรทางธรรมชาติ (Natural Selection) ที่คัดเลือกให้สิ่งมีชีวิตที่เข้มแข็ง แข็งแรง และมีความสามารถปรับตัวเข้ากับสิ่งแวดล้อมได้ดีกว่ามีโอกาสอยู่รอดและขยายเผ่าพันธุ์ได้มากกว่าสิ่งมีชีวิตที่อ่อนแอ สังเกตได้จากสัตว์เพศผู้ที่แข็งแรงกว่าจะมีโอกาสได้เป็นจ่าฝูงและมีโอกาสผสมพันธุ์กับเพศเมียได้มากกว่า

จากหลักการของการวิวัฒนาการทางธรรมชาติที่น่าอัศจรรย์ใจนี้เอง ที่ทำให้เกิดการคิดค้นกระบวนการทางคอมพิวเตอร์ที่ใช้ค้นหาคำตอบโดยเลียนแบบธรรมชาติขึ้น ขั้นตอนวิธีเชิงวิวัฒนาการยืมวิธีการทางธรรมชาติที่สร้างประชากรจำนวนหนึ่งขึ้นมา แล้วมีกระบวนการคัดสรรเพื่อเลือกสิ่งมีชีวิตที่เหมาะสมให้ขยายเผ่าพันธุ์ต่อไป

ขั้นตอนวิธีเชิงวิวัฒนาการมีกระบวนการทำงานโดยเริ่มต้นจากการสร้างตัวอย่างคำตอบขึ้นมาจำนวนหนึ่ง (เรียกว่า ประชากร) ประชากรแต่ละตัวจะถูกประเมินค่าความเหมาะสม (Fitness Value) แล้วผ่านกระบวนการคัดเลือก โดยประชากรที่มีค่าความเหมาะสมที่ดีกว่าจะมีโอกาสถูกเลือกเพื่อนำไปเป็นต้นแบบในการปรับปรุงคุณภาพคำตอบให้ดียิ่งขึ้นในรุ่นถัดไป และกระบวนการนี้จะถูกทำซ้ำจากรุ่นสู่รุ่น (Generation) จนกว่าจะได้คำตอบที่เหมาะสม

สิ่งหนึ่งที่เป็นหัวใจสำคัญที่ทำให้ขั้นตอนวิธีเชิงวิวัฒนาการประสบความสำเร็จ ก็คือ การคัดเลือกประชากรที่มีค่าความเหมาะสมที่ดี แล้วหยิบชิ้นส่วนที่ดี (Building Block) ที่เป็นองค์ประกอบของคำตอบจากประชากรเหล่านั้น มาเป็นต้นแบบสำหรับสร้างประชากรในรุ่นถัดไป ซึ่งก็มีสมมุติฐานสำคัญอันหนึ่ง ที่เรียกว่า “building block hypothesis” Goldberg (1989) กล่าวว่า “*Short, low order, and highly fit schemata are sampled, recombined, and re-sampled to form strings of potentially higher fitness*.” หลักการอันนี้เอง ที่นักวิจัยส่วนใหญ่นำมาออกแบบขั้นตอนวิธีเชิงวิวัฒนาการของตนเอง ให้ค้นคำตอบไปในทิศทางที่จะทำให้คำตอบมีค่าความเหมาะสมสูงขึ้น

แต่อย่างไรก็ดี การเลือกชิ้นส่วนที่น่าจะดีที่ปรากฏอยู่ในคำตอบตัวที่มีค่าความเหมาะสมสูงอาจจะไม่ได้นำไปสู่คำตอบที่ดีที่สุดเสมอไป ตัวอย่างเช่น สำหรับบางปัญหาที่คำตอบมีค่าดีสุดเฉพาะที่ (Local Optimum) ขั้นตอนวิธีที่ถูกออกแบบมาให้เลือกหยิบชิ้นส่วนที่ดีมาประกอบกันเป็นคำตอบใหม่ อาจนำไปสู่กับดักในตำแหน่งที่ดีสุดเฉพาะที่ได้ และขั้นตอนวิธีที่มีลักษณะการทำงานแบบนี้ก็จะหนีออกจากสถานการณ์ดังกล่าวได้ยาก เพราะเหตุนี้ จึงมีนักวิจัยกลุ่มหนึ่งหันมาสนใจวิธีการออกแบบขั้นตอนวิธีที่พิจารณาความรู้จากคำตอบที่ค่าความเหมาะสมต่ำร่วมด้วย เช่น ในปี ค.ศ. 1994 Baluja ได้เสนอวิธีการใช้ความรู้จากตัวอย่างคำตอบที่มีค่าความเหมาะสมน้อยที่สุดร่วมด้วยแทนที่จะใช้คำตอบที่ดีที่สุดเพียงอย่างเดียว ในการปรับปรุงค่าเวกเตอร์ความน่าจะเป็น (Probability Vector) สำหรับขั้นตอนวิธีการเรียนรู้เพิ่มขึ้นแบบอาศัยประชากร (Population Based Incremental Learning: PBIL)

โครงการวิจัยนี้ เล็งเห็นประโยชน์จากการนำความรู้ของคำตอบที่มีค่าความเหมาะสมต่ำเพื่อมาเติมเต็มความรู้ที่ได้จากชิ้นส่วนของคำตอบดี ๆ จึงทำการศึกษา โดยมุ่งเน้นไปที่ขั้นตอนวิธีประมาณการแจกแจงแบบง่ายที่สุดคือแบบที่ไม่มีความขึ้นต่อกันในตัวแปรแต่ละตัว โดยจะศึกษาขั้นตอนวิธีการเรียนรู้เพิ่มขึ้นแบบอาศัยประชากร และขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับ (Compact Genetic Algorithm: cGA) และทำการวิเคราะห์ขั้นตอนวิธีเหล่านี้ โดยรายละเอียดของการทดลองและผลการวิเคราะห์จะกล่าวในบทที่ 3 และ 4

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ขั้นตอนวิธีประมาณการแจกแจง

ขั้นตอนวิธีประมาณการแจกแจง (Estimation of Distribution Algorithms: EDAs) หรือขั้นตอนวิธีเชิงพันธุกรรมแบบการสร้างแบบจำลองความน่าจะเป็น (Probabilistic Model Building Genetic Algorithms: PMBGAs) นำเสนอโดย Mühlenbein และ Paaß (1996), Pelikan, Goldberg และ Lobo (1999) ในขั้นตอนวิธีนี้แต่ละตัวแปร (อาจมองเทียบได้กับแต่ละบิตในขั้นตอนวิธีเชิงพันธุกรรมแบบเดิม) จะมีค่าความน่าจะเป็นในการเป็นคำตอบ รวมทั้งความสัมพันธ์กับตัวแปรอื่น และจะใช้การปรับค่าความน่าจะเป็นเพื่อปรับปรุงการกระจายตัวของคำตอบให้ไปในทิศทางที่ผลเฉลยจะมีค่าความเหมาะสมสูงขึ้น

ขั้นตอนวิธีประมาณการแจกแจงสามารถแบ่งได้เป็น 3 กลุ่มหลัก ๆ ตามลักษณะการขึ้นต่อกันของตัวแปร คือ ขั้นตอนวิธีประมาณการแจกแจงแบบที่ตัวแปรไม่ขึ้นต่อกัน (Univariate Estimation of Distribution Algorithm) ขั้นตอนวิธีประมาณการแจกแจงแบบที่ตัวแปรขึ้นต่อกันเป็นคู่ (Bivariate Estimation of Distribution Algorithm) และขั้นตอนวิธีประมาณการแจกแจงแบบตัวแปรหลายตัวขึ้นต่อกัน (Multivariate Estimation of Distribution Algorithm) ซึ่งแต่ละกลุ่มก็มีขั้นตอนวิธีแต่ละแบบที่ถูกนำเสนอในหลายปีที่ผ่านมา ในที่นี้จะยกตัวอย่างขั้นตอนวิธีประมาณการแจกแจงแบบง่ายที่สุดคือไม่มีความขึ้นต่อกันในตัวแปรแต่ละตัว เช่น ขั้นตอนวิธีการเรียนรู้เพิ่มขึ้นแบบอาศัยประชากร (Population Based Incremental Learning: PBIL) และ ขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับ (Compact Genetic Algorithm: cGA) ซึ่งมีรายละเอียดดังนี้

2.1.1 ขั้นตอนวิธีการเรียนรู้เพิ่มขึ้นแบบอาศัยประชากร

อัลกอริทึมนี้นำเสนอโดย Baluja (1994) ซึ่งมีแนวคิดในการใช้เวกเตอร์ความน่าจะเป็น (Probability Vector) แทนที่การใช้กลุ่มประชากรแบบเดิม เวกเตอร์ความน่าจะเป็นนี้จะเป็นตัวแบบในการหาการกระจายตัวของคำตอบดี ๆ โดยแต่ละมิติ (Dimension) ของเวกเตอร์เป็นค่าความน่าจะเป็นที่แต่ละบิตจะเป็น 1 การปรับปรุงค่าความน่าจะเป็นนี้ทำได้โดยทำการสุ่มสร้างตัวอย่างแล้วปรับค่าความน่าจะเป็นให้เข้าใกล้ตัวอย่างที่ดีที่สุด ขั้นตอนการทำงานของการเรียนรู้เพิ่มขึ้นแบบอาศัยประชากรแสดงในรูปที่ 1 เมื่อ l เป็นจำนวนมิติ (จำนวนบิต) p_i คือ ค่าความน่าจะเป็นในแต่ละมิติของเวกเตอร์, α คือ อัตราการเรียนรู้ซึ่งมีค่าอยู่ระหว่าง (0, 1] และ $best_i$ คือ ค่าในแต่ละบิต (ค่า 1 หรือ 0) ของตัวอย่างตัวที่ดีที่สุด ส่วนค่า *mutation_shilf* เป็นค่าบอกปริมาณว่าจะกลายพันธุ์ไปมากน้อยเพียงใด

การทำงานจะเริ่มต้นด้วยการสร้างเวกเตอร์ความน่าจะเป็นขึ้นมา ในที่นี้กำหนดให้แต่ละมิติของเวกเตอร์มีค่าความน่าจะเป็นเท่ากับ 0.5 นั่นคือให้การกระจายตัวของคำตอบซึ่งเป็น บิต 0 หรือ 1 มีโอกาสเกิดขึ้นได้เท่า ๆ กัน จากนั้นจะสุ่มสร้างตัวอย่างที่เป็นตัวแทนของคำตอบขึ้นมาจากเวกเตอร์ความน่าจะเป็นนี้ *M* ตัวอย่าง ทำการประเมินค่าความเหมาะสมของตัวอย่างเหล่านี้แล้วเลือกคำตอบที่ดีที่สุดมา 1 ตัว แล้วจะนำตอบตอบที่เลือกมาได้นี้เป็นต้นแบบในการปรับปรุงค่าความน่าจะเป็นในแต่ละมิติเพื่อให้การกระจายตัวของคำตอบในรุ่นถัดไปเข้าใกล้ตัวอย่างนี้ การปรับปรุงค่าความน่าจะเป็นในเวกเตอร์ทำตามสูตรในขั้นตอนที่ 4 และ 5 ของรูปที่ 2.1

Procedure population-based incremental learning

1: Initialize probability vector (p) with 0.5 at each position.

2: Generate M individuals from the vector

3: Select the best individual (*best*)

4: Update the probability vector p

for i = 1 to l do

$p_i = p_i \times (1.0 - \alpha) + (best_i \times \alpha)$

5: Mutate probability vector

for i = 1 to l do

if(random(0, 1] < *mutation_probability*)

$p_i = p_i \times (1.0 - mutation\_shift)$ + random(0.0 or 1.0) $\times$ *mutation_shift*

6: Go to step 2 until a termination criterion is met.

รูปที่ 2.1 ขั้นตอนวิธีของการเรียนรู้เพิ่มขึ้นแบบอาศัยประชากร

ขั้นตอนการปรับปรุงเวกเตอร์ความน่าจะเป็นข้างต้น เป็นขั้นตอนวิธีที่ใช้ความรู้จากตัวอย่างที่ดีที่สุดเพียงตัวเดียว (ในขั้นตอนที่ 3 ในรูปที่ 2.1 ทำการเลือกตัวอย่างที่ดีที่สุดมาใช้เป็นต้นแบบในการปรับปรุงค่าของเวกเตอร์ความน่าจะเป็น)

สำหรับแนวทางการปรับปรุงขั้นตอนวิธีดังกล่าว Baluja (1994) ยังได้เสนอวิธีการอีกหลายแบบเพื่อทำให้ขั้นตอนวิธีมีประสิทธิภาพดีขึ้น หนึ่งในนั้นคือ การใช้ความรู้จากตัวอย่างคำตอบที่มีค่าความเหมาะสมน้อยที่สุดร่วมด้วยแทนที่จะใช้คำตอบที่ดีที่สุดเพียงอย่างเดียว โดยเปลี่ยนวิธีการปรับปรุงค่าเวกเตอร์ความน่าจะเป็น จากขั้นตอนที่ 4 ในรูปที่ 2.1 เป็นดังรูปที่ 2.2

4: Update the probability vector p

for i = 1 to l do

if ($best_i \neq worst_i$) then

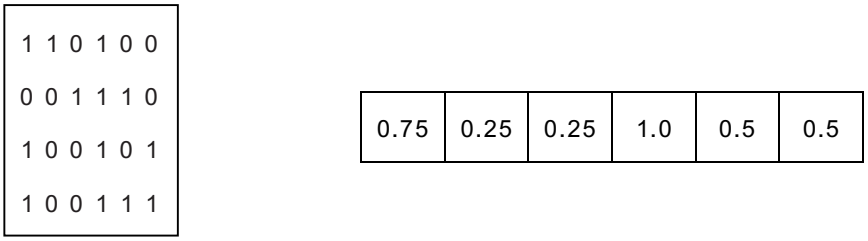
$p_i = p_i \times (1.0 - \alpha) + (best_i \times \alpha)$

รูปที่ 2.2 การปรับค่าเวกเตอร์ความน่าจะเป็นโดยใช้คำตอบแย่สุดร่วมด้วย

2.1.2 ขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับ

ขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับ เป็นหนึ่งในขั้นตอนวิธีเชิงวิวัฒนาการแบบใหม่ ที่มีแนวความคิดในการใช้ตัวแบบความน่าจะเป็น (Probabilistic Model) แทนการใช้กลุ่มประชากรแบบเดิมในการค้นหาคำตอบ แนวความคิดนี้ทำให้ขั้นตอนวิธีเชิงพันธุกรรมใช้หน่วยความจำในการเก็บประชากรน้อยลง เนื่องจากไม่มีการใช้ประชากรอีกต่อไป อีกทั้งยังไม่ต้องอาศัยการดำเนินการเชิงพันธุกรรม เช่น การไขว้เปลี่ยน หรือ การกลายพันธุ์ ทำให้การประมวลผลทำได้รวดเร็วยิ่งขึ้น โดยที่ยังคงความสามารถเทียบเท่ากับขั้นตอนวิธีเชิงพันธุกรรมอย่างง่ายที่ใช้อยู่เดิม ซึ่งข้อพิสูจน์นี้ปรากฏในงานวิจัยของ Harik *et al*. (1999)

การแทนคำตอบของขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับ จะอยู่ในรูปแบบของเวกเตอร์ความน่าจะเป็น (Probability Vector) ซึ่งจะใช้เป็นตัวแบบในการหาการกระจายตัวของคำตอบโดยแต่ละมิติ (Dimension) ของเวกเตอร์เป็นค่าความน่าจะเป็นที่แต่ละบิตจะเป็น 1 ตัวอย่างเช่น สมมุติว่าประชากรของขั้นตอนวิธีเชิงพันธุกรรมแบบเดิมมีโครโมโซมยาว 6 บิตดังรูปที่ 2.3 (ก) ตัวอย่างของเวกเตอร์ความน่าจะเป็น ของขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับอาจเป็นดังรูปที่ 2.3 (ข) ซึ่งจะมีจำนวนมิติคือ 6 ตามความยาวของโครโมโซมในขั้นตอนวิธีเชิงพันธุกรรมอย่างง่าย จากรูปที่ 2.3 จะเห็นได้ว่าการแทนคำตอบโดยใช้เวกเตอร์ความน่าจะเป็นนี้ สามารถลดหน่วยความจำที่ใช้จัดเก็บประชากรลงไปได้อย่างมาก เช่น จากตัวอย่างนี้แทนที่จะใช้หน่วยความจำในการเก็บประชากร 4 ตัว ก็จะเหลือเพียงเวกเตอร์จำนวนจริงเพียงเวกเตอร์เดียว โดยในเวกเตอร์ความน่าจะเป็นนี้ แต่ละมิติจะเป็นค่าความน่าจะเป็นของการเกิดบิตที่เป็น 1 เช่น ค่า 0.75 แทนความน่าจะเป็นที่โครโมโซมบิตแรกจะเป็น 1 จากประชากรทั้งหมด เป็นต้น



(ก) (ข)

รูปที่ 2.3 การแทนประชากรของขั้นตอนวิธีเชิงพันธุกรรมเทียบกับเวกเตอร์ความน่าจะเป็น

การปรับปรุงคำตอบของขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับ สามารถทำได้โดยการปรับค่าความน่าจะเป็นในแต่ละมิติของเวกเตอร์ความน่าจะเป็นตามคำตอบที่ดีกว่า โดยในการทำงานของอัลกอริทึมนี้จะมีการสุ่มสร้างตัวอย่าง 2 ตัว ขึ้นมาจากเวกเตอร์ความน่าจะเป็น โดยโครโมโซมของตัวอย่างที่สุ่มขึ้นมานี้ในแต่ละบิตจะเป็น 0 หรือ 1 ขึ้นกับค่าความน่าจะเป็นในแต่ละมิติของเวกเตอร์ความน่าจะเป็น เช่น ถ้าเวกเตอร์ความน่าจะเป็นมีค่า 0.5 ในทุกมิติ ตัวอย่างประชากรที่สุ่มสร้างขึ้นมา จะมีโอกาสเป็นบิต 1 หรือ 0 เท่า ๆ กัน แต่ถ้าเวกเตอร์ความน่าจะเป็นมีค่า 1.0 ทุกมิติ ตัวอย่างที่สุ่มออกมาได้จะเป็นโครโมโซมที่เป็นบิต 1 ทั้งหมด

เมื่อสุ่มสร้างตัวอย่างขึ้นมาแล้วก็จะพิจารณาว่าคำตอบตัวใดมีค่าความเหมาะสมมากกว่ากัน เวกเตอร์ความน่าจะเป็นในแต่ละมิติ จะถูกปรับตามบิตของคำตอบตัวที่ดีกว่า ถ้าบิตนั้นมีค่าเป็น 1 ความน่าจะเป็นในมิตินั้นก็จะปรับเข้าใกล้ 1 แต่ถ้าเป็น 0 ค่าความน่าจะเป็นในมิตินั้นก็จะลดค่าลง โอกาสที่บิตนั้นจะถูกสร้างมาเป็น 0 ในรอบถัดไปก็จะเพิ่มตามด้วย ดังนั้นเมื่อผ่านกระบวนการวิวัฒนาการไประยะหนึ่ง เวกเตอร์ความน่าจะเป็นก็จะเป็นรูปแบบการกระจายตัวของคำตอบที่ดี โดยการทำงานของขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับมีขั้นตอนโดยสรุปดังนี้

1. กำหนดค่าเริ่มต้นของตัวแปรในเวกเตอร์ความน่าจะเป็นในทุก ๆ มิติให้มีค่าเป็น 0.5
2. สุ่มสร้างตัวอย่างคำตอบจากเวกเตอร์ความน่าจะเป็นขึ้นมา 2 ตัว
3. คำนวณค่าความเหมาะสมของตัวอย่างที่สุ่มมาได้ แล้วพิจารณาว่าตัวใดเป็นผู้ชนะและผู้แพ้
4. ปรับค่าเวกเตอร์ความน่าจะเป็นตามผู้ชนะ โดยพิจารณาเฉพาะบิตที่ผู้ชนะและผู้แพ้ไม่เหมือนกัน
   - ถ้าบิตตำแหน่งที่ *i* ของผู้ชนะ เป็น 1 จะปรับค่าเวกเตอร์ความน่าจะเป็นที่ตำแหน่ง *i* โดยนำค่าความน่าจะเป็นเดิมบวกกับ 1/*n* เมื่อ *n* คือจำนวนประชากรที่ถูกกำหนดใช้ในขั้นตอนวิธีเชิงพันธุกรรมอย่างง่าย
   - ในทางกลับกัน ถ้าบิตตำแหน่งที่ *i* ของผู้ชนะ เป็น 0 จะปรับค่าเวกเตอร์ความน่าจะเป็นที่ตำแหน่ง *i* โดยนำค่าความน่าจะเป็นเดิมลบด้วย 1/*n*

5. ตรวจสอบว่าเวกเตอร์ความน่าจะเป็นลู่เข้าสู่คำตอบแล้วหรือไม่ โดยพิจารณาจากค่าความน่าจะเป็นของแต่ละมิติว่าเป็น 0.0 หรือ 1.0 ครบแล้วหรือไม่ ถ้าค่าความน่าจะเป็นลู่เข้าสู่ 0.0 หรือ 1.0 หมดแล้ว ให้จบการทำงาน แต่ถ้ายังลู่เข้าไม่ครบ ให้สุ่มตัวอย่างจากเวกเตอร์ความน่าจะเป็นขึ้นมาใหม่แล้วทำซ้ำขั้นตอนที่ 2 ถึง 5 ต่อไป

2.2 ปัญหาทดสอบที่ใช้ในการทดลอง

2.2.1 ปัญหาจำนวนบิตหนึ่งมากสุด (OneMax)

ปัญหานี้เป็นปัญหาง่ายสำหรับขั้นตอนวิธีเชิงพันธุกรรม และมักจะถูกใช้เป็นปัญหาทดสอบพฤติกรรมของอัลกอริทึมเพื่อเปรียบเทียบความสามารถในการแก้ปัญหาง่าย ค่าความเหมาะสมจะขึ้นอยู่กับจำนวนบิตที่เป็น 1 และค่าสูงสุดที่เป็นไปได้คือทุกบิตเป็น 1 ทั้งหมด ในกรณีนี้ค่าความเหมาะสมที่มากที่สุดจะเท่ากับความยาวของโครโมโซม

กำหนดให้ $\vec{x} = \{x_1, x_2, ..., x_N\}$ เป็นโครโมโซมความยาว N, โดยที่แต่ละบิต $x_i \in \{0,1\}$ การหาค่าความเหมาะสมของโครโมโซมนี้สามารถคำนวณได้จากสมการที่ 2.1

$$F(\vec{x}) = \sum_{i=1}^{N} x_i \qquad (2.1)$$

ตัวอย่างเช่น ถ้าคำตอบของอัลกอริทึมเป็น 111011 ค่าความเหมาะสมที่คำนวณได้ คือ 5 (เนื่องจากมีจำนวนบิตที่เป็น 1 อยู่ 5 บิต)

2.2.2 ปัญหาจำนวนบิตที่ตรงกับเป้าหมายแบบสุ่มมากสุด (RandomMax)

ปัญหานี้คล้ายกับปัญหาจำนวนบิตหนึ่งมากสุดที่ได้กล่าวไปข้างต้น แต่ต่างกันตรงที่แทนที่จะค้นหาคำตอบดีสุดที่คำตอบเป็นบิตหนึ่งทั้งหมด ก็จะค้นคำตอบไปในทิศทางที่ทำให้คำตอบตรงกับเป้าหมายที่สุ่มขึ้นมาแทน ซึ่งการพิจารณาค่าความเหมาะสมของคำตอบก็จะเทียบคำตอบกับเป้าหมายที่สุ่มรูปแบบมาตั้งแต่ต้น โดยเทียบคำตอบบิตต่อบิต ถ้าบิตคำตอบในตำแหน่งใดตรงกันก็จะนับคะแนนเพิ่มขึ้นหนึ่งแต้ม การที่ต้องทำเช่นนี้ก็เนื่องจากปัญหานี้ถูกออกแบบมา เพื่อทดสอบอัลกอริทึมโดยไม่ต้องการให้มีการเอื้อไปในทิศทางของศูนย์หรือหนึ่งเพียงด้านเดียว เช่น สมมติว่าเป้าหมายถูกสุ่มได้เป็น 011011 และคำตอบจากอัลกอริทึมได้เป็น 110011 ค่าความเหมาะสมที่คำนวณได้ คือ 4 (บิตที่ตรงกับเป้าหมาย คือ บิตที่ 2, 4, 5 และ 6)

2.2.3 ปัญหากับดัก (Trap)

ปัญหากับดัก (Goldberg, 1987) เป็นปัญหายากสำหรับขั้นตอนวิธีเชิงพันธุกรรม เนื่องจากฟังก์ชันหาค่าความเหมาะสมจะให้ค่ากับโครโมโซมที่ประกอบด้วยบิต 0 มากกว่าบิต 1 แต่ค่าสูงสุดของฟังก์ชันกลับได้มาจากโครโมโซมที่ประกอบด้วยบิต 1 ทั้งหมด

ปัญหากับดักนี้จะประกอบด้วยหน่วยสร้าง (Building Block) ย่อย ๆ นำมาประกอบต่อกันเป็นกับดักที่ยาวขึ้น หน่วยสร้างเล็ก ๆ เหล่านี้ยาว k บิต และสามารถหาค่าความเหมาะสมของแต่ละหน่วยสร้างได้ตามสมการที่ 2.2

$$F_k(b_0 \ldots b_{k-1}) = \begin{cases} f_{\text{high}} & ; \quad \text{if } u = k \\ f_{\text{low}} - u\dfrac{f_{\text{low}}}{k-1} & ; \quad \text{otherwise} \end{cases} \tag{2.2}$$

