กลไกการจัดตารางงานบนพื้นฐานของเซลลูลาร์ออโตมาตาและเจเนติกอัลกอริทึม

TASK SCHEDULING MECHANISM BASED ON CELLULAR AUTOMATA AND GENETIC ALGORITHM

อิงขวัญ วชิระชูเดช
INGKWAN VASHIRASHUDEJ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2560
KMITL-2017-EN-M-070-84

กลไกการจัดตารางงานบนพื้นฐานของเซลลูลาร์ออโตมาตาและเจเนติกอัลกอริทึม

TASK SCHEDULING MECHANISM BASED ON CELLULAR AUTOMATA AND GENETIC ALGORITHM

อิงขวัญ วชิระชูเดช
INGKWAN VASHIRASHUDEJ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2560

KMITL-2017-EN-M-070-84

TASK SCHEDULING MECHANISM BASED ON CELLULAR AUTOMATA AND GENETIC ALGORITHM

INGKWAN VASHIRASHUDEJ

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN COMPUTER ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2017

KMITL-2017-EN-M-070-84

คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ กลไกการจัดตารางงานบนพื้นฐานของเซลลูลาร์ออโตมาตาและเจเนติกอัลกอริทึม
Thesis Title Task Scheduling Mechanism Based on Cellular Automata and Genetic Algorithm
นักศึกษา นางสาวอิงขวัญ วชิระชูเดช (นักศึกษาในโครงการปริญญาตรีก้าวหน้า)
รหัสประจำตัว 55611513
ปริญญา วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์ ผศ.ดร.ศักดิ์ชัย ทิพย์จักษุรัตน์
หมายเลขวิทยานิพนธ์ KMITL-2017-EN-M-070-84

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
ผศ.ดร.ภุชงค์	อุทโยภาศ	
ผศ.ดร.สุรินทร์	กิตติธรกุล	
ดร.ธนัญชัย	ตรีภาค	
รศ.ดร.เกียรติกูล	เจียรนัยธนะกิจ	
ผศ.ดร.ศักดิ์ชัย	ทิพย์จักษุรัตน์	

วัน / เดือน/ ปี ที่สอบ วันอังคารที่ 18 กรกฎาคม พ.ศ. 2560 เวลา 15.00-17.00 น.
สถานที่สอบ ณ อาคาร A ชั้น 5 ห้องประชุม 2

คณะวิศวกรรมศาสตร์ รับรองแล้ว

(รองศาสตราจารย์ ดร. คมสัน มาลีสี)

คณบดี คณะวิศวกรรมศาสตร์
วันที่ 18 กรกฎาคม พ.ศ. 2560

หัวข้อวิทยานิพนธ์	กลไกการจัดตารางงานบนพื้นฐานของเซลลูลาร์ออโตมาตาและเจเนติกอัลกอริทึม
นักศึกษา	นางสาวอิงขวัญ วชิระชูเดช
รหัสประจำตัว	55611513
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
พ.ศ.	2560
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ผศ. ดร.ศักดิ์ชัย ทิพย์จักษุรัตน์

บทคัดย่อ

การจัดตารางงานเป็นหนึ่งในปัญหาที่สำคัญในระบบประมวลผลแบบกลุ่มเมฆ โดยมีจุดมุ่งหมายเพื่อกำหนดงานที่เข้ามาในระบบให้กับคอมพิวเตอร์เสมือน (Virtual Machine: VM) และลดระยะเวลาในการประมวลผลของงานทั้งหมดให้เหลือน้อยลงที่สุดเพื่อให้ประสิทธิภาพในการทำงานของระบบดีขึ้น ในงานวิจัยนี้ได้นำเสนอการประยุกต์เซลลูลาร์ออโตมาตาสำหรับการจัดตารางงานในระบบประมวลผลแบบกลุ่มเมฆหรือวิธีแคทส์ โดยการทำงานของแคทส์จะใช้วิธีสร้างกฎต่างๆสำหรับการหาวิธีการจัดตารางงานที่ดีที่สุด และกฎที่มีประสิทธิภาพจะถูกค้นพบโดยเจเนติกอัลกอริทึม สำหรับการประเมินประสิทธิผลของวิธีแคทส์สามารถทำได้โดยการจำลองสถานการณ์ ซึ่งตัวชี้วัดประสิทธิภาพของการทำงานจะอยู่ในเทอมของเวลาที่ใช้ในการประมวลผลของงานทั้งหมด (Makespan) โดยจะถูกนำไปเปรียบเทียบกับวิธีการจัดตารางงานในระบบการประมวลผลแบบกลุ่มเมฆโดยใช้วิธีมาก่อนได้ก่อน วิธีเจเนติกอัลกอริทึม และวิธีระบบอาณานิคมมดหรือเอซีโอ จากผลการจำลองสถานการณ์ได้แสดงให้เห็นว่าวิธีแคทส์สามารถทำให้เวลาที่ใช้ในการประมวลผลของงานทั้งหมดต่ำกว่าวิธีการอื่นๆเมื่อมีจำนวนของทรัพยากรน้อยในทุกๆจำนวนงานที่เข้ามาในระบบ วิธีแคทส์จะมีประสิทธิภาพที่ดีที่สุดกรณีจำนวนคอมพิวเตอร์เสมือนมีจำนวนไม่เกินแปดเครื่อง โดยประสิทธิภาพจะต่ำกว่าวิธีเอซีโอเมื่อมีจำนวนคอมพิวเตอร์เสมือนมากกว่าแปดเครื่อง อย่างไรก็ตามในกรณีที่มีจำนวนคอมพิวเตอร์เสมือนมากกว่าแปดเครื่อง วิธีแคทส์ยังคงให้ประสิทธิภาพที่ดีกว่าวิธีมาก่อนได้ก่อนและวิธีเจเนติกอัลกอริทึม

Thesis	Task Scheduling Mechanism Based on Cellular Automata and Genetic Algoirthm
Student	Miss Ingkwan Vashirashudej
Student ID.	55611513
Degree	Master of Engineering
Program	Computer Engineering
Year	2017
Thesis Advisor	Asst.Prof.Dr.Sakchai Thipchaksurat

ABSTRACT

Task scheduling is one of the major challenging issues in cloud computing system, which aims to assign tasks to virtual machines (VM) and minimizes the total execution time of tasks called makespan. In this research, we propose the application of cellular automata to the task scheduling in clodu computing systems. Our proposed scheme is called Cellular Automata for Task Scheduling or CATS scheme. In CATS scheme, the cellular automata (CA) is used for setting the task scheduling rules to find the optimal allocation for scheduling. The effective rule can be discovered from genetic algorithm (GA). The effectiveness of CATS scheme is evaluated by means of the simulation. We evaluate the performance of CATS in the terms of makespan by comparing with those of First-Come-First-Served (FCFS) scheme, genetic algorithm scheme, and Ant Colony Optimization (ACO) scheme. The simulation results show that CATS scheme can provide the lowest makespan compared to those schemes when the maximum number of VMs is 8 VMs for every number of tasks in the system. However, when the number of VMs is greater than 8 VMs such as 16 VMs or 32 VMs, CATS provides the better performance than those of FCFS and GA scheme but they are worse than ACO scheme.

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จลุล่วงไปได้ด้วยความกรุณาและความอนุเคราะห์จากบุคคลหลายฝ่าย โดยเฉพาะอย่างยิ่งข้าพเจ้าขอกราบขอบพระคุณอาจารย์ที่ปรึกษา ผศ.ดร.ศักดิ์ชัย ทิพย์จักษุรัตน์ ที่ให้ความกรุณาให้คำปรึกษา แนะนำ ตรวจสอบและแก้ไขข้อบกพร่องต่างๆ ตลอดจนช่วยแก้ปัญหาและให้กำลังใจข้าพเจ้าอย่างดียิ่งตลอดมา

ขอขอบคุณมนัสวี พี่ศุภกิจ พี่พนา จากห้องวิจัย 801 ที่ให้ความช่วยเหลือในการศึกษาครั้งนี้ รวมถึงคำแนะนำดีๆสำหรับการทำงานวิจัย

ท้ายที่สุดขอกราบขอบพระคุณบิดาและมารดาของข้าพเจ้า และญาติๆทุกท่านที่คอยให้การสนับสนุนมาโดยตลอด

ประโยชน์และคุณค่าของวิทยานิพนธ์เล่มนี้ ข้าพเจ้าขอมอบให้ทุกท่านที่มีส่วนเกี่ยวข้องในการช่วยดำเนินการให้วิทยานิพนธ์เสร็จสมบูรณ์ โดยเฉพาะครอบครัวของข้าพเจ้าซึ่งเป็นที่รักและเคารพยิ่ง ตลอดจนครูอาจารย์ที่เคารพทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้และถ่ายทอดประสบการณ์ที่ดีให้แก่ข้าพเจ้าเสมอมา

อิงขวัญ วชิระชูเดช

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันระบบการประมวลผลแบบกลุ่มเมฆได้รับความนิยมกันอย่างมากในเทคโนโลยีสารสนเทศ (Information Technology: IT) โดยจะย้ายการประมวลผลและข้อมูลจากคอมพิวเตอร์ไปยังดาต้าเซ็นเตอร์ (Data Center) ขนาดใหญ่ และการประมวลผลแบบกลุ่มเมฆยังหมายถึงแอพพลิเคชัน (Application) ที่จะถูกส่งเป็นบริการผ่านทางอินเตอร์เน็ต (Internet) และโครงสร้างพื้นฐานของกลุ่มเมฆ (Cloud Infrastructure) ผู้ใช้บริการสามารถทำงานบางอย่างได้ด้วยตนเอง เช่น เพิ่มความสามารถในการรองรับตามความต้องการที่เพิ่มขึ้น ทำการทดลองกับบริการใหม่ๆ หรือแม้กระทั่งลดความสามารถในการรองรับที่ไม่จำเป็นให้น้อยลง ประโยชน์ที่ได้จากการประมวลผลแบบกลุ่มเมฆคือ ค่าใช้จ่ายที่ใช้ในการปรับใช้กับธุรกิจหรือความสามารถทางเทคโนโลยีใหม่ๆ รวมถึงค่าใช้จ่ายในการเพิ่มประสิทธิภาพทางเศรษฐกิจจากการพัฒนาโครงสร้างพื้นฐานที่เคยมีราคาแพงจะลดลง

ซึ่งการปรับใช้โมเดลของการประมวลผลแบบกลุ่มเมฆสามารถทำได้สามรูปแบบ ได้แก่ พับบลิคคลาวด์ (Public Cloud) ไพรเวทคลาวด์ (Private Cloud) และไฮบริดคลาวด์ (Hybrid Cloud) การให้บริการและโครงสร้างพื้นฐานของพับบลิคคลาวด์นั้นถูกให้บริการจะอยู่ภายนอกโดยจะผ่านทางอินเตอร์เน็ต ในขณะที่บริการและโครงสร้างพื้นฐานของไพรเวทคลาวด์จะถูกสร้างอยู่ภายในเครือข่ายส่วนตัว (Private Network) และยังถูกออกแบบสำหรับการเข้าถึงอย่างจำกัดในองค์กรด้วย ดังนั้นไพรเวทคลาวด์จึงมีระดับของความปลอดภัยและการควบคุมที่สูงกว่า ส่วนไฮบริดคลาวด์จะเป็นการผสมผสานกันระหว่างไพรเวทคลาวด์และพับบลิคคลาวด์เพื่อให้มีคุณสมบัติที่มีความหลากหลายอยู่ภายในองค์กรเดียวกัน ตามระดับการกำหนดสาระสำคัญ (abstraction level) ของขีดความสามารถและการบริการของผู้ให้บริการ ทำให้การประมวลผลแบบกลุ่มเมฆได้ถูกแบ่งออกเป็นสามประเภท ได้แก่ Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (Paas) และ Software-as-a-Service (Saas)

ลักษณะเด่นของการประมวลผลแบบกลุ่มเมฆคือ การสร้างทรัพยากรเสมือนด้วยซอฟต์แวร์หรือเวอร์ชวลไลซ์เซชั่น (Virtualization) การกระจายการประมวลผล (Distribution) และความสามารถในการขยายระบบแบบไดนามิก (Dynamically Scalability) การสร้างทรัพยากรเสมือนจะเป็นลักษณะเด่นหลักๆในระบบการประมวลผลแบบกลุ่มเมฆ เมฆหรือคลาวด์ (Cloud) จะถูกสร้างจากคอมพิวเตอร์กายภาพ (Physical Machine) เป็นจำนวนมากซึ่งสามารถรันคอมพิวเตอร์เสมือน (Virtual Machine: VM) ได้หลายเครื่องในเวลาเดียวกัน โดยจะเป็นแอพพลิเคชันเลเยอร์ (Application Layer) ของระบบ ตัวอย่างของผู้ให้บริการประเภทนี้ได้แก่ Amazon Elastic Compute Cloud (EC2)

ปัญหาของระบบการประมวลผลแบบกลุ่มเมฆนั้นมีหลายปัญหา ซึ่งหนึ่งในปัญหาหลักเหล่านั้นคือการจัดตารางงาน (Task Scheduling) โดยในแพลตฟอร์ม (Platform) ของการประมวลผลแบบกลุ่มเมฆนั้น อัลกอริทึม (Algorithm) ของการจัดตารางงานแบบดั้งเดิมที่ใช้กันอยู่คือแบบการทำงานแบบตามลำดับก่อนหลัง (First-Come-First-Served: FCFS) แต่ว่าวิธีนี้ไม่ใช่วิธีที่เหมาะสมที่สุด (Optimal) และไม่สามารถใช้ทรัพยากรที่มีอยู่ให้เกิดประโยชน์สูงสุด (Utilize) ในแบบขนาน (Parallel) ได้

แต่อย่างไรก็ตาม เป็นการยากที่จะจัดการกำหนดงานไปยังทรัพยากรที่ใช้ในการประมวลผลในกลุ่มเมฆได้ด้วยตนเอง จึงได้มีการออกแบบและพัฒนาวิธีการและอัลกอริทึมต่างๆสำหรับการจัดตารางงานเพื่อให้การจับคู่ทรัพยากรที่เหมาะสมกับงานที่เข้ามาทำได้ดีขึ้น โดยปัญหาของการจัดตารางงานจะถูกจัดให้เป็นปัญหาประเภท NP-hard optimization

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

1.2.1 เพื่อศึกษาโครงสร้างพื้นฐานของระบบการประมวลผลแบบกลุ่มเมฆ

1.2.2 เพื่อศึกษาวิธีการจัดตารางงานในระบบต่างๆ

1.2.3 เพื่อนำเสนอวิธีการจัดตารางงานในระบบการประมวลผลแบบกลุ่มเมฆ โดยใช้เซลลูลาร์ออโตมาตาเพื่อลดเวลาทั้งหมดที่ใช้การประมวลผลของงานที่เข้ามาในระบบให้น้อยลง

1.2.4 เพื่อศึกษาการจำลองการทำงานของระบบการประมวลผลแบบกลุ่มเมฆด้วยไลบรารีคลาวด์ซิม (CloudSim)

1.3 แนวคิดที่ใช้ในงานวิจัย

การจัดตารางงานในระบบมีจุดมุ่งหมายหลักคือเพื่อลดระยะเวลาการประมวลผลของงานที่เข้ามาในระบบทั้งหมดหรือเมคสแปน (Makespan) ให้เหลือน้อยลงที่สุด จึงมีวิธีการต่างๆถูกคิดค้นขึ้นมารวมถึงวิธีการแบบเมตาฮิวริสติก ไม่ว่าจะเป็นเจเนติกอัลกอริทึมหรือระบบอาณานิคมมด เพื่อใช้ในการหาคำตอบที่ให้ผลลัพธ์ที่ดีที่สุด และได้มีการนำเซลลูลาร์ออโตมาตาซึ่งเป็นแบบจำลองทางคณิตศาสตร์มาประยุกต์ใช้ร่วมกับเจเนติกอัลกอริทึมสำหรับการจัดตารางงานในระบบมัลติโพรเซสเซอร์แบบสองโพรเซสเซอร์

ดังนั้นแนวคิดนี้จึงได้ถูกนำไปประยุกต์ใช้กับงานวิจัยสำหรับการจัดตารางงานในระบบการประมวลผลแบบกลุ่มเมฆ โดยสามารถมองว่าคอมพิวเตอร์เสมือนหนึ่งเครื่องในระบบการประมวลผลแบบกลุ่มเมฆคือหนึ่งโพรเซสเซอร์ในระบบมัลติโพรเซสเซอร์ได้ วิธีการที่ใช้ในงานวิจัยได้ถูกเรียกว่า CATS (Cellular Automata for Task Scheduling) และเนื่องจากคอมพิวเตอร์เสมือนมักจะมีจำนวนตั้งแต่สองเครื่องเป็นต้นไป จึงได้ทำการปรับปรุงให้วิธีการสามารถนำมาใช้งานจริงได้ในระบบการประมวลผลแบบกลุ่มเมฆ

1.4 การเปรียบเทียบระหว่างวิธีการที่นำเสนอกับวิธีที่มีอยู่เดิม

วิธีการที่จะนำมาเปรียบเทียบกับวิธี CATS ได้แก่ วิธีแบบมาก่อนได้ก่อน (First-Come-First-Served: FCFS) วิธีเจเตนิกอัลกอริทึม (Genetic Algorithm: GA) และวิธีระบบอาณานิคมมด (Ant Colony Optimization: ACO) ประสิทธิภาพที่ถูกประเมินคือเมคสแปน (Makespan) เพื่อเปรียบเทียบหาระยะการจัดตารางงานที่ใช้ระยะเวลาในการประมวลผลงานทั้งหมดที่น้อยที่สุด

1.5 ขอบเขตการวิจัย

1.5.1 การจำลองการทำงานของงานวิจัยนี้ใช้จำนวนของคอมพิวเตอร์เสมือนทั้งหมด 2, 4, 8, 16, 32 เครื่อง และมีดาต้าเซนเตอร์ทั้งหมด 10 ดาต้าเซนเตอร์

1.5.2 การจำลองการทำงานจะจำลองงานที่เข้ามาในระบบตั้งแต่จำนวน 100, 200, 300, 400, และ 500 งาน

1.5.3 งานที่เข้ามาในระบบทั้งหมดจะไม่มีความสัมพันธ์ต่อกัน (Independent Tasks) นั่นคือ แต่ละงานจะเป็นอิสระต่อกันสามารถถูกส่งไปประมวลผลได้โดยไม่ขึ้นกับงานอื่น