เมื่อ $b_i \in \{0, 1\}$, $u = \sum_{i=0}^{k-1} b_i$, และ $f_{\text{high}} > f_{\text{low}}$ โดยปกติแล้ว f_{high} จะมีค่าเท่ากับ k และ f_{low} มีค่าเท่ากับ k-1

เมื่อนำหน่วยสร้างนี้มาประกอบต่อกันให้ยาวขึ้น ค่าความเหมาะสมสามารถคำนวณได้จากการนำค่าความเหมาะสมของแต่ละหน่วยสร้างย่อย ๆ มาบวกกัน ฟังก์ชันหาค่าความเหมาะสมของการนำหน่วยสร้างขนาด k มาต่อกัน m หน่วย แสดงในสมการที่ 2.3

$$F_{k\times m}(B_0 \ldots B_{m-1}) = \sum_{i=0}^{m-1} F_k(B_i),\ B_i \in \{0,1\}^k \tag{2.3}$$

ตัวอย่างเช่น ถ้ากำหนดหน่วยสร้างขนาด 3 (k=3) และนำแต่ละหน่วยสร้างมาต่อกันทั้งหมด 5 ชุด (3 x 5 trap problem) และมีตัวอย่างคำตอบคือ 111 001 110 000 100 ค่าความเหมาะสมที่คำนวณได้คือ 3 + 1 + 0 + 2 + 1 = 7

2.2.4 ปัญหารอยัลโรด (Royal Road)

ปัญหานี้เป็นการพิจารณากลุ่มของรูปแบบบิตที่ประกอบกันเป็นคำตอบ โดยที่รูปแบบบิตเหล่านี้จะเรียกว่า สคีมา (Schema) ซึ่งสำหรับปัญหารอยัลโรดแบบ 64 บิตที่ใช้ในการทดลองนี้ จะมีรูปแบบสคีมากำหนดอยู่ทั้งหมด 15 รูปแบบ ดังแสดงในรูปที่ 2.4

Schema 1 = 11111111**********; $s1 = 8$
Schema 2 = ********11111111**********; $s2 = 8$
Schema 3 = ****************11111111**********; $s3 = 8$
Schema 4 = ************************11111111**********; $s4 = 8$
Schema 5 = ********************************11111111**********; $s5 = 8$
Schema 6 = **11111111**********; $s6 = 8$
Schema 7 = **11111111********; $s7 = 8$
Schema 8 = **11111111; $s8 = 8$
Schema 9 = 1111111111111111**********; $s9 = 16$
Schema10 =****************1111111111111111**********; $s10 = 16$
Schema11 =********************************1111111111111111**********; $s11 = 16$
Schema12 =**1111111111111111; $s12 = 16$
Schema13 =11111111111111111111111111111111**********; $s13 = 32$
Schema14 =********************************11111111111111111111111111111111; $s14 = 32$
Schema15 =1111111111111111111111111111111111111111111111111111111111111111; $s15 = 64$

รูปที่ 2.4 รูปแบบสคีมาในปัญหารอยัลโรด 64 บิต

วิธีการคำนวณค่าความเหมาะสมของปัญหารอยัลโรด 64 บิต จะทำการเทียบรูปแบบของคำตอบที่ได้จากอัลกอริทึมกับรูปแบบของสคีมาทีละสคีมา ถ้าคำตอบมีรูปแบบของบิตตรงตามสคีมาใดก็จะได้คะแนนตามที่ระบุในสคีมานั้น เช่น ถ้ามีรูปแบบบิตตามสคีมาที่ 1 ก็จะได้คะแนน 8 แต้ม และถ้าคำตอบมีรูปแบบตรงกับสคีมาที่ 9 ก็จะได้คะแนน 16 แต้ม (คะแนนที่ได้นี้จะบวกสะสมรวมกัน) คำตอบที่ดีที่สุดสำหรับปัญหานี้คือทุก ๆ บิตเป็นหนึ่งทั้งหมด ซึ่งกรณีนี้การคำนวณค่าความเหมาะสมทำได้โดย $(8 \times 8) + (4 \times 16) + (2 \times 32) + 64 = 256$

2.3 งานวิจัยที่เกี่ยวข้อง

Harik และคณะ (1999) นำเสนอขั้นตอนวิธีใหม่ในการทำงานของขั้นตอนวิธีเชิงพันธุกรรมเรียกว่าขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับ (Compact Genetic Algorithm)โดยนำเสนอการใช้ตัวแบบความน่าจะเป็น (Probability Vector) แทนกลุ่มประชากรในการหาคำตอบ ซึ่งทำให้สามารถลดการใช้หน่วยความจำจากการจัดเก็บกลุ่มประชากรได้ เนื่องจากวิธีนี้จะทำการจัดเก็บเพียงตัวแบบความน่าจะเป็นแค่ตัวเดียวเท่านั้น และวิธีนี้ยังทำให้การประมวลผลเป็นไปได้อย่างรวดเร็ว เนื่องจากไม่จำเป็นจะต้องอาศัยการดำเนินการเชิงพันธุกรรม เช่น การไขว้เปลี่ยนและการกลายพันธุ์ เป็นต้นโดยที่ยังคงความสามารถในการหาคำตอบได้เทียบเท่ากับขั้นตอนวิธีแบบเดิม

หลังจากที่ขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับถูกนำเสนอ ได้มีงานวิจัยจำนวนหนึ่งที่ปรับปรุงขั้นตอนวิธีดังกล่าว เช่น Ahn และ Ramakrishna (2003) นำเสนอขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับที่เลือกเก็บประชากรที่ดีที่สุดแบบถาวร (Persistent Elitist Compact Genetic Algorithm: pe-cGA) และขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับที่เลือกเก็บประชากรที่ดีที่สุดแบบไม่ถาวร (Nonpersistent Elitist Compact Genetic Algorithm: ne-cGA)

ขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับที่เลือกเก็บประชากรที่ดีที่สุดแบบถาวร ได้นำประชากรตัวที่ดีที่สุดมาใช้ในการปรับปรุงค่าความน่าจะเป็น ซึ่งขั้นตอนวิธีนี้มีการปรับปรุงการทำงานของขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับใน 2 ขั้นตอนด้วยกัน ได้แก่ ขั้นตอนการสร้างประชากร และขั้นตอนการพิจารณาหาประชากรที่มีค่าความเหมาะสมที่ดีกว่า

การสร้างประชากรของขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับที่เลือกเก็บประชากรที่ดีที่สุดแบบถาวรนั้น มีการสร้างประชากร และประชากรที่ดีที่สุดอย่างละ 1 ประชากรเท่านั้นโดยที่ประชากรที่ดีที่สุดจะสร้างขึ้นมาเฉพาะในประชากรรุ่นแรก หลังจากประชากรรุ่นแรกขั้นตอนวิธีจะทำการสร้างประชากรขึ้นมาเพียงตัวเดียวเท่านั้น

การพิจารณาหาประชากรที่มีค่าเหมาะสมที่ดีกว่านั้น ขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับที่เลือกเก็บประชากรที่ดีที่สุดแบบถาวร ได้นำประชากรกับประชากรที่ดีที่สุดมาคำนวณค่าความเหมาะสมเพื่อนำมาเปรียบเทียบหาประชากรที่มีค่าความเหมาะสมที่ดีกว่า ซึ่งถ้าหากประชากรมีค่าความเหมาะสมที่มากกว่าประชากรที่ดีที่สุด ประชากรนี้ก็จะกลายเป็นประชากรผู้ชนะ และนำมาแทนที่ประชากรที่ดีที่สุดตัวเก่า

ขั้นตอนการทำงานของขั้นตอนวิธีมีดังนี้

1. กำหนดค่าความน่าจะเป็นเริ่มต้นในแต่ละมิติของเวกเตอร์ความน่าจะเป็น(p) โดยที่ lคือความยาวของโครโมโซม

   for i := 1 to l do

   $p[i]$:= 0.5;

2. สร้างประชากรและประชากรที่ดีที่สุด โดยที่ *Generation* คือรุ่นของประชากร

   if (*Generation* = 1) then

   elite := generate(p);

   individual := generate(p);

3. พิจารณาหาประชากรที่มีค่าความเหมาะสมที่ดีกว่า

   winner, *loser* := compete(*elite*, *individual*);

   if (winner is better than elite)

   elite := *winner;*

4. ปรับปรุงค่าความน่าจะเป็นโดยที่ 1/*psize* คือค่าที่ใช้ปรับค่าความน่าจะเป็นในแต่ละครั้ง ซึ่งค่า *psize* สามารถเทียบเคียงได้กับจำนวนประชากรในขั้นตอนวิธีเชิงพันธุกรรมอย่างง่าย (Simple GA)

```
for i := 1 to l
begin
        if winner[i] ≠ loser[i] then
                if winner[i] = 1 then
                        p[i] := p[i] + 1/psize;
                else
                        p[i] := p[i] – 1/psize;
end
```

5. ทำซ้ำตามขั้นตอนที่ 2 ถึง 4 จนกว่าค่าความน่าจะเป็นในแต่ละมิติของเวกเตอร์ความน่าจะเป็นจะเป็น 0.0 หรือ 1.0 ทุกมิติ

ขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับที่เลือกเก็บประชากรที่ดีที่สุดแบบไม่ถาวร เป็นขั้นตอนวิธีที่นำประชากรที่ดีที่สุดมาใช้ในการปรับปรุงค่าความน่าจะเป็น เช่นเดียวกับขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับที่เลือกเก็บประชากรที่ดีที่สุดแบบถาวร แต่มีการสร้างตัวแปรขึ้นมาเพื่อใช้ในการควบคุมไม่ให้ประชากรที่ดีที่สุดเป็นประชากรที่ดีที่สุดแบบถาวร โดยกำหนดค่า *N* เป็นจำนวนรุ่นสูงสุดในการเป็นประชากรที่ดีที่สุด และกำหนดให้ *Control Parameter* เป็นตัวแปรที่ใช้ในการนับช่วงการเป็นประชากรที่ดีสุด

ในขั้นตอนการสร้างประชากรนั้น จะสร้างประชากร และประชากรที่ดีที่สุดขึ้นมาอย่างละ 1 ตัว โดยประชากรที่ดีที่สุดจะสร้างขึ้นมาในประชากรรุ่นแรกเท่านั้น และกำหนดให้ *Control Parameter* มีค่าเท่ากับ 0 หลังจากประชากรรุ่นแรกขั้นตอนวิธีจะทำการสร้างประชากรขึ้นมาเพียงตัวเดียวเท่านั้น

ในขั้นตอนการพิจารณาหาประชากรที่มีค่าความเหมาะสมที่ดีกว่านั้น ขั้นตอนวิธีได้นำประชากรกับประชากรที่ดีที่สุดมาคำนวณค่าความเหมาะสมเพื่อนำมาเปรียบเทียบหาประชากรที่มีค่าความเหมาะสมที่ดีกว่า หรือผู้ชนะ ซึ่งถ้าหากประชากรที่ดีกว่าเป็นผู้ชนะ และ *Control Parameter* ยังมีค่าไม่ถึงจำนวนรุ่นสูงสุดในการเป็นประชากรที่ดีที่สุดที่กำหนดไว้แล้วนั้น ให้ *Control Parameter* มีค่าเพิ่มขึ้น 1 แต่ถ้าประชากรที่ดีที่สุดอยู่มาจนครบ *N* รุ่นแล้ว ขั้นตอนวิธีจะทำการสร้างประชากรที่ดีที่สุดขึ้นมาใหม่โดยสุ่มแต่ละบิตในโครโมโซมด้วยค่าความน่าจะเป็น 0.5 และกำหนดค่า *Control Parameter* กลับไปเป็น 0

โดยขั้นตอนการทำงานของขั้นตอนวิธีมีดังนี้

1. กำหนดค่าความน่าจะเป็นเริ่มต้นในแต่ละมิติของเวกเตอร์ความน่าจะเป็น (p) โดยที่ l คือความยาวของโครโมโซม

for*i*:= 1 to *l* do

p[*i*] := 0.5;

2. สร้างประชากร และประชากรที่ดีที่สุด พร้อมทั้งกำหนดค่าเริ่มต้นของ *Control Parameter*

if (*Generation* = 1) then

Control Parameter := 0;

elite := generate(*p*);

individual := generate(*p*);

3. พิจารณาหาประชากรที่มีค่าความเหมาะสมที่ดีกว่าโดย *N* คือ จำนวนรุ่นสูงสุดในการเป็นประชากรที่ดีที่สุด

winner, *loser* := compete(*elite*, *individual*);

if (*Control Parameter* <= *N* and *winner* = *elite*) then

elite := *winner;*

Control Parameter++;

else

elite = generate(with prob = 0.5);

Control Parameter := 0;

4. ปรับปรุงค่าความน่าจะเป็น โดยที่ 1/*psize* คือค่าที่ใช้ปรับค่าความน่าจะเป็นในแต่ละครั้ง ซึ่งค่า *psize* สามารถเทียบเคียงได้กับจำนวนประชากรในขั้นตอนวิธีเชิงพันธุกรรมอย่างง่าย (Simple GA)

for *i* :=1 to *l*

begin

if *winner*[*i*] ≠ *loser*[*i*] then

if *winner*[*i*] = *1* then

p[*i*] := *p*[*i*] + 1/*psize*;

else

p[*i*] := *p*[*i*] – 1/*psize*;

end

5. ทำซ้ำตามขั้นตอนที่ 2 ถึง 4 จนกว่าค่าความน่าจะเป็นในแต่ละมิติของเวกเตอร์ความน่าจะเป็นจะเป็น 0.0 หรือ 1.0 ทุกมิติ

และในปี ค.ศ. 2006 Sunisa Rimcharoen และคณะ ได้นำเสนอขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับด้วยค่าเฉลี่ย (Moving Average Compact Genetic Algorithm: mcGA) ในการปรับปรุงขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับ โดยใช้ค่าเฉลี่ยของค่าความน่าจะเป็นในการสร้างประชากรรุ่นถัดไป ซึ่งค่าเฉลี่ยนี้ได้มาจากการนำค่าความน่าจะเป็นที่เกิดขึ้นจากประชากรแต่ละรุ่นมารวมกันและคำนวณเป็นค่าเฉลี่ยออกมา โดยขั้นตอนวิธีมีการกำหนดจำนวนรุ่นที่ใช้ในการจัดเก็บค่าความน่าจะเป็นเอาไว้เรียกว่า Window Size ซึ่งขั้นตอนวิธีได้ทำการปรับปรุงขั้นตอนการทำงานของขั้นตอนวิธีเชิงพันธุกรรมในขั้นตอนการปรับปรุงค่าความน่าจะเป็น ซึ่งแยกได้เป็นสองการทำงาน ได้แก่ คำนวณอัตราการปรับปรุงค่าความน่าจะเป็น และคำนวณอัตราการปรับปรุงค่าความน่าจะเป็นโดยเฉลี่ย โดยขั้นตอนการทำงานของขั้นตอนวิธีมีดังนี้

1. กำหนดค่าความน่าจะเป็นเริ่มต้นในแต่ละมิติของเวกเตอร์ความน่าจะเป็น (p) โดยที่ l คือความยาวของโครโมโซม

   for i := 1 to l do

   $p[i]$:= 0.5;

2. สร้างประชากร

   individual1 := generate(p);

   individual2 := generate(p);

3. ประเมินค่าความเหมาะสม

   winner, *loser* := compete(*individual1*, *individual2*);

4. ปรับปรุงค่าความน่าจะเป็น

   4.1. คำนวนอัตราการปรับปรุงค่าความน่าจะเป็นในแต่ละมิติของเวกเตอร์ความน่าจะเป็น (q)

   for i := 1 to l do

   if $winner[i] \neq loser[i]$ then

   if $winner[i]$ = 1 then $q[i] := q[i] + 1/n$

   else $q[i] := q[i] - 1/n$

   4.2. คำนวนอัตราการปรับปรุงค่าความน่าจะเป็นโดยเฉลี่ยในแต่ละมิติของเวกเตอร์ความน่าจะเป็น (p) โดย *movavg* คือค่าความน่าจะเป็นสะสม และ M คือขนาดของ Window Size

```
for i := 1 to I do
        for m := 1 to M do
                movavg := movavg + q[i][m]
        movavg := movavg / M;
p[i] := movage;
```

5. ทำซ้ำตามขั้นตอนที่ 2 ถึง 4 จนกว่าค่าความน่าจะเป็นในแต่ละมิติของเวกเตอร์ความน่าจะเป็นจะเป็น 0.0 หรือ 1.0 ทุกมิติ

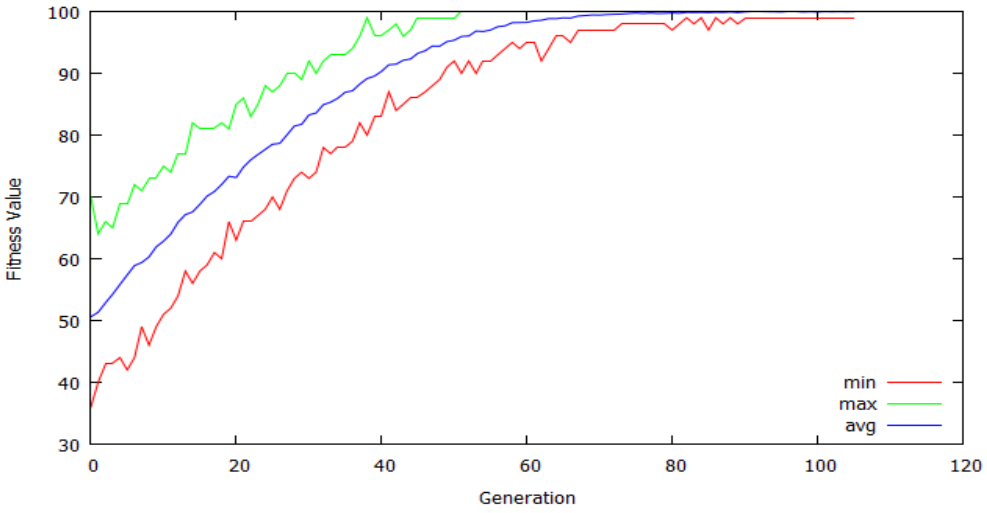
บทที่ 3

การศึกษาและทดลองขั้นตอนวิธีการเรียนรู้เพิ่มขึ้นแบบอาศัยประชากร

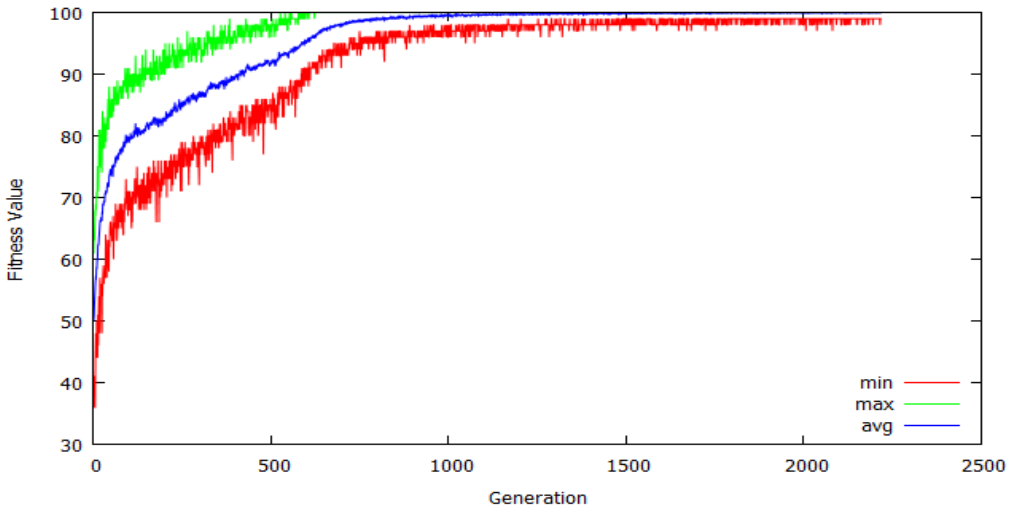
การดำเนินงานเบื้องต้น ได้ศึกษาและทดลองทดสอบขั้นตอนวิธีการเรียนรู้เพิ่มขึ้นแบบอาศัยประชากรกับปัญหาจำนวนบิตหนึ่งมากสุด (ตัวแทนปัญหาง่าย) และปัญหากับดัก (ตัวแทนปัญหายาก) เพื่อสังเกตพฤติกรรมในการค้นคำตอบของขั้นตอนวิธีดังกล่าว ในกรณีที่ใช้วิธีการปรับค่าความน่าจะเป็นโดยอาศัยคำตอบดีสุดเพียงตัวเดียว และการใช้คำตอบที่แย่ที่สุดร่วมด้วย ในการทดลอง ผู้วิจัยได้กำหนดจำนวนประชากรที่สุ่มคำตอบขึ้นมาในแต่ละรุ่น คือ 50, กำหนดอัตราการเรียนรู้ (α) คือ 0.1 โดยในการทดลองไม่ได้กำหนดให้มีการกลายพันธุ์ ส่วนจำนวนรอบสูงสุดของการทำงานกำหนดไว้ที่ 5000 รอบ (แต่จะหยุดก่อนถ้าค่าในเวกเตอร์ความน่าจะเป็นมีค่าเป็น 1.0 หรือ 0.0 แล้ว) ผลการทดลองแสดงดังต่อไปนี้

การทดลองกับปัญหาจำนวนบิตหนึ่งมากสุด (OneMax)

ในการทดลองได้กำหนดจำนวนบิตไว้ คือ 100 บิต ผลการทดลองพบว่าทั้งการใช้คำตอบที่ดีที่สุดเพียงอย่างเดียวในการปรับปรุงค่าเวกเตอร์ความน่าจะเป็น และการใช่คำตอบตัวที่แย่ที่สุดร่วมด้วยพบคำตอบที่ดีที่สุดทั้งคู่ แต่การใช้คำตอบที่ดีที่สุดเพียงอย่างเดียวในการปรับปรุงค่าเวกเตอร์ความน่าจะเป็นจะเจอคำตอบได้เร็วกว่า รูปที่ 3.1 และ 3.2 แสดงพฤติกรรมการลู่เข้าสู่คำตอบของทั้งสองวิธี โดยรูปที่ 3.1 คือ รูปการลู่เข้าของการใช้คำตอบที่ดีที่สุดในการปรับปรุงค่าเวกเตอร์ความน่าจะเป็น ส่วนรูปที่ 3.2 คือ รูปการลู่เข้าเมื่อมีการใช้คำตอบแย่สุดร่วมด้วย

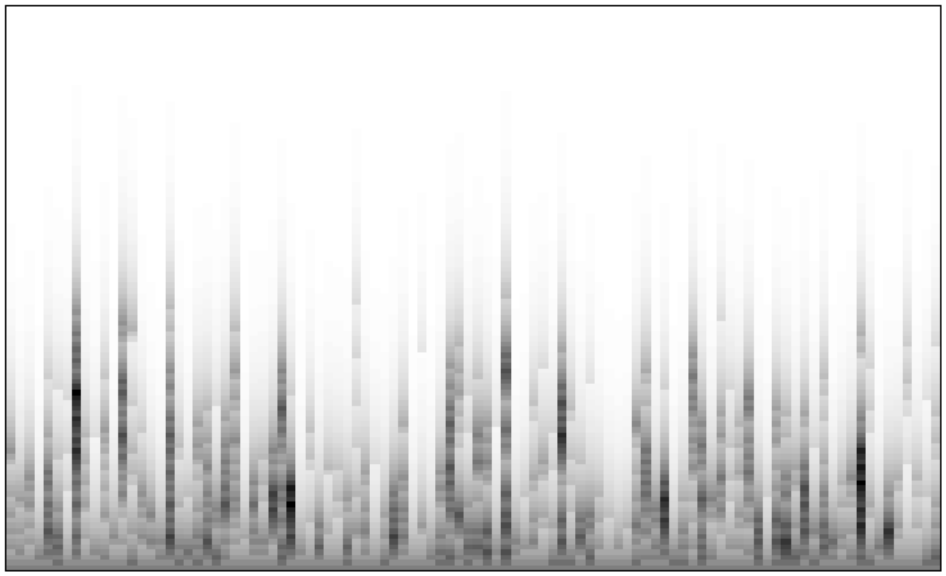


รูปที่ 3.1 การลู่เข้าของการใช้คำตอบที่ดีที่สุดในการปรับปรุงค่าเวกเตอร์ความน่าจะเป็น

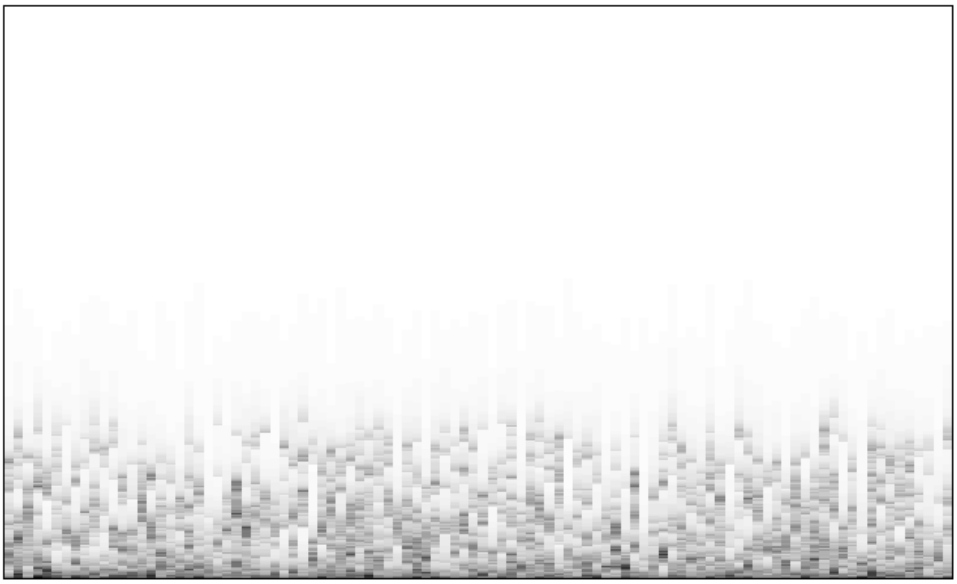


รูปที่ 3.2 การลู่เข้าของการใช้คำตอบแย่สุดร่วมด้วยในการปรับปรุงค่าเวกเตอร์ความน่าจะเป็น