1.6 ขั้นตอนการศึกษา

1.6.1 ศึกษาทฤษฎีของระบบการประมวลผลแบบกลุ่มเมฆ

1.6.2 ศึกษาทฤษฎีของเซลลูลาร์ออโตมาตา

1.6.3 ศึกษาทฤษฎีของเจเนติกอัลกอริทึม

1.6.4 ศึกษาวิธีการจัดตารางงานในระบบต่างๆ

1.6.5 ศึกษาวิธีการจัดตารางงานในระบบการประมวลผลแบบกลุ่มเมฆ

1.6.6 ศึกษาวิธีการจัดตารางงานโดยใช้ระบบอาณานิคมมด

1.6.7 ศึกษาการเขียนโปรแกรมด้วยภาษาจาวาและการใช้งานไลบรารี CloudSim

1.6.8 จำลองการทำงานของวิธีเจเนติกอัลกอริทึม

1.6.8 จำลองการทำงานของวิธีระบบอาณานิคมมด

1.6.9 จำลองการทำงานของวิธี CATS และหาพารามิเตอร์ที่เหมาะสมกับการใช้งาน

1.6.10 เปรียบเทียบประสิทธิภาพของ CATS กับวิธี FCFS, GA, ACO

1.6.11 สรุปผลการจำลองการทำงาน

1.6.12 รายงานผลการวิจัย อภิปรายผลการทดลองและข้อเสนอแนะ

1.7 ประโยชน์ที่คาดว่าจะได้รับ

1.7.1 สามารถนำเซลลูลาร์ออโตมาตามาทำงานร่วมกับเจเนติกอัลกอริทึมและเพื่อประยุกต์ใช้สำหรับการจัดตารางงานในระบบการประมวลผลแบบกลุ่มเมฆ

1.7.2 สามารถลดเวลาที่ใช้ในการประมวลผลของงานที่เข้ามาในระบบให้น้อยลง

1.7.3 การนำเซลลูลาร์ออโตมาตามาทำงานร่วมกับเจเนติกอัลกอริทึมจะเป็นการปรับปรุงประสิทธิภาพของเจเนติกอัลกอริทึมให้ทำงานได้ผลลัพธ์ที่ดีขึ้น

1.8 เนื้อหาของวิทยานิพนธ์

วิทยานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 6 บท ประกอบด้วย

บทที่ 1 บทนำ

อธิบายถึง ความเป็นมาของงานวิจัย ความมุ่งหมายและวัตถุประสงค์ แนวคิดที่ใช้ในงานวิจัย การเปรียบเทียบระหว่างวิธีการที่นำเสนอกับวิธีที่มีอยู่เดิม ขอบเขตของการวิจัย ขั้นตอนการศึกษา และประโยชน์ที่คาดว่าจะได้รับ

บทที่ 2 ความรู้พื้นฐานที่เกี่ยวข้องกับงานวิจัย

อธิบายถึง การความรู้พื้นฐาน ทฤษฎี โครงสร้างของระบบและรวมถึงสถาปัตยกรรมต่างๆ ที่ได้ถูกนำเสนอในระบบการประมวลผลแบบกลุ่มเมฆ และทฤษฎีการจัดตารางงานในระบบ

บทที่ 3 งานวิจัยที่เกี่ยวข้อง

อธิบายถึง การนำเซลลูลาร์ออโตมาตามาประยุกต์ใช้งานในรูปแบบต่างๆ รวมถึงขั้นตอนการทำงานของวิธีการจัดสรรงานโดยใช้เซลลูลาร์ออโตมาตาร่วมกับเจเนติกอัลกอริทึมบนระบบมัลติโพรเซสเซอร์ และการใช้ระบบอาณานิคมมดในการจัดสรรงานบนระบบการประมวลผลแบบกลุ่มเมฆ

บทที่ 4 กลไกการจัดตารางงานบนพื้นฐานของเซลลูลาร์ออโตมาตาสำหรับระบบการประมวลผลแบบกลุ่มเมฆ

อธิบายถึง วิธีการที่นำเสนอ จากการนำเซลลูลาร์ออโตมาตามาประยุกต์ใช้ร่วมกับเจเนติกอัลกอริทึมเพื่อค้นหากฎที่มีประสิทธิภาพในการลดเวลาในการประมวลผลของงานที่เข้ามาในระบบ เรียกโดยย่อว่า CATS

บทที่ 5 การประเมินประสิทธิภาพ

แสดงให้เห็นถึง แบบจำลองการทำงาน เพื่อวัดประสิทธิภาพเปรียบเทียบระหว่างวิธี CATS กับวิธีแบบมาก่อนได้ก่อน (FCFS) วิธีเจเนติกอัลกอริทึม (GA) ระบบอาณานิคมมด (ACO)

บทที่ 6 สรุปผลการวิจัยและข้อเสนอแนะ

สรุปผลการวิจัยจากการจำลองการทำงาน และข้อเสนอต่างๆ พร้อมทั้งปัญหาที่เกิดขึ้น

บทที่ 2
ความรู้พื้นฐานที่เกี่ยวข้องกับงานวิจัย

ในบทนี้จะอธิบายเกี่ยวกับเทคโนโลยีการประมวลผลในรูปแบบต่างๆ รวมถึงทฤษฎีพื้นฐานและสถาปัตยกรรมของระบบการประมวลผลแบบกลุ่มเมฆ (Cloud Computing Systems)

2.1 ประเภทของการประมวลผล

2.1.1 Utility Computing

Utility computing เป็นแนวคิดที่พัฒนาโดยจอห์น แมคคาร์ที (John McCarthy) เป็นการใช้งานทรัพยากรการประมวลผลในลักษณะเดียวกันกับการใช้สาธารณูปโภคที่มีค่าใช้จ่ายตามการใช้งาน ไม่ว่าจะเป็นการซื้อ ใช้งาน และการบำรุงรักษาอุปกรณ์และแอพลิเคชันต่างๆ

2.1.2 Distributed Computing

แนวคิดของการประมวลผลแบบการจาย (Distributed Computing) เป็นการนำคอมพิวเตอร์หลายๆ เครื่องมาเชื่อมต่อกันผ่านทางระบบเครือข่ายเพื่อนำไปใช้ประมวลผลร่วมกัน โดยรันร่วมกันเป็นระบบเดียว งานที่เข้ามาจะถูกแบ่งออกเป็นหลายๆส่วนเพื่อกระจายไปยังคอมพิวเตอร์ที่อยู่ในระบบเพื่อให้ช่วยกันประมวลผล จึงมีลักษณะเป็นการประมวลผลแบบกระจายจากศูนย์กลาง (Decentralize) ทำให้มีความรวดเร็วในการประมวลผลมากยิ่งขึ้น

นอกจากนี้การประมวลผลแบบกระจายยังมีคุณสมบัติของความยืดหยุ่น (Scalability) โดยสามารถเพิ่มและลดจำนวนคอมพิวเตอร์ได้โดยง่าย รวมถึงเมื่อคอมพิวเตอร์เครื่องใดเครื่องหนึ่งไม่สามารถทำงานได้ไม่มีผลทำให้ระบบหยุดการทำงาน โมเดลของการทำงานแบบกระจายจะมีลักษณะเป็นแม่ข่าย-ลูกข่าย (Client-Server) ซึ่งระบบที่พัฒนาจากการประมวลผลแบบกระจายมีดังนี้

1) คลัสเตอร์ (Clusters)

คอมพิวเตอร์ในระบบจะเชื่อมต่อกันแบบ LAN (Local Area Network) และแชร์ทรัพยากรร่วมกัน เช่น ซีพียู ที่เก็บข้อมูล เป็นต้น โดยมีซอฟต์แวร์ที่ช่วยบริหารจัดการ ดังนั้นจึงสามารถใช้คลัสเตอร์เสมือนกับซุปเปอร์คอมพิวเตอร์ได้ โดยการนำไปประมวลผลงานเดียวร่วมกันแบบขนาด หรือนำหลายๆงานที่เป็นอิสระจากกันมาประมวลผลได้ โดยเน้นไปที่ความเร็วในการประมวลผล

2) กริด (Grids)

เป็นการเชื่อมทรัพยากรคอมพิวเตอร์ระหว่างหน่วยงานหรือองค์กรหลายๆระบบมาเชื่อมต่อกันเพื่อให้ทำงานเหมือนระบบใหญ่เพียงระบบเดียวผ่าน WAN (Wide Area Network) ที่มีแบนด์วิธต่ำ โดยเน้นไปที่การใช้ทรัพยากรการประมวลผลให้เกิดประโยชน์สูงสุด และระบบย่อยภายในกริดสามารถแยกจัดการอย่างอิสระได้

3) กลุ่มเมฆ (Clouds)

เป็นการประมวลผลที่ย้ายจากการประมวลผลแบบเดิมไปประมวลผลผ่านเครือข่ายอินเทอร์เน็ตซึ่งจะคล้ายกับกริดและมีความยืดหยุ่นรวมถึงปรับให้เหมาะสมกับกิจกรรมทางธุรกิจมากขึ้น ซึ่งคอมพิวเตอร์ที่เชื่อมต่อกันไม่จำเป็นต้องมีฮาร์ดแวร์และระบบปฏิบัติการที่เหมือนกันทั้งหมด โดยอิงกับความต้องการของผู้ใช้งานเป็นหลัก ซึ่งสามารถระบุความต้องการไปยังซอฟต์แวร์ที่ใช้ควบคุมจัดการระบบได้ เนื่องจากมีความหลากหลายวิธีและการพัฒนาระบบการประมวลผลแบบกลุ่มเมฆ นิยามจึงแตกต่างกันไปตามเทคโนโลยีและวิธีการที่ใช้ในการพัฒนา โดยรายละเอียดจะอธิบายในส่วนถัดไป

2.2 ระบบการประมวลผลแบบกลุ่มเมฆ

2.2.1 ประวัติความเป็นมา

ย้อนกลับไปในปี ค.ศ.1960 จอห์น แมคคาร์ธี (John McCarthy) ได้เสนอความคิดเห็นว่า ในวันหนึ่งการประมวลผลจะถูกทำให้สามารถใช้งานได้แบบสาธารณะ ซึ่งในเวลาต่อมาแนวความคิดนี้ได้กลายเป็นหลักการระบบการประมวลผลแบบกลุ่มเมฆในปัจจุบัน

โดยในช่วง ค.ศ.1960-1970 ตามบริษัทหลายๆที่ได้มีเมนเฟรมคอมพิวเตอร์ขนาดใหญ่ (mainframe computer) ซึ่งเป็นคอมพิวเตอร์ที่มีสมรรถนะสูงมาก เพื่อใช้ในการให้บริการกับพนักงานในบริษัทในการเข้าถึงระบบโดยผ่านคอมพิวเตอร์เทอมินอล (Dumb Terminal) ซึ่งเป็นเครื่องที่รับหรือส่งข้อมูลได้แต่เพียงอย่างเดียว คอมพิวเตอร์เมนเฟรมเหล่านี้จะเก็บข้อมูลทั้งหมดและทำการประมวลผล และเมื่อผู้ใช้งานต้องการรับหรือส่งข้อมูลจะทำการส่งรายการจากคอมพิวเตอร์เทอมินอลไปยังคอมพิวเตอร์เมนเฟรม จากนั้นข้อมูลก็จะถูกส่งคืนกลับมายังคอมพิวเตอร์เทอมินอล แต่ทว่าในยุคนั้นคอมพิวเตอร์เมนเฟรมเหล่านี้จะมีราคาที่แพงมาก

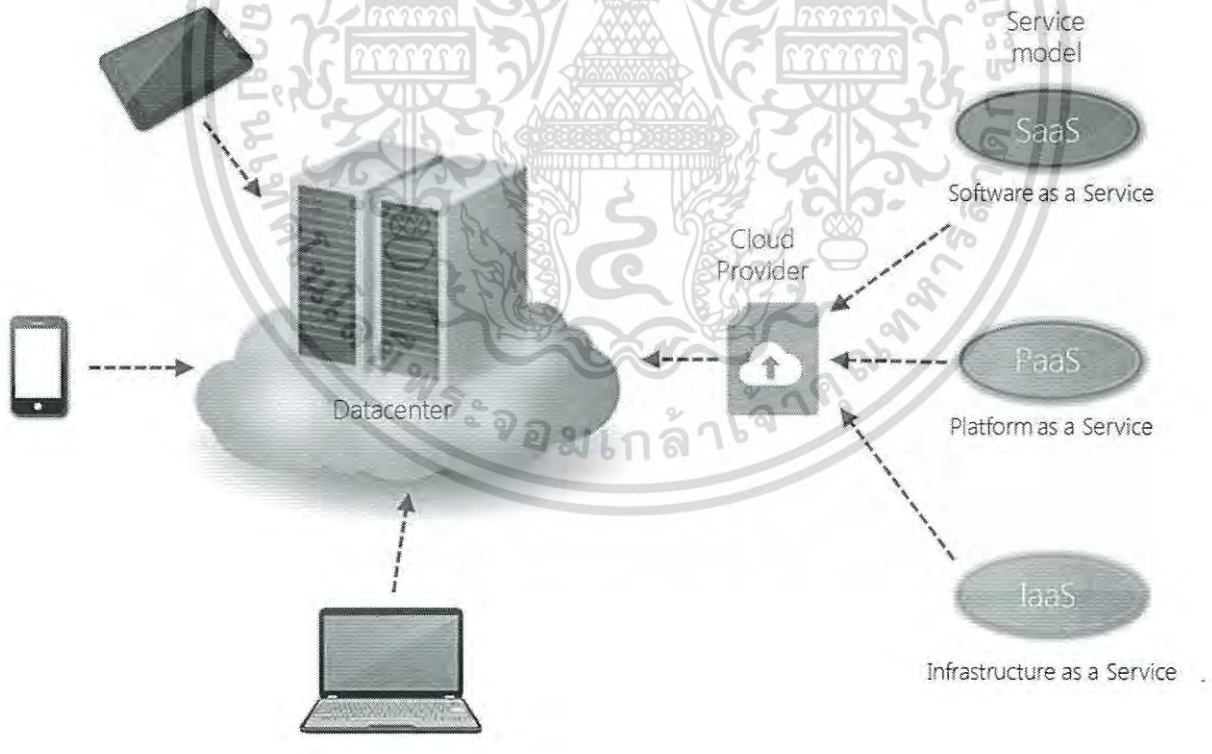
ในปี ค.ศ.1980 บริษัทต่างๆได้ตระหนักว่าคอมพิวเตอร์ธรรมดาที่เป็นแบบเซิร์ฟเวอร์เบส (Server-Based) จะสามารถติดตั้งใช้งานได้ในราคาที่ถูกกว่าแบบคอมพิวเตอร์เมนเฟรม ดังนั้นคอมพิวเตอร์เมนเฟรมเหล่านี้จึงถูกแทนที่ด้วยคอมพิวเตอร์แบบเซิร์ฟเวอร์เบส

ในช่วงต้นปี ค.ศ.1990 เมื่อมีการเริ่มใช้งานอินเทอร์เน็ต (Internet) กันอย่างแพร่หลาย การเข้าใช้งานไปยังเซิร์ฟเวอร์ขนาดใหญ่จึงกลับมาอีกครั้ง และในปี ค.ศ.1999 บริษัทเซลฟอร์ส (Salesforce) ได้ก่อตั้งขึ้นโดยมาร์ค เบอนีออฟ (Marc Benioff) และปาร์คเกอร์ แฮร์ริส (Parker Harris) พวกเข้าใช้เทคโนโลยีหลายอย่างที่พัฒนาโดยบริษัทเช่นกูเกิ้ล (Google) และยาฮู (Yahoo) เพื่อนำไปประยุกต์ใช้ในเชิงธุรกิจ นอกจากนี้ยัง

ใช้แนวคิดของออนดีมานด์ (On demand) และเอสเอเอเอส (SaaS) กับธุรกิจของตนเอง ซึ่งเป็นหนึ่งในลักษณะของระบบการประมวลผลแบบกลุ่มเมฆ

ในปี ค.ศ.2002 บริษัทอะเมซอน (Amazon) ได้เปิดตัวอะเมซอนเว็บเซอร์วิส (Amazon Web Services: AWS) โดยให้บริการทรัพยากรที่ใช้ในการประมวลผล เช่น ซีพียู (CPU), หน่วยความจำ (Memory), เครือข่าย รวมถึงแพลตฟอร์ม (Platform) และบริการอื่นๆ เป็นแนวคิดในการให้บริการที่คิดตามค่าใช้จ่ายตามการใช้งานจริง เพื่อช่วยลดต้นทุนและสร้างรายได้ให้กับผู้ใช้บริการ จากนั้นในปี 2006 อะเมซอนได้เปิดตัวบริการ อะเมซอนอีซีทู (Amazon Elastic Compute Cloud: EC2) ให้กับบริษัทขนาดเล็กและผู้ใช้งานอื่น โดยผู้ใช้งานได้สามารถให้บริการในการรันซอฟต์แวร์ของตนเองบนกลุ่มเมฆ (Cloud) ตามความต้องการใช้บริการ

ในปี ค.ศ.2007 กูเกิ้ลและไอบีเอ็ม รวมถึงมหาวิทยาลัยอีกจำนวนหนึ่งได้เริ่มวิจัยระบบการประมวลผลแบบกลุ่มเมฆกัน และปีถัดมาไมโครซอฟต์ (Microsoft) ได้เข้ามาสู่อุตสาหกรรมระบบการประมวลผลแบบกลุ่มเมฆ โดยใช้ชื่อผลิตภัณฑ์ว่าอะซัว (Azure) ซึ่งเป็นระบบปฏิบัติการ และมีการพัฒนามาอย่างต่อเนื่องจนถึงทุกวันนี้ ตัวอย่างของระบบการประมวลผลแบบกลุ่มเมฆแสดงให้เห็นดังรูปที่ 2.1



รูปที่ 2.1 ตัวอย่างระบบการประมวลผลแบบกลุ่มเมฆ

2.2.2 ทฤษฎีและหลักการพื้นฐาน

นิยามของระบบการประมวลผลแบบกลุ่มเมฆนั้นยังไม่แน่นอนและมีความหลากหลายมาก แต่ความหมายโดยรวม คือ ระบบการประมวลผลแบบกลุ่มเมฆคือวิธีการประมวลผลที่อิงกับความต้องการกับผู้ใช้ โดยผู้ใช้จะระบุความต้องการไปยังซอฟต์แวร์ของระบบการประมวลผลแบบกลุ่มเมฆ จากนั้นซอฟต์แวร์จะร้องขอให้ระบบจัดสรรทรัพยากรและบริการให้ตรงกับความต้องการของผู้ใช้งาน ทั้งนี้ระบบยังสามารถเพิ่มและลดจำนวนของทรัพยากร รวมถึงเสนอบริการให้พอเหมาะกับความต้องการของผู้ใช้งานโดยที่ผู้ใช้ไม่จำเป็นต้องทราบว่าการทำงานเบื้องหลังของระบบเป็นอย่างไร และระบบการประมวลผลแบบกลุ่มเมฆนั้นจะมองว่าทุกแอพพลิชันหรือส่วนประกอบของระบบจะถูกนำมาใช้เป็นบริการหรือส่วนหนึ่งของบริการให้กับผู้ใช้งาน ดังนั้นจะเห็นว่าระบบการประมวลผลแบบกลุ่มเมฆนั้นจะเป็นการนำเทคโนโลยีและธุรกิจมาประกอบเข้าด้วยกัน

2.2.3 ลักษณะสำคัญ

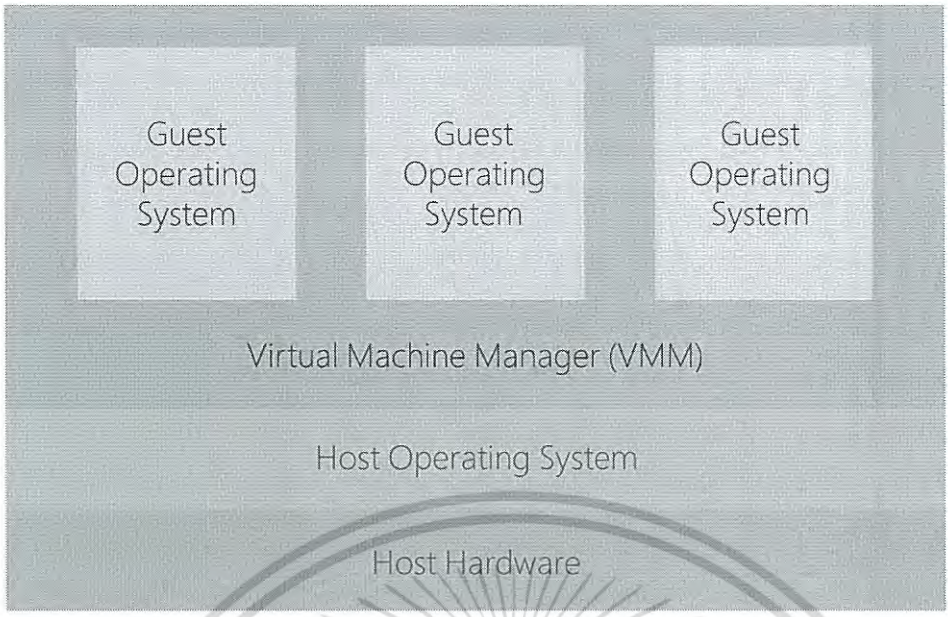
ระบบการประมวลผลแบบกลุ่มเมฆจะมีลักษณะสำคัญทั้งหมดสามประการ ดังนี้

1) ความสามารถในการปรับขนาด (Scalability) ผู้ใช้งานสามารถเพิ่มและลดทรัพยากรได้ง่ายผ่านซอฟต์แวร์โดยซอฟต์แวร์ก็จะไปร้องขอบริการจากระบบ และสะดวกเพื่อรองรับโหลดที่เข้ามาในระบบ ตัวอย่างเช่น ผู้ใช้งานได้ใช้บริการเช่าพื้นที่สร้างเว็บไซต์บนระบบการประมวลผลแบบกลุ่มเมฆ เมื่อถึงช่วงเวลาหนึ่งที่มีผู้เข้ามาใช้บริการเว็บไซต์เป็นจำนวนมากเป็นประจำ เจ้าของเว็บไซต์สามารถกำหนดให้เพิ่มจำนวนทรัพยากรให้เพียงพอต่อความต้องการใช้งาน และเมื่อผ่านช่วงเวลานั้นก็สามารถลดทรัพยากรที่มีอยู่ออกไปได้เช่นกัน

2) เทคโนโลยีเวอร์ชวลไลเซชั่น (Virtualization) โดยทรัพยากรสำหรับการประมวลผลจะถูกแบ่งเป็นคอมพิวเตอร์เสมือน (Virtual Machine: VM) จำนวนหลายๆเครื่อง โดยมีซอฟต์แวร์ทำหน้าที่ควบคุมและจัดการ แต่ละเครื่องจะสามารถทำงานได้อย่างอิสระ ทำให้ประหยัดในเรื่องต้นทุนของฮาร์ดแวร์และโครงสร้างพื้นฐาน นอกจากนี้ยังมีความยืดหยุ่นโดยสามารถย้ายคอมพิวเตอร์เสมือนไปยังอีกเครื่องหรือเพิ่มลดได้อย่างรวดเร็ว

สถาปัตยกรรมเบื้องต้นของของเทคโนโลยีเวอร์ชวลไลเซชั่นเป็นไปตามรูปที่ 2.2 ซึ่งแสดงให้เห็นถึงพื้นฐานของเทคโนโลยีเวอร์ชวลไลเซชั่น มีซอฟต์แวร์ที่ใช้ควบคุมจัดการเรียกว่า Virtual Machine Manager หรือ VMM และสามารถสร้างระบบปฏิบัติการบนคอมพิวเตอร์เสมือน (Guest Operating System) ได้หลายระบบตามความสามารถของเครื่องโฮสต์ที่นำมาใช้งาน

ซึ่งระบบปฏิบัติการเสมือนทั้งหมดจะทำงานบนระบบปฏิบัติการของโฮสต์ (Host Operating System) และทำงานอยู่บนฮาร์ดแวร์ของโฮสต์อีกที (Host Hardware) ตัวอย่างของผลิตภัณฑ์เวอร์ชวลไลเซชั่น เช่น Vmware, Citrix XenServer, Microsoft Hyper-V



รูปที่ 2.2 สถาปัตยกรรมเบื้องต้นของเทคโนโลยีเวอร์ชวลไลเซชั่น

3) ความน่าเชื่อถือ (Reliability) เป็นสิ่งที่สำคัญในการให้บริการ ระบบที่ดีจะต้องสามารถทำงานและให้บริการได้ตลอดเวลา นั่นคือมี Downtime ต่ำ ในระบบการประมวลผลแบบกลุ่มเมฆจะสามารถดึงทรัพยากรจากหลายแหล่งมาใช้ในการประมวลผล เป็นการป้องกันการล่มของระบบ ทำให้ระบบพร้อมที่จะให้บริการตลอดเวลา

2.2.4 ประเภทของกลุ่มเมฆ

1) พับลิคคลาวด์ (Public cloud)

พับลิคคลาวด์เป็นระบบที่ทุกคนสามารถเข้าถึงได้ผ่านทางอินเทอร์เน็ต ซึ่งผู้ให้บริการสร้างทรัพยากรไว้ให้บริการเพื่อให้ผู้ใช้งานใช้ได้ตามความต้องการ แต่ผู้ใช้จะมีสิทธิในการควบคุมที่จำกัดโดยจะขึ้นอยู่กับสิทธิที่ผู้ให้บริการได้มอบให้ โดยการใช้บริการพับลิคคลาวด์อาจจะมีทั้งเสียและไม่เสียค่าใช้จ่าย ตัวอย่างเช่น Amazon Elastic Compute Cloud (EC2)

2) ไพรเวทคลาวด์ (Private cloud)

เป็นระบบที่เป็นแบบส่วนตัว โดยองค์กรจะสร้างขึ้นมาเพื่อนำไปใช้งานภายในองค์กรและมีการบริหารจัดการระบบกันเอง ซึ่งจะมุ่งเน้นเรื่องความปลอดภัยของข้อมูลเป็นหลักเพราะสามารถควบคุมและปรับปรุงระบบความปลอดภัยได้ด้วยตนเอง อีกทั้งผู้ใช้งานภายนอกไม่สามารถเข้าถึงระบบได้

3) ไฮบริดคลาวด์ (Hybrid cloud)

เป็นการผสมผสานกันระหว่างพับลิคและไพรเวทคลาวด์ โดยบางส่วนจะสร้างขึ้นมาเองและบางส่วนจะใช้บริการจากผู้ให้บริการภายนอก

2.2.5 การให้บริการ (Services)

รูปแบบการให้บริการของระบบการประมวลผลแบบกลุ่มเมฆจะแบ่งออกได้เป็นสามรูปแบบ ซึ่งจะประมวลผลตามความต้องการของผู้ใช้ซึ่งทำให้เรียกใช้งานได้อย่างง่ายและสะดวก ซึ่งจะแสดงให้เห็นระดับชั้นของการให้บริการดังรูปที่ 2.3

1) การให้บริการซอฟต์แวร์ (Software as a Service: SaaS)

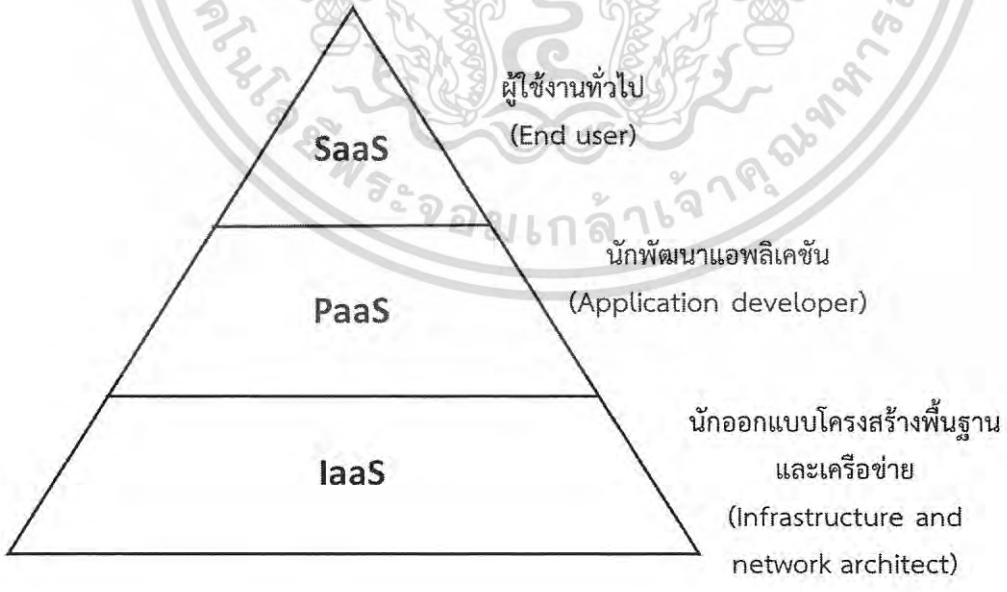
บริการเหล่านี้คือการที่ให้ผู้ใช้งานเรียกใช้แอพพลิเคชันต่างๆบนอินเตอร์เน็ตโดยไม่จำเป็นต้องติดตั้งแอพลิเคชันลงบนคอมพิวเตอร์ของตนเอง โดยปกติแล้วผู้ใช้งานจะทำการรันแอพพลิเคชันเหล่านั้นผ่านเว็บบราวเซอร์ ซึ่งผู้ใช้งานจะไม่รู้เบื้องหลังการทำงานของแอพลิเคชันเหล่านี้ ตัวอย่างของบริการแบบเอสเอเอเอส เช่น Google Docs

2) การให้บริการแพลตฟอร์ม (Platform as a Service: PaaS)

บริการนี้จะมุ่งเน้นไปที่การให้บริการเช่าแพลตฟอร์ม (Platform) ซึ่งประกอบไปด้วยฮาร์ดแวร์ (Hardware) ระบบปฏิบัติการ (Operating System) ที่จัดเก็บข้อมูล (Storage) ผ่านทางอินเทอร์เน็ต ทำให้ผู้ใช้งานสามารถปรับเปลี่ยนความสามารถของระบบได้ง่ายและช่วยลดค่าใช้จ่ายในการดูแลระบบ ตัวอย่างของบริการแบบพีเอเอเอส เช่น Google App engine

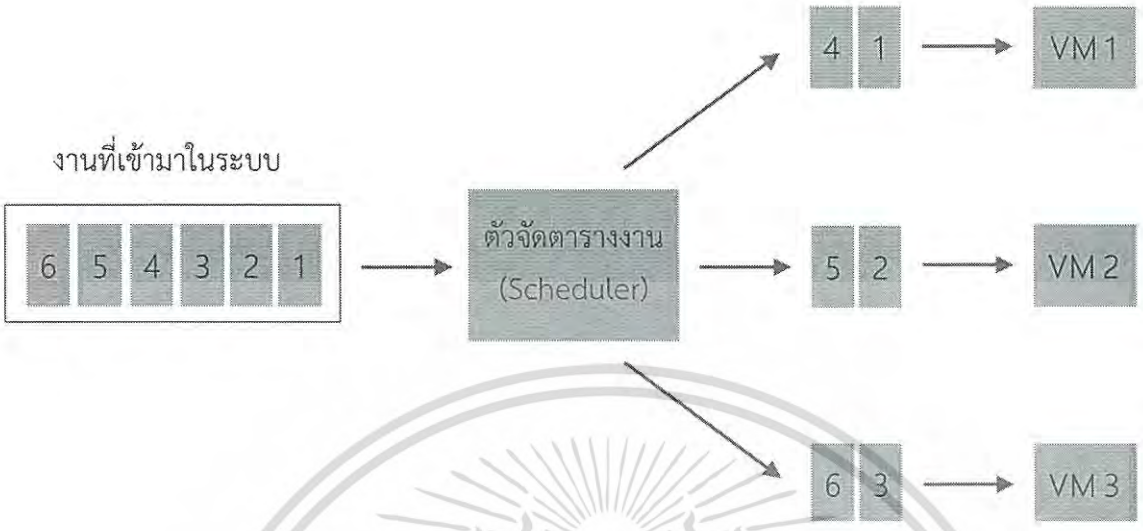
3) การให้บริการโครงสร้างพื้นฐาน (Infrastructure as a Service: IaaS)

เป็นบริการที่เกี่ยวกับโครงสร้างพื้นฐานของระบบคอมพิวเตอร์ ไม่ว่าจะเป็นเซิร์ฟเวอร์ (Server) การเชื่อมต่อ (Connection) ฮาร์ดแวร์และแอพลิเคชัน รวมถึงทรัพยากรจะถูกจัดหามาให้โดยผู้ให้บริการ โดยผู้ให้บริการจะเป็นเจ้าของอุปกรณ์และรับผิดชอบการทำงาน การบำรุงรักษา



รูปที่ 2.3 สถาปัตยกรรมการให้บริการบนระบบการประมวลผลแบบกลุ่มเมฆ

2.3 การจัดตารางงานในระบบการประมวลผลแบบกลุ่มเมฆ



รูปที่ 2.4 ตัวอย่างการจัดตารางงานในระบบการประมวลผลแบบกลุ่มเมฆ

ที่ผ่านมาได้มีอัลกอริทึมในการจัดตารางงานหลายรูปแบบที่มีการใช้งานบนระบบแบบกระจาย (Distributed System) ถูกนำมาประยุกต์ใช้กับระบบการประมวลผลแบบกลุ่มเมฆโดยมีการพัฒนาปรับปรุงให้เหมาะสมกับการใช้งาน [18] โดยจุดมุ่งหมายหลักของอัลกอริทึมในการจัดตารางงานคือเพื่อลดเวลาที่ใช้ในการประมวลผลให้น้อยลง และเพิ่มประสิทธิภาพในการทำงานให้กับระบบการประมวลผล โดยการจัดตารางงานในระบบการประมวลผลแบบกลุ่มเมฆมีลักษณะดังรูป 2.4 ตัวอย่างของวิธีการในการจัดตารางงานแบบพื้นฐานมีดังนี้

2.3.1 First Come First Serve algorithm (FCFS)

งานจะถูกนำไปประมวลผลตามลำดับที่เข้ามาในระบบ นั่นคือมาก่อนได้ทำงานก่อน โดยงานที่เข้ามาในระบบจะถูกไปเก็บไว้ในคิว (Queue) เพื่อรอรับการประมวลผล อัลกอริทึมนี้สามารถนำไปใช้งานได้ง่ายและไม่ซับซ้อน

2.3.2 Round Robin algorithm (RR)

ในการจัดตารางงานแบบ RR นั้น แต่ละงานจะได้รับเวลาที่จำกัดสำหรับซีพียูสำหรับใช้ในการประมวลผล โดยเรียกเวลานั้นว่า ไทม์สไลด์ (Time-Slice) หรือ ควอนตัม (Quantum) ถ้าหากว่างานนั้นไม่สามารถทำให้เสร็จได้ภายในเวลาที่กำหนดแล้วก็จะถูกงานถัดไปที่อยู่ในคิว (Queue) เข้ามาทำแทนที่และงานนั้นก็จะถูกนำไปต่อยังท้ายสุดในคิว

2.3.3 Min-Min algorithm

อัลกอริทึมนี้จะเลือกงานที่มีขนาดเล็กที่สุดก่อนแล้วส่งไปยังคอมพิวเตอร์เสมือน จึงทำให้งานที่มีขนาดใหญ่ต้องใช้เวลาในการรอถูกประมวลผลนานกว่า

2.3.4 Max-Min algorithm

อัลกอริทึมนี้จะเลือกงานที่มีขนาดใหญ่ที่สุดก่อนแล้วส่งไปยังคอมพิวเตอร์เสมือน จึงทำให้งานที่มีขนาดเล็กต้องใช้เวลาในการรอถูกประมวลผลนานd;jk

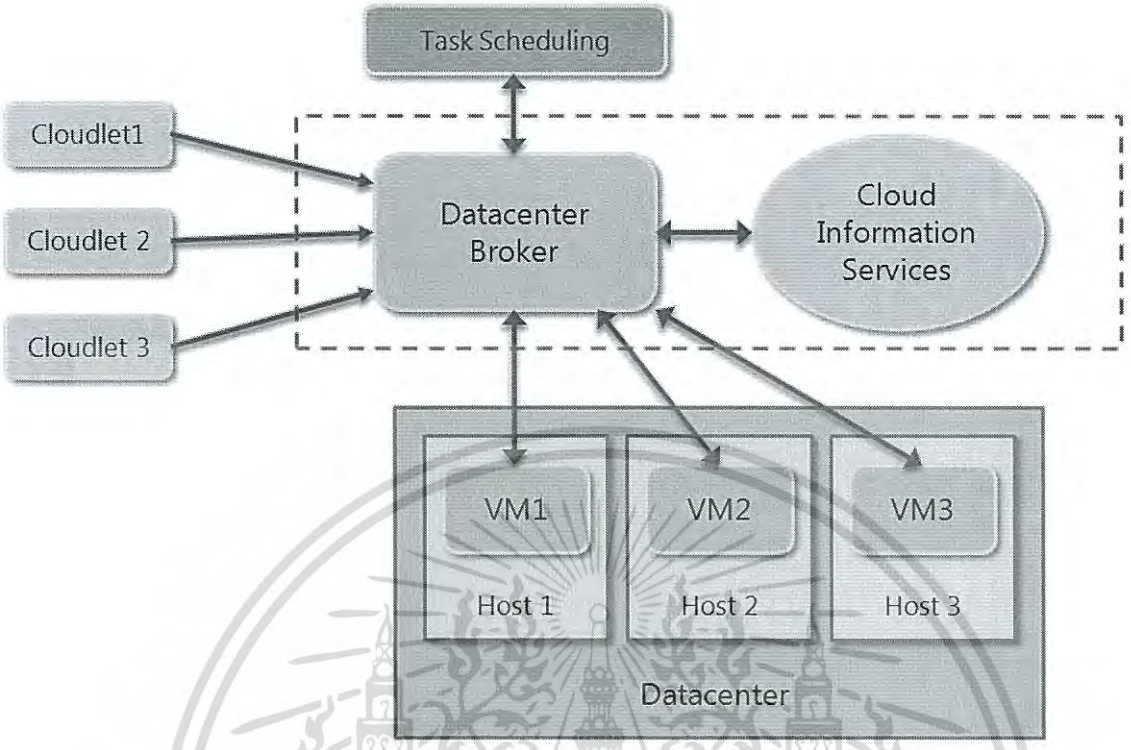
2.3.5 Priority scheduling algorithm

แต่ละงานจะถูกกำหนดลำดับความสำคัญ (Priority) โดยการพิจารณาจากพารามิเตอร์ (Parameter) ต่างๆ และจะถูกนำไปประมวลผลโดยเลือกงานที่มีความสำคัญมากที่สุดก่อน