เมื่อพิจารณาการเปลี่ยนแปลงของค่าความน่าจะเป็นแต่ละตำแหน่งในเวกเตอร์ความน่าจะเป็น จะพบว่าการใช้คำตอบที่ดีที่สุดเพียงอย่างเดียวในการปรับปรุงค่าเวกเตอร์ความน่าจะเป็น จะทำให้เวกเตอร์ความน่าจะเป็นลู่เข้าสู่ค่า 1.0 ค่อนข้างเร็ว ในขณะที่ถ้าใช้คำตอบที่แย่สุดพิจารณาร่วมด้วย การเปลี่ยนแปลงของค่าความน่าจะเป็นจะมีการเพิ่มขึ้น/ลดลงอยู่นานก่อนที่จะลู่เข้าสู่ค่า 1.0 รูปที่ 3.3 และ 3.4 แสดงให้เห็นถึงค่าการเปลี่ยนแปลงของเวกเตอร์ความน่าจะเป็นในแต่ละตำแหน่งเมื่อมีการทำงานผ่านไปในรุ่นต่างๆ สีดำแทนค่า 0.0 และสีขาวจะแทนค่า 1.0 โดยเวกเตอร์ความน่าจะเป็นของรุ่นแรกจะอยู่ด้านล่างสุด ไล่ขึ้นมาจนรุ่นสุดท้ายอยู่บนสุด โดยรูปที่ 3.3 แสดงการเปลี่ยนแปลงค่าเวกเตอร์ความน่าจะเป็นของการใช้คำตอบดีสุดในการปรับค่า และรูปที่ 3.4 เป็นการใช้คำตอบแย่สุดร่วมด้วย



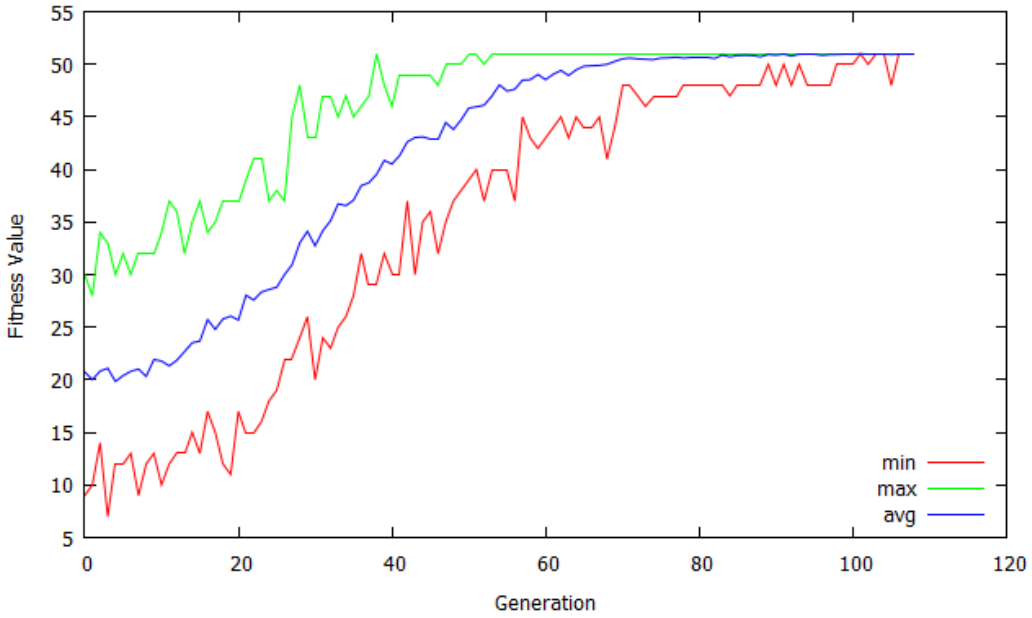
รูปที่ 3.3 การเปลี่ยนแปลงค่าเวกเตอร์ความน่าจะเป็นของการใช้คำตอบดีสุดในการปรับค่า



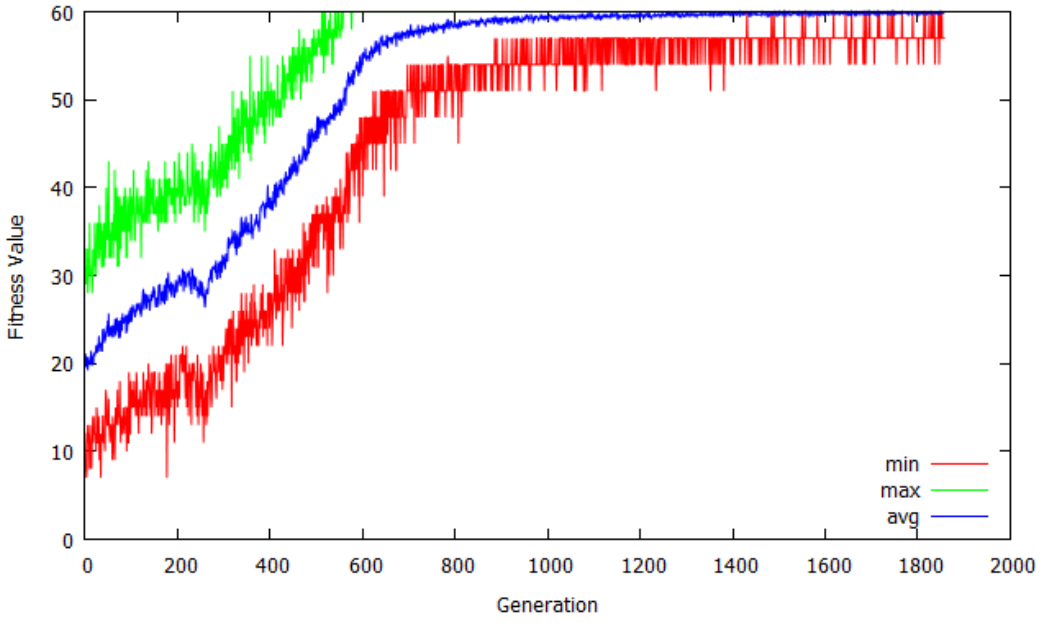
รูปที่ 3.4 การเปลี่ยนแปลงค่าเวกเตอร์ความน่าจะเป็นโดยใช้คำตอบแย่สุดร่วมด้วย

การทดลองกับปัญหากับดัก (Trap)

ในการทดลอง ผู้วิจัยได้กำหนดจำนวนบิตไว้ คือ 60 บิต (ปัญหากับดักขนาด 3 จำนวนทั้งสิ้น 20 ชุดต่อกัน) ผลการทดลองพบว่าการใช้คำตอบที่ดีที่สุดเพียงอย่างเดียวในการปรับปรุงค่าเวกเตอร์ความน่าจะเป็นไม่เจอคำตอบที่ดีที่สุด แต่วิธีการที่ใช้คำตอบตัวที่แย่สุดร่วมด้วยสามารถพบคำตอบที่ดีที่สุด ดังจะเห็นได้จากรูปที่ 3.5 การลู่เข้าของการใช้คำตอบที่ดีที่สุดในการปรับปรุงค่าเวกเตอร์ความน่าจะเป็น จะลู่เข้าเร็วกว่าแต่ในตอนท้ายของการค้นหาคำตอบก็ได้คำตอบที่มีค่าความเหมาะสมแย่กว่า ส่วนรูปที่ 3.6 เป็นผลลัพธ์จากการใช้คำตอบแย่สุดร่วมด้วยแม้ว่าจะใช้เวลานานกว่า แต่ในที่สุดก่อนที่เวกเตอร์ความน่าจะเป็นจะลู่เข้าสู่ศูนย์หรือหนึ่งทั้งหมดขั้นตอนวิธีก็พบคำตอบที่มีคุณภาพดีกว่า



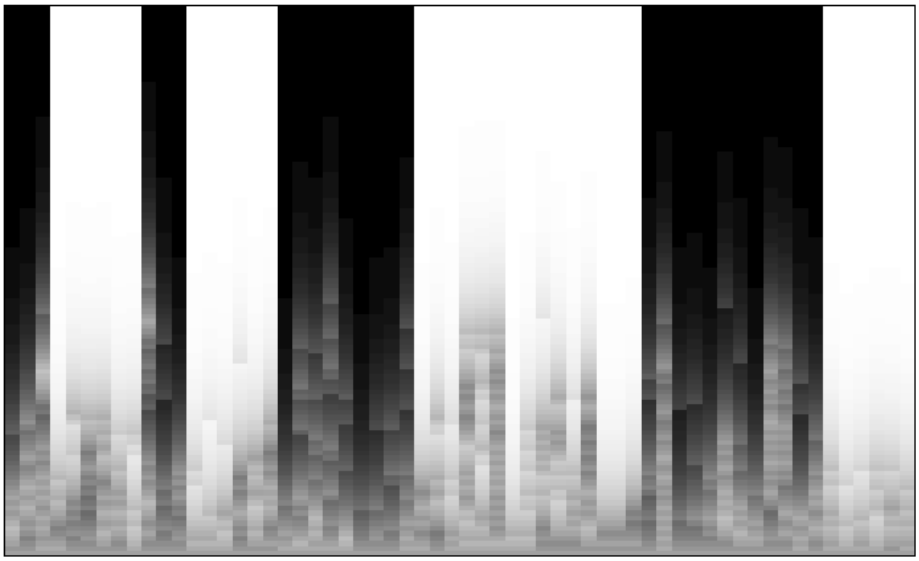
รูปที่ 3.5 การลู่เข้าจากการใช้คำตอบที่ดีที่สุดในการปรับค่า



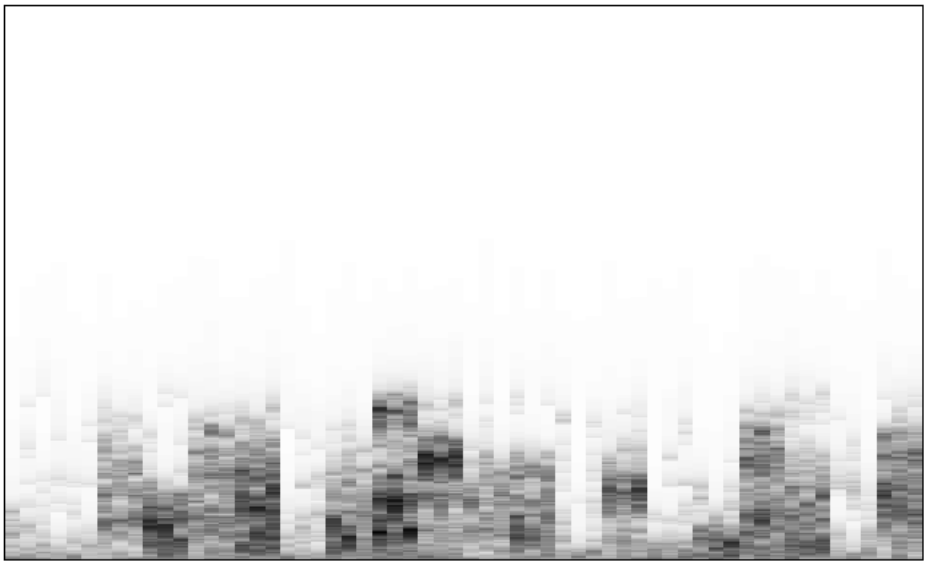
รูปที่ 3.6 การลู่เข้าจากการใช้คำตอบที่แย่ที่สุดร่วมด้วยในการปรับค่า

เมื่อพิจารณาค่าการเปลี่ยนแปลงของค่าความน่าจะเป็นแต่ละตำแหน่งในเวกเตอร์ความน่าจะเป็น จะพบว่าการใช้คำตอบที่ดีที่สุดเพียงอย่างเดียวในการปรับปรุงค่าเวกเตอร์ความน่าจะเป็น จะทำให้เวกเตอร์ความน่าจะเป็นลู่เข้าสู่ค่า 1.0 หรือ 0.0 ค่อนข้างเร็ว ในขณะที่ถ้าใช้คำตอบที่แย่สุดพิจารณาร่วมด้วย การเปลี่ยนแปลงของค่าความน่าจะเป็นจะมีการเพิ่มขึ้น/ลดลงอยู่นาน

ก่อนที่จะลู่เข้า ซึ่งนำไปสู่การค้นคำตอบและเห็นตัวอย่างจำนวนมากขึ้น ทำให้พบคำตอบที่มีคุณภาพดีกว่าการใช้คำตอบดีสุดเพียงอย่างเดียวในการปรับปรุงทิศทางการค้นคำตอบ รูปที่ 3.7 และ 3.8 แสดงให้เห็นถึงค่าการเปลี่ยนแปลงของเวกเตอร์ความน่าจะเป็น เมื่อมีการทำงานผ่านไปในรุ่นต่าง ๆ (หนึ่งแถวคือ 1 รุ่น) สีดำแทนค่า 0.0 และสีขาวจะแทนค่า 1.0 โดยเวกเตอร์ความน่าจะเป็นของรุ่นแรกจะอยู่ด้านล่างสุด ไล่ขึ้นมาจนรุ่นสุดท้ายอยู่บนสุด รูปที่ 3.7 เป็นการเปลี่ยนแปลงค่าเวกเตอร์ความน่าจะเป็นของการใช้คำตอบดีสุดในการปรับค่า และรูปที่ 3.8 เป็นการใช้คำตอบแย่สุดร่วมด้วย จากรูปจะเห็นได้ว่าแทบที่เป็นสีดำ คือ กลุ่มบิตที่เป็น 000 (ถูกหลอก) รูปที่ 3.7 ที่เป็นการใช้คำตอบดีสุดในการปรับทิศทางในการค้นหา จะถูกหลอกได้ง่ายกว่า ดังจะเห็นจากรูปได้ว่ามีจำนวนกลุ่มบิตที่ติดกับดักถึง 9 ชุด ในขณะที่รูปที่ 3.8 ที่เป็นการใช้คำตอบแย่สุดร่วมด้วย พบคำตอบที่ดีที่สุด (ไม่ติดกับดักเลย)

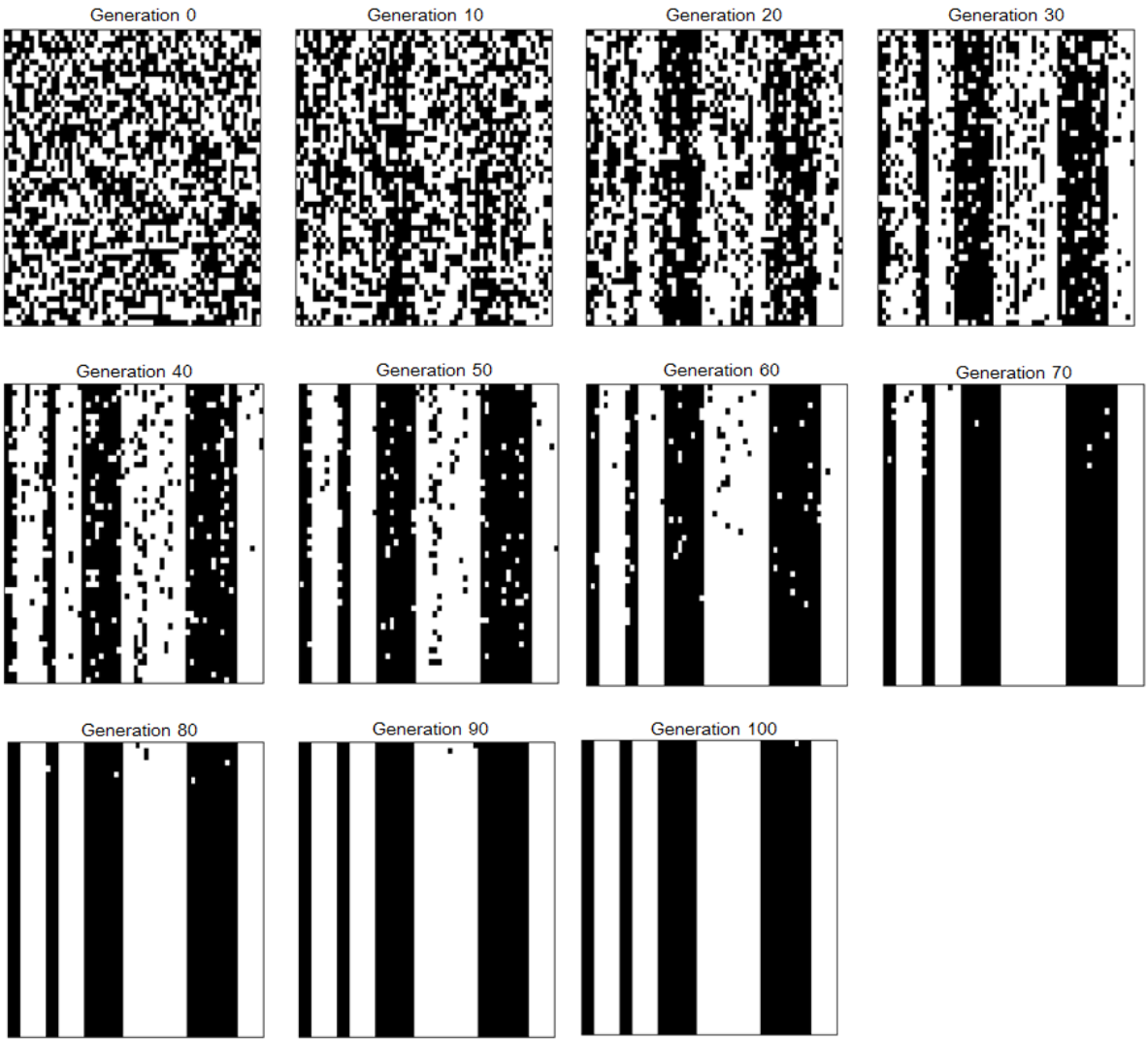


รูปที่ 3.7 การเปลี่ยนแปลงค่าเวกเตอร์ความน่าจะเป็นของการใช้คำตอบดีสุดในการปรับค่า

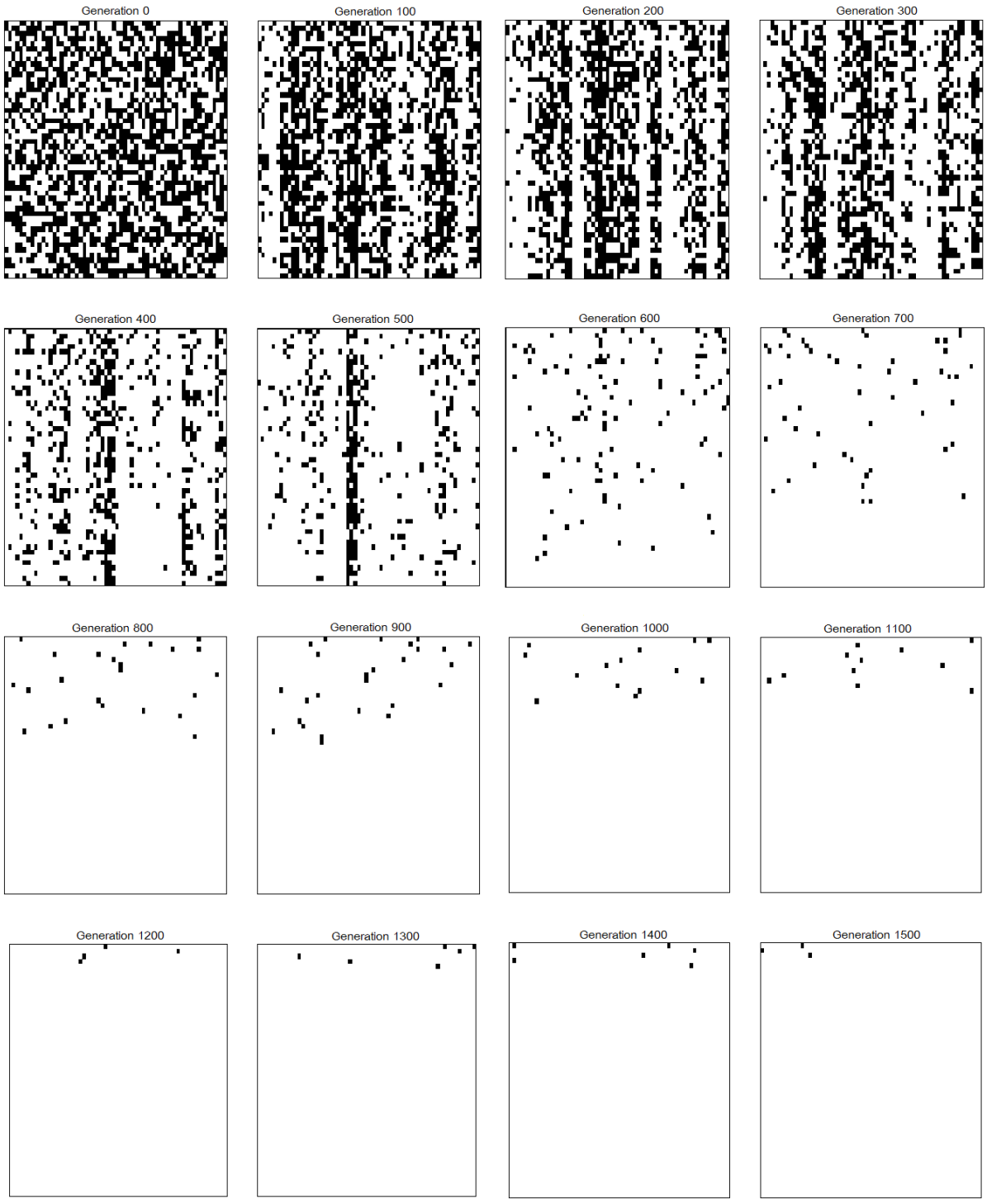


รูปที่ 3.8 การเปลี่ยนแปลงค่าเวกเตอร์ความน่าจะเป็นเมื่อใช้คำตอบแย่สุดร่วมด้วยในการปรับค่า

เมื่อทำการวิเคราะห์ประชากรที่ถูกสุ่มสร้างขึ้นมาในแต่ละรุ่น จะเห็นว่าการปรับค่าเวกเตอร์ความน่าจะเป็นโดยใช้คำตอบดีสุดเพียงอย่างเดียว ทำให้ค่าความน่าจะเป็นในแต่ละตำแหน่งลู่เข้าตามคำตอบดีสุด (ซึ่งอาจถูกหลอก) ทำให้ประชากรรุ่นหลัง ๆ หลงทางและมุ่งไปสู่กับดัก ในขณะที่ประชากรที่ถูกสุ่มจากการปรับค่าเวกเตอร์ความน่าจะเป็นโดยใช้คำตอบแย่สุดร่วมด้วย มีการปรับค่าความน่าจะเป็นขึ้น/ลง ซึ่งแม้อาจจะใช้เวลานานกว่าจะลู่เข้าสู่คำตอบ แต่ทำให้มีโอกาสที่จะเห็นคำตอบอื่นที่ดีกว่า และทำให้กระบวนการค้นหาคำตอบไม่ติดกับดัก ดังจะเห็นได้จากรูปที่ 3.9 และ 3.10 ที่แสดงประชากรแต่ละตัวที่ถูกสุ่มสร้างขึ้น (แต่ละแถวในรูปแทนประชากรแต่ละตัว) โดยตำแหน่งที่เป็นสีดำคือบิตที่เป็น 0 และสีขาวคือบิตที่เป็น 1



รูปที่ 3.9 การเปลี่ยนแปลงของประชากรที่สุ่มสร้างขึ้นเมื่อปรับค่าเวกเตอร์ความน่าจะเป็นโดยการใช้คำตอบดีสุดเพียงอย่างเดียวในปัญหา TRAP 3 x 20

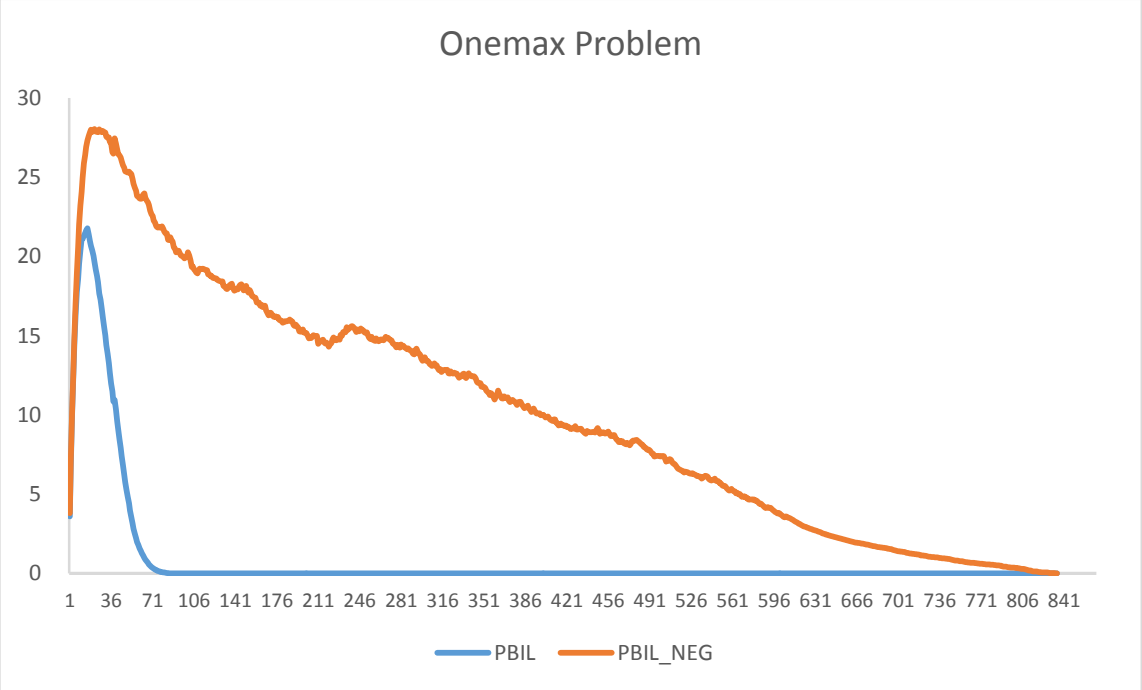


รูปที่ 3.10 การเปลี่ยนแปลงของประชากรที่สุ่มสร้างขึ้นเมื่อปรับค่าเวกเตอร์ความน่าจะเป็นโดยการใช้คำตอบดีสุดร่วมกับคำตอบแย่สุดในปัญหา TRAP 3 x 20