2.3.6 Most fit task scheduling algorithm

เป็นการเลือกประมวลผลงานที่มีความเหมาะสมที่สุดในคิวก่อน ซึ่งความเหมาะสมนั้นจะพิจารณาเพียงบางพารามิเตอร์เท่านั้น

โดยเมื่อมีการเลือกวิธีการจัดตารางงานให้กับระบบประมวลผลแบบกลุ่มเมฆเป็นที่เรียบร้อยแล้ว จะมีกระบวนการจัดตารางงาน (Scheduling Process) ซึ่งสามารถแบ่งออกเป็นทั้งหมดสามขั้นตอนหลัก ได้แก่ ขั้นตอนการค้นหาและคัดกรองทรัพยากร (Resource Discovering and Filtering) ขั้นตอนการเลือกทรัพยากร (Resource Selection) และขั้นตอนการส่งมอบงาน (Task Submission)



รูปที่ 2.5 กระบวนการจัดตารางงานบนระบบประมวลผลแบบกลุ่มเมฆ

ขั้นตอนของกระบวนการจัดตารางงานสามารถอธิบายได้ดังรูปที่ 2.5

1) ขั้นตอนการค้นหาและคัดกรองทรัพยากร (Resource Discovering And Filtering)

Datacenter Broker จะทำการค้นหาทรัพยากรทั้งหมดที่อยู่ในระบบ และเก็บข้อมูลรวมถึงสถานะต่างๆที่เกี่ยวข้องเพื่อนำไปใช้งาน

2) ขั้นตอนการเลือกทรัพยากร (Resource Selection)

ในการเลือกทรัพยากรเพื่อนำไปใช้ในการประมวลผล จะถูกเลือกโดยขึ้นอยู่กับพารามิเตอร์ต่างๆของงานที่เข้ามาในระบบและทรัพยากรที่มีอยู่ โดยขั้นตอนนี้จะเป็นขั้นตอนการตัดสินใจในการจัดการเลือก VM ให้กับงาน

3) ขั้นตอนการส่งมอบงาน (Task Submission)

งานที่เข้ามาในระบบทั้งหมดจะถูก Datacenter Broker ส่งให้กับ VM เพื่อไปประมวลผลต่อไป

2.4 วิธีการเมตาฮิวริสติก

ปัญหาทั่วไปในทางการวิจัยและดำเนินงานนั้นมีมากมาย ซึ่งปัญหาที่เป็นปัญหามาตรฐานนั้นจะใช้ในการพัฒนาหาวิธีการหาคำตอบใหม่ๆซึ่งมีประสิทธิภาพสูงขึ้น ในขณะที่เวลาที่ใช้ในการหาคำตอบลดลงเรื่อยๆ

แต่เมื่อปัญหามีขนาดใหญ่ขึ้นจะมีจำนวนคำตอบที่เป็นไปได้มากขึ้น ซึ่งการคำนวณหาคำตอบที่ดีที่สุดจะเสียเวลามาก แต่หากใช้วิธีการทางฮิวริสติก (Heuristic) อาจจะใช้เวลาเพียงไม่กี่นาทีเท่านั้น แม้ว่าคำตอบที่ได้อาจจะไม่ใช่คำตอบที่ดีที่สุด แต่คุณภาพของคำตอบก็ดีเพียงพอต่อความต้องการ ซึ่งเป็นข้อดีของการใช้วิธีการฮิวริสติกในการแก้ปัญหาแทนวิธีการที่ได้คำตอบที่ดีที่สุด

วิธีการฮิวริสติกเป็นระเบียบวิธีแบบอิสระที่สามารถสร้างวิธีหรือขั้นตอนใดๆก็ได้ โดยการออกแบบวิธีการทางฮิวริสติกต้องออกแบบให้ใช้งานง่ายและมีประสิทธิภาพใช้งานได้จริง วิธีการสร้างฮิวริสติกอีกประเภทหนึ่งเรียกว่า เมตาฮิวริสติก (Meta-Heuristic) ซึ่งหมายถึง ชุดของลำดับขั้นการแก้ปัญหาหรืออัลกอริทึม (Algorithm) แบบฮิวริสติกชนิดหนึ่งที่สามารถนำหลักการเดียวกันไปใช้แก้ปัญหาได้หลากหลายปัญหา ซึ่งในปัจจุบัน วิธีการออกแบบฮิวริสติกโดยอาศัยหลักการทางเมตาฮิวริสติกนี้ได้รับความนิยมเป็นอย่างสูงเนื่องจากคำตอบที่ได้จากวิธีการนี้ให้ผลที่ดี แก้ปัญหาได้รวดเร็วและนำไปใช้งานได้ง่าย

Blum และ Roli [19] ได้กล่าวถึงหลักการเบื้องต้นของเมตาฮิวริสติกดังนี้

1) เมตาฮิวริสติกมีระเบียบวิธีในการค้นหาคำตอบที่ดีภายในพื้นที่ของคำตอบที่เป็นไปได้ (Feasible Region)
2) เมตาฮิวริสติกมีจุดประสงค์เพื่อหาคำตอบที่ดีที่สุดหรือคำตอบที่ใกล้เคียงคำตอบที่ดีที่สุดภายในระยะเวลาอันสั้น
3) เมตาฮิวริสติกเป็นขั้นตอนการประมาณคำตอบ
4) เมตาฮิวริสติกอาจจะเกิดจากการรวมหลากหลายเทคนิค เพื่อค้นหาคำตอบที่ดีที่สุดภายในพื้นที่คำตอบที่เป็นไปได้

ในปัจจุบัน วิธีการทางด้านเมตาฮิวริสติกมีอยู่หลากหลายให้เลือกใช้ ซึ่งแต่ละวิธีการจะมีจุดดีและจุดด้อยที่แตกต่างกัน บางวิธีการให้ผลดีแต่ใช้เวลาในการคำนวณ บางวิธีการทำงานได้รวดเร็วแต่ให้ผลที่แย่กว่าวิธีการอื่น

ตัวอย่างของวิธีการเมตาฮิวริสติกที่เกิดจากแรงบันดาลใจตามธรรมชาติ ได้แก่ เจเนติกอัลกอริทึม (Genetic Algorithm) และวิธีระบบอาณานิคมมด (Ant Colony Optimization) จะถูกกล่าวถึงในส่วนถัดไป

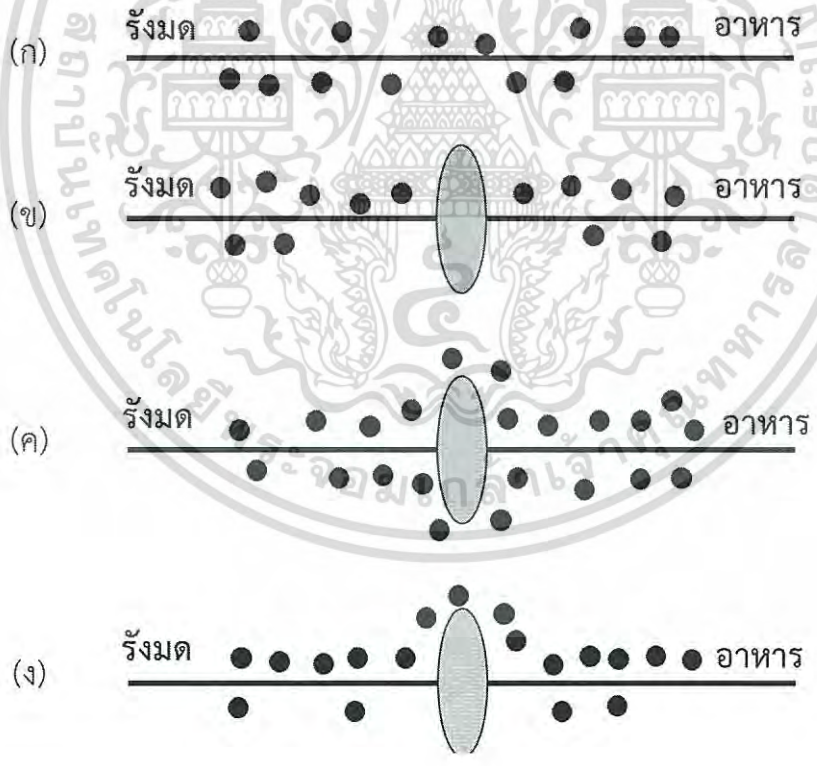
2.4.1 วิธีระบบอาณานิคมมด

มาร์โค โดริโก (Marco Dorigo) และคณะ ได้เสนอแนวคิดวิธีการหาคำตอบโดยใช้ระบบอาณานิคมมด (Ant Colony Optimization: ACO) [20] ซึ่งเป็นอัลกอริทึมด้านวิวัฒนาการ (Evolutionary Algorithm) โดยมีขั้นตอนในการทำงานหลักคือการสร้างกลุ่มของตัวค้นหาผลลัพธ์ (Population of Solution) แล้วใช้หลักเกณฑ์ที่ต่างกันในการค้นหาแบบสโทแคสติก (Stochastic Search) ที่ทำการลอกเลียนแบบการพัฒนาตัวเองตามธรรมชาติ

โดยระบบอาณานิคมมด จะเลียนแบบพฤติกรรมการหาอาหารของมดโดยการค้นหาเส้นทางที่สั้นที่สุดระหว่างรังกับแหล่งอาหาร โดยมดจะเดินทางจากรังไปสู่แหล่งอาหารและกลับมาที่รังอีกครั้งหลักจากได้อาหาร

แล้ว ในระหว่างเดินทาง มดจะปล่อยสารเคมีที่ชื่อว่าฟีโรโมน (Pheromone) เพื่อให้มดตัวอื่นๆเดินทางตามกลิ่นหรือร่องรอยของฟีโรโมนนั้นๆ เมื่อระยะเวลาผ่านไป ฟีโรโมนนี้จะสามารถระเหยไปได้ตามคุณสมบัติทางเคมี ดังนั้นระยะทางที่ยาวจนเกินไปจะทำให้ฟีโรโมนระเหยหมดระหว่างเดินทาง ในขณะเดียวกันหากระยะทางสั้นจะทำให้มีโอกาสเพิ่มฟีโรโมนในขณะเดินทางไปและกลับได้ก่อนที่ฟีโรโมนจะระเหย เมื่อมีการเดินทางของมดหลายๆตัวหรือมีการเดินทางหลายๆรอบ จะทำให้ฟีโรโมนเพิ่มความเข้มข้นขึ้นเรื่อยๆ ในทางตรงกันข้ามเส้นทางเดินที่มีระยะยาวกว่า จะมีแรงดึงดูดหรือระดับความเข้มข้นของฟีโรโมนน้อยกว่าเนื่องจากมีการระเหยมากกว่า และหากมีมดเดินทางผ่านเป็นจำนวนน้อยจะทำให้ฟีโรโมนมีความเจือจางและไม่ดึงดูดมดตัวอื่นๆในที่สุด

การทำงานของระบบอาณานิคมมดจะแสดงให้เห็นดังรูปที่ 2.6 โดย (ก) มดจะเริ่มต้นเดินจากรังไปยังแหล่งอาหารพร้อมกับปล่อยฟีโรโมนไว้ระหว่างทาง และเมื่อมีสิ่งกีดขวางดัง (ข) มดจะทำการสุ่มเส้นทางไปทั้งด้านบนและด้านล่างพร้อมกับปล่อยฟีโรโมนไว้ในระหว่างทางเช่นเดียวกัน ในรูป (ค) เมื่อเวลาผ่านไปมดจะทำการเลือกเส้นทางตามความเข้มข้นของฟีโรโมนที่มดตัวอื่นได้ปล่อยไว้ และ (ง) คือผลลัพธ์ที่ได้โดยมดจะทำการเลือกเดินทางตามเส้นทางนี้ตลอดซึ่งจะเป็นระยะทางที่สั้นที่สุดในการเดินทางจากรังไปยังแหล่งอาหาร



รูปที่ 2.6 จำลองการทำงานของระบบอาณานิคมมด

ตัวอย่างของปัญหาที่นำระบบอาณานิคมมดมาใช้เพื่อหาคำตอบ เช่น การหาเส้นทางที่สั้นที่สุดของเซลส์แมน (Travelling Salesman Problem: TSP) การจัดตารางการผลิตแบบตามสั่ง (Job-Shop Scheduling) เป็นต้น

อัลกอริทึมของวิธีระบบอาณานิคมมดสามารถเขียนเป็นรหัสเทียม (Pseudo Code) ได้ดังนี้

ตั้งค่าเริ่มต้นที่จำเป็น
เมื่อวนซ้ำยังไม่ครบจำนวนรอบที่กำหนดหรือเงื่อนไขอื่นๆที่ส่งผลให้หยุดการวนซ้ำยังไม่ครบกำหนด ดำเนินการดังนี้
    เมื่อจำนวนมดยังไม่ครบตามจำนวนที่ตั้งค่าไว้
        สร้างคำตอบเริ่มต้น
        ปรับปรุงคำตอบ
    สิ้นสุดการสร้างคำตอบจากมดแต่ละตัว
    ปรับปรุงค่าฟีโรโมน
สิ้นสุดการวนซ้ำ

อัลกอริทึมการทำงานของระบบอาณาจักรมด เริ่มแรกมดจะทำการสุ่มประชากรจำนวนหนึ่ง โดยที่มดแต่ละตัวจะแทนผลลัพธ์ มดจะทำการเดินทางจะโหนดหนึ่งไปยังอีกโหนดหนึ่งหรือเรียกว่าการเปลี่ยนสถานะ ซึ่งมีสองทางเลือกคือ การเลือกสำรวจเพื่อหาผลเฉลย (Exploration) หรือการเลือกผลเฉลยโดยอาศัยความรู้เดิมที่มีอยู่ก่อนแล้ว (Exploitation) มดตัวที่ k ที่อยู่ที่โหนด i จะทำการเลือกโหนดถัดไปที่จะเดินทางด้วยฟังก์ชันความน่าจะเป็นดังสมการที่ 1

$$P_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}]^{\beta}}{\sum_{l\in N_i^k}[\tau_{il}(t)]^{\alpha}\cdot[\eta_{il}]^{\beta}} & if\ \ j\in N_i^k \\ 0, & otherwise\end{cases} \qquad \text{---------- (1)}$$

$\tau_{ij}(t)$ คือ ค่าฟีโรโมนระหว่างโหนด i และโหนด j ในเวลา t

$\eta = 1/d_{ij}$ คือ ค่าฮิวริสติกในเวลา t โดย d_{ij} คือค่าของผลลัพธ์ที่ต้องการ

โดยค่า $[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}$ คือค่าความน่าสนใจ (Attractiveness) ของการเดินทางจากโหนด *i* ไปยังโหนด *j* ซึ่งค่าความสนใจจะมี 2 พารามิเตอร์ย่อย คือ นอกจากนี้ค่าแอลฟา α และบีตา β ซึ่งเป็นค่าคงที่ค่าหนึ่งที่ระบุถึงการให้น้ำหนัก (Weight) กับค่าฟีโรโมนกับระยะทางระหว่างการเชื่อมของโหนดสองโหนด โดยแอลฟาจะเป็นน้ำหนักที่ให้กับฟีโรโมน และบีตาจะเป็นน้ำหนักที่ให้กับค่าฮิวริสติกระหว่างสองโหนด

ในการอัพเดทค่าฟีโรโมน (Pheromone updating) จะทำหลังจากเมื่อมดทำการสร้างเส้นทางเรียบร้อยแล้วเพื่อหลีกเลี่ยงการสะสมของฟีโรโมนที่มีมากเกินไป โดยมดทุกตัวจะมีสิทธิ์ปล่อยฟีโรโมน แต่การปล่อยจะมีปริมาณที่ไม่เท่ากัน โดยสมการที่ใช้ในการปรับค่าฟีโรโมนจะแสดงให้ดังสมการที่ 2

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \qquad \text{---------- (2)}$$

ρ คือ อัตราการระเหยของฟีโรโมน (Evaporation rate) มีค่าระหว่าง $0 \le \rho \le 1$

$\Delta\tau_{ij}$ คือ ปริมาณของฟีโรโมนที่มดทุกตัวจะเพิ่มให้กับเส้นทางที่ได้เดินผ่านมาแล้ว สามารถคำนวณได้จากสมการที่ 3

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \qquad \text{---------- (3)}$$

และ $\Delta\tau_{ij}^{k}$ คือ ปริมาณของฟีโรโมนที่มดตัวที่ k จะเพิ่มให้กับเส้นทาง ซึ่งสามารถคำนวณได้ดังสมการที่ 4

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L^{k}(t)} & if\ (i,j) \in T^{k}(t) \\ 0 & if\ (i,j) \notin T^{k}(t) \end{cases} \qquad \text{---------- (4)}$$

$T^{k}(t)$ คือ เส้นทางที่สร้างโดยมด k ในเวลา t

$L^{k}(t)$ คือ ผลลัพธ์คาดหวังสำหรับเส้นทางที่สร้างโดยมด k ในเวลา t

Q คือ ค่าคงที่ใดๆ

และมดตัวที่ให้คำตอบที่ดีที่สุดจะทำการเพิ่มปริมาณฟีโรโมนบนเส้นทางอีกครั้ง เรียกว่า การอัพเดทฟีโรโมนแบบโกลบอล (Global pheromone updating) ดังแสดงในสมการที่ 6

$$\tau_{ij}(t) = \tau_{ij}(t) + \frac{Q}{L^{+}} \quad if\ (i,j) \in T^{+} \qquad \text{---------- (6)}$$

L^+ คือ ผลลัพธ์ที่ได้ของเส้นทางที่ดีที่สุด

T^+ คือ เส้นทางที่สร้างมดที่ให้คำตอบที่ดีที่สุด

โดยค่าความน่าจะเป็นนี้ขึ้นอยู่กับค่าของฟังก์ชันเป้าหมายและปริมาณของฟีโรโมน และเมื่อเวลาผ่านไปปริมาณฟีโรโมนจะระเหยเพื่อไม่ให้ลู่เข้าหาคำตอบเร็วเกินไป (Convergence) เมื่อมดทุกตัวเดินทางไปยังโหนดต่อไปจะมีการปรับค่า (Update) ปริมาณของฟีโรโมนสำหรับช่วงเวลาถัดไป เมื่อเวลาผ่านไปกระบวนการ Positive Feedback จะทำให้มดทุกตัวเลือกเดินทางในเส้นที่สั้นที่สุดและได้คำตอบที่ต้องการ

2.4.2 เจเนติกอัลกอริทึม

เจเนติกอัลกอริทึม (Genetic Algorithm) หรือเรียกอย่างย่อว่าจีเอ (GA) เป็นวิธีการเมตาฮิวริสติกที่นิยมนำมาใช้กันอย่างแพร่หลาย ถูกคิดค้นและนำเสนอโดยจอห์น เฮนรี ฮอลแลนด์ (John Henry Holland) ในปีค.ศ. 1970 โดยสามารถนำมาใช้ในการแก้ปัญหาหรือค้นหาคำตอบเพื่อให้ได้วิธีการหรือคำตอบที่ดีที่สุด ซึ่งได้รับแรงบันดาลใจมาจากวิธีการวิวัฒนาการ (Evolution) ของสิ่งมีชีวิตตามทฤษฎีของชาร์ล ดาร์วิน (Charles Darwin) โดยในปีค.ศ. 1859 เขาได้เสนอทฤษฎีการวิวัฒนาการของสิ่งมีชีวิตโดยอธิบายกระบวนการในการคัดเลือกสิ่งมีชีวิตให้อยู่รอดสำหรับรุ่นถัดไป

ชาร์ล ดาร์วินได้กล่าวไว้ว่าแต่ละรุ่นของการวิวัฒนาการนั้นจะมีการคัดเลือกผู้ที่มีความเหมาะสมที่จะอยู่รอด (Survival of The Fittest) เพื่อทำการผสมพันธุ์และสร้างประชากรรุ่นใหม่ที่ดีกว่าเดิมขึ้นมา

ต่อมาเมนเดล (Mendel) ได้เสนอการทดลองการข้ามผสมพันธุ์ในปี ค.ศ. 1901 เป็นการอธิบายเพิ่มเติมสำหรับทฤษฎีของดาร์วิน โดยสรุปว่าในการวิวัฒนาการของสิ่งมีชีวิตตามธรรมชาตินั้นเป็นไปเพื่อปรับปรุงให้เป็นสิ่งมีชีวิตที่ดีขึ้นกว่าเดิมตามธรรมชาติ โดยมีลักษณะของพันธุกรรมที่ถ่ายทอดต่อกันมาจากรุ่นสู่รุ่นในลักษณะของยีน (Gene) โดยยีนจะอยู่ในโครโมโซม (Chromosome) ของสิ่งมีชีวิต ดังนั้นโดยส่วนใหญ่แล้วในขั้นตอนของการวิวัฒนาการนั้น ลูกมักจะมีคุณสมบัติที่ดีกว่าพ่อแม่หรือบรรพบุรุษอยู่เสมอ จึงได้สิ่งมีชีวิตที่มีวิวัฒนาการที่ดีขึ้นเรื่อยๆ

ดังนั้นเจเนติกอัลกอริทึมเป็นแนวคิดในการแก้ปัญหาและค้นหาคำตอบด้วยการเลียนแบบกระบวนการทางธรรมชาติ ซึ่งเป็นการผสมผสานกันระหว่างการค้นหาแบบสุ่มและการเปรียบเทียบคำตอบที่ได้คล้ายการค้นหาแบบ Hill Climbing Technique แต่ต่างกันที่เจเนติกอัลกอริทึมใช้การผสมชุดคำตอบที่มีความเหมาะสมและคัดเลือกคำตอบที่ดีกว่าเพื่อสร้างชุดคำตอบที่ดียิ่งขึ้น

องค์ประกอบหลักของเจเนติกอัลกอริทึม มีดังนี้

1) การออกแบบโครโมโซมแทนคำตอบ (Chromosome Encoding)

2) ประชากรเริ่มต้น (Initial Population)

3) ฟังก์ชันวัตถุประสงค์ (Fitness Function)

4) วิธีการถ่ายทอดทางพันธุกรรม (Genetic Operator)

โดยรายละเอียดในแต่ละองค์ประกอบจะอธิบายในส่วนถัดไป

1) การออกแบบโครโมโซมแทนคำตอบ (Chromosome Encoding)

เป็นขั้นตอนการออกแบบโครโมโซมเพื่อให้สอดคล้องกับปัญหาจริงที่ต้องการพัฒนาวิธีการแก้ปัญหาด้วยเจเนติกอัลกอริทึม วิธีการออกแบบโครโมโซมเพื่อแทนคำตอบมีหลายวิธี ดังรายละเอียดต่อไปนี้

1.1) การออกแบบโครโมโซมแบบไบนารี (Binary Encoding) เป็นการออกแบบโครโมโซมที่แทนที่ด้วย 0 หรือ 1 เท่านั้น

1.2) การออกแบบโครโมโซมแบบลำดับ (Permutation Encoding) เป็นการออกแบบโครโมโซมที่ใช้เป็นตัวเลขทั่วไปได้ เช่น ตัวเลขที่บอกถึงลำดับขั้นตอนในการทำงานหรือเดินทาง

1.3) การออกแบบโครโมโซมแบบใช้ค่าหรือเครื่องหมายจริง (Value Encoding) เป็นการออกแบบโครโมโซมที่ใช้เลขจำนวนจริงหรือใช้อักษรที่เป็นตัวแทนของคำตอบจริงมาใช้ในการแทนค่าในโครโมโซม

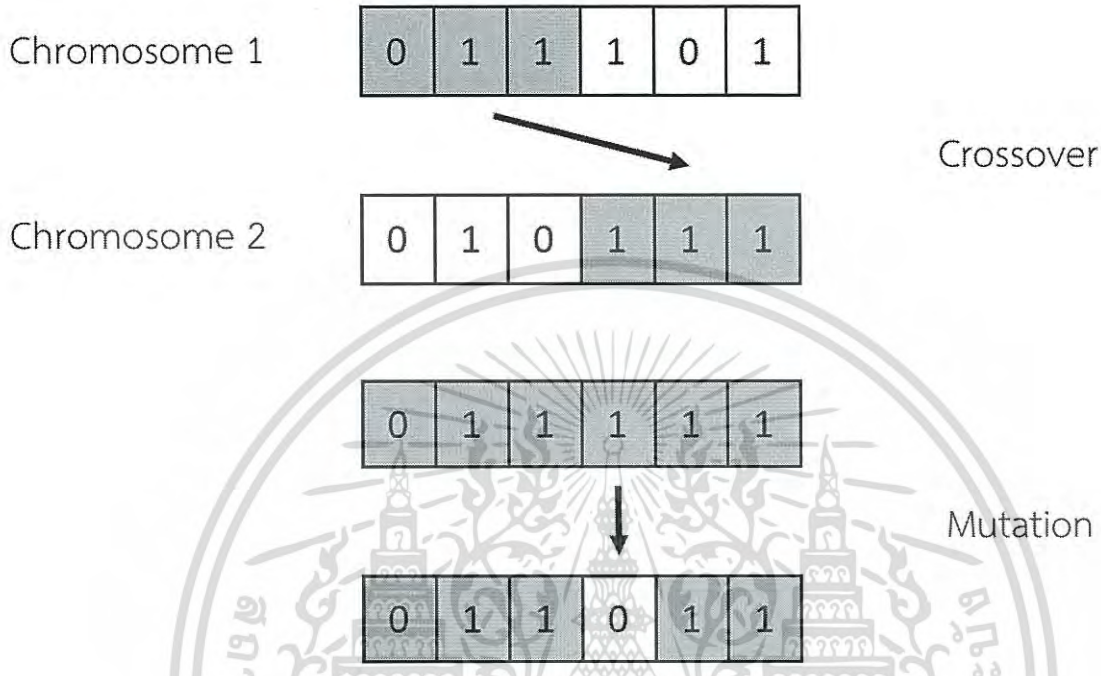
2) การสร้างประชากรเริ่มต้น (Initial Population)

สร้างประชากรขึ้นมาโดยใช้โครโมโซม (Chromosome) เพื่อใช้แทนประชากรแต่ละตัว (Individual) โดยแต่ละโครโมโซมจะประกอบด้วยยีนต่างๆเพื่อถ่ายทอดคุณสมบัติเฉพาะของตนไปยังประชากรรุ่นถัดไปได้ โดยขั้นตอนนี้จะเรียกว่า Genotype Representation โดยนิยมใช้อาร์เรย์ (Array) ในการแทนโครโมโซม มักกำหนดในรูปแบบของแถวของอักขระ (String of Alphabet) หรือแถวของเลขฐานสอง (Bit String) ตัวอย่างการกำหนดโครโมโซมอย่างง่าย เช่น {100101} โดยตำแหน่งของยีนในแต่ละโครโมโซมจะแทนลักษณะขององค์ประกอบย่อยของชุดคำตอบของปัญหา แต่อย่างไรก็ดี สามารถกำหนดยีนและโครโมโซมในรูปแบบอื่นได้ขึ้นอยู่กับลักษณะของปัญหาและคำตอบที่ต้องการ

3) การกำหนดฟังก์ชันวัตถุประสงค์ (fitness Function)

การวัดความเหมาะสมของแต่ละโครโมโซมจะได้ค่าออกมาค่าหนึ่งที่เรียกว่าค่าฟิตเนส (Fitness Value) เพื่อให้คะแนนคำตอบต่างๆที่เป็นไปได้ เป็นการบ่งบอกว่าโครโมโซมนั้นๆเหมาะสมที่ถูกถ่ายทอดไปยังอีกรุ่นหนึ่งหรือไม่ ซึ่งค่าฟิตเนสนั้นยิ่งมากจะเป็นการบอกว่าโครโมโซมดังกล่าวมีคุณภาพมากกว่า โดยฟังก์ชันที่ใช้ในการหาค่าฟิตเนสจะเรียกว่าฟิตเนสฟังก์ชัน (fitness function) โดยใช้ข้อมูลจากโครโมโซม ได้แก่ ยีนและตำแหน่งของยีนในโครโมโซม นอกจากนี้ยังสามารถคำนวณข้อจำกัด (Constraints) อื่นๆ การกำหนดฟิตเนสฟังก์ชันจะขึ้นอยู่กับคำตอบของปัญหาที่ต้องการหา

ตัวอย่างของการครอสโอเวอร์และมิวเทชันแสดงให้เห็นดังรูป 2.7 โดยจะทำการครอสโอเวอร์แบบการแลกเปลี่ยนยีนข้ามโครโมโซมแบบหนึ่งจุด



รูปที่ 2.7 ตัวอย่างการครอสโอเวอร์และมิวเทชัน

5) การทำซ้ำ (Repetition)

กระบวนการทั้งหมดจะถูกทำซ้ำจนกว่าจะตรงตามเงื่อนไขที่ได้กำหนดไว้ โดยรูปที่ 2.8 จะแสดงแผนผังการทำงานของเจเนติกอัลกอริทึมแบบพื้นฐาน

4) ตัวดำเนินการทางพันธุกรรม (Genetic Operator)

ตัวดำเนินการทางพันธุกรรม คือ ตัวดำเนินการต่างๆเพื่อให้เกิดการถ่ายทอดจากประชากรรุ่นหนึ่งไปสู่อีกรุ่นหนึ่ง ไม่ว่าจะเป็นการคัดเลือกสายพันธุ์ (Selection) การแลกเปลี่ยนยีนข้ามโครโมโซมหรือครอสโอเวอร์ (Crossover) การปรับเปลี่ยนยีนภายในโครโมโซมหรือมิวเทชั่น (Mutation)

นอกจากวิธีการทั้งสามแล้ว การกำหนดค่าพารามิเตอร์ที่เกี่ยวเนื่องกับวิธีการทั้งสามนี้ก็มีผลกับคำตอบเช่นเดียวกัน พารามิเตอร์ที่จำเป็นต่อการดำเนินการต่างๆทางพันธุกรรม เช่น ขนาดของประชากรในแต่ละรุ่น ความน่าจะเป็นในการเลือกใช้วิธีการถ่ายทอดทางพันธุกรรมทั้งสามแบบ เป็นต้น รายละเอียดของค่าพารามิเตอร์ต่างๆสามารถอธิบายได้ตามลำดับต่อไปนี้

การคัดเลือกประชากรที่เหมาะสมที่จะนำไปเป็นประชากรรุ่นถัดไป จะคัดเลือกมาเพียงส่วนหนึ่งเท่านั้น วิธีการทั่วไปที่ใช้สำหรับการคัดเลือกประชากร เช่น การคัดเลือกแบบ Roulette Wheel คือการสุ่มเลือกด้วยความน่าจะเป็นในการถูกคัดเลือกตามสัดส่วนของคะแนนความเหมาะสมของประชากรจากคะแนนรวมทั้งหมด เป็นต้น

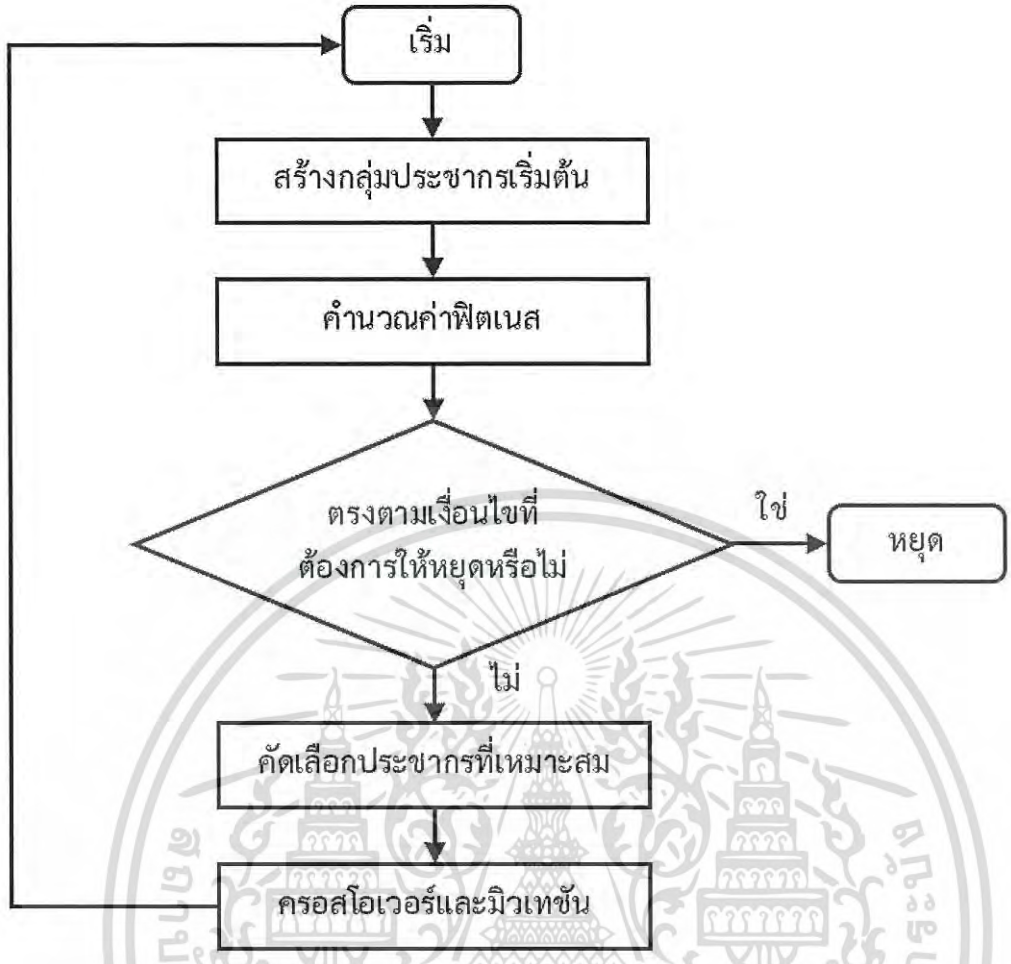
จำนวนประชากรที่เหลือจะถูกสร้างขึ้นมาด้วยการครอสโอเวอร์และมิวเทชั่นเพื่อเป็นการสร้างประชากรรูปแบบใหม่ขึ้นมาและอาจมีโอกาสที่ได้ประชากรที่มีคุณภาพมากกว่าเดิม ซึ่งวิธีการทั้งสองแบบมีกระบวนการทำงานดังนี้

1) ครอสโอเวอร์ (Crossover)

การครอสโอเวอร์คือการที่สองโครโมโซมแลกยีนกันและได้โครโมโซมใหม่ขึ้นมา หรือโครโมโซมรุ่นลูก (Offspring) ซึ่งจะมีส่วนที่เหมือนกับโครโมโซมรุ่นพ่อแม่ (Parent) โดยวิธีการในการแลกยีนกันก็มีหลากหลายรูปแบบ ตัวอย่างเช่น การแลกเปลี่ยนยีนข้ามโครโมโซมแบบหนึ่งจุด (Single Point Crossover) การแลกเปลี่ยนยีนข้ามโครโมโซมแบบสองจุด (Two Points Crossover) การแลกเปลี่ยนยีนข้ามโครโมโซมแบบลำดับ (Order Crossover) การแลกเปลี่ยนยีนข้ามโครโมโซมแบบยึดตำแหน่งเป็นหลัก (Position Based Crossover) การแลกเปลี่ยนยีนข้ามโครโมโซมแบบวนรอบ (Cycle Crossover) การแลกเปลี่ยนยีนข้ามโครโมโซมแบบการจับคู่ปรับบางส่วน (Partial Mapped Crossover) เป็นต้น

2) มิวเทชัน (Mutation)