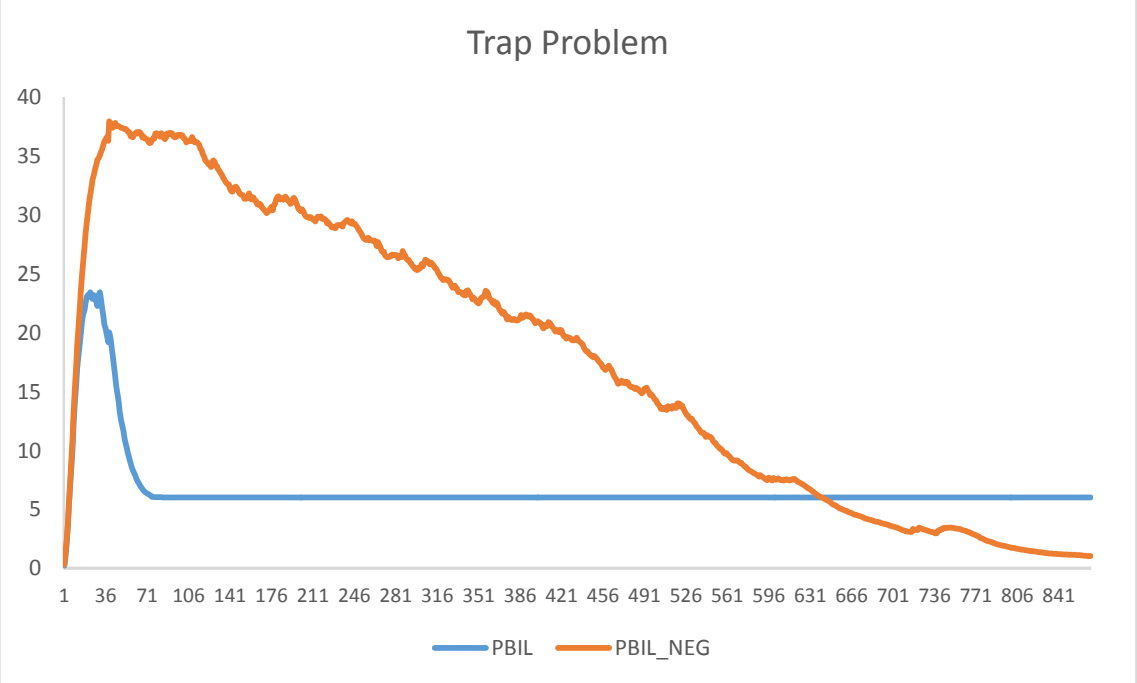
จากผลการทดลองข้างต้นแสดงให้เห็นว่า เมื่อขั้นตอนวิธีการเรียนรู้เพิ่มขึ้นแบบอาศัยประชากรใช้ความรู้จากคำตอบที่มีค่าความเหมาะสมต่ำร่วมด้วยจะทำให้สามารถพบคำตอบของปัญหายากได้ (ถึงแม้จะใช้เวลามากกว่า) ในขณะที่ถ้าใช้ความรู้จากคำตอบที่มีค่าความเหมาะสมสูงเพียงอย่างเดียวจะไม่พบคำตอบที่ดีที่สุด

ผู้วิจัยได้ทดลองหาค่าความแตกต่างในแง่มูลค่าของข้อมูลที่ใช้ในการตัดสินใจ ของการใช้ขั้นตอนวิธีการเรียนรู้เพิ่มขึ้นแบบอาศัยประชากรที่มีการใช้ความรู้จากตัวอย่างคำตอบที่มีค่าความเหมาะสมสูงเพียงอย่างเดียว เทียบกับวิธีการที่มีการใช้ความรู้จากตัวอย่างคำตอบที่มีค่าความเหมาะสมต่ำร่วมด้วย โดยมูลค่าของข้อมูลคำนวณจากการนำค่าที่ได้จากการตัดสินใจที่ดีที่สุด (ในที่นี้คือการตัดสินใจโดยที่สมมุติว่ารู้คำตอบของปัญหาอยู่แล้ว) ลบด้วย ค่าที่ได้จากการตัดสินใจของขั้นตอนวิธีที่ใช้ความรู้จากคำตอบดี ๆ เพียงอย่างเดียว และการใช้คำตอบที่มีค่าความเหมาะสมต่ำร่วมด้วย

มูลค่าของข้อมูลในที่นี้เป็นมูลค่าของความแตกต่างของค่าคาดหวัง (Expected Value) จากการตัดสินใจที่ดีที่สุดเทียบกับค่าคาดหวังจากการตัดสินใจของขั้นตอนวิธี รูปที่ 3.11 และ 3.12 แสดงมูลค่าของข้อมูลสำหรับปัญหา OneMax และ ปัญหา Trap ตามลำดับ แกนนอนของกราฟคือ generation ส่วนแกนตั้งคือค่าคาดหวังของค่าความเหมาะสม เส้นกราฟสีน้ำเงิน (เส้นที่อยู่ด้านล่าง) คือมูลค่าข้อมูลเมื่อเทียบกับการตัดสินใจที่ดีที่สุดของการใช้ขั้นตอนวิธีการเรียนรู้เพิ่มขึ้นแบบอาศัยประชากรโดยใช้คำตอบที่มีค่าความเหมาะสมสูงเพียงอย่างเดียว ส่วนเส้นสีส้ม (เส้นกราฟด้านบน) เป็นกรณีที่ใช้ความรู้จากประชากรที่มีค่าความเหมาะสมต่ำร่วมด้วย



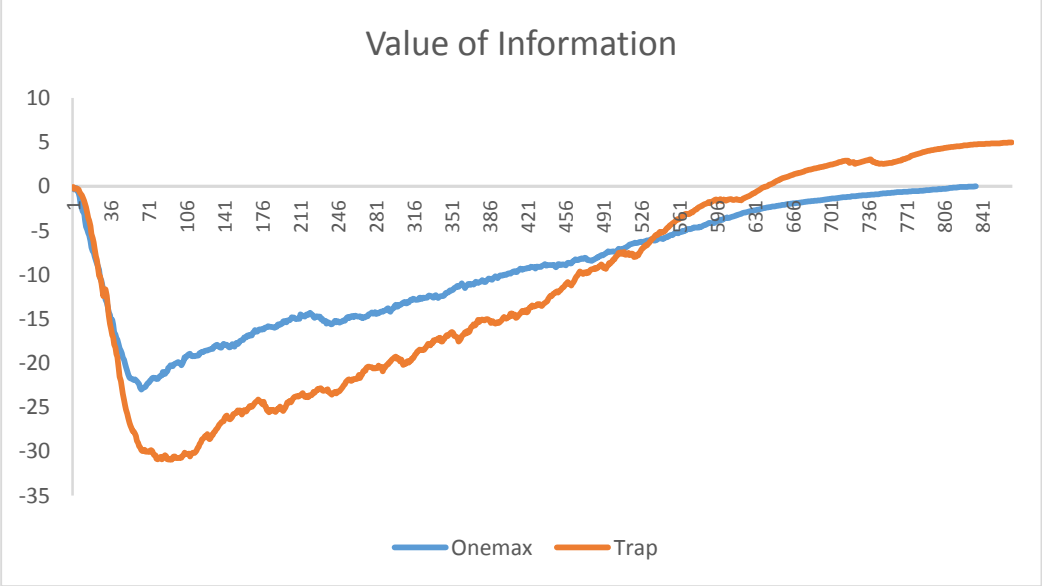
รูปที่ 3.11 มูลค่าของข้อมูลในการทดลองกับปัญหา OneMax



รูปที่ 3.12 มูลค่าของข้อมูลในการทดลองกับปัญหา Trap

จากกราฟในรูปที่ 3.11 และ 3.12 ข้างต้น จะเห็นได้ว่า ถ้าเรามีความรู้ว่าคำตอบหรือวิธีการตัดสินใจที่ดีที่สุดในแต่ละ generation เป็นอย่างไร มูลค่าข้อมูลที่ได้จากขั้นตอนวิธีที่ใช้คำตอบดี ๆ เพียงอย่างเดียว ข้อมูลความรู้นี้จะมีค่าน้อยกว่า เมื่อเทียบกับขั้นตอนวิธีที่มีการใช้ความรู้จากประชากรที่มีค่าความเหมาะสมต่ำ

เมื่อพิจารณาถึงความแตกต่างของมูลค่าข้อมูล ทำให้เห็นความแตกต่างระหว่างการใช้ขั้นตอนวิธีที่ใช้คำตอบดี ๆ เพียงอย่างเดียวกับการใช้ประชากรที่มีค่าความเหมาะสมต่ำร่วมด้วย กล่าวคือ สำหรับปัญหาง่าย (ปัญหา OneMax) มูลค่าข้อมูลจากการใช้ประชากรที่มีค่าความเหมาะสมต่ำ มีค่าติดลบตลอดการค้นหาคำตอบ (ทุก generation) ดังจะเห็นได้จากเส้นกราฟสีน้ำเงินในรูปที่ 3.13 ส่วนมูลค่าข้อมูลในกรณีที่เป็นปัญหายาก (ปัญหา Trap) ซึ่งแสดงด้วยเส้นกราฟสีส้ม ถึงแม้ว่าในช่วงต้นของกระบวนการวิวัฒนาการ ใน generation แรก ๆ มูลค่าข้อมูลจะติดลบ แต่เมื่อผ่านกระบวนการวิวัฒนาการไประยะหนึ่ง มูลค่าความรู้สำหรับปัญหายากนี้จะเป็นค่าบวก นั่นหมายถึงว่าการใช้ความรู้จากประชากรที่มีค่าความเหมาะสมต่ำร่วมด้วย จะช่วยให้ความรู้กับกระบวนการค้นหา โดยเฉพาะอย่างยิ่ง ในช่วงปลายของการค้นหาซึ่งโดยส่วนมากแล้วประชากรจะมีความหลากหลายน้อยลง การที่ขั้นตอนวิธีเชื่อความรู้จากคำตอบดี ๆ เพียงอย่างเดียว อาจพาให้อัลกอริทึมหลงทาง หรือติดกับดักได้ ในขณะที่ความรู้จากคำตอบที่มีค่าความเหมาะสมต่ำจะเข้ามามีบทบาทในการเพิ่มความหลากหลาย และพาไปสู่การค้นหาคำตอบในเส้นทางอื่นที่อาจนำไปสู่คำตอบที่ดีกว่า ดังนั้นความรู้จากคำตอบที่มีค่าความเหมาะสมต่ำนี้จึงมีมูลค่ามากกว่าคำตอบดี ๆ ในช่วงท้ายของการค้นหาคำตอบ



รูปที่ 3.13 มูลค่าของข้อมูล จากการทดลองกับปัญหา OneMax และ ปัญหา Trap

การทดลองนี้แสดงให้เห็นว่า การใช้ความรู้จากตัวอย่างคำตอบที่มีค่าความเหมาะสมต่ำจะมีมูลค่ามากกว่าความรู้จากตัวอย่างดี ๆ ในช่วงปลายของการค้นหาคำตอบสำหรับปัญหายาก ซึ่งผลการทดลองดังกล่าวนี้อาจจะเป็นจุดเริ่มต้นของการวิเคราะห์ว่าทำไมคำตอบที่ดูเหมือนว่าจะไม่ดีในช่วงแรกๆ (และขั้นตอนวิธีส่วนใหญ่เลือกที่จะคัดคำตอบเหล่านี้ทิ้ง) กลับให้มูลค่าความรู้ที่สูงกว่าเมื่อทำงานไปสักระยะหนึ่ง และถ้าเราสามารถค้นพบวิธีการที่จะนำความรู้จากคำตอบที่ดูเหมือนแย่เหล่านี้ออกมาใช้ได้ตั้งแต่ช่วงต้นของกระบวนการวิวัฒนาการ เราจะได้ความรู้ใหม่และขั้นตอนวิธีใหม่ในการค้นคำตอบที่เจอคำตอบเร็วกว่าเดิม และมีโอกาสพบคำตอบที่ดีที่สุดด้วย

บทที่ 4

การศึกษาและทดลองขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับ

งานวิจัยนี้นำเสนอ ขั้นตอนวิธีการปรับปรุงเวกเตอร์ความน่าจะเป็นของขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับด้วยค่าความถี่ (fb-cGA) โดยในการปรับปรุงค่าความน่าจะเป็นในแต่ละมิติของเวกเตอร์ความน่าจะเป็นนั้นมีความไม่แน่นอนเกิดขึ้นแตกต่างกันไป บางมิติวิ่งเข้าหา 1.0 อย่างต่อเนื่อง บางมิติวิ่งเข้าหา 0.0 อย่างต่อเนื่อง แต่บางมิติที่ยังมีความไม่แน่นอนอยู่มากนั้นไม่สามารถที่จะตัดสินใจได้ว่ามีทิศทางไปทางใด ในระหว่างการทำงานของขั้นตอนวิธี จะมีการจัดเก็บข้อมูลค่าความถี่ ในการปรับปรุงค่าความน่าจะเป็นของแต่ละมิติในเวกเตอร์ความน่าเป็นเพื่อนำมาใช้เป็นข้อมูลประกอบการตัดสินใจในการปรับปรุงค่าความน่าจะเป็นในแต่ละมิติ โดยค่าความถี่ในการปรับปรุงค่าความน่าจะเป็นนี้ สามารถที่จะบอกได้ถึงทิศทางของการปรับปรุงค่าความน่าจะเป็นในมิตินั้น ๆ ว่ามีทิศทางไปในทางใดมากกว่ากัน ระหว่างการปรับค่าความน่าจะเป็นเข้าหา 1.0 และ 0.0 ถ้าหากในการปรับปรุงค่าความน่าจะเป็นในครั้งนั้นมีทิศทางที่ตรงกันกับค่าความถี่ของการปรับปรุงค่าความน่าจะเป็นแล้ว ก็มีความเป็นไปได้ว่าในการปรับปรุงค่าความน่าจะเป็นครั้งนี้เป็นไปในทิศทางของตัวอย่างคำตอบดี ๆ ส่วนใหญ่

นอกจากนี้ ยังเสนอวิธีการใช้การจัดเก็บข้อมูลค่าความต่อเนื่องในการปรับปรุงค่าความน่าจะเป็นของแต่ละมิติในเวกเตอร์ความน่าจะเป็น เพื่อนำมาใช้ในการปรับปรุงค่าความน่าจะเป็นในแต่ละมิติของเวกเตอร์ความน่าจะเป็น โดยค่าความต่อเนื่องในการปรับปรุงค่าความน่าจะเป็นนี้สามารถที่จะบอกได้ถึงความขัดแย้งที่เกิดขึ้นในการปรับปรุงค่าความน่าจะเป็นในแต่ละมิติ ซึ่งถ้าหากในมิติใดมีค่าความต่อเนื่องในการปรับปรุงค่าความน่าจะเป็นน้อย แสดงว่าในมิตินั้นก็มีความขัดแย้งในการปรับปรุงค่าความน่าจะเป็นมาก และจะทำการปรับค่าความต่อเนื่องให้เป็นศูนย์ (Reset) เมื่อมีการปรับปรุงไปในทิศทางตรงกันข้าม แต่ถ้าหากในมิติใดมีค่าความต่อเนื่องในการปรับปรุงค่าความน่าจะเป็นมาก แสดงว่าในมิตินั้นก็มีความขัดแย้งในการปรับปรุงค่าความน่าจะเป็นน้อย นั่นหมายความว่าเราสามารถที่จะเชื่อได้ว่าการปรับปรุงค่าความน่าจะเป็นในมิตินั้นมีการปรับปรุงไปในทิศทางที่ถูกต้อง

4.1 การปรับปรุงวิธีปรับค่าในเวกเตอร์ความน่าจะเป็น

งานวิจัยนี้นำเสนอวิธีการปรับปรุงค่าของเวกเตอร์ความน่าจะเป็น โดยนับค่าความถี่และความต่อเนื่องในการปรับค่าความน่าจะเป็น ทั้งจำนวนครั้งในการปรับค่าความน่าจะเป็นขึ้นไปในทิศทางที่เข้าใกล้ 1 และในทิศทางที่ปรับให้ค่าความน่าจะเป็นเข้าใกล้ 0 ซึ่งจากข้อสังเกตในการนับจำนวนครั้งในการปรับค่าความน่าจะเป็นในแต่ละมิติ ทำให้สามารถออกแบบขั้นตอนวิธีในการปรับค่าของเวกเตอร์ความน่าจะเป็นได้ดังรูปที่ 4.1

Updating strategy of fb-cGA

```
1:  for i := 1 to l
2:  begin
3:      if winner[i] ≠ loser[i] then
4:          if winner[i] = 1 then
5:              Ufreq[i] := Ufreq[i] + 1;
6:              Ucon[i] := Ucon[i] + 1;
7:              Dcon [i] := 0;
8:              if (Ufreq[i] > Dfreq[i] AND Gen > (psize/3)) then
9:                  p[i] := p[i] + ((1/psize)+(p[i] * (Ucon[i]/100));
10:             else
11:                 p[i] := p[i] + (1/ psize);
12:         else
13:             Dfreq[i] := Dfreq[i] + 1;
14:             Dcon [i] := Dcon [i] + 1;
15:             Ucon[i] := 0;
16:             if (Dfreq[i] > Ufreq[i] AND Gen > (psize/3)) then
17:                 p[i] := p[i] - ((1/psize) + (p[i]*(Dcon [i] / 100));
18:             else
19:                 p[i] := p[i] - (1/psize);
20:  endfor
```

Parameters:

Ufreq : number of stepping-up updates
Dfreq : number of stepping-down updates
Ucon : number of consecutive stepping-up updates
Dcon : number of consecutive stepping-down updates
Gen : generation number

รูปที่ 4.1 Pseudo-code ของขั้นตอนวิธี fb-cGA

กรณีที่บิตของผู้ชนะมีค่าเป็น 1

ในกรณีที่ผู้ชนะมีค่าเป็น 1 ถ้าหากค่าความถี่ในการปรับปรุงค่าความน่าจะเป็นในทิศทางเข้าใกล้ 1 มากกว่าค่าความถี่ในการปรับปรุงค่าความน่าจะเป็นในทิศทางเข้าใกล้ 0 และจำนวนการวิวัฒนาการมากกว่าหนึ่งในสามของขนาดประชากรแล้ว หมายความว่าการปรับปรุงค่าความน่าจะเป็นในมิตินี้มีความน่าเชื่อถือที่ทำให้เชื่อถือได้ว่าเป็นไปในทิศทางที่ถูกต้อง และเมื่อจำนวนการวิวัฒนาการผ่านไปมากกว่าหนึ่งในสามของขนาดประชากรแล้ว ขั้นตอนวิธีนี้จะทำการปรับปรุงค่าความน่าจะเป็นด้วยค่าความต่อเนื่องในการปรับปรุงค่าของมิตินี้ แต่ถ้าหากค่าความถี่ในการปรับปรุงค่าความน่าจะเป็นขึ้นน้อยกว่าค่าความถี่ในการปรับปรุงค่าความน่าจะเป็นลง และจำนวนของการวิวัฒนาการน้อยกว่าหนึ่งในสามของขนาดประชากร ขั้นตอนวิธีนี้จะทำการปรับปรุงค่าความน่าจะเป็นด้วยอัตราการปรับปรุงปกติ

กรณีที่บิตของผู้ชนะมีค่าเป็น 0

ในกรณีที่ผู้ชนะมีค่าเป็น 0 ถ้าหากค่าความถี่ในการปรับปรุงค่าความน่าจะเป็นในทิศทางเข้าใกล้ 0 มากกว่าค่าความถี่ในการปรับปรุงค่าความน่าจะเป็นในทิศทางเข้าใกล้ 1 และจำนวนการวิวัฒนาการมากกว่าหนึ่งในสามของขนาดประชากรแล้ว หมายความว่าการปรับปรุงค่าความน่าจะเป็นในมิตินี้มีความน่าเชื่อถือที่ทำให้เชื่อถือได้ว่าเป็นไปในทิศทางที่ถูกต้อง และเมื่อจำนวนการวิวัฒนาการผ่านไปมากกว่าหนึ่งในสามของขนาดประชากรแล้ว ขั้นตอนวิธีนี้จะทำการปรับปรุงค่าความน่าจะเป็นด้วยค่าความต่อเนื่องในการปรับปรุงค่าของมิตินี้ แต่ถ้าหากค่าความถี่ในการปรับปรุงค่าความน่าจะเป็นลงน้อยกว่าค่าความถี่ในการปรับปรุงค่าความน่าจะเป็นขึ้น และจำนวนของการวิวัฒนาการน้อยกว่าหนึ่งในสามของขนาดประชากร ขั้นตอนวิธีนี้จะทำการปรับปรุงค่าความน่าจะเป็นด้วยอัตราการปรับปรุงปกติ

4.2 การศึกษาพารามิเตอร์

จากรูปที่ 4.1 จะเห็นได้ว่า ขั้นตอนวิธีในบรรทัดที่ 8 และ 16 จะมีค่าพารามิเตอร์ *psize*/3 ถูกกำหนดอยู่ ในการนี้ ผู้วิจัยจึงทำการศึกษาว่าค่าพารามิเตอร์นี้ควรจะกำหนดเป็นเท่าไรถึงจะเหมาะสม การทดลองนี้จึงทดสอบด้วยค่าต่าง ๆ กล่าวคือ *psize*/2, *psize*/3, *psize*/4 และ *psize*/5 โดยทดสอบกับปัญหาทดสอบทั้ง 4 ปัญหาที่ได้กล่าวรายละเอียดไว้ในบทที่ 2 ผลการทดลองแสดงดังตารางที่ 4.1

ตารางที่ 4.1 ค่า Efficiency ratio จากการทดลองปรับค่าพารามิเตอร์รูปแบบต่าง ๆ

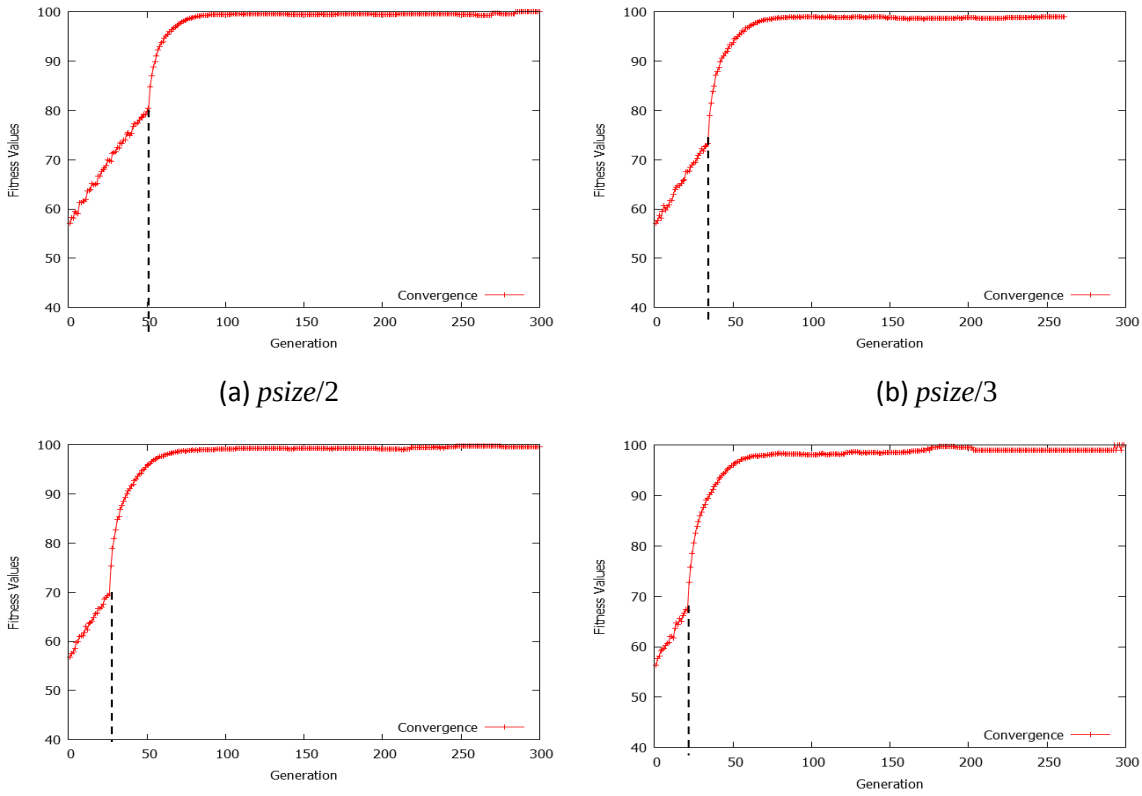
Problem		Efficiency ratio				Average
		One-Max	Random Max	Royal Road	Trap	
Tournament Size 2	*psize*/2	56.63	32.51	4.97	0.37	23.62
	psize/3	53.14	32.23	5.65	0.42	22.86
	psize/4	56.79	33.68	5.35	0.50	24.08
	psize/5	54.48	34.00	5.35	0.52	23.59
Tournament Size 4	*psize*/2	69.08	41.56	10.03	0.80	30.37
	psize/3	78.88	41.61	10.53	0.95	32.99
	psize/4	89.20	44.76	9.87	0.95	36.20
	psize/5	80.36	49.07	9.65	1.03	35.03
Tournament Size 8	*psize*/2	86.95	42.38	17.51	0.69	36.89
	psize/3	83.52	40.06	14.47	0.91	34.74
	psize/4	87.17	47.69	16.01	1.06	37.99
	psize/5	91.19	47.28	14.03	1.15	38.42