การมิวเทชันคือการเปลี่ยนรูปแบบของยีนบางส่วนเพื่อให้เกิดโครโมโซมที่มีลักษณะใหม่ขึ้นมาที่อาจให้คำตอบที่ดีขึ้นกว่าเดิม สามารถทำได้ด้วยการสุ่มเปลี่ยนยีนในโครโมโซมในอัตราความน่าจะเป็นในช่วง 0-100 โดยทั่วไปค่าความน่าจะเป็นของการปรับเปลี่ยนยีนภายในโครโมโซมจะถูกกำนดไว้ให้อยู่ในช่วง 0-1% ของจำนวนตำแหน่งโครโมโซม หากปรับเปลี่ยนยีนภายในโครโมโซมให้เป็น 100% จะทำให้ทุกตำแหน่งในโครโมโซมมีการเปลี่ยนแปลงทั้งหมด ซึ่งปกติแล้วในเจเนติกอัลกอริทึมอาจเกิดกรณีแบบนี้ขึ้นได้แต่ไม่บ่อยนัก เพราะจะทำให้การค้นหาในเจเนติกอัลกอริทึมกลายเป็นการค้นหาแบบสุ่ม (Random Search) ตัวอย่างของวิธีการปรับเปลี่ยนยีนภายในโครโมโซม เช่น การปรับเปลี่ยนภายในโครโมโซมแบบอินเวิร์ส (Inversion Mutation) การปรับเปลี่ยนโครโมโซมด้วยการแทรก (Insertion Mutation)



รูปที่ 2.8 เจเนติกอัลกอริทึมแบบพื้นฐาน

6) ตรวจสอบเงื่อนไขหยุดการทำงาน

เงื่อนไขในการหยุดการทำงานของเจเนติกอัลกอริทึมมีได้หลายรูปแบบ เช่น ทำงานครบตามจำนวนรอบ (รุ่น) ที่ต้องการ หรือคำตอบที่ได้เป็นคำตอบที่น่าพอใจแล้ว

2.6 เซลลูลาร์ออโตมาตา

เซลลูลาร์ออโตมาตา (Cellular Automata: CA) ถูกพัฒนาโดย John Von Neumann และ Stanisław Ulam ในปีค.ศ. 1940 โดยเซลลูลาร์ออโตมาตาเป็นระบบที่มีเวลาแบบไม่ต่อเนื่อง (discrete-time system) ที่สามารถใช้ในการสร้างแบบจำลองปรากฏการณ์หลายอย่างทั้งในระบบแบบกายภาพ (physical system) และแบบธรรมชาติ (natural system)

โมเดลของเซลลูลาร์ออโตมาตาจะอยู่ในรูปของเซลล์ (Cell) ประกอบกันจำนวนตั้งแต่หนึ่งเซลล์ขึ้นไป ซึ่งมีลักษณะสำคัญสามประการดังนี้

1) กริด (Grid) เซลล์ทุกๆเซลล์ของเซลล์ลาร์ออโตมาตาอาศัยอยู่บนกริด

2) สถานะ (State) แต่ละเซลล์มีสถานะกำหนดอยู่ โดยจำนวนสถานะของเซลล์จะเป็นจำนวนที่มีขอบเขต (finite) และสถานะของเซลล์ที่น้อยที่สุดที่สามารถมีได้คือสองสถานะ ได้แก่ 0 และ 1 ซึ่งจะกล่าวในส่วนถัดไป

3) เพื่อนบ้าน (Neighborhood) แต่ละเซลล์จะมีเพื่อนบ้านซึ่งก็คือเซลล์ที่อยู่ข้างเคียง

ในส่วนถัดไปจะกล่าวถึงรูปแบบของเซลลูลาร์ออโตมาตาที่เป็นรูปแบบพื้นฐานและเรียบง่ายที่สุด นั่นคือเซลลูลาร์ออโตมาตาแบบพื้นฐาน (Elementary Cellular Automata)

2.6.1 เซลลูลาร์ออโตมาตาแบบพื้นฐาน (Elementary Cellular Automata)

เซลลูลาร์ออโตมาตาแบบพื้นฐาน หรือเรียกอีกชื่อหนึ่งว่า เซลลูลาร์ออโตมาตาแบบไบนารีหนึ่งมิติ (One-Dimensional Binary Cellular Automata) ซึ่งเป็นเซลลูลาร์ออโตมาตาที่มีรูปแบบอย่างง่ายที่สุด

ส่วนประกอบสำคัญหลักสามประการของเซลลูลาร์ออโตมาตาแบบพื้นฐานมีดังนี้

1) กริด (Grid)

โดยกริดในรูปแบบพื้นฐานที่สุดคือแบบหนึ่งมิติ (One-Dimensional Gird) ดังรูป 2.9 และหนึ่งช่องของกริดจะเรียกว่า เซลลูลาร์ออโตเมตอน (Cellular Automaton) หรือเซลล์ (Cell)

รูปที่ 2.9 ตัวอย่างของกริดในเซลลูลาร์ออโตมาตา

2) สถานะ (State)

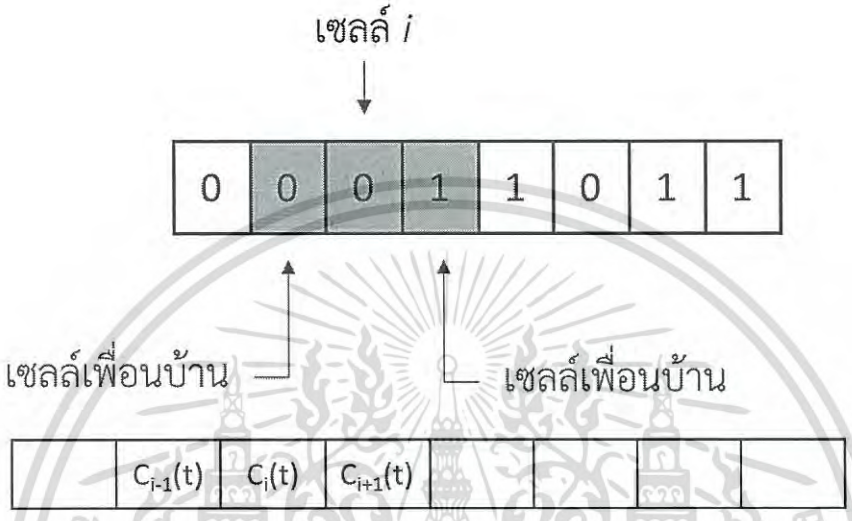
จำนวนสถานะของเซลล์ในรูปแบบที่ง่ายที่สุดคือสองสถานะ ได้แก่ 0 หรือ 1 ดังรูป 2.10

0	0	0	1	1	0	1	1

รูปที่ 2.10 ตัวอย่างของสถานะในเซลลูลาร์ออโตมาตา

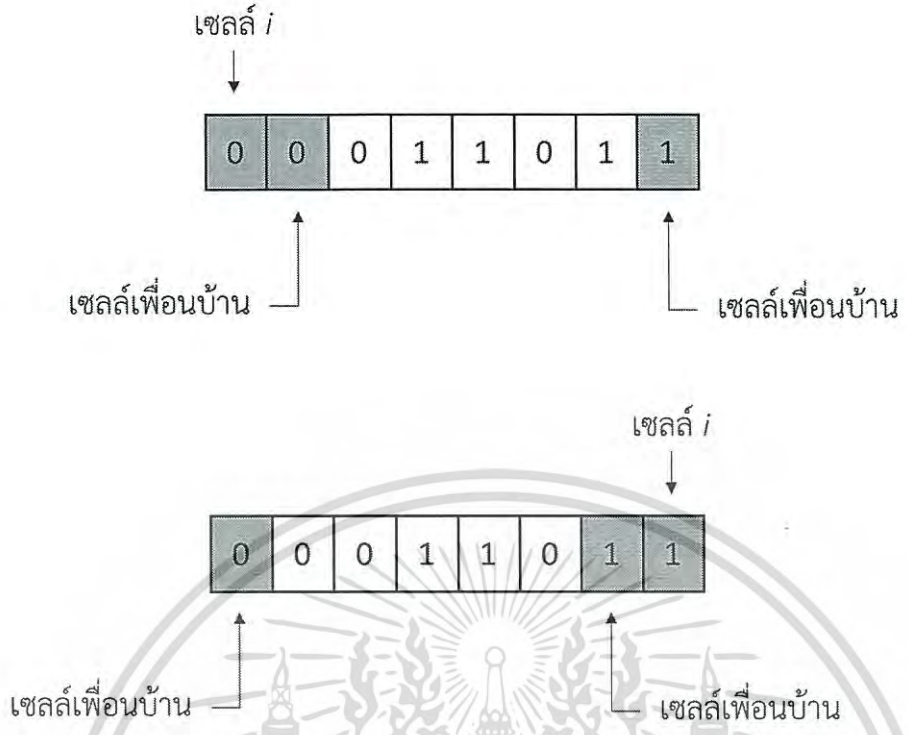
3) เพื่อนบ้าน (Neighborhood)

เพื่อนบ้านของเซลล์คือเซลล์ข้างเคียงที่อยู่ติดกันกับเซลล์ที่สนใจ กำหนดให้สถานะของเซลล์ i ณ เวลา t คือเซลล์ $C_i(t)$ ดังนั้นสถานะของเพื่อนบ้านของเซลล์ i คือ $C_{i-1}(t)$ และ $C_{i+1}(t)$ โดยรูป 2.11 แสดงให้เห็นตัวอย่างของเพื่อนบ้านของเซลล์ที่สาม ซึ่งจะมีเซลล์เพื่อนบ้านอยู่ติดกันทั้งหมดสองเซลล์



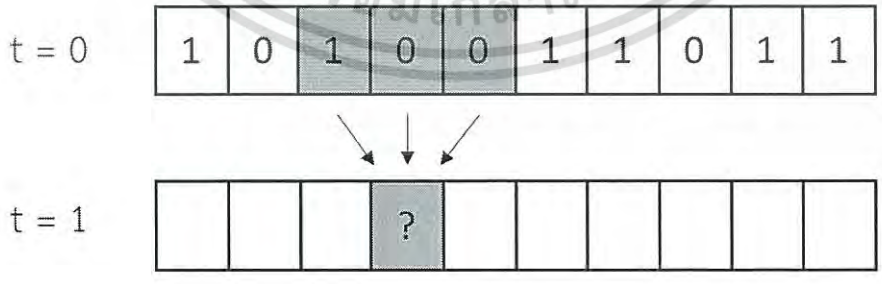
รูปที่ 2.11 ตัวอย่างของเพื่อนบ้านของเซลล์ในเซลลูลาร์ออโตมาตา

ในกรณีที่เซลล์ที่สนใจเป็นเซลล์แรก เซลล์เพื่อนบ้านคือเซลล์ทางด้านขวาและเซลล์สุดท้าย ในทางตรงกันข้ามสำหรับเซลล์สุดท้าย เซลล์เพื่อนบ้านคือเซลล์ทางด้านซ้ายและเซลล์แรก นั่นหมายความว่าจะมองการจัดเรียงของเซลลูลาร์ออโตมาตาในรูปแบบของวงกลม (Circular) ดังรูป 2.12



รูปที่ 2.12 ตัวอย่างของเพื่อนบ้านของเซลล์แรกและเซลล์สุดท้าย

จากที่ได้กล่าวมาว่าสถานะที่เป็นไปได้จะมีทั้งหมดสองสถานะ คือ 0 หรือ 1 เซลลูลาร์ออโตมาตาจะมีการวิวัฒนาการตามเวลาที่เปลี่ยนแปลง โดยการวิวัฒนาการคือการเปลี่ยนแปลงสถานะของเซลล์ในเวลา *t+1* โดยสถานะใหม่ของเซลล์ในเวลาที่ *t+1* จะขึ้นอยู่กับกฎที่ถูกเลือกใช้ กฎการเปลี่ยนสถานะของเซลล์คือตารางที่ระบุสถานะถัดไปที่เวลา *t+1* ซึ่งจะพิจารณาจากสถานะปัจจุบันของตัวมันเองและเซลล์เพื่อนบ้านที่เวลา *t* ดังรูป 2.13 เวลาเริ่มต้นคือที่ *t = 0* และดำเนินต่อเนื่องไปเรื่อยๆเป็น *t = 1,2,3,...*



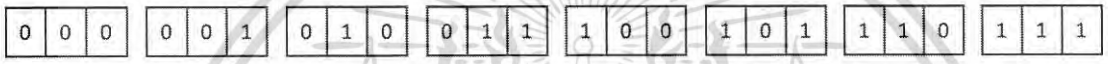
รูปที่ 2.13 วิวัฒนาการของเซลลูลาร์ออโตมาตาแบบพื้นฐาน

ดังนั้นวิวัฒนาการของเซลล์ i ที่มีสถานะคือ $C_i(t)$ ในเวลา $t+1$ จะทำให้เกิดสถานะใหม่ คือ คือ $C_i(t+1)$ สามารถอธิบายได้ดังสมการที่ 7

$$C_i(t+1) = \varphi[C_{i-1}(t), C_i(t), C_{i+1}(t)] \qquad \text{---------- (7)}$$

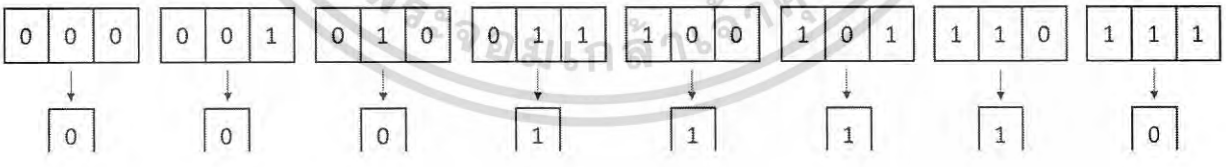
นั่นคือสถานะใหม่ของเซลล์ i จะเป็นสถานะใดขึ้นอยู่กับสถานะปัจจุบันของตัวมันเองและเซลล์เพื่อนบ้านที่อยู่ติดกัน

เนื่องจากเซลล์ที่ถูกนำมาพิจารณาในการเปลี่ยนสถานะของแต่ละเซลล์จะมีทั้งหมดสามเซลล์ แต่ละเซลล์จะมีสถานะที่เป็นไปได้คือ 0 หรือ 1 ทำให้รูปแบบของการจัดเรียงสถานะของทั้งสามเซลล์ที่เป็นไปได้มีทั้งหมดแปดรูปแบบ (2^3) ดังรูป 2.14

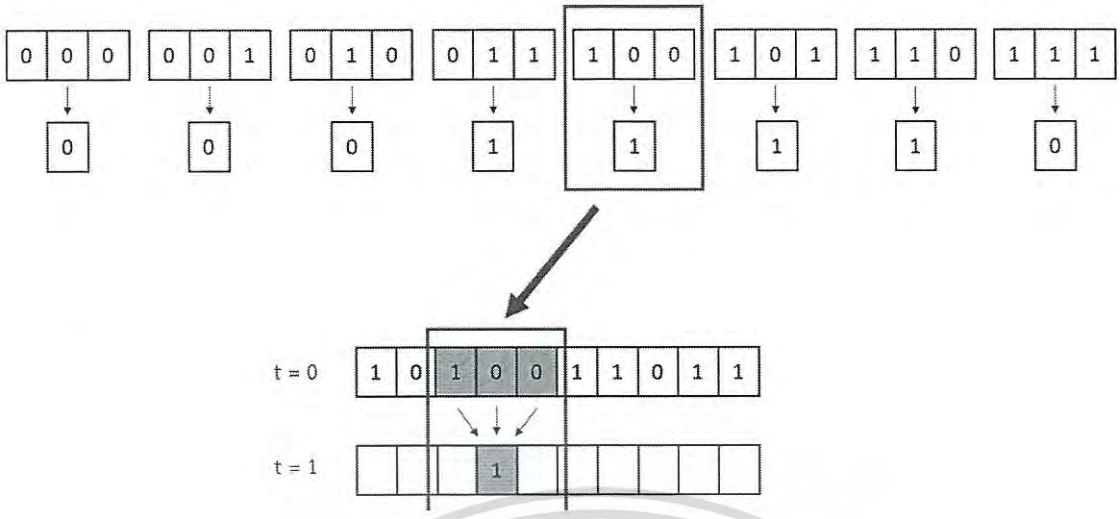


รูปที่ 2.14 รูปแบบการจัดเรียงสถานที่เป็นไปได้ของเซลล์สามเซลล์

เมื่อกำหนดรูปแบบของการจัดเรียงสถานะเป็นไปได้ของเซลล์สามเซลล์ทั้งหมดแล้ว จากนั้นกำหนดผลลัพธ์ (Outcome) ที่ได้จากการพิจารณาทั้งสามเซลล์ซึ่งก็คือสถานะถัดไปของเซลล์ที่อยู่ตรงกลางแต่ละรูปแบบ ดังรูป 2.15 ซึ่งเป็นตัวอย่างของกฎที่นำมาใช้กับเซลลูลาร์ออโตมาตา โดยชื่อของกฎจะเรียกตามสถานะใหม่ที่เกิดจากรูปแบบทั้งหมด ในตัวอย่างเรียกว่า กฎ 30 (00011110_2) และตัวอย่างของการนำกฎ 30 มาใช้งานกับเซลลูลาร์ออโตมาตาจะแสดงให้เห็นดังรูป 2.16

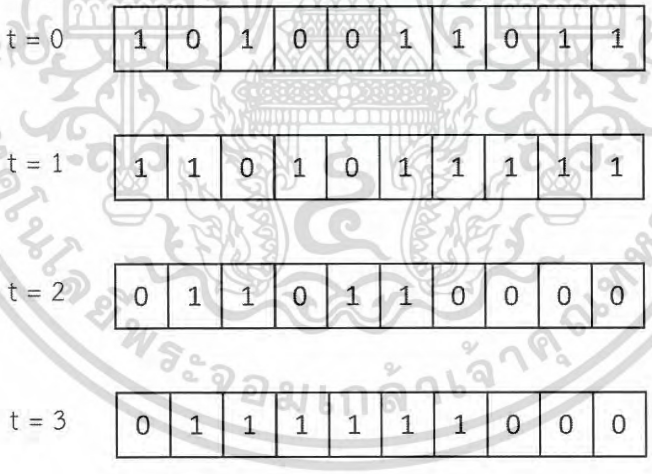


รูปที่ 2.15 กฎ 30



รูปที่ 2.16 ตัวอย่างการเปลี่ยนสถานะของเซลล์โดยใช้กฎ 30

เมื่อนำมากฎ 30 ใช้กับเซลลูลาร์ออโตมาตาในรูป 2.20 ที่เวลา $t = 0$ จะทำให้มีวิวัฒนาการดังรูป 2.17 เมื่อเวลาผ่านไป



รูปที่ 2.17 วิวัฒนาการของเซลลูลาร์ออโตมาตาเมื่อใช้กฎ 30

บทที่ 3
งานวิจัยที่เกี่ยวข้อง

3.1 ทบทวนงานวิจัยที่เกี่ยวข้อง

เนื่องจากระบบการประมวลผลแบบกลุ่มเมฆได้พัฒนามาจากระบบการประมวลผลแบบกระจาย (Distributed Computing) การประมวลผลแบบขนาน (Parallel Computing) และการประมวลผลแบบกริด (Grid Computing) ทำให้สามารถนำวิธีการจัดสรรงานจากระบบอื่นมาประยุกต์ใช้ในระบบการประมวลผลแบบกลุ่มเมฆได้ ซึ่งการจัดตารางงานในระบบการประมวลผลแบบกลุ่มเมฆเป็นสิ่งสำคัญที่ต้องมีการพัฒนาในดียิ่งขึ้นเพื่อลดระยะเวลาในการประมวลผลของงานที่เข้ามาในระบบทั้งหมดและเป็นการเพิ่มประสิทธิภาพในการทำงานให้กับระบบ จึงได้มีการพัฒนาวิธีการต่างๆเพื่อใช้ในการจัดตารางงานขึ้นมาหลายวิธี

ปัญหาการจัดตารางงานเป็นปัญหาแบบ Np-Hard ที่มีความยากในการแก้ปัญหา การที่จะแก้ปัญหาได้อย่างมีประสิทธิภาพนิยมใช้วิธีการเมตาฮิวริสติก (Meta-Heuristic) ซึ่งถูกพัฒนาขึ้นมาเพื่อใช้ในการแก้ปัญหาที่มีความยากและซับซ้อน ช่วยลดระยะเวลาในการคำนวณให้น้อยลงและได้คำตอบที่ดีที่สุด มีผู้คิดค้นวิธีการเมตาฮิวริสติกต่างๆมากมาย เช่น เจเนติกอัลกอริทึม (Genetic Algorithm) ที่อาศัยหลักการจากทฤษฎีวิวัฒนาการจากชีววิทยาและการคัดเลือกตามธรรมชาติ วิธีอาณานิคมมด (Ant Colony Optimization) เป็นการเลียนแบบจากการหาอาหารของมด เป็นต้น

สำหรับเจเนติกอัลกอริทึมได้เป็นที่นิยมและมีผู้นำไปพัฒนากันอย่างหลากหลายเพื่อใช้ในการแก้ปัญหาจัดสรรงาน ผู้เขียนใน [12] ได้นำเจเนติกอัลกอริทึมมาใช้ในระบบมัลติโพรเซสเซอร์เพื่อลดระยะเวลาในการประมวลผลของงานทั้งหมดโดยกำหนดให้โครโมโซมคือการจัดสรรงานให้กับแต่ละโพรเซสเซอร์ และค่าฟิตเนสคือเวลาที่ใช้ในการประมวลผลงานที่เข้ามา

ในการประมวลผลแบบกริดได้มีผู้นำเจเนติกอัลกอริทึมมาใช้เช่นกัน ดังผู้เขียนใน [13] และ [15] รวมถึงในระบบการประมวลผลแบบกระจายในงานวิจัย [16] และ [17] โดยจะมีการทำงานพื้นฐานที่เหมือนกันคือกำหนดให้การจัดสรรงานเป็นโครโมโซมในอัลกอริทึม

และนอกจากนี้เจเนติกอัลกอริทึมยังถูกนำไปใช้ทำงานร่วมกับวิธีการอื่นๆเพื่อเพิ่มประสิทธิภาพการทำงานของอัลกอริทึมให้ดียิ่งขึ้น หนึ่งในวิธีการเหล่านั้นคือเซลลูลาร์ออโตมาตา

ผู้เขียนใน [4] และ [5] ได้นำเสนอการจัดการตารางงานที่ใช้เซลลูลาร์ออโตมาตาทำงานร่วมกับเจเนติกอัลกอริทึมในการหาวิธีการจัดสรรงานให้กับทรัพยากรที่มีอยู่ให้สามารถลดเวลาในการประมวลผลของงานแบบขนานให้เหลือน้อยที่สุดในระบบมัลติโพรเซสเซอร์ที่มีสองโพรเซสเซอร์ (Two-Processor System) เซลลูลาร์ออโตมาตาได้รับความสนใจเนื่องจากพฤติกรรมที่มีความซับซ้อนของมันที่เกิดขึ้นจากการทำงานร่วมกันแบบง่าย และอีกทั้งยังสามารถนำเซลลูลาร์ออโตมาตามาประยุกต์ใช้กับการแก้ปัญหาการจัดตารางงาน

โดยใช้กฎที่เจอ กฎที่มีประสิทธิภาพสำหรับเซลลูลาร์ออโตมาตานั้นสามารถถูกค้นพบได้โดยการใช้เจเนติกอัลกอริทึมในการค้นหา

ในส่วนถัดไปจะอธิบายถึงทฤษฎีและวิธีการต่างๆที่เกี่ยวข้องกับวิธีที่นำเสนอ ได้แก่ การประยุกต์ใช้งานเซลลูลาร์ออโตมาตาสำหรับงานวิจัย [23], [24] การพัฒนาเซลลูลาร์ออโตมาตาในการจัดตารางงานบนระบบมัลติโพรเซสเซอร์ [4], [5] และวิธีที่นำมาเปรียบเทียบเพื่อวัดประสิทธิภาพ ได้แก่วิธีแบบมาก่อนได้ก่อน (First-Come-First-Served: FCFS) วิธีเจเนติกอัลกอริทึม (Genetic Algorithm: GA) และวิธีการใช้ระบบอาณานิคมมดมาจัดสรรงานในระบบการประมวลผลแบบกลุ่มเมฆ (Ant Colony Optimization: ACO) [14]

3.2 การประยุกต์ใช้งานเซลลูลาร์ออโตมาตาสำหรับงานวิจัย

ในการวิจัยนั้นเซลลูลาร์ออโตมาตาสามารถนำไปใช้ประยุกต์ใช้งานได้อย่างกว้างขวางในหลายๆด้าน [23], [24] โดยตัวอย่างของการนำเซลลูลาร์ออโตมาตาไปใช้งานมีดังนี้

3.2.1 เกมเซลลูลาร์ออโตมาตา (CA Games)

เซลลูลาร์ออโตมาได้ถูกนำไปใช้ในการสร้างแบบจำลองสำหรับเกมหลายเกม โดยเกมที่เป็นที่รู้จักกันมากนั่นคือ Game of Life ได้ถูกนำเสนอโดย John H. Conway [25] โดยได้แสดงให้เห็นถึงการที่เซลลูลาร์ออโตมาตาที่มีความเรียบง่ายสามารถสร้างรูปแบบ (Pattern) ที่มีความซับซ้อนได้มาก ซึ่งเกมดังกล่าวเป็นเกมที่ไม่ต้องใช้ผู้เล่น (Zero-Player Game) และการวิวัฒนาการของเกมจะถูกกำหนดด้วยสถานเริ่มต้นของตัวเองและเซลล์เพื่อนบ้าน ซึ่งจะทำให้เกิดรูปแบบที่มีความซับซ้อนเป็นจำนวนมาก

นอกเหนือจาก Game of Life แล้วยังเกมอื่นที่นำเซลลูลาร์ออโตมาตามาประยุกต์ใช้งาน เช่น Firing Squad [26], Queen Bee [27] รวมถึงมีการนำไปใช้ในการจำลองสถานการณ์กับทฤษฎีเกม (Game Theory) ที่มีชื่อเสียง ตัวอย่างการนำไปใช้งานในทฤษฎีเกมคือ เกมความลำบากใจของนักโทษ (Prisoner's Dilemma) [28]

3.2.2 การออกแบบจำลองสำหรับระบบทางกายภาพ (Physical) และชีวภาพ (Biology)

นักฟิสิกส์ได้พัฒนาเซลลูลาร์ออโตมาตาให้เป็นทางเลือกสำหรับสมการเชิงอนุพันธ์ (Differential Equation) ซึ่งเป็นสมการที่แสดงความสัมพันธ์กันระหว่างฟังก์ชั่นของตัวแปรอิสระ ตัวแปรตาม และอนุพันธ์ของตัวแปรตามเทียบกับตัวแปรอิสระนั้นๆสำหรับกฎการสร้างแบบจำลองทางฟิสิกส์ [29] และเซลลูลาร์ออโตมาตายังได้ถูกนำไปใช้กับกระบวนการทางเคมีในอย่างหลากหลาย เช่น การแพร่กระจายระหว่างอะตอมของวัสดุสองชนิด [30] การก่อตัวของโลหะผสมโดยการใช้เซลลูลาร์ออโตมาตาเป็นแบบจำลอง [31]

มีงานวิจัยหลายงานที่นำเซลลูลาร์ออโตมาตาไปประยุกต์ใช้กับงานด้านชีววิทยาและประสบความสำเร็จในการใช้งาน ได้แก่ การทำแบบจำลองสำหรับระบบภูมิคุ้มกัน (Immune System) ถูกคิดค้นโดย Celada, Seiden, และ De Boer [32] การพัฒนายาเพื่อการรักษาการติดเชื้อเอชไอวี (HIV) [33] การพัฒนา

แบบจำลองเซลลูลาร์ออโตมาตาสำหรับการเกิดเนื้องอก [34] การตรวจจับความผิดปกติทางพันธุกรรมของเซลล์มะเร็ง [35] รวมถึงมีงานวิจัยเป็นจำนวนมากที่ได้นำไปใช้กับหัวข้อที่เกี่ยวข้องกับแบบจำลองลำดับของดีเอ็นเอ (Dna)

ในนิเวศวิทยา (Ecology) ได้มีการใช้งานเซลลูลาร์ออโตมาตาเช่นกัน ทั้งการตรวจจับการอพยพของปลาในแม่น้ำตามธรรมชาติ [36] การจำลองการเติบโตของประชากรผัก [37] การจำลองระบบนิเวศวิทยาของผู้ล่าและผู้ถูกล่า [38]

3.2.3 การใช้งานด้านสังคมศาสตร์ (Social Science)

ในด้านสังคมศาสตร์มีการนำเซลลูลาร์ออโตมาตาไปใช้งานเช่นเดียวกัน โดย James M. Sakoda เป็นผู้นำเซลลูลาร์ออโตมาตาไปพัฒนาแบบจำลองด้านสังคมศาสตร์เป็นคนแรกในปี 1971 [39] โดยได้สร้างแบบจำลองกระดานหมากรุก (Checkerboard Model) ซึ่งเป็นการจำลองสถานการณ์โดยใช้คอมพิวเตอร์สำหรับการสร้างปฏิสัมพันธ์ทางสังคมระหว่างสองกลุ่มโดยมีจุดประสงค์คือเพื่อให้เข้าใจการสร้างกลุ่ม (Group Formation)

หลังจากนั้นได้มีงานวิจัยในด้านสังคมศาสตร์อีกหลายงานที่ใช้เซลลูลาร์ออโตมาตาเพื่อใช้อธิบายพฤติกรรมทางสังคม รวมทั้งพฤติกรรมทางด้านเศรษฐศาสตร์ (Economics) เช่น การวิเคราะห์การตั้งราคาโดยใช้เซลลูลาร์ออโตมาตาแบบหนึ่งมิติ [40] การวิเคราะห์พลวัตของความร่วมมือจากเฟรมเวิร์คของเซลลูลาร์ออโตมาตา [41] เป็นต้น

3.2.4 การรู้จำแบบ (Pattern Recognition)

มีการค้นคว้าเกี่ยวกับการจดจำรูปแบบโดยมีพื้นฐานจากไวยากรณ์ (Syntactic Approach) ซึ่งเซลลูลาร์ออโตมาตาแบบจำกัด (Finite Cellular Automata) จะถูกมองให้เป็นตัวยอมรับภาษา (Language Acceptor) สำหรับการพิจารณานั้นจะพิจารณาจากค่าเริ่มต้นของข้อมูลนำเข้าแบบสายอักขระ (String) โดยการยอมรับหรือการปฏิเสธจะถูกกำหนดโดยเซลล์ของเซลลูลาร์ออโตมาตา มีงานวิจัยที่ได้แสดงให้เห็นว่าเซลลูลาร์ออโตมาตาสามารถยอมรับภาษาที่เป็น Context-Free และ Non Context-Free และ Context-Sensitive [42], [43] และมีการศึกษาความสามารถในการจำแนกความหนาแน่นโดยการใช้เซลลูลาร์ออโตมาตา (Classification)

3.2.5 การเข้ารหัสข้อมูล (Crptography)

Wolfram Stephen ได้นำเสนอการนำเซลลูลาร์ออโตมาตามาใช้ร่วมกับการเข้ารหัสในหนังสือ Cellular Automata And Complexity: Collected Papers โดยเซลลูลาร์ออโตมาตาที่ใช้จะเป็นแบบหนึ่งมิติ ซึ่งเป็นรูปแบบที่เรียบง่ายที่สุดไปใช้ในการเข้ารหัสข้อมูลโดยการสร้างคีย์เพื่อเข้ารหัส [24] และมีงานวิจัยที่นำหลักการไปพัฒนาต่อยอด [44] โดยใช้กับการเข้ารหัสแบบบล็อก (Block Cipher) และเข้ารหัสแบบสตรีม

(Stream Cipher) บนพื้นฐานของการเปลี่ยนสถานะของเซลลูลาร์ออโตมาตาแบบหนึ่งมิติที่กฎ 51, 153 และ 195 งานวิจัยด้านการสร้างการเข้ารหัสโดยการใช้คีย์แบบสมมาตร (Symmetric Key Cryptography) บนพื้นฐานของการเข้ารหัสแบบ Vernam [45]

การเปลี่ยนสถานะของเซลลูลาร์ออโตมาตาสามารถให้กำเนิดรูปแบบแบบสุ่มได้เป็นจำนวนมากและมีความซับซ้อน จึงนิยมนำไปใช้ในการสร้างลำดับตัวเลขแบบสุ่มในงานด้านการเข้ารหัสข้อมูล [46] โดยได้มีงานวิจัยที่ได้วิเคราะห์การสร้างลำดับเลขแบบสุ่มจากเซลลูลาร์ออโตมาตาเพื่อวิเคราะห์การกระจายตัวของผลลัพธ์ที่ได้ [47]

3.2.6 การสร้างวงจรรวมขนาดใหญ่มาก (Very Large Scale Integration)

การสร้างวงจรรวมขนาดใหญ่มากหรือเรียกกันว่า Vlsi คือกระบวนการในการสร้างวงจรรวมโดยการนำทรานซิสเตอร์เป็นจำนวนมากให้อยู่ภายใต้ชิปตัวเดียวกัน กลายเป็นแผงวงจรที่มีขนาดใหญ่และทำให้เกิดไมโครโพรเซสเซอร์ (Microprocessor) ตัวแรกของโลก ทำให้คอมพิวเตอร์มีขนาดเล็กลง

เนื่องจากความเรียบง่ายไม่ซับซ้อนของโครงสร้าง Vlsi ทำให้สามารถนำเซลลูลาร์ออโตมาตามาปรับเพื่อใช้งานร่วมกันได้ มีการใช้งานที่หลากหลายที่ได้ถูกนำเสนอขึ้นมา เช่น การออกแบบและทดสอบ Vlsi โดยขึ้นอยู่กับคุณสมบัติทางด้านสถิติของรูปแบบที่ถูกสร้าง Wolfram ได้จัดหมวดหมู่ของเซลลูลาร์ออโตมาตาแบบสามเซลล์เพื่อนบ้านออกเป็นทั้งหมด 4 หมวดหลัก ซึ่งพบว่ากฎของเซลลูลาร์ออโตมาตาแบบ Class 3 จะเหมาะสมในการสร้างรูปแบบแบบสุ่มมากที่สุด [48]

Hortensius ได้นำเสนอ Prpg (Psudo Random Pattern Generator) บนพื้นฐานของเซลลูลาร์ออโตมาตาแบบผสม เพื่อใช้ในสร้างวงจร Vlsi [49] และหลังจากนั้นประสิทธิภาพของ Prpg ได้ถูกนำไปเปรียบเทียบกับงานวิจัยในกับงานเป็นจำนวนมาก

3.2.7 การประมวลผลภาพ (Image Processing)

เซลลูลาร์ออโตมาตาได้ประสบความสำเร็จในการนำไปพัฒนาเพื่อใช้ร่วมกับการประมวลผลภาพ โดยจุดประสงค์หลักคือเพื่อปรับปรุงคุณภาพของภาพที่ได้จากการประมวลผลให้มีคุณภาพที่ดีขึ้น มีงานวิจัยที่มองว่าภาพดิจิตอลนั้นมีลักษณะเป็นสองมิติและมีขนาดเป็น N X N พิกเซล และสามารถนำเซลลูลาร์ออโตมาตาแบบสองมิติไปปรับใช้ได้ [50] โดยผลลัพธ์ที่ได้คือให้คุณภาพของภาพที่ดีขึ้น งานวิจัยใน [51] ได้มีการทดลองโดยค้นหาความเป็นไปได้ในการเทรนเซลลูลาร์ออโตมาตาสำหรับการใช้งานเพื่อประมวลผลภาพในหลายๆงาน

งานส่วนใหญ่ที่ใช้เซลลูลาร์ออโตมาตาจะเกี่ยวข้องกับภาพแบบไบนารี แต่มีงานวิจัยที่จัดการกับความเข้มของสีภาพ มีงานวิจัยที่ประยุกต์ใช้เซลลูลาร์ออโตมาตากับงานประมวลผลภาพประเภทการลดสัญญาณรบกวน (Denoising) และการตรวจหาคุณลักษณะเด่น (Feature Detection) โดยการเพิ่มสถานะของเซลล์ในเซลลูลาร์ออโตมาตาให้เป็น 3 สถานะ

3.2.8 โทรคมนาคม (Telecommunication)

ในด้านของโทรคมนาคมมีหลายส่วนที่นำเซลลูลาร์ออโตมาตาไปใช้งานกับระบบเครือข่าย เช่น ระบบเซนเซอร์แบบไร้สาย (Wireless Sensor Network) โดยการนำเซลลูลาร์ออโตมาตาไปใช้เพื่อสร้างตัวจำลองสถานการณ์สำหรับระบบขนาดใหญ่เพื่อควบคุมโทโปโลยี [52] และมีผู้นำไปพัฒนาต่อสำหรับการทำเราท์ติ้ง (Routing) บนระบบเซนเซอร์แบบไร้สาย [53], [54] งานวิจัยเพื่อการพัฒนาโปรโทคอลแบบกระจายชื่อ DV-CAST บนระบบเครือข่ายแอดฮอค (Ad Hoc Network) [55] การพัฒนาเซลลูลาร์ออโตมาตาสำหรับการจัดการโลเคชั่นบนระบบเครือข่ายมือถือที่มีการใช้งานร่วมกันกับเจเนติกอัลกอริทึมเพื่อค้นหากฎที่ดีที่สุด [56]

3.2.9 ระบบประมวลผล (Computing Systems)