ข้อมูลตัวเลขที่ปรากฏอยู่ในตารางที่ 4.1 คือ ค่า Efficiency Ratio ซึ่งผู้วิจัยได้ใช้เป็นตัววัดประสิทธิภาพเพื่อเปรียบเทียบผลลัพธ์ที่ได้จากขั้นตอนวิธี เมื่อกำหนดพารามิเตอร์เป็นค่าต่างๆ โดยค่า Efficiency Ratio นี้คำนวณตามสมการที่ 4.1

$$\text{Efficiency Ratio} = (\text{solution quality} / \text{number of evaluations}) \times 1000 \qquad (4.1)$$

จากผลการทดลองในตารางที่ 4.1 จะเห็นได้ว่าสำหรับปัญหาง่ายอย่างเช่นปัญหา OneMax และ ปัญหา RandomMax ค่า Efficiency Ratio จะค่อนข้างดีเมื่อกำหนด *psize*/4 และ *psize*/5 แต่ในปัญหาที่ยากขึ้นมา เช่น ปัญหารอยัลโรด การกำหนดค่า *psize*/2 และ *psize*/3 ให้ค่า Efficiency Ratio ที่ดีกว่า แต่ถ้าพิจารณาในกรณีเฉลี่ย จะเห็นได้ว่าค่า *psize*/4 ให้ผลลัพธ์ที่ดีที่สุดจากการทดลองนี้

คำถามหนึ่งที่เกิดขึ้นในระหว่างการศึกษาวิจัยก็คือ ค่า *psize*/*n* มีความสำคัญอย่างไรและส่งผลต่อการทำงานของขั้นตอนวิธีที่นำเสนออย่างไรบ้าง ผู้วิจัยขออธิบายความสำคัญของพารามิเตอร์ตัวนี้โดยแสดงกราฟการเปลี่ยนแปลงของค่าความเหมาะสมในระหว่างที่ขั้นตอนวิธีทำงานด้วยการกำหนดค่าพารามิเตอร์ที่ต่างกัน ดังรูปที่ 4.2



(a) *psize*/2 (b) *psize*/3

(c) *psize*/4 (d) *psize*/5

รูปที่ 4.2 กราฟการลู่เข้าเมื่อกำหนดพารามิเตอร์ต่าง ๆ

จากรูปที่ 4.2 ให้สังเกตที่ตำแหน่งเส้นประ ณ ตำแหน่งนั้น คือ ตำแหน่งเริ่มต้นที่ขั้นตอนวิธีจะทำการตัดสินใจใช้การปรับปรุงค่าความน่าจะเป็นในเวกเตอร์ความน่าจะเป็นตามที่นำเสนอ ซึ่งค่ากำหนดค่า *n* ไว้น้อย การเริ่มต้นตัดสินใจก็จะช้า (แต่ก็มีข้อดีในแง่ที่ขั้นตอนวิธีจะเห็นตัวอย่างจำนวนมาก ซึ่งอาจนำไปสู่การตัดสินใจที่ดีขึ้น) แต่ถ้ากำหนดค่า *n* มาก การตัดสินใจก็จะเกิดใน Generation แรก ๆ ของการหาคำตอบ ซึ่งมีข้อดีในแง่ของการประหยัดจำนวนครั้งในการประเมินค่าความเหมาะสม แต่ก็อาจนำไปสู่การตัดสินใจที่ผิดพลาด เนื่องจากอาจจะตัดสินใจเร็วเกินไปและยังเห็นตัวอย่างคำตอบไม่มากนัก

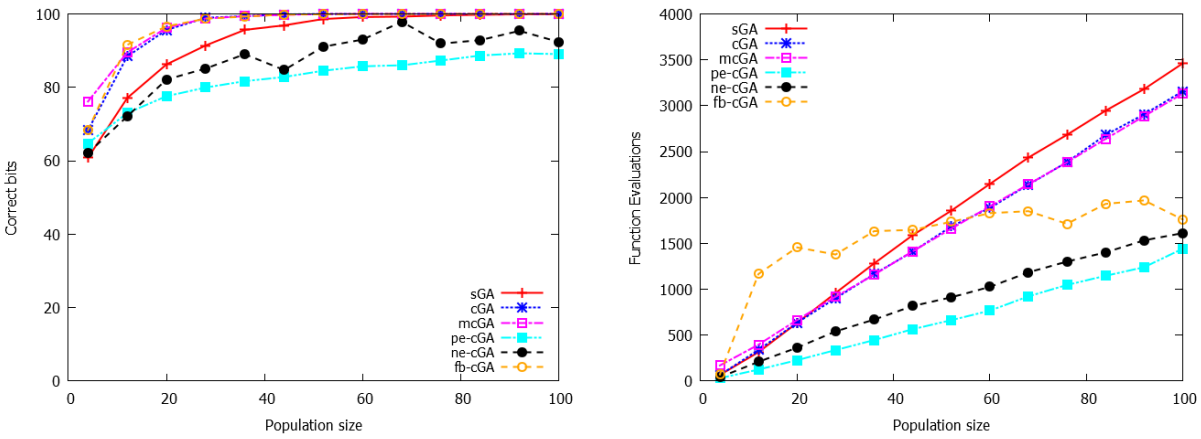
4.3 การเปรียบเทียบประสิทธิภาพ

สำหรับการทดลองเพื่อวัดประสิทธิภาพการค้นคำตอบ ของขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับด้วยค่าความถี่เทียบกับวิธีการปรับปรุงขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับจากงานวิจัยก่อนหน้า โดยได้ทำการทดลองกับปัญหา OneMax, RandomMax, Royal Road และปัญหา Trap และวัดผลลัพธ์ในแง่ของคุณภาพคำตอบ (ความถูกต้องของผลลัพธ์) กับในแง่ของต้นทุนการคำนวณที่เสียไป (นับจากจำนวนครั้งในการประเมินค่าความเหมาะสม) ซึ่งผลการทดลองมีดังนี้

4.3.1 การทดลองกับปัญหา OneMax

ในการทดลองนี้ กำหนดความยาวของโครโมโซม คือ 100 บิต และได้ทดลองปรับเปลี่ยนค่าจำนวนประชากรที่ขนาดต่าง ๆ ผลการทดลองแสดงดังรูปที่ 4.3

จากรูปที่ 4.3 จะเห็นได้ว่าขั้นตอนวิธีเกือบทุกวิธีสามารถค้นหาคำตอบที่ดีที่สุดพบยกเว้นขั้นตอนวิธี pe-cGA กับ ne-cGA ที่ไม่ได้คำตอบที่ดีที่สุด แต่ถ้าสังเกตในแง่ของจำนวนครั้งในการประเมินค่าความเหมาะสม จะพบว่าสองวิธีนี้ใช้จำนวนครั้งในการประเมินค่าความเหมาะสมน้อยที่สุด ในขณะที่ขั้นตอนวิธีที่นำเสนอ คือ fb-cGA ใช้จำนวนครั้งในการประเมินค่าความเหมาะสมอยู่กลาง ๆ แต่เมื่อคำนึงถึงว่าขั้นตอนวิธีนี้สามารถค้นพบคำตอบที่ดีที่สุดได้ก็น่าจะเป็นทางเลือกที่ดี



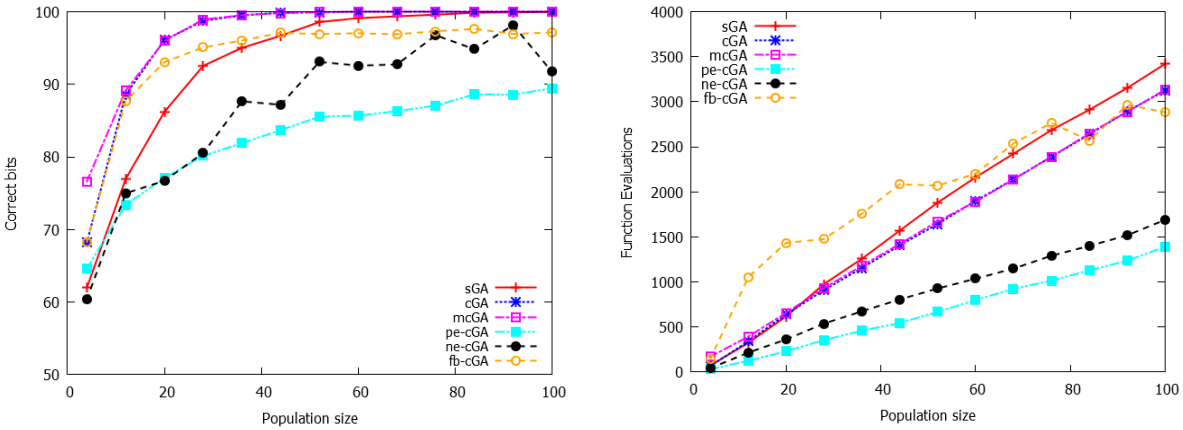
(ก) solution quality (ข) the number of function evaluations

รูปที่ 4.3 ผลการทดลองสำหรับปัญหา OneMax

4.3.2 การทดลองกับปัญหา RandomMax

ในการทดลองนี้ ได้กำหนดความยาวของโครโมโซม คือ 100 บิต เช่นเดียวกับปัญหา OneMax ข้างต้น ผลการทดลองในรูปที่ 4.4 แสดงผลลัพธ์ในแง่คุณภาพคำตอบที่ได้ เทียบกับจำนวนครั้งในการประเมินค่าความเหมาะสม

จะเห็นได้ว่า สำหรับปัญหานี้ ขั้นตอนวิธีที่นำเสนอมีการใช้จำนวนครั้งในการประเมินค่าความเหมาะสมมากขึ้นเมื่อเทียบกับปัญหา OneMax



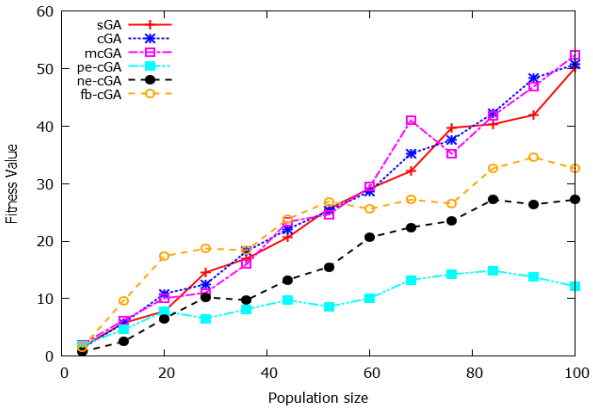
(ก) solution quality (ข) the number of function evaluations

รูปที่ 4.4 ผลการทดลองสำหรับปัญหา RandomMax

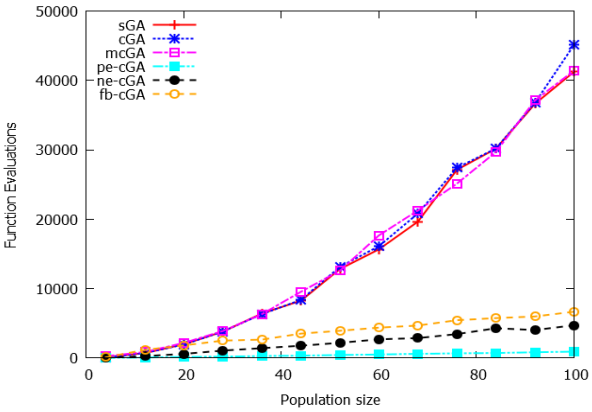
4.3.3 การทดลองกับปัญหา Royal Road

ในการทดลองนี้ ได้ทดสอบขั้นตอนวิธีต่าง ๆ กับปัญหารอยัลโรดขนาด 64 บิต โดยทำการปรับค่าจำนวนประชากรด้วยค่าต่าง ๆ พร้อมกับกำหนด tournament size ขนาด 2, 4 และ 8 ตามลำดับ ผลการทดลองแสดงในรูปที่ 4.5

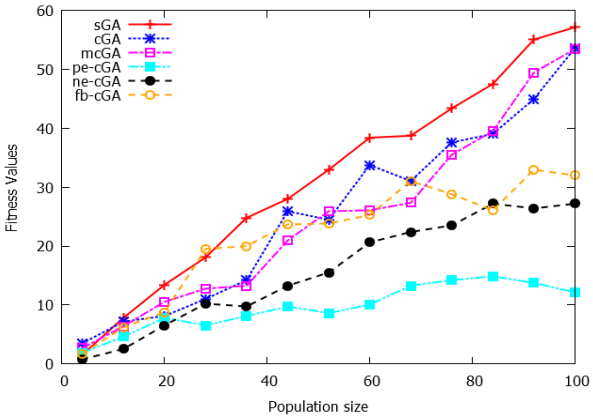
จากรูปที่ 4.5 จะเห็นได้ว่า ขั้นตอนวิธี fb-cGA ที่นำเสนอ ให้ค่าความถูกต้องอยู่ในระดับกลาง แต่เมื่อพิจารณากราฟทางขวา จะพบว่า fb-cGA ใช้จำนวนครั้งในการประเมินค่าความเหมาะสมน้อยกว่าหลาย ๆ ขั้นตอนวิธี



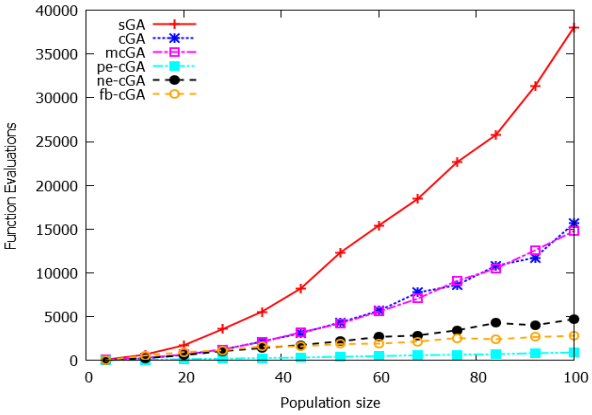
(a) solution quality (tournament size 2)



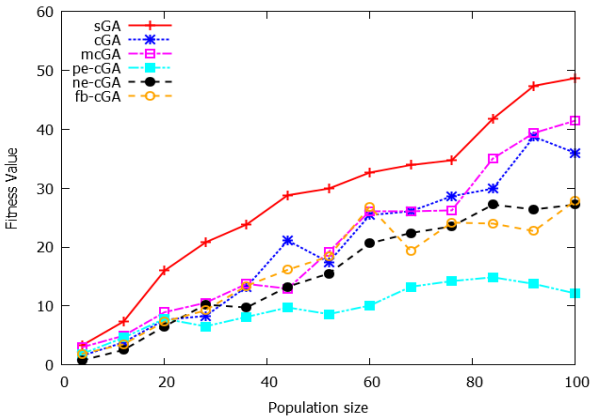
(b) the number of function evaluations (tournament size 2)



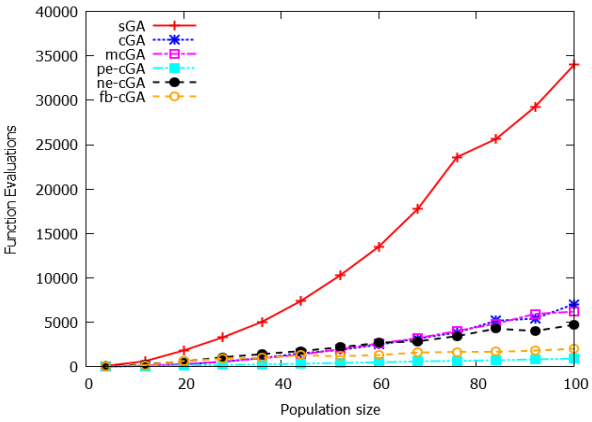
(c) solution quality (tournament size 4)



(d) the number of function evaluations (tournament size 4)



(e) solution quality (tournament size 8)



(f) the number of function evaluations (tournament size 8)

รูปที่ 4.5 ผลการทดลองสำหรับปัญหา Royal Road

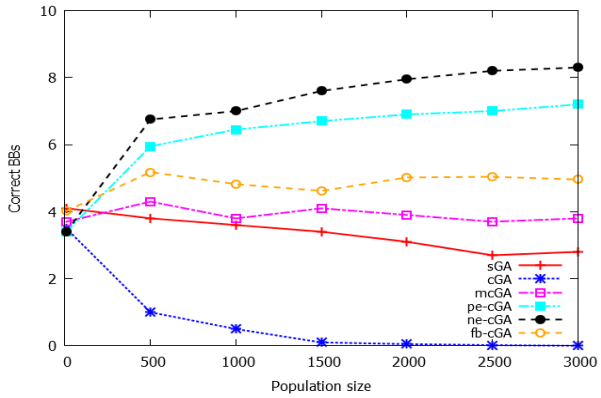
4.3.4 การทดลองกับปัญหา Trap

ในการทดลองนี้ ได้ทดสอบขั้นตอนวิธีต่าง ๆ กับปัญหา Trap ขนาด 3 x 10 (กับดักขนาด 3 บิต นำมาต่อกัน 10 ชุด) ทำการปรับเปลี่ยนค่าจำนวนประชากรขนาด 8, 500, 1000, 1500, 2000 และ 3000 ตามลำดับ รวมทั้งทำการเปลี่ยนค่า tournament size ขนาด 2, 4 และ 8 เพื่อศึกษาผลลัพธ์ที่ได้เมื่อมีการปรับพารามิเตอร์ ผลการทดลองแสดงดังรูปที่ 4.6

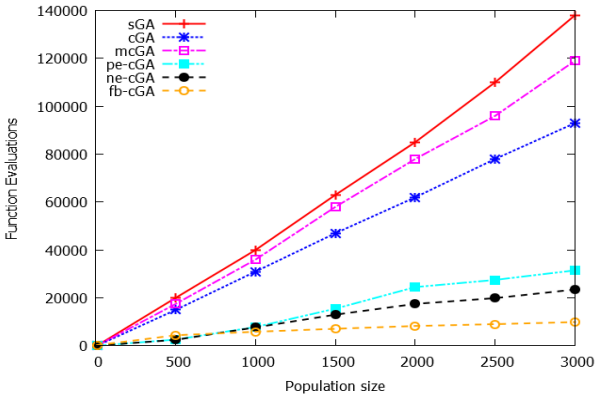
จากรูปที่ 4.6 กราฟทางด้านซ้าย เป็นคุณภาพคำตอบที่ได้จากขั้นตอนวิธีต่าง ๆ โดยนำเสนอเป็นจำนวน Building Block ที่ถูกต้อง นั่นก็คือ นับจำนวนกลุ่มบิตที่ตรงกับคำตอบที่ต้องการ (ในที่นี้คือแต่ละกลุ่มจะต้องเป็น 111) จากภาพรวมของกราฟ จะเห็นได้ว่า ขั้นตอน fb-cGA ที่นำเสนอได้ผลลัพธ์คุณภาพคำตอบค่อนข้างดี โดยเฉพาะเมื่อพิจารณาเทียบกับต้นทุนการคำนวณที่ใช้ไป (กราฟทางขวา) ขั้นตอนวิธีที่นำเสนอนี้ ใช้จำนวนครั้งในการประเมินค่าความเหมาะสมน้อยที่สุด เมื่อเปรียบเทียบกับขั้นตอนวิธีอื่น

ตัวอย่างเช่น เมื่อกำหนด tournament size เท่ากับ 2 รูปกราฟด้านบนสุด ขั้นตอนวิธีที่นำเสนอให้ผลลัพธ์ดีกว่าขั้นตอนวิธีเชิงพันธุกรรมอย่างง่าย (sGA) และดีกว่าขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับ (cGA) อย่างเห็นได้ชัด และที่สำคัญก็คือ fb-cGA ยังใช้จำนวนครั้งในการคำนวณค่าความเหมาะสมน้อยกว่าหลายเท่า ในที่นี้ fb-cGA ให้คำตอบดีกว่า sGA และใช้จำนวนครั้งในการประเมินค่าความเหมาะสมน้อยกว่าประมาณ 14 เท่า ส่วนในกรณีที่เทียบกับขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับ fb-cGA ให้คำตอบดีกว่าอย่างชัดเจน และจำนวนครั้งที่ใช้ในการประเมินค่าความเหมาะสม ยังใช้จำนวนครั้งน้อยกว่า cGA ถึง 9 เท่า

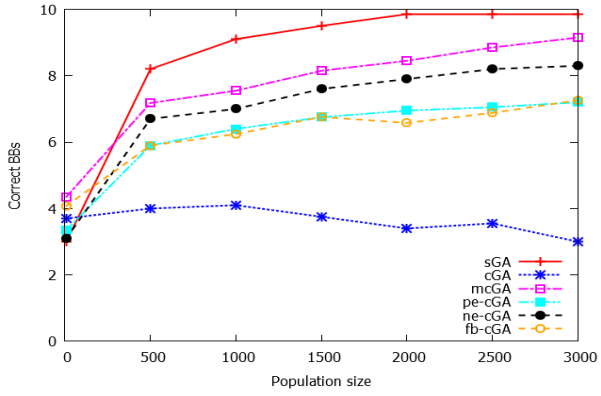
จากผลการทดลองดังกล่าว เป็นที่น่าสนใจว่าการนับความถี่ จำนวนครั้งในการปรับค่าความน่าจะเป็นในเวกเตอร์ความน่าจะเป็น สามารถช่วยให้ขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับได้คำตอบที่ดีขึ้นในปัญหายากที่มีการหลอก ดังเช่นปัญหา Trap ที่นำมาทดสอบนี้ อีกทั้งยังใช้จำนวนครั้งในการประเมินค่าความเหมาะสมที่ต่ำกว่าขั้นตอนอื่น



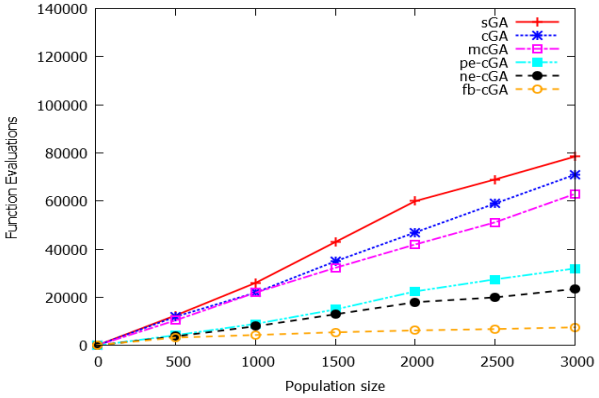
(a) solution quality (tournament size 2)



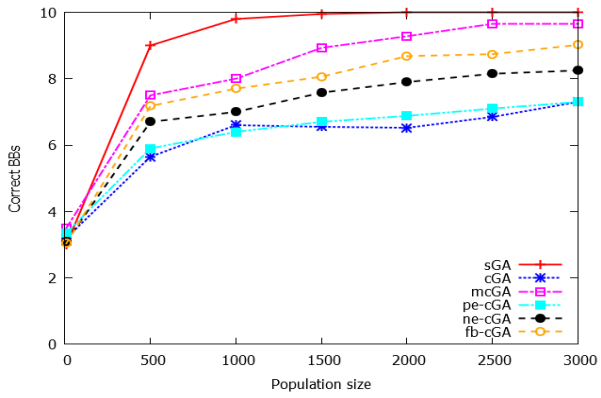
(b) the number of function evaluations (tournament size 2)



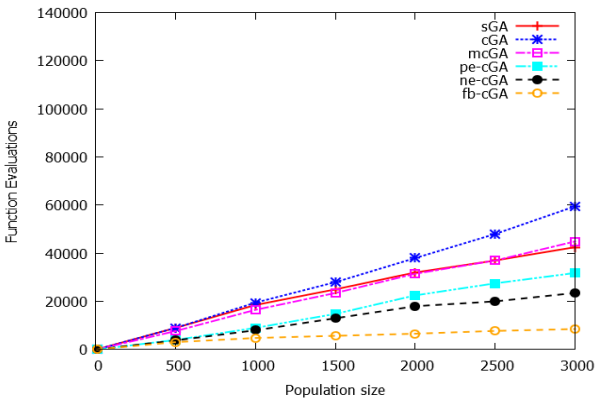
(c) solution quality (tournament size 4)



(d) the number of function evaluations (tournament size 4)



(e) solution quality (tournament size 8)



(f) the number of function evaluations (tournament size 8)

รูปที่ 4.6 ผลการทดลองสำหรับปัญหา Trap

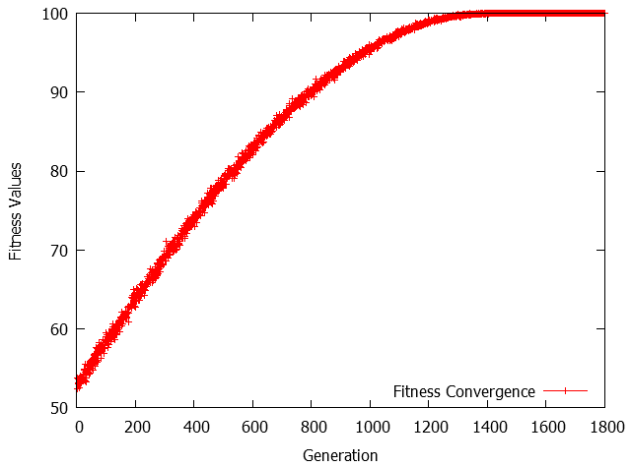
4.4 การวิเคราะห์

จากผลการทดลองที่กล่าวข้างต้น ผู้วิจัยได้ทำการวิเคราะห์เพื่อดูพฤติกรรมการค้นคำตอบของขั้นตอนวิธีที่นำเสนอ เทียบกับขั้นตอนวิธีต่าง ๆ ในงานวิจัยก่อนหน้า รูปที่ 4.7 แสดงพฤติกรรมของขั้นตอนวิธีที่นำมาเปรียบเทียบต่าง ๆ โดยยกตัวอย่างพฤติกรรมการทำงานสำหรับปัญหา OneMax กราฟทางด้านซ้ายแสดงการลู่เข้า โดยแกน x คือ generation และแกน y คือ ค่าความเหมาะสม กราฟด้านขวาแสดงค่าความน่าจะเป็นในมิติต่าง ๆ ของเวกเตอร์ความน่าจะเป็น แกน x คือ ตำแหน่งของบิตต่าง ๆ ในเวกเตอร์ความน่าจะเป็น (ในที่นี้คือปัญหา OneMax ความยาว 100 บิต) แกน y คือ generation จากรูปนี้จึงทำให้เห็นการเปลี่ยนแปลงของค่าความน่าจะเป็นตลอดการทำงานของขั้นตอนวิธี สีที่แสดงในรูปคือสีที่ใช้แสดงค่าความน่าจะเป็น โดยที่สีดำแทนความน่าจะเป็นเท่ากับ 0 และสีขาวแทนค่าความน่าจะเป็นเท่ากับ 1

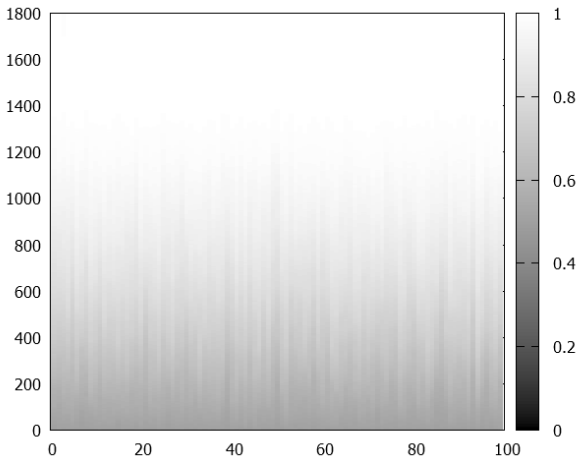
จากรูปที่ 4.7 จะเห็นได้ว่าขั้นตอนวิธี cGA, mcGA และ fb-cGA สามารถค้นหาคำตอบที่ดีที่สุดได้ (คำตอบดีสุดในที่นี้คือทุกบิตเป็น 1 ทั้งหมด นั่นคือรูปแสดงค่าความน่าจะเป็นทางด้านขวาจะเป็นสีขาวในตอนท้าย) ถ้าพิจารณาถึงจำนวนครั้งในการประเมินค่าความเหมาะสมก็จะเห็นได้ว่าขั้นตอนวิธี cGA กับ mcGA ใช้เวลาใกล้เคียงกัน แต่ขั้นตอนวิธี fb-cGA ที่นำเสนอ ได้คำตอบที่ดีที่สุดเร็วกว่า ใช้จำนวนครั้งในการประเมินค่าความเหมาะสมที่น้อยกว่า

การที่พฤติกรรมของ fb-cGA เป็นเช่นนั้นก็เนื่องจาก ขั้นตอนวิธีนี้ถูกออกแบบมาให้ดูค่าความถี่และความต่อเนื่องของการปรับปรุงค่าความน่าจะเป็นในเวกเตอร์ความน่าจะเป็น ทั้งในทิศทางที่ปรับขึ้น (เข้าใกล้ 1.0) และทิศทางที่ปรับลง (เข้าใกล้ 0.0) การมีข้อมูลเช่นนี้ ทำให้ขั้นตอนวิธีสามารถใช้ข้อมูลนี้มาทำการตัดสินใจได้เร็วขึ้น และปรับค่าความน่าจะเป็นไปในที่ทางที่น่าจะไปสู่คำตอบได้มากขึ้น

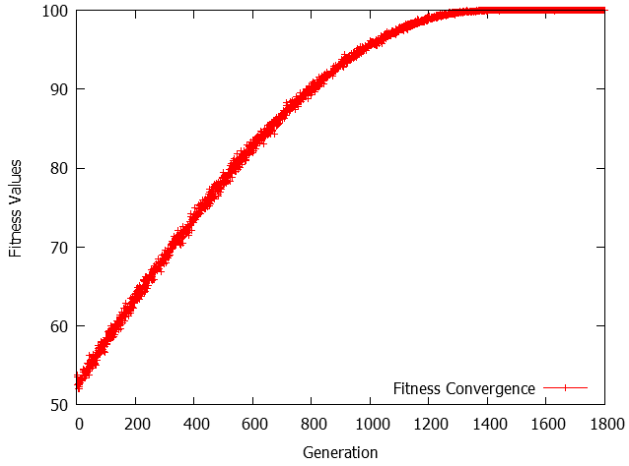
ส่วนขั้นตอนวิธีที่มีการเก็บตัวเก่งที่สุดเอาไว้ อย่างเช่นขั้นตอนวิธี pe-cGA และ ne-cGA มีบางบิตที่ขั้นตอนวิธีปรับค่าความน่าจะเป็นไปผิดทิศทาง ทั้ง ๆ ที่ปัญหาทดสอบนี้เป็นปัญหาง่าย จุดสังเกตที่น่าสนใจที่ทำให้ขั้นตอนวิธีที่ใช้ข้อมูลจากตัวเก่งมากเกินไปไม่สามารถค้นพบคำตอบที่ดีที่สุดได้ อาจเนื่องมาจากขั้นตอนวิธีเหล่านี้พอค้นหาเจอคำตอบที่มีค่าความเหมาะสมค่อนข้างสูงก็จะเชื่อว่าคำตอบนี้ดี และพยายามที่จะปรับค่าเวกเตอร์ความน่าจะเป็นไปในทิศทางนั้น แต่ในบางครั้งถ้าคำตอบที่ได้มาเป็นเพียงตัวอย่างคำตอบที่ค่อนข้างดี หรือเป็นคำตอบดีสุดเฉพาะที่ ก็อาจทำให้อัลกอริทึมหลงทางได้ ดังนั้น การนำความรู้จากคำตอบที่ดีมาก ๆ มาเป็นแนวทางในการค้นคำตอบอาจจะไม่ดีเสมอไป บางครั้งการใช้คำตอบจากตัวอย่างคำตอบที่มีค่าความเหมาะสมไม่สูงมากนักมาช่วยพิจารณา อาจช่วยให้ขั้นตอนวิธีหลุดจากการติดที่จุดดีสุดเฉพาะที่และนำไปสู่คำตอบที่ดีขึ้นได้



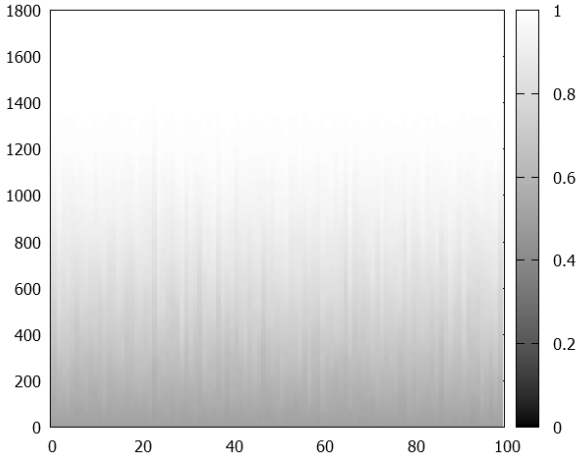
(a) Convergence (the cGA)



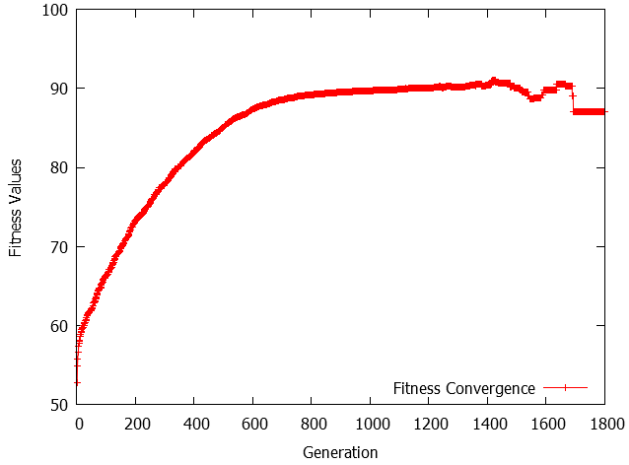
(b) Probability (the cGA)



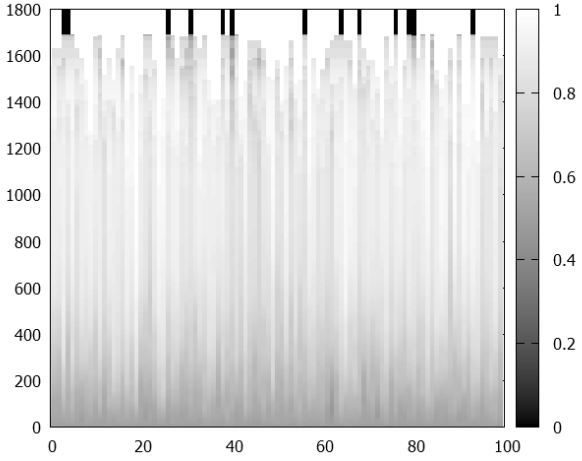
(c) Convergence (the mcGA)



(d) Probability (the mcGA)

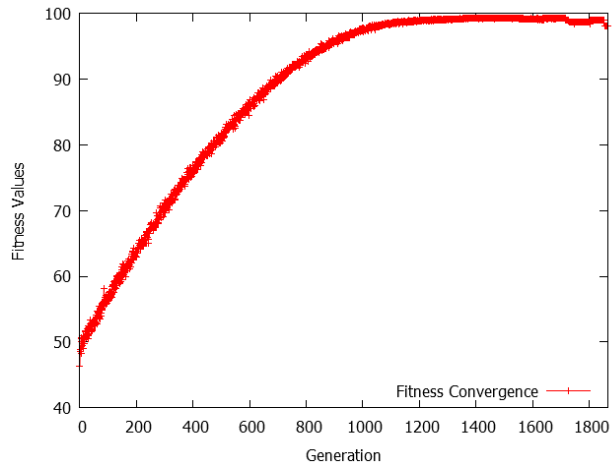


(e) Convergence (the pe-cGA)

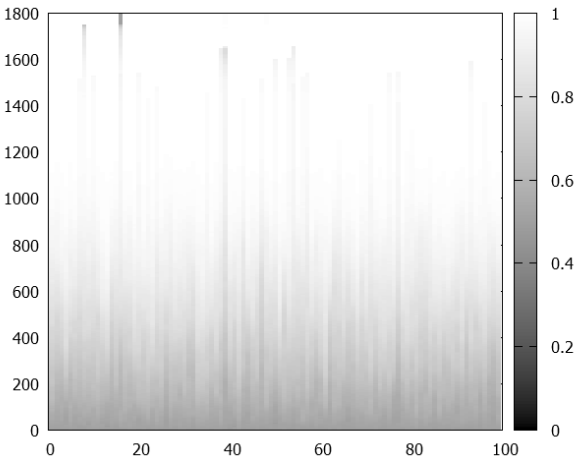


(f) Probability (the pe-cGA)

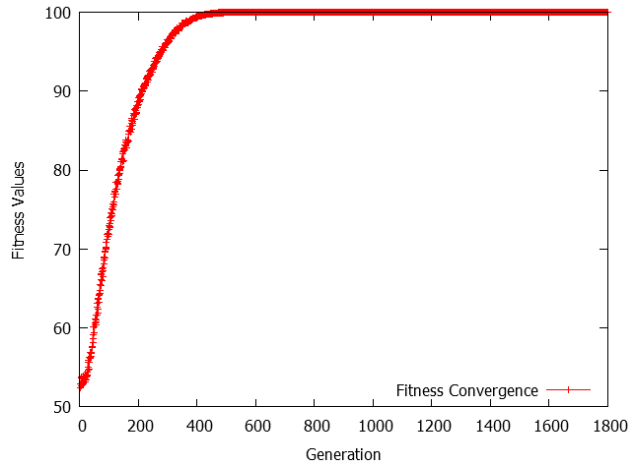
รูปที่ 4.7 กราฟการลู่เข้าและการวิเคราห์เวกเตอร์ความน่าจะเป็น



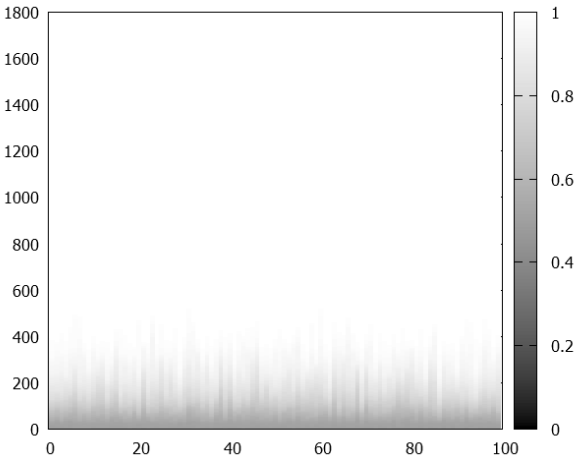
(g) Convergence (the ne-cGA)



(h) Probability (the ne-cGA)



(i) Convergence (the fb-cGA)



(j) Probability (the fb-cGA)

รูปที่ 4.7 (ต่อ) กราฟการลู่เข้าและการวิเคราะห์เวกเตอร์ความน่าจะเป็น

บทที่ 5

สรุปและวิจารณ์

โครงการวิจัยนี้ ได้ทำการศึกษาและวิเคราะห์การเรียนรู้จากตัวอย่างคำตอบที่มีค่าความเหมาะสมสูง และการเรียนรู้จากตัวอย่างคำตอบที่มีค่าความเหมาะสมต่ำร่วมด้วย โดยมุ่งเน้นศึกษากับขั้นตอนวิธีประมาณการแจกแจงอย่างง่ายที่ตัวแปรไม่ขึ้นต่อกัน ขั้นตอนวิธีที่ได้ทำการศึกษาวิเคราะห์ ได้แก่ ขั้นตอนวิธีการเรียนรู้เพิ่มขึ้นแบบอาศัยประชากร และ ขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับ

ผู้วิจัยได้ศึกษาพฤติกรรมของขั้นตอนวิธีการเรียนรู้เพิ่มขึ้นแบบอาศัยประชากร แบบที่ใช้ความรู้จากคำตอบที่ดีที่สุดในการปรับปรุงเวกเตอร์ความน่าจะเป็น เปรียบเทียบกับการใช้ตัวอย่างคำตอบที่แย่ที่สุดร่วมด้วย จากการทดลองจะเห็นได้ว่าการใช้คำตอบแย่สุดร่วมด้วยในการพิจารณาปรับค่าความน่าจะเป็นในเวกเตอร์ความน่าจะเป็น ส่งผลให้คำตอบที่ได้มีคุณภาพดีขึ้น ซึ่งก็สอดคล้องกับงานวิจัยก่อนหน้า

ผู้วิจัยได้ศึกษา และทำการปรับปรุงขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับ ที่มีการใช้ตัวอย่างคำตอบที่ชนะ และตัวอย่างคำตอบตัวที่แพ้มาพิจารณาว่าควรจะปรับค่าในเวกเตอร์ความน่าจะเป็นไปในทิศทางใด งานวิจัยนี้ได้ทดลองและนำเสนอขั้นตอนวิธีเชิงพันธุกรรมแบบกระชับด้วยค่าความถี่ (fb-cGA) พร้อมทั้งนำเสนอผลการวิเคราะห์ถึงพฤติกรรมการทำงาน ที่แสดงให้เห็นว่าการนำความรู้จากการนับจำนวนครั้งและความต่อเนื่องในการปรับค่าความน่าจะเป็นมาใช้นั้น ส่งผลให้ขั้นตอนวิธีสามารถตัดสินใจลู่เข้าสู่คำตอบที่ดีได้เร็วขึ้น

งานวิจัยนี้ อาจจะเป็นแนวทางให้ผู้ที่สนใจจะนำความรู้ของตัวอย่างคำตอบที่มีค่าความเหมาะสมไม่สูงนัก มาใช้ประโยชน์เพื่อเติมเต็มความรู้ที่ได้จากตัวอย่างที่มีค่าความเหมาะสมสูง และความรู้นี้อาจช่วยให้ขั้นตอนวิธีสามารถค้นหาคำตอบไปในทิศทางที่ถูกต้องมากขึ้น สามารถหลุดออกจากคำตอบดีสุดเฉพาะที่ และสามารถลดจำนวนครั้งในการประเมินค่าความเหมาะสมลงได้ ซึ่งทำให้การคำนวณมีความรวดเร็วและมีประสิทธิภาพมากยิ่งขึ้น

บรรณานุกรม

B. V. Ha, R. E. Zich, M. Mussetta, P. Pirinoli and C. N. Dao, “Improved compact genetic algorithm for EM complex system design”, Proceedings of the International Conference on Communications and Electronics, **2012**, Vietnam.

C. Aporntewan and P. Chongstitvatana, “A hardware implementation of the compact genetic algorithm”, Proceedings of the IEEE Congress on Evolutionary Computation, **2001**, Seoul, Korea, pp. 624-629.

C. W. Ahn and R. S. Ramakrishna, “Elitism-based compact genetic algorithms”, *IEEE Trans. Evol. Comp.*, **2003**, 367-385.

C. Zhou, K. Meng and Z. Qiu, “Compact genetic algorithm mutated by bit”, Proceedings of 4th World Congress on Intelligent Control and Automation, **2002**, Shanghai, China, pp. 1836-1839.

G. R. Harik, F. G. Lobo and D. E. Goldberg, “The compact genetic algorithm”, *IEEE Trans. Evol. Comp.*, **1999**, 3, 287-297.

H. Mühlenbein and G. Paaß, “From recombination of genes to the estimation of distributions I. Binary parameters”, *Parallel Problem Solving from Nature*, LNCS 1141, Berlin: Springer, **1996**.

J. Gallagher and S. Vigraham, “A modified compact genetic algorithm for the intrinsic evolution of continuous time recurrent neural networks”, Proceedings of the Genetic and Evolutionary Computation Conference, **2002**, USA, pp. 163-170.

J. Gallagher, S. Vigraham and G. Kramer, “A family of compact genetic algorithms for intrinsic evolvable hardware”, *IEEE Trans. Evol. Comp.*, **2004**, 8, 111-126.

J. Y. Lee, M. S. Kim and J. J. Lee, “Compact genetic algorithms using belief vectors”, *APPL SOFT COMPUT*, **2011**, 11, 3385-3401.

J. Y. Lee, S. M. Im and J. J. Lee, “Bayesian network-based non-parametric compact genetic algorithm”, Proceedings of IEEE International Conference on Industrial Informatics, **2008**, Daejeon, Korea, pp. 359-364.

K. M. Timmerman, “A hardware compact genetic algorithm for hover improvement in an insect-scale flapping-wing micro air vehicle”, *Master Thesis*, **2010**, Wright State University.

M. Pelikan, D. E. Goldberg and F. G. Lobo, “A survey of optimization by building and using probabilistic models”, Urbana, IL: *University of Illinois Genetic Algorithms Laboratory* (IlliGAL Report No. 99018), **1999**.

S. Baluja, “Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning”, *Technical Report CMU-CS-95-163*, Carnegie Mellon University, **1994**.

S. Phiromlap and S. Rimcharoen, “A frequency-based updating strategy in compact genetic algorithm”, Proceedings of the International Computer Science and Engineering Conference, **2013**, Bangkok, Thailand, pp. 212-216.

S. Rimcharoen, D. Sutivong and P. Chongstitvatana, “Updating strategy in compact genetic algorithm using moving average approach”, Proceedings of the IEEE International Conferences on Cybernetics and Intelligent Systems, **2006**, Bangkok, Thailand.

S. Rimcharoen, S. Phiromlap, and N. Leelathakul, “Analysis of frequency-based compact genetic algorithm (fb-cGA)”, *Maejo International Journal of Science and Technology*, **2015**, 9(1), pp. 121-135.

ภาคผนวก

(ผลงานตีพิมพ์ในวารสารวิชาการนานาชาติ)

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 121-135; doi: 10.14456/mijst.2015.10

Maejo International Journal of Science and Technology

ISSN 1905-7873

Available online at www.mijst.mju.ac.th

Full Paper

Analysis of frequency-based compact genetic algorithm (fb-cGA)

Sunisa Rimcharoen, Srichol Phiromlap and Nutthanon Leelathakul *

Faculty of Informatics, Burapha University, Chon Buri, 20131, Thailand

* Corresponding author, e-mail: nutthanon@buu.ac.th

Received: 19 June 2014 / Accepted: 30 March 2015 / Published: 9 April 2015

Abstract: A behaviour analysis of frequency-based compact genetic algorithm (fb-cGA) is proposed. The fb-cGA is a version of compact genetic algorithm (cGA) enhanced by the use of a new updating strategy. The algorithm counts the number of probability updates and the continuities of probability-update directions and uses them to adaptively update the algorithm's step sizes. This method requires fewer function evaluations and achieves solutions that are more accurate than those from the conventional cGA. It has been shown that fb-cGA can reduce the number of function evaluations to only one ninth of the number obtained from cGA on ten copies of a 3-bit trap function using a tournament size of 2. We conduct parameter studies and show that the use of one fourth of the population size (*psize/4*) as the algorithm's starting threshold can improve the overall efficiency of fb-cGA. The behaviour of fb-cGA on various problems is also examined. The results of the analysis show that information from the algorithm's past experience (i.e. the numbers of probability updates and continuities) can help the fb-cGA to update the probability vector towards a more promising direction, requiring fewer function evaluations.

Keywords: compact genetic algorithm, updating strategy, update frequency, update continuity

INTRODUCTION

Compact genetic algorithm (cGA) was proposed by Harik et al. [1]. It has been widely applied to various fields such as pipe network optimisation [2], parameter optimisation [3, 4], inventory planning [5], image recognition [6], traffic transportation management [7], communication [8-10], container loading [11], grid computing [12] and biology [13, 14]. The main contribution of this algorithm is to replace a whole set of candidate solutions (the so-called population) used by simple genetic algorithm (sGA) with a probability distribution. cGA requires

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 121-135; doi: 10.14456/mijst.2015.10

much less memory, as it does not need to maintain the population throughout the evolution process. The concept of cGA can be easily translated to hardware implementation by using the common very-large-scale integration [15-17]. Therefore, it opens up the application of genetic algorithm to new fields such as embedded systems. For example, Timmerman [18] used cGA to develop an insect-sized flapping-wing micro air vehicle.

However, for more difficult problems, cGA does not provide acceptable solutions. There have been many attempts to modify and improve cGA's probability updating strategy. Zhou et al. [19] proposed an improved cGA using mutation and named the algorithm mutated-by-bit-compact genetic algorithm (MBBCGA). At each generation, MBBCGA generates only one individual and then mutates this individual bit by bit. Ha et al. [20] proposed the use of more than one probability vector (PV) to enhance the exploration properties of the algorithm. Rimcharoen et al. [21] improved the updating strategy of cGA by using a moving average technique (mcGA). Ahn and Ramakrishna [22] adopted 'elitism', i.e. the idea of reserving the best solution in each generation. They proposed two variants: a persistent elitist compact genetic algorithm (pe-cGA) and a non-persistent elitist compact genetic algorithm (ne-cGA). The former stores the current best solution until a better solution is found, while the latter keeps the best solution just for a certain lifetime. In 2008 Lee et al. [23] introduced a new update strategy using augmented Bayesian networks. A few years later, they proposed compact genetic algorithm using a belief vector (cGABV) [24]. The new technique uses a belief vector (BV) instead of a probability vector. The difference between BV and PV is that each element of the BV stores a probability distribution (represented by associated mean and variance), whereas each of the PV keeps a probability value.

In our previous work [25], we proposed the usage of a frequency-based updating technique as the updating strategy of cGA. The technique collects and utilises information from the algorithm's past experience. Specifically, for each probability in the PV, the number of probability updates (in both up and down directions) are counted and used to adjust probability-updating step sizes, turning the vector towards the promising direction faster. Comparison results show that the frequency-based compact genetic algorithm (fb-cGA) requires substantially (up to nine times) fewer function evaluations when compared with traditional cGA. However, in-depth explanation and analysis of why this algorithm outperforms others remained lacking. Accordingly, in this paper, we conduct parameter studies and analyse how the algorithm behaves while solving various problems.

FREQUENCY-BASED COMPACT GENETIC ALGORITHM (fb-cGA)

cGA is one of various evolutionary algorithms. Instead of evolving the population for searching solutions, it employs a probabilistic model, PV, which requires relatively small amount of memory. Furthermore, the algorithm eliminates genetic operators such as crossover and mutation.

cGA keeps a PV over a chromosome to represent the population. The number of probabilities in the vector is equal to the chromosome length. Each probability is defined as the probability with the associated bit being equal to 1. The pseudo-code of cGA is shown in Figure 1. The two parameters are the chromosome length (l) and the population size (*psize*), which are used to further specify an updating step size (i.e. step size defined as 1 / *psize*). (Note that the relation between *psize* and the updating size in the cGA is analogous to the one between population size and evolving speed in sGA.)

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 121-135; doi: 10.14456/mijst.2015.10

```
initialise(p)
while (p does not converge) do
    individual1 := generate(p)
    individual2 := generate(p)
    evaluate(individual1, individual2)
    winner, loser := compete(individual1,
                             individual2)
    for i:=1 to l
    begin
        if winner[i] ≠ loser[i] then
            if winner[i] = 1 then
                p[i] := p[i] + 1/psize
            else
                p[i] := p[i] – 1/psize
    endfor
endwhile
```

Figure 1. Pseudo-code of cGA

```
s := tournament size
initialise(p)
while (p does not converge) do
    create(p, S[], s)
    evaluate(S[])
    rearrange(S[]) // S[1] is the best individual
    for i := 2 to s
    begin
        winner, loser := compete(S[1], S[i])
        for i:=1 to l
        begin
            if winner[i] ≠ loser[i] then
                if winner[i] = 1 then
                    p[i] := p[i] + 1/psize
                else
                    p[i] := p[i] – 1/psize
        endfor
    endfor
endwhile
```

Figure 2. Pseudo-code of tournament cGA

First, the cGA initially sets each of the probabilities in the vector to 0.5. According to the PV, the algorithm randomly generates two candidate solutions, denoted as *individual1* and *individual2*. Next, the solutions are evaluated, i.e. assigned fitness values. The winner, the one with the greater fitness value, is selected. In step 5, the PV is then updated towards the winner. The value of each probability changes if the winner's associated bit is not equal to the loser's: either increasing when the winner's bit is one, or decreasing otherwise. The loop continues to run until the PV converges, meaning that each probability in the vector is either zero or one.

Harik et al. [1] also modified cGA by adding more candidates, called tournament cGA, shown in Figure 2. The modified version randomly generates a set of s candidate solutions, denoted by an array *S* in the pseudo-code, and uses a tournament selection to choose the winner, which will be stored in *S*[1]. The PV is then updated by comparing *S*[1] with *S*[i] (for all *i* not equal to 1) in the same manner as the original cGA.

Both of the cGAs update each probability in the vector towards either one or zero. Some probabilities gradually increase while others drop. However, some might fluctuate, reflecting uncertainty in updating the PV. It is known that the PV fluctuates during the beginning period and converges to a certain direction at the end. The algorithms seem to work well in the case where problems have consistent information, leading the algorithms to turn the vector towards only one direction. However, if the problems are deceptive, they might delude the algorithms into searching for solutions in the wrong directions. Consequently, the cGAs could not provide the desired solution quality in spite of spending much of searching time. There has been much research aimed at modifying and improving the cGAs in such case.

In our previous work [25], we applied a frequency-based technique to update the PV, using the numbers of updates and the continuities of preceding updates as criteria. (The update continuity is

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 121-135; doi: 10.14456/mijst.2015.10

defined as the number of consecutive updates moving towards the same direction). We measured the uncertainty by observing the direction of each probability in the vector: if the direction is the same for a long time (high continuity), the uncertainty is low. The monitored continuities serve as a guideline or a promising trend that quickly leads to vector convergence.

Specifically, for each probability in the vector, the frequencies of two types of updates: stepping-up (increasing the probability towards 1) and stepping-down (decreasing the probability towards 0), were counted. Likewise, two types of update continuities were collected. The stepping-up continuity is reset to zero if the current update moves towards 0 and the stepping-down continuity is reset to zero if the current update moves towards 1. The fb-cGA technique is shown in Figure 3.

Updating strategy of fb-cGA

```
1:  for i := 1 to l
2:  begin
3:      if winner[i] ≠ loser[i] then
4:          if winner[i] = 1 then
5:              Ufreq[i] := Ufreq[i] + 1;
6:              Ucon[i] := Ucon[i] + 1;
7:              Dcon [i] := 0;
8:              if (Ufreq[i] > Dfreq[i] AND Gen > (psize/3)) then
9:                  p[i] := p[i] + ((1/psize)+(p[i] * (Ucon[i]/100));
10:             else
11:                 p[i] := p[i] + (1/ psize);
12:         else
13:             Dfreq[i] := Dfreq[i] + 1;
14:             Dcon [i] := Dcon [i] + 1;
15:             Ucon[i] := 0;
16:             if (Dfreq[i] > Ufreq[i] AND Gen > (psize/3)) then
17:                 p[i] := p[i] - ((1/psize) + (p[i]*(Dcon [i] / 100));
18:             else
19:                 p[i] := p[i] - (1/psize);
20:  endfor
```

Parameters:

Ufreq : number of stepping-up updates
Dfreq : number of stepping-down updates
Ucon : number of consecutive stepping-up updates
Dcon : number of consecutive stepping-down updates
Gen : generation number (incremented in Step 6)

Figure 3. Pseudo-code of fb-cGA

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 121-135; doi: 10.14456/mijst.2015.10

Figure 3 presents the pseudo-code of the frequency-based updating strategy in fb-cGA. *Ufreq* denotes the number of probability updates towards one (i.e. stepping-up updates). *Dfreq* denotes the number of probability updates towards zero (i.e. stepping-down updates). *Gen* denotes the generation number whose value is increased incrementally in step 6. The proposed updating strategy is performed when *Gen* is greater than 1/3 of the population size (*psize*). For the first third of the generations, fb-cGA works like the original method to explore solutions and find the right direction. It waits until the generation number reaches *psize*/3 because it needs time to gather sufficient information to see the trend. For the last two-thirds of the generations, the i^{th} probability is updated when the i^{th} bit of the winner (*winner*[*i*]) and the one of the loser (*loser*[*i*]) are not equal. If *winner*[*i*] is 1, the algorithm checks whether, from past experience, this probability is updated towards 1 most of the time (i.e. *Ufreq* greater than *Dfreq*). If so, the probability vector should be updated according to the majority with a larger step size. The step size can be determined by adding the term *Ucon*/100 multiplied by the previous value of i^{th} probability, where *Ucon* denotes the number of consecutive stepping-up updates. In contrast, when *winner*[*i*] is 0, the algorithm performs in a similar manner but considers *Dfreq* and *Dcon* instead. The i^{th} probability is updated by decreasing towards zero.

In this paper, we study the effects of the *psize* parameter and show that using *psize*/4 can improve the efficiency of fb-cGA. Thus, we use *psize*/4 instead of the previously proposed *psize*/3 [25] throughout the experiments conducted and presented in this paper.

PARAMETER STUDIES

As mentioned earlier, the proposed method performs the new updating strategy when the number of generations is greater than *psize*/4. The reason behind this strategy is that the statistics obtained during the beginning period are not reliable enough to capture the trend. In this section, empirical experiments are presented to explain why we set this parameter as *psize*/4. The algorithm on 4 benchmark problems, viz. 100-bits One-Max, 100-bits Random Max, 64-bits Royal Road and ten copies of 3-bits Trap problems, were tested. The characteristics of the four problems are explained below.

The One-Max problem is quite simple. The objective is to find the solution which is a bit string whose bits are all one. The fitness value is equal to the number of 1-bits in the bit string. The Random Max problem is similar to the One-Max problem in finding a bit-string solution whose bit pattern is exactly the same as the one of the target. However, instead of being all one, the target bit pattern is selected randomly. Obtained by comparing bit by bit, the fitness value is the number of bits equal to the associated ones of the target. Notice that this problem is designed to determine whether an algorithm is biased against one or zero.

The Royal Road is a group of bit patterns built up from sequences of short bit patterns. The bit pattern is called schema. There are 15 schemas for 64-bits royal road as shown in Figure 4. After comparing the bit string with each schema, the fitness value is calculated by summing up the numbers of bits equal to those of s_i for all i. For example, a fitness value of a bit string that contains all one (the optimum solution) is $(8 \times 8) + (4 \times 16) + (2 \times 32) + 64 = 256$.

The Trap problem is one of many difficult problems used for testing GAs. It is designed to fool gradient-based optimisers that favour zeroes, but the optimal solution is composed of all 1-bits. We can create a $k \times m$ Trap problem by combining the m groups of a k-bits trap. The fitness value is calculated by summing up the scores associated with all groups. For instance, a 3-bit Trap problem

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 121-135; doi: 10.14456/mijst.2015.10

gives a score of 3, 0, 1 and 2 for a group of three, two, one and zero 1-bits respectively. For example, a candidate solution '111 001 110 000 100' has a fitness value of 3 + 1 + 0 + 2 + 1 = 7.

Schema 1 = 11111111**********; $s1 = 8$
Schema 2 = ********11111111**********; $s2 = 8$
Schema 3 = ****************11111111**********; $s3 = 8$
Schema 4 = ************************11111111**********; $s4 = 8$
Schema 5 = ********************************11111111**********; $s5 = 8$
Schema 6 = **11111111**********; $s6 = 8$
Schema 7 = **11111111********; $s7 = 8$
Schema 8 = **11111111; $s8 = 8$
Schema 9 = 111111111111111**********; $s9 = 16$
Schema10 =****************111111111111111**********; $s10 = 16$
Schema11 =********************************111111111111111**********; $s11 = 16$
Schema12 =**111111111111111; $s12 = 16$
Schema13 =11111111111111111111111111111111**********; $s13 = 32$
Schema14 =********************************1111111111111111111111111111111; $s14 = 32$
Schema15 =111111111111111111111111111111111111111111111111111111111111111; $s15 = 64$

Figure 4. Royal Road problem

We ran the proposed algorithm with all benchmark problems described above. For each tournament size of 2, 4 and 8, the parameter was varied among *psize*/2, *psize*/3, *psize*/4 and *psize*/5. The results shown in Table 1 are efficiency ratios [= (solution quality / number of evaluations) × 1000]. The efficiency ratio is used as a quantitative measurement to quantify a quality rate: the higher the rate, the better the efficiency. When the value of *n* is varied from 2 to 4, the efficiency ratio is better when *n* is large (4 or 5) in the case of solving the easy problems (i.e. One-Max and Random Max). For the harder but non-deceptive problem (i.e. Royal Road), a small value of *n* (2 or

Table 1. Efficiency ratio of varying tournament and population sizes in One-Max, Random Max, Royal Road and Trap problems

Problem		Efficiency ratio				Average
		One-Max	Random Max	Royal Road	Trap	
Tournament Size 2	*psize*/2	56.63	32.51	4.97	0.37	23.62
	psize/3	53.14	32.23	5.65	0.42	22.86
	psize/4	56.79	33.68	5.35	0.50	24.08
	psize/5	54.48	34.00	5.35	0.52	23.59
Tournament Size 4	*psize*/2	69.08	41.56	10.03	0.80	30.37
	psize/3	78.88	41.61	10.53	0.95	32.99
	psize/4	89.20	44.76	9.87	0.95	36.20
	psize/5	80.36	49.07	9.65	1.03	35.03
Tournament Size 8	*psize*/2	86.95	42.38	17.51	0.69	36.89
	psize/3	83.52	40.06	14.47	0.91	34.74
	psize/4	87.17	47.69	16.01	1.06	37.99
	psize/5	91.19	47.28	14.03	1.15	38.42

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 121-135; doi: 10.14456/mijst.2015.10

3) yields a slightly better ratio. This can be interpreted that the proposed algorithm needs more time to collect more diverse and higher fitness-valued samples before increasing its updating step size. For the deceptive problem (i.e. Trap), the efficiency ratio tends to be relatively high when n is large. This is because in the Trap problem the fb-cGA cannot find a good solution no matter what parameters are ─ the fitness value might remain similar. Therefore, the efficiency depends on the number of fitness evaluations more than the fitness value. In terms of tournament size, a larger size tends to provide a larger efficiency ratio. Overall, almost all of the best quality rates come from *psize*/4 and *psize*/5 (highlighted in Table 1). The average rate of *psize*/4 from all problems and all sizes of the tournament is 32.76, and that of *psize*/5 is 32.35. The *psize*/4 is therefore more desirable in terms of efficiency.

As shown in Figure 5, the convergence graphs, obtained from the One-Max problem experiments (*psize* = 100), reflect the algorithm behaviour. The dash lines are plotted at the generation numbers equal to *psize*/*n* (x = *psize*/*n*), showing when the proposed updating strategy is triggered. If n is larger, the proposed strategy starts sooner. Passing this line, the algorithm updates the PV with a larger step size when the winner's bit conforms to the majority direction (i.e. meeting the condition on line 8 or 16 in Figure 3). As the graphs show, the fitness values gradually improve in the early generations (generation number < *psize*/*n*) but increase abruptly after the trigger. This behaviour explains why the algorithm's PV converges to a solution using fewer function evaluations.

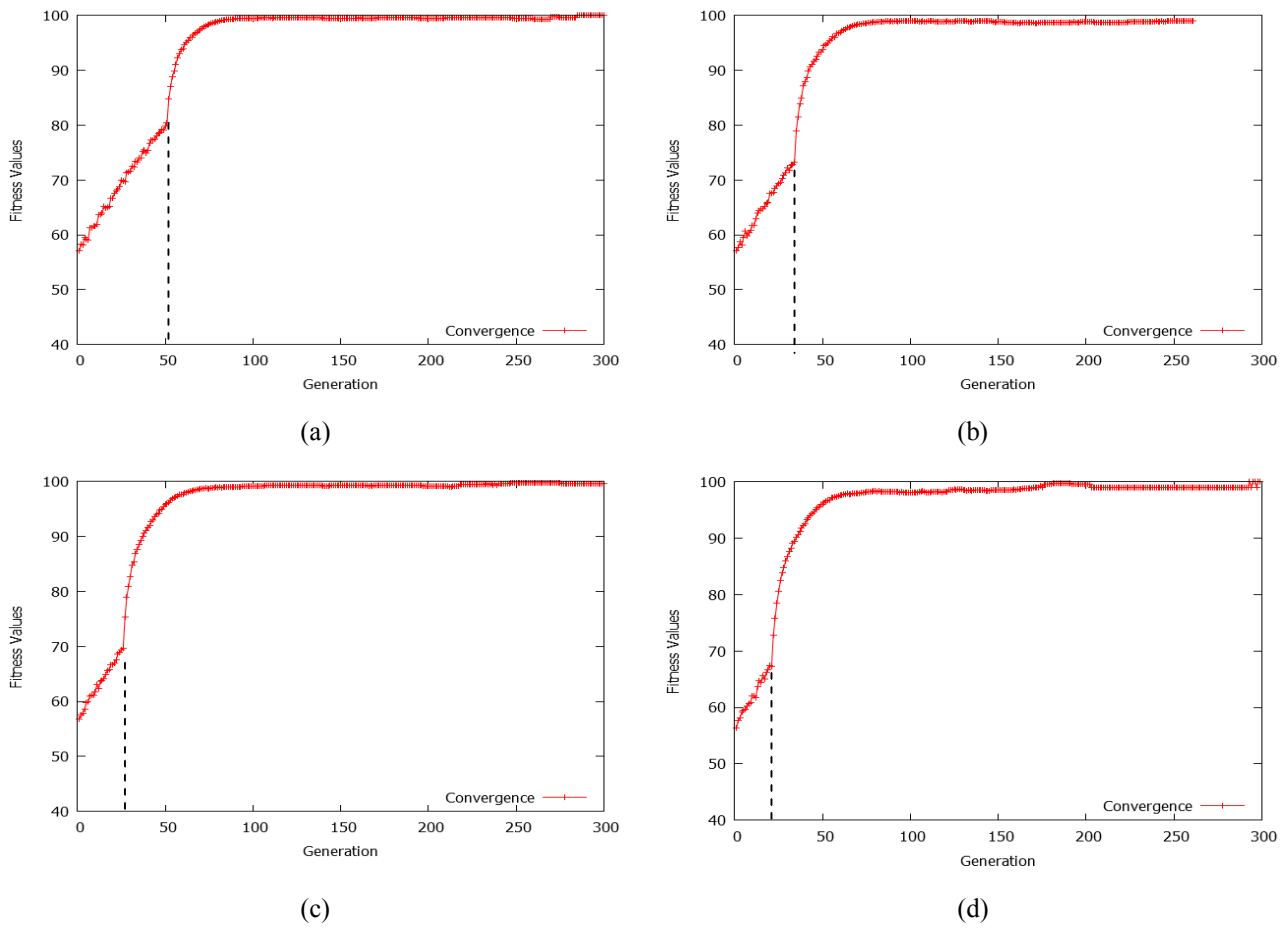


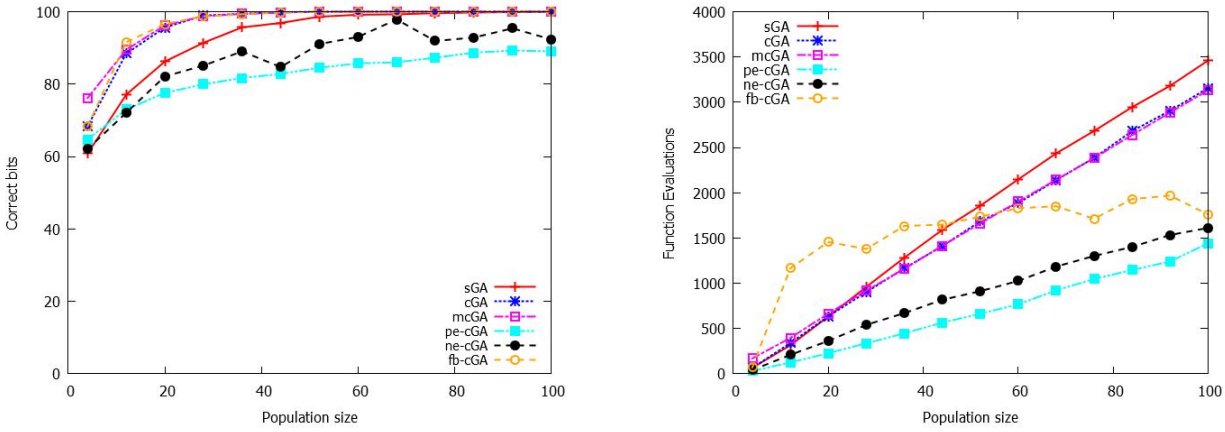
Figure 5. Convergence graphs of experiments with parameters: (a) psize/2, (b) psize/3, (c) psize/4 and (d) psize/5

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 121-135; doi: 10.14456/mijst.2015.10

PERFORMANCE COMPARISONS

At first, we tested all of the algorithms – sGA, cGA, mcGA, pe-cGA, ne-cGA and fb-cGA – with the 100-bit One-Max problem. Each graph in Figure 6 shows the results when the parameter *psize* (population size) varies between 4-100 with a step value of 8. All algorithms used the tournament size of 2. Each line shows an average result from 50 runs. In general, when the population size becomes larger, GAs take more function evaluations but yield better solutions.

Figure 6a shows that the solution quality (the numbers of correct bits) obtained from fb-cGA is comparable with those obtained from sGA, cGA and mcGA, while pe-cGA and ne-cGA have lower solution qualities. In terms of the number of function evaluations, Figure 6b shows that fb-cGA outperforms sGA, cGA and mcGA if the population size is large. When it is small (*psize* < 40), fb-cGA needs more function evaluations than do others. The example where *psize* is equal to 4 is used to explain this situation; the algorithm collects the statistics merely from one generation (*psize*/4 = 1) as a guide to update the PV with a large step size. The triggering time may be too early, leading the vector to a wrong direction. Consequently, the algorithm would spend more time (i.e. a larger number of function evaluations) searching before coming back to the right direction. Nevertheless, the number of fitness evaluations of fb-cGA does not increase as much as the population size and is comparable to those of pe-cGA and ne-cGA when the population size is 100.



(a) Solution quality (b) Number of function evaluations

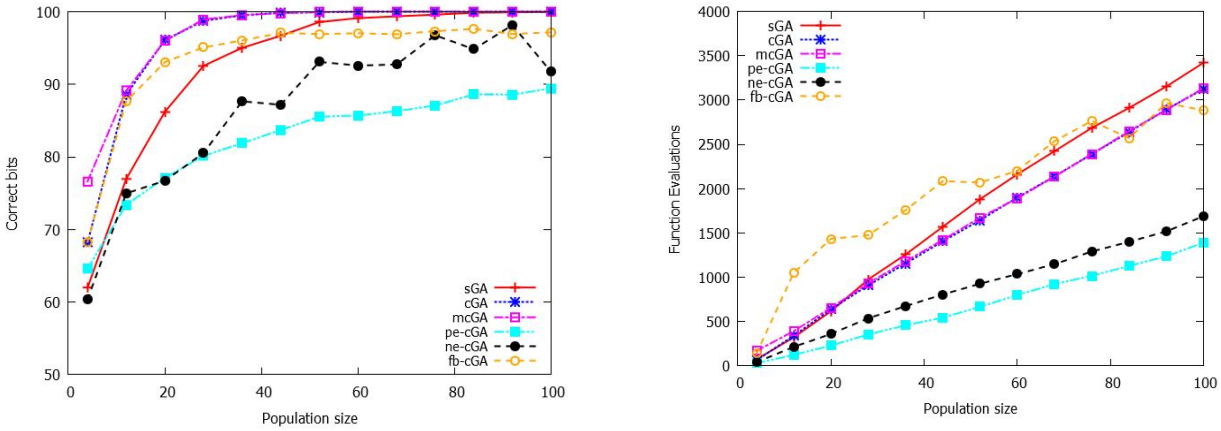
Figure 6. Correct bits and function evaluations (of all the algorithms) in One-Max problem

Figure 7 shows the performance of all the algorithms on the Random Max problem. The fb-cGA performance is moderate when compared with other techniques. It requires a large number of function evaluations to find the solution in the case of a small population, but when the population size increases the numbers of function evaluations tend to be comparable to those in cGA and mcGA.

Figure 8 shows the performance of all the algorithms on the Royal Road problem using tournament sizes of 2, 4 and 8. The number of population sizes varies between 4-100 with a step value of 8. Figures 8a, 8c and 8e show the solution quality in terms of fitness value. Figures 8b, 8d and 8f show the numbers of function evaluations. The fb-cGA yields comparable results in terms of solution quality with those from sGA, cGA and mcGA while requiring a much smaller number of function evaluations. When compared with ne-cGA in the case of tournament size of 2, fb-cGA has a higher fitness value than that of ne-cGA but requires more function evaluations. However, for

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 121-135; doi: 10.14456/mijst.2015.10

tournament sizes of 4 and 8, fb-cGA yields comparable fitness values with those of ne-cGA and needs fewer fitness evaluations. In this problem, pe-cGA requires the smallest number of function evaluations but yields lowest fitness values.



(a) Solution quality

(b) Number of function evaluations

Figure 7. Correct bits and function evaluations (of all the algorithms) in Random Max problem

Figure 9 shows the algorithms' performance on the 3-Trap problem – the Trap problem with a group of 3 bits (k=3) – using tournament sizes of 2, 4 and 8, and population sizes of 8, 500, 1000, 1500, 2000, 2500 and 3000. Figures 9a, 9c and 9e show the solution quality in terms of the number of correct building blocks (the number of 3-bits blocks containing all 1-bits). Figures 9b, 9d and 9f show the numbers of function evaluations taken to find the solution. Figure 9a shows that the solution quality of fb-cGA is higher than that of sGA, cGA and mcGA, but lower than that of pe-cGA and ne-cGA. The fb-cGA requires the smallest number of function evaluations, using them approximately 14, 9, 12, 3 and 2 times fewer than do sGA, original cGA, mcGA, pe-cGA and ne-cGA respectively. This cofirms the efficiency of the proposed method in terms of the number of function evaluations saved.

ANALYSIS OF ALGORITHMS' CONVERGENCE AND BEHAVIOUR

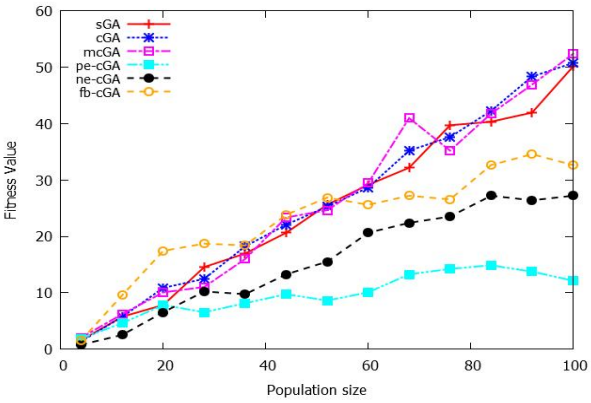
The convergence analysis was carried out by using plots of the fitness values over time (generations). The behaviour analysis was performed through the graphic representation of all probability values in the PV from the first to the last generation to track how the probabilities change.

Figure 10 shows the fitness values and the probability values of cGA, mcGA, pe-cGA, ne-cGA and fb-cGA for the One-Max problem with tournament sizes of 2 and *psize* of 100. The graphs on the right show probability values in density of greyscale. Bearing in mind that the objective of the One-Max problem is to find a solution in which all bits are 1, all the shades representing probability values shown in the graphs on the right should fade to white (probability = 1) in the final generation.

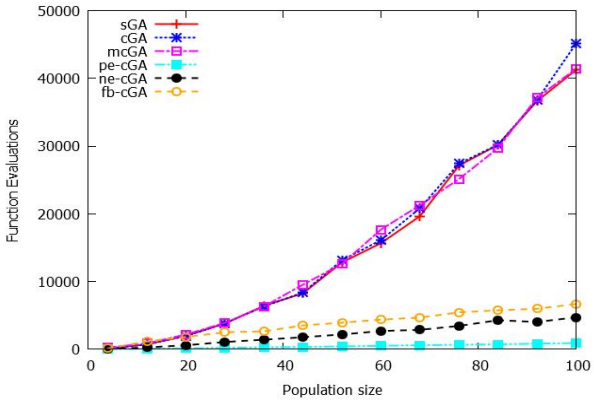
The cGA, mcGA and fb-cGA can find the optimal solution (fitness value = 100), the ne-cGA yields a result very close to the optimal, while the pe-cGA's PV converges to one far from the optimal. The convergence graphs of cGA and mcGA are very similar. Their PV converges to the solution at nearly the same generation. However, the way in which the probabilities change is slightly different. The shades of mcGA fade quicker and more smoothly than do those of cGA. The

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 121-135; doi: 10.14456/mijst.2015.10

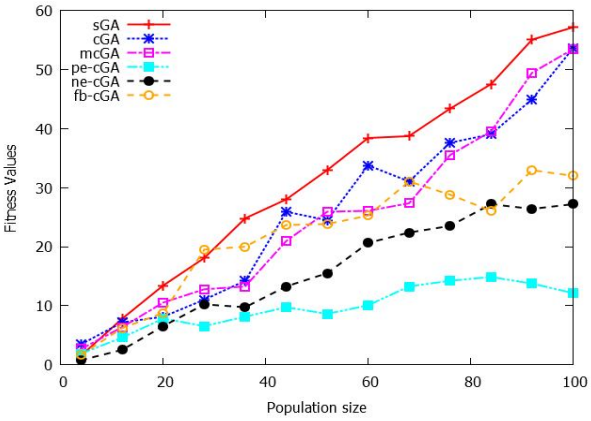
smooth change in the probability values of mcGA is in accordance with its updating rule in that the moving average approach waits to see the trend, thus slowing down the increase or decrease in the probability values.



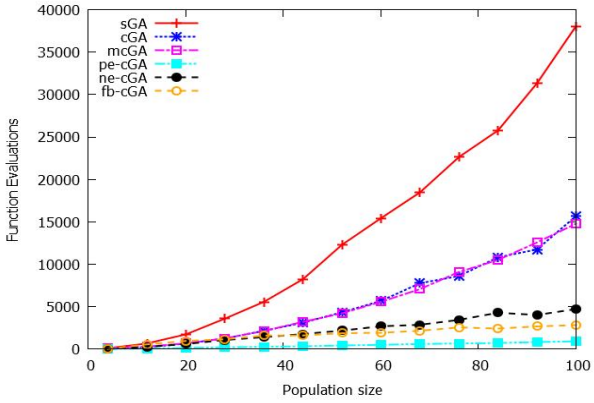
(a) Solution quality (tournament size 2)



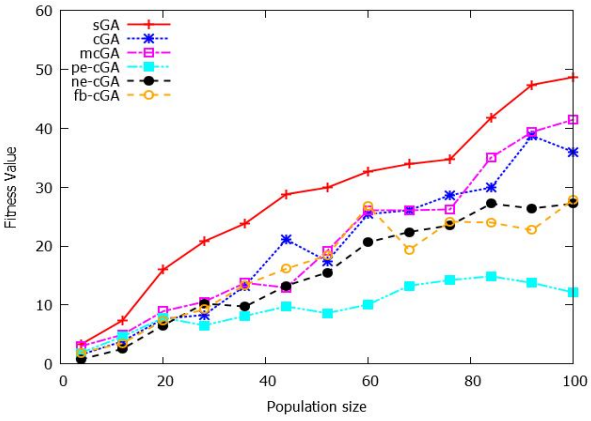
(b) Number of function evaluations (tournament size 2)



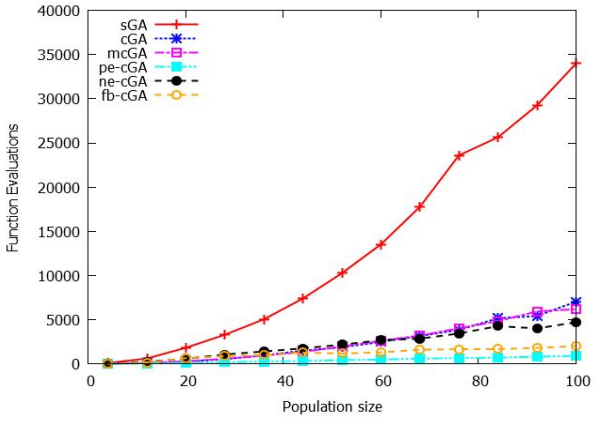
(c) Solution quality (tournament size 4)



(d) Number of function evaluations (tournament size 4)



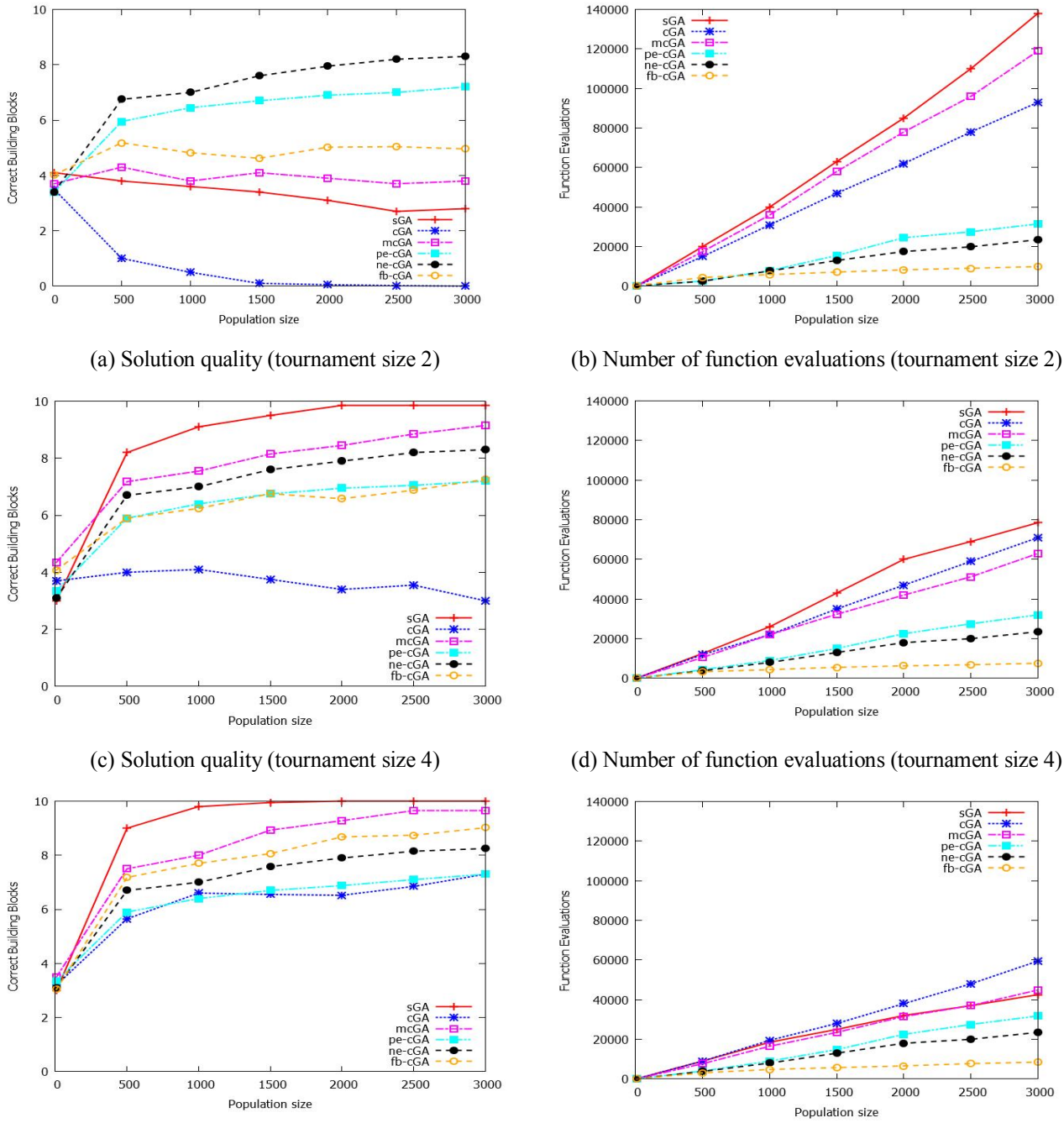
(e) Solution quality (tournament size 8)



(f) Number of function evaluations (tournament size 8)

Figure 8. Fitness values and function evaluations (of all the algorithms) in Royal Road problem

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 121-135; doi: 10.14456/mijst.2015.10



(a) Solution quality (tournament size 2)

(b) Number of function evaluations (tournament size 2)

(c) Solution quality (tournament size 4)

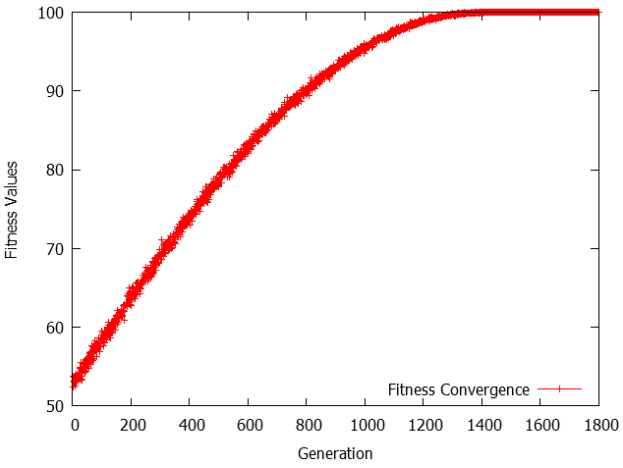
(d) Number of function evaluations (tournament size 4)

(e) Solution quality (tournament size 8)

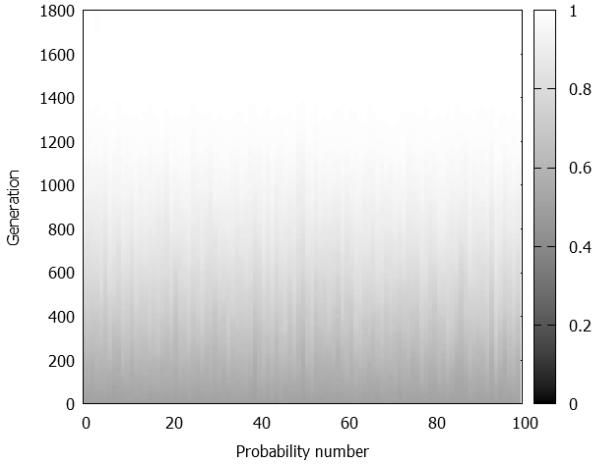
(f) Number of function evaluations (tournament size 8)

Figure 9. Correct building blocks and function evaluations (of all the algorithms) in Trap problem

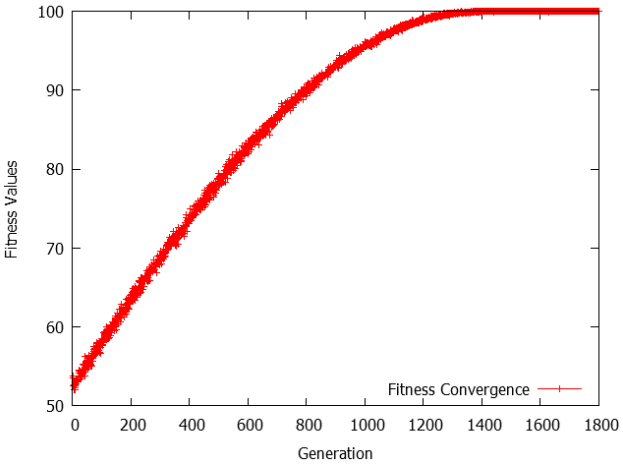
Maejo Int. J. Sci. Technol. **2015**, *9*(01), 121-135; doi: 10.14456/mijst.2015.10



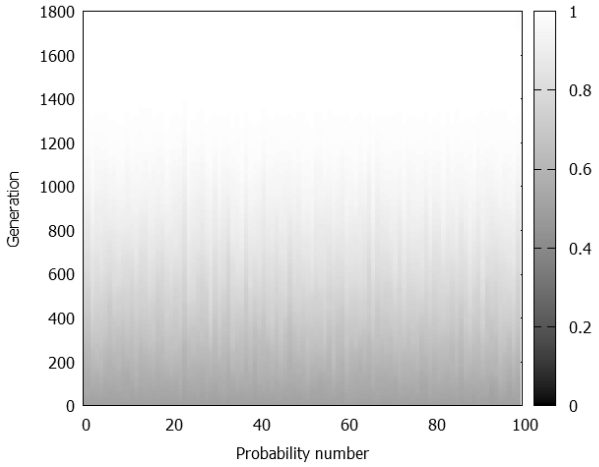
(a) Convergence of fitness values (cGA)



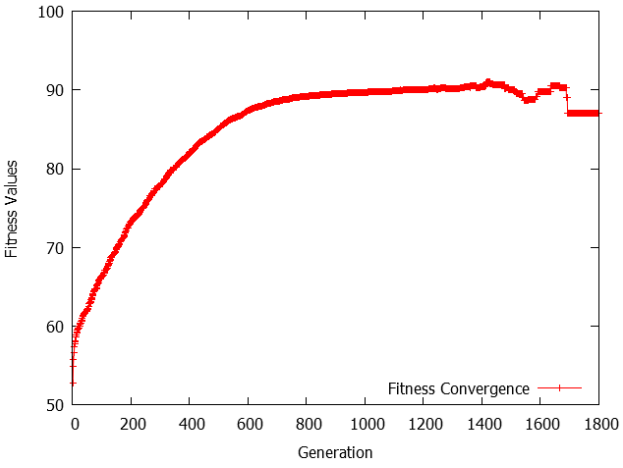
(b) 100 probability values of probability vector (cGA)



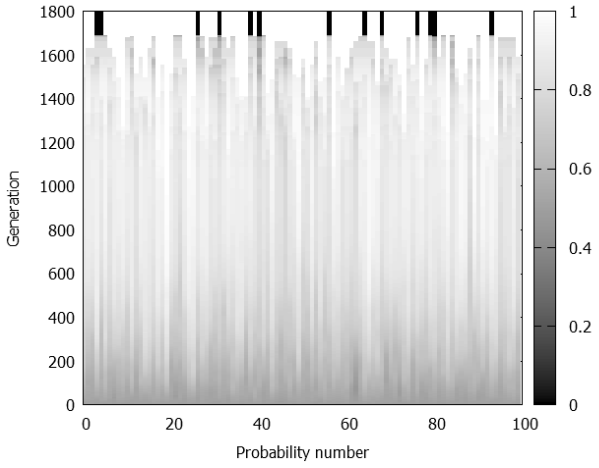
(c) Convergence of fitness values (mcGA)



(d) 100 probability values of probability vector (mcGA)



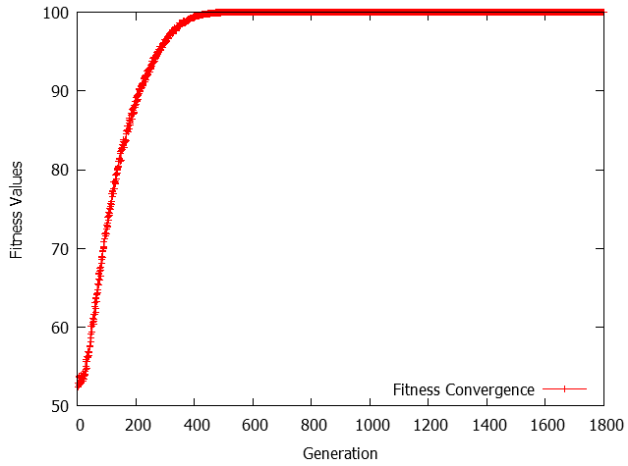
(e) Convergence of fitness values (pe-cGA)



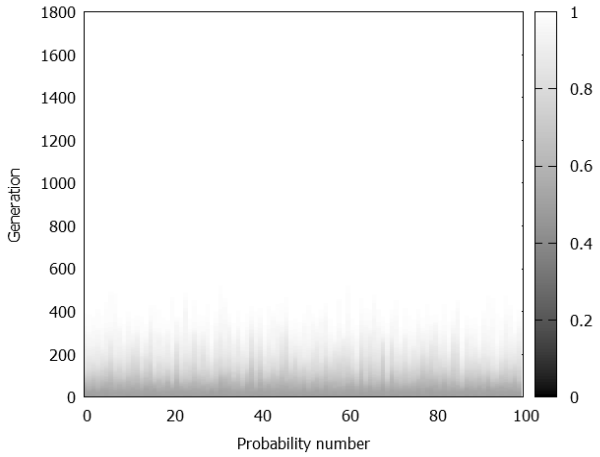
(f) 100 probability values of probability vector (pe-cGA)

Figure 10. Convergence of fitness values and change in 100 probability values in the probability vectors at each generation. Darker shading represents the probability closer to 0 while white represents the probability of 1. All probabilities are initialised to 0.5. All plots show the average values from 50 runs.

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 121-135; doi: 10.14456/mijst.2015.10



(i) Convergence (fb-cGA)



(j) Probability (fb-cGA)

Figure 10 (continued). The convergence and probability analysis

Both the pe-cGA's and ne-cGA's shades turn light grey faster than do the cGA's and mcGA's, as shown in the early generations of Figures 10f and 10h. However, the pe-cGA's final PV is unfavourable and the ne-cGA's approaches, but does not quite reach, the optimum. This is characteristic of elitism. The elite (the best solution so far) often contains zero bits in the chromosome, which deceives the algorithm into updating the probability towards zero. In the case where newly generated candidate solutions are worse than the elite, the PV is updated towards the elite again and the probability at the associated position may come closer to zero. This situation may lead the elitism-based algorithms to update the PV towards the wrong direction. Elitism affects the pe-cGA's performance more than the ne-cGA's due to its everlasting elite.

The fb-cGA's PV converges to the solution faster than the other algorithms (Figure 10i). The shade representing probability values in Figure 10j turns white in a small number of generations. To efficiently apply our proposed technique to a real-world problem, the chromosome should have all of its bits uncorrelated with one another. It is important to realise that the fb-cGA evolves its PV by updating all associated probabilities of all bits. The update is done only one bit at a time without taking into account the information of the other bits. The results of One-Max problem shown in Figure 10 serve as an example that supports this claim: fb-cGA can solve the problem using far fewer number of generations (approximately 400, instead of about 1300 generations required by the traditional cGA). This significant outperformance stems from the new dynamic updating strategy: the proposed technique decides to update the probabilities with a larger step size based on the collected statistics. However, if the chromosome bits of the real-world problem are correlated with one another, the proposed algorithm might not find the best solution, as shown in the Royal Road and the Trap problem.

For a real-world optimisation problem that has multiple local optima, there is a higher chance that the fb-cGA may get stuck at a local optimum. For example, in the field of computational vision, efficient algorithms such as cGA may be used to recognise objects in an image. However, if the input image is complex, it might introduce various local optima in the search space. Because fb-cGA uses a trend in early generations to quickly decide to update PV with a larger updating step size, it might prematurely decide to search towards a seemingly promising direction at a certain time. Once the proposed algorithm gets stuck, it is hard to escape from the local optimum because it already has a strong bias in favour of either zero or one. To handle this kind of problem, we should wait longer to

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 121-135; doi: 10.14456/mijst.2015.10

see a correct trend before using a large step size. In addition, the step size should be incrementally increased during the evolution.

CONCLUSIONS

This paper presents a behaviour analysis of fb-cGA. To update the PV, the fb-cGA collects and utilises the update number of each probability in both up and down directions. The numbers of updates are used to adjust probability-updating step sizes, turning the vector towards the promising direction faster. When the effect of parameter *psize*/*n* on the algorithm performance was investigated, the results suggested that the newly proposed updating strategy should be used when the generation number is greater than *psize*/4. The analysis, through graphic representation of all probabilities from the first to the last generation, shows that the fb-cGA updates the PVs towards the solution quicker than the other algorithms and also requires fewer function evaluations.

ACKNOWLEDGEMENTS

This work was funded by Thailand Research Fund, the Office of Higher Education Commission and the Faculty of Informatics, Burapha University (Grant No. TRG5680073). We also thank our mentor, Prof. Prabhas Chongstitvatana, for his guidance, support and encouragement.

REFERENCES

1. G. R. Harik, F. G. Lobo and D. E. Goldberg, "The compact genetic algorithm", *IEEE Trans. Evol. Comput.*, **1999**, *3*, 287-297.
2. M. H. Afshar, "Application of a compact genetic algorithm to pipe network optimization problems", *Transact. A*: *Civil Eng.*, **2009**, *16*, 264-271.
3. R. D. Al-Dabbagh, M. S. Baba, S. Mekhilef and A. Kinsheel, "The compact genetic algorithm for likelihood estimator of first order moving average model", Proceedings of 2nd International Conference on Digital Information and Communication Technology and Its Applications, **2012**, Bangkok, Thailand, pp.474-481.
4. R. D. Al-Dabbagh, A. Kinsheel, M. S. Baba and S. Mekhilef, "An integration of compact genetic algorithm and local search method for optimizing ARMA (1, 1) model of likelihood estimator", Proceedings of 2nd International Conference on Computer Science and Computational Mathematics, **2013**, Kuala Lumpur, Malaysia, pp.60-67.
5. C. F. M. Toledo, M. S. Arantes, R. R. R. Oliveira and A. C. B. Delbem, "A hybrid compact genetic algorithm applied to the multi-level capacitated lot sizing problem", Proceedings of 28th Annual ACM Symposium on Applied Computing, **2013**, Coimbra, Portugal, pp.200-205.
6. R. R. Silva, H. S. Lopes and C. R. E. Lima, "A compact genetic algorithm with elitism and mutation applied to image recognition", *Lect. Notes Comput. Sci.*, **2008**, *5227*, 1109-1116.
7. P. Olarthichachart, S. Kaitwanidvilai and S. Karnprachar, "Trip frequency scheduling for traffic transportation management based on compact genetic algorithm", Proceedings of International MultiConference of Engineers and Computer Scientists, **2010**, Hong Kong, pp.1072-1074.
8. R. D. H. Al-Dabbagh, "Compact genetic algorithm for cryptanalysis trapdoor 0-1 knapsack cipher", *J. Al-Nahrain Univ.*, **2009**, *12*, 137-145.
9. A. Azouaoui, A. Berkani and M. Belkasmi, "An efficient soft decoder of block codes based on compact genetic algorithm", *Int. J. Comput. Sci. Iss.*, **2012**, *9*, 431-438.
10. H. Xing and R. Qu, "A compact genetic algorithm for the network coding based resource minimization problem", *Appl. Intell.*, **2012**, *36*, 809-823.

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 121-135; doi: 10.14456/mijst.2015.10

11. P. Gupta and R. Tiwari, "Solving three dimensional bin packing problem using elitism based genetic algorithm", *Int. J. Adv. Res. Comput. Eng. Technol.*, **2012**, *1*, 471-475.
12. P. K. Singh and N. Sahu, "Task scheduling in grid computing environment using compact genetic algorithm", *Int. J. Sci. Eng. Technol. Res.*, **2014**, *3*, 107-110.
13. Y. C. Huang, C. F. Chang, C. H. Chan, T. J. Yeh, Y. C. Chang, C. C. Chen and C. Y. Kao, "Integrated minimum-set primers and unique probe design algorithms for differential detection on symptom-related pathogens", *Bioinformatics*, **2005**, *21*, 4330-4337.
14. A. Bade, I. M. Aref, B. M. Hussien and Y. Eman, "Solving protein folding problem using elitism-based compact genetic algorithm", *J. Comput. Sci.*, **2008**, *4*, 525-529.
15. C. Aporntewan and P. Chongstitvatana, "A hardware implementation of the compact genetic algorithm", Proceedings of IEEE Congress on Evolutionary Computation, **2001**, Seoul, Korea, pp.624-629.
16. J. C. Gallagher and S. Vigraham, "A modified compact genetic algorithm for the intrinsic evolution of continuous time recurrent neural networks", Proceedings of Genetic and Evolutionary Computation Conference, **2002**, New York, USA, pp.163-170.
17. J. C. Gallagher, S. Vigraham and G. Kramer, "A family of compact genetic algorithms for intrinsic evolvable hardware", *IEEE Trans. Evol. Comput.*, **2004**, *8*, 111-126.
18. K. M. Timmerman, "A hardware compact genetic algorithm for hover improvement in an insect-scale flapping-wing micro air vehicle", *Master Thesis*, **2012**, Wright State University, USA.
19. C. Zhou, K. Meng and Z. Qiu, "Compact genetic algorithm mutated by bit", Proceedings of 4th World Congress on Intelligent Control and Automation, **2002**, Shanghai, China, pp.1836-1839.
20. B. V. Ha, R. E. Zich, M. Mussetta, P. Pirinoli and C. N. Dao, "Improved compact genetic algorithm for EM complex system design", Proceedings of 4th International Conference on Communications and Electronics, **2012**, Hue, Vietnam, pp.381-392.
21. S. Rimcharoen, D. Sutivong and P. Chongstitvatana, "Updating strategy in compact genetic algorithm using moving average approach", Proceedings of IEEE Conference on Cybernetics and Intelligent Systems, **2006**, Bangkok, Thailand, pp.690-695.
22. C. W. Ahn and R. S. Ramakrishna, "Elitism-based compact genetic algorithms", *IEEE Trans. Evol. Comput.*, **2003**, *7*, 367-385.
23. J. Y. Lee, S. M. Im and J. J. Lee, "Bayesian network-based non-parametric compact genetic algorithm", Proceedings of 6th IEEE International Conference on Industrial Informatics, **2008**, Daejeon, Korea, pp.359-364.
24. J. Y. Lee, M. S. Kim and J. J. Lee, "Compact genetic algorithms using belief vectors", *Appl. Soft Comput.*, **2011**, *11*, 3385-3401.
25. S. Phiromlap and S. Rimcharoen, "A frequency-based updating strategy in compact genetic algorithm", Proceedings of International Computer Science and Engineering Conference, **2013**, Nakorn Pathom, Thailand, pp.207-211.

© 2015 by Maejo University, San Sai, Chiang Mai, 50290 Thailand. Reproduction is permitted for noncommercial purposes.