เซลลูลาร์ออโตมาตาได้ถูกนำไปใช้ในระบบประมวลผลแบบขนาน (Parallel Processing) [57], [58] ซึ่งสามารถนำไปใช้ในการจำลองระบบการประมวลผล (Computational Simulation) เนื่องจากความสามารถสร้างความซับซ้อนของระบบโดยเซลลูลาร์ออโตมาตา และสามารถนำไปประยุกต์ใช้เพื่อจำลองในรูปแบบของโลกความจริงได้

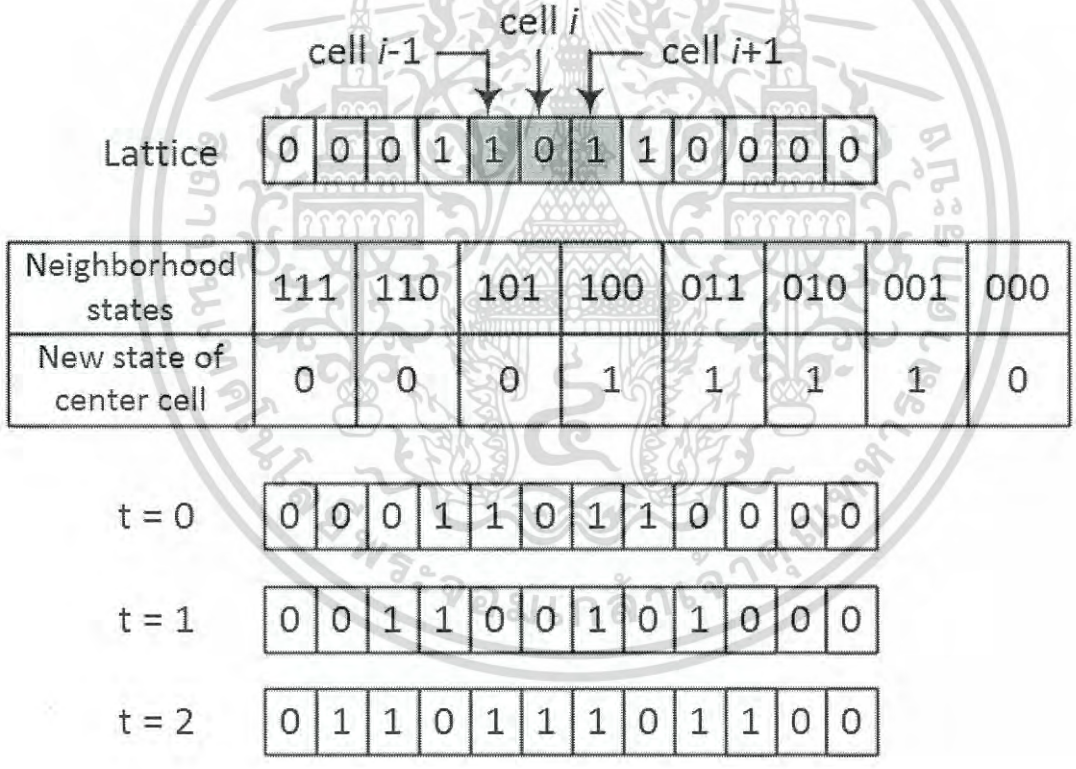
นอกจากนี้ยังการนำไปพัฒนาการบนระบบมัลติโพรเซสเซอร์เพื่อใช้ในการจัดตารางงานโดยจะอธิบายในหัวข้อ 3.3

3.3 การพัฒนาเซลลูลาร์ออโตมาตาในการจัดตารางงานบนระบบมัลติโพรเซสเซอร์

3.3.1 เซลลูลาร์ออโตมาตา

เซลลูลาร์ออโตมาตาที่ถูกนำมาใช้ในงานวิจัย [4] และ [5] คือ เซลลูลาร์ออโตมาตาแบบไบนารีหนึ่งมิติ (One-Dimensional Binary Cellular Automata) ซึ่งจะมีสถานะที่เป็นไปได้ในแต่ละเซลล์ *I* คือ 0 และ 1 (*k* = 2) และมีกฎการเปลี่ยนสถานะ (Transition Rule) สำหรับใช้ในการเปลี่ยนแปลงสถานะของเซลล์ในเซลลูลาร์ออโตมาตา โดยกฎจะกำหนดสถานะใหม่ของแต่ละเซลล์ในเซลลูลาร์ออโตมาตาจากพิจารณาสถานะของเซลล์นั้นๆและเซลล์เพื่อนบ้านอีกสองเซลล์

จากรูปที่ 3.1 ตัวอย่างที่ใช้คือกฎ 30 สามารถเขียนเป็นเลขฐานสองได้ 00011110 แต่ละเซลล์จะมีเซลล์เพื่อนบ้าน ทำให้มีทั้งหมด 8 รูปแบบ (2^3) โดยรูปจะแสดงให้เห็นถึงสถานะที่เป็นไปได้ กฎการเปลี่ยนสถานะ และการพัฒนาของเซลลูลาร์ออโตมาตา



รูปที่ 3.1 ตัวอย่างของกฎ 30 สำหรับเซลลูลาร์ออโตมาตาแบบไบนารีหนึ่งมิติ

3.3.2 การนำเซลลูลาร์ออโตมาตาไปใช้ในการจัดตารางงานในระบบมัลติโพรเซสเซอร์

ผู้เขียนใน [4] และ [5] ได้นำเซลลูลาร์ออโตมาตาไปใช้งานร่วมกับเจเนติกอัลกอริทึมในการจัดตารางงานในระบบมัลติโพรเซสเซอร์เนื่องจากสถานะของเซลล์สามารถแสดงถึงวิธีการจัดสรรงานในแต่ละโพรเซสเซอร์ได้ ซึ่งจำนวนโพรเซสเซอร์ที่ใช้ในการทดลองมีจำนวนสองเครื่อง และใช้เจเนติกอัลกอริทึมในการค้นพบกฎที่มีประสิทธิภาพสำหรับเซลลูลาร์ออโตมาตาเพื่อที่จะได้ลดเวลาที่ใช้ในการประมวลผลของงานทั้งหมดหรือเมคสแปน (Makespan)

โดยวิธีนี้จะแบ่งออกเป็นสองขั้นตอน ขั้นตอนแรก เจเนติกอัลกอริทึมจะถูกใช้ในการค้นหากฎที่เหมาะสมสำหรับเซลลูลาร์ออโตมาตาสำหรับการแก้ปัญหาในการจัดตารางงาน และขั้นตอนถัดมาคือ กฎที่ถูกค้นพบสำหรับเซลลูลาร์ออโตมาตาจะถูกใช้ในการหาวิธีการจัดสรรงานที่เหมาะสมในระบบมัลติโพรเซสเซอร์และให้เวลาที่น้อยลงเมื่อเปรียบเทียบกับวิธีการแบบดั้งเดิมคือตามลำดับก่อนหลัง (First-Come-First-Served: FCFS)

แต่อย่างไรก็ตาม ในระบบการประมวลผลแบบกลุ่มเมฆนั้นมีจำนวนของคอมพิวเตอร์เสมือนเป็นจำนวนมากในระบบ จึงทำให้ปัญหามีความซับซ้อนเพิ่มมากยิ่งขึ้นเนื่องจากความยาวของกฎของเซลลูลาร์ออโตมาตาที่เพิ่มขึ้นตามจำนวนของโพรเซสเซอร์ที่จำนวน k เมื่อพิจารณาปัญหาของการจัดตารางงานในกรณีที่มีจำนวนโพรเซสเซอร์ที่มากกว่าสองโพรเซสเซอร์ ความยาวของกฎจะเพิ่มขึ้นอย่างรวดเร็วจากค่า k ซึ่งความยาวของกฎสามารถคำนวณได้เท่ากับ k^3 ดังที่ได้แสดงไว้ในตารางที่ 3.1 โดยกำหนดให้ค่า k มีค่า จะเห็นว่าเป็นการยากที่เจเนติกอัลกอริทึมจะสามารถค้นหากฎที่เหมาะสมได้ในพื้นที่ (Search Space) ที่มีขนาดใหญ่มาก

k	ความยาวของกฎ
2	8
3	27
4	64
5	125
10	1000
20	8000
30	27000
50	125000

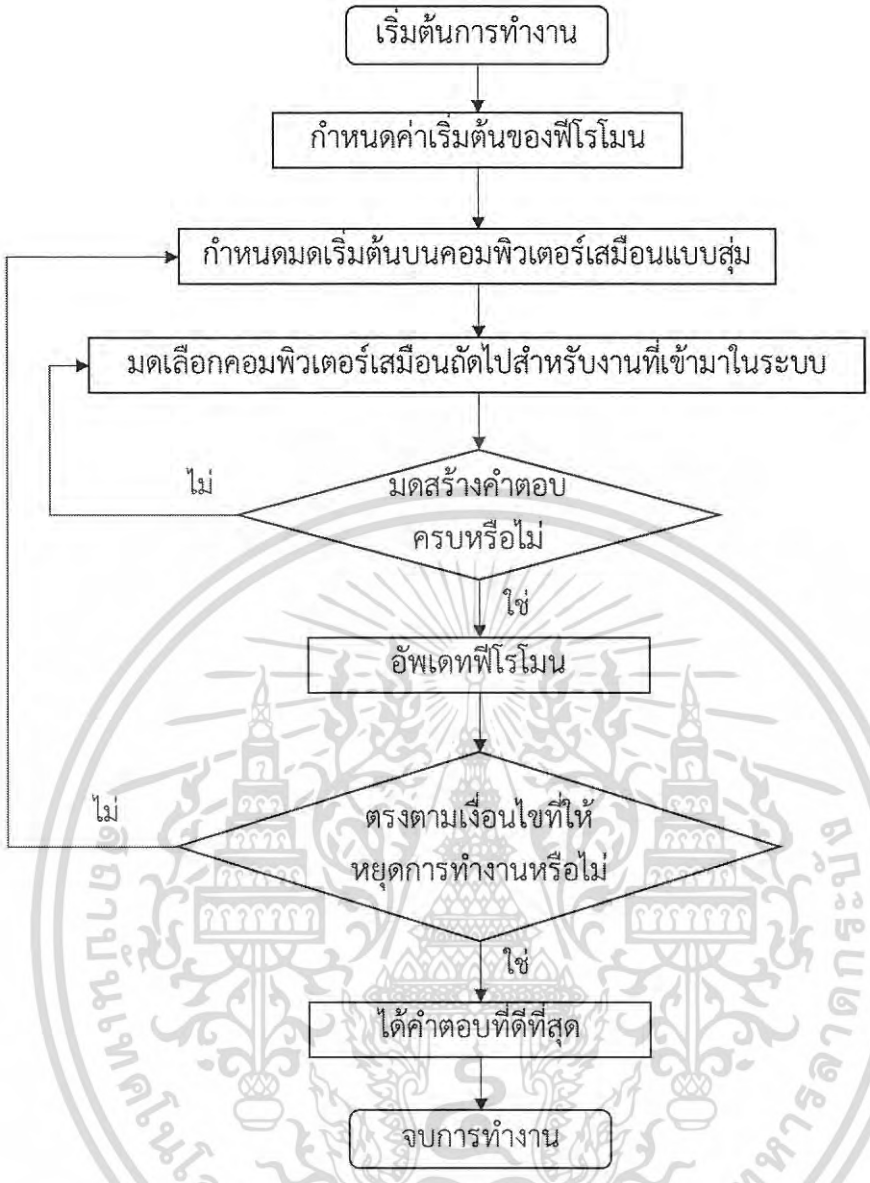
ตารางที่ 3.1 ความยาวของกฎที่ค่า k ต่างๆ

3.4 การจัดสรรงานบนระบบการประมวลผลแบบกลุ่มเมฆโดยใช้ระบบอาณานิคมมด

ผู้เขียนใน [14] ได้นำเสนอวิธีการจัดสรรงานที่เข้ามาในบนระบบการประมวลผลแบบกลุ่มเมฆโดยใช้ระบบอาณานิคมมดมาช่วยค้นหาวิธีการจัดตารางงานที่ดีที่สุด จุดมุ่งหมายของงานวิจัยคือเพื่อต้องการลดระยะเวลาในการประมวลผลของงานทั้งหมดหรือเมคสแปน (Makespan) ให้เหลือน้อยลงที่สุดโดยใช้ระบบอาณานิคมแบบมดมาช่วยในการแก้ปัญหา

ปัญหาของระบบการจัดสรรงานในระบบได้ถูกอธิบายด้วยกราฟ $G=(N,E)$ โดย N คือ จำนวนโหนดหรือคอมพิวเตอร์เสมือน (VM) และงานที่เข้ามาในระบบคือเซตของ E

มดทุกตัวจะเริ่มต้นการทำงานที่คอมพิวเตอร์เสมือนแบบสุ่ม ในระหว่างการทำงานมดจะมีการสร้างคำตอบโดยการย้ายจากคอมพิวเตอร์เสมือนเครื่องหนึ่งไปยังคอมพิวเตอร์และทำซ้ำจนกระทั่งจบการทำงานนั่นคือเมื่อการจัดสรรงานได้เสร็จสมบูรณ์แล้ว โดยเงื่อนไขในการหยุดการทำงานของมดคือเมื่อครบรอบในการวนซ้ำ $t=t\max$ เมื่อ $t\max$ คือจำนวนรอบที่ได้กำหนดไว้ และใช้เวลาในการประมวลผลที่คาดหวัง (Expected Execution Time) เป็นค่าฮิวริสติกของงาน *i* ที่ถูกประมวลผลบนคอมพิวเตอร์เสมือน *j* โดยขั้นตอนการทำงานของอัลกอริทึมจะแสดงตามรูปที่ 3.2



รูปที่ 3.2 แสดงอัลกอริทึมของวิธีการจัดสรรงานโดยใช้ระบบอาณานิคมมด

ในการกำหนดค่าเริ่มต้นของฟีโรโมน $\tau_{ij}(t)$ ให้กับการเส้นทาง (Edge) ระหว่างงาน i และคอมพิวเตอร์เสมือน j จะสมมติให้ค่าเริ่มต้นของฟีโรโมนเท่ากันหมดโดยให้เป็นค่าที่น้อยมากๆที่มากกว่าศูนย์ ระหว่างการทำงานมด k จะทำการเลือกคอมพิวเตอร์เสมือนเครื่องถัดไปสำหรับงานที่เข้ามาทำด้วยความน่าจะเป็นตามสมการที่ 1

$$P_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}]^{\beta}}{\sum_{s\in allowed_k}[\tau_{is}(t)]^{\alpha}\cdot[\eta_{is}]^{\beta}} & if\ \ j\in allowed_k \\ 0, & otherwise\end{cases} \qquad \text{---------- (1)}$$

$\tau_{ij}(t)$ คือ ค่าฟีโรโมนระหว่างงาน i และคอมพิวเตอร์เสมือน j ในเวลา t

$\eta_{ij}=1/d_{ij}$ คือ ค่าฮิวริสติกในเวลา t โดย d_{ij} คือเวลาในการประมวลที่คาดหวังของงาน i บนคอมพิวเตอร์เสมือน j โดยคำนวณได้จากสมการที่ 2

$$d_{ij}=\frac{TL\_Task_i}{Pe\_num * Pe\_mips_j}+\frac{InputFileSize}{VM\_bw_j} \qquad \text{---------- (2)}$$

TL_Task_i คือ ความยาว (Length) ของงาน i ที่ถูกส่งไปยังคอมพิวเตอร์เสมือน j

Pe_num_j คือ จำนวนโพรเซสเซอร์ (Processor) ของคอมพิวเตอร์เสมือน j

Pe_mips_j คือ MIPS (Million Instructions per Second) คอมพิวเตอร์เสมือน j

$InputFileSize$ คือ ขนาดอินพุตของงานก่อนส่งเข้าระบบ

VM_bw_i คือ แบนด์วิธ (Bandwidth) ของคอมพิวเตอร์เสมือน j

α และ β เป็นค่าที่ใช้สำหรับถ่วงน้ำหนัก (Weight) ให้กับฟีโรโมนและค่าฮิวริสติก

ในการอัพเดทค่าฟีโรโมน (Pheromone Updating) จะทำหลังจากเมื่อมดทำการเลือกคอมพิวเตอร์เสมือนครบสำหรับการจัดสรรงานที่เข้ามาในระบบดังสมการที่ 3 เพื่อหลีกเลี่ยงการสะสมของฟีโรโมนบนที่มีมากเกินไป

$$\tau_{ij}(t)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t) \qquad \text{---------- (3)}$$

ρ คือ อัตราการระเหยของฟีโรโมน (Evaporation rate) มีค่าระหว่าง $0\le\rho\le1$

$\Delta\tau_{ij}$ คือ ปริมาณของฟีโรโมนที่มดทุกตัวจะเพิ่มให้กับเส้นทางที่ได้เดินผ่านมาแล้ว โดยที่ปริมาณของฟีโรโมนที่มดตัวที่ k จะเพิ่มให้กับเส้นทาง ซึ่งคำนวณได้จากสมการที่ 4

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k} \qquad \text{---------- (4)}$$

และ $\Delta\tau_{ij}^{k}$ คือ ปริมาณของฟีโรโมนที่มดตัวที่ k จะเพิ่มให้กับเส้นทาง ซึ่งสามารถคำนวณได้ดังสมการที่ 5

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L^{k}(t)} & if\,(i,j) \in T^{k}(t) \\ 0 & if\,(i,j) \notin T^{k}(t) \end{cases} \qquad \text{---------- (5)}$$

$T^{k}(t)$ คือ เส้นทางที่สร้างโดยมด k ในเวลา t

$L^{k}(t)$ คือ คือ เมคสแปนที่คาดหวังสำหรับเส้นทางที่สร้างโดยมด k ในเวลา t

Q คือ Adaptive Parameter

และมดจะทำงานวนซ้ำจนจบการทำงานตามเงื่อนไขที่ได้กำหนดไว้ ผลลัพธ์ที่ได้คือผลเฉลยของอัลกอริทึม นั่นคือวิธีการจัดสรรตารางงานสำหรับงานที่เข้ามาในระบบ

บทที่ 4
กลไกการจัดตารางงานบนพื้นฐานของเซลลูลาร์ออโตมาตาและเจเนติกอัลกอริทึม

ในบทนี้จะกล่าวถึงวิธีการที่ได้นำเสนอ คือ กลไกการจัดตารางงานบนพื้นฐานของเซลลูลาร์ออโตมาตา ซึ่งได้ถูกพัฒนาเพื่อใช้งานในระบบการประมวลผลแบบกลุ่มเมฆ หรือเรียกย่อว่าCATS (CATS) โดยเนื้อหาภายในจะประกอบด้วยขั้นตอนการทำงานต่างๆของวิธีการCATS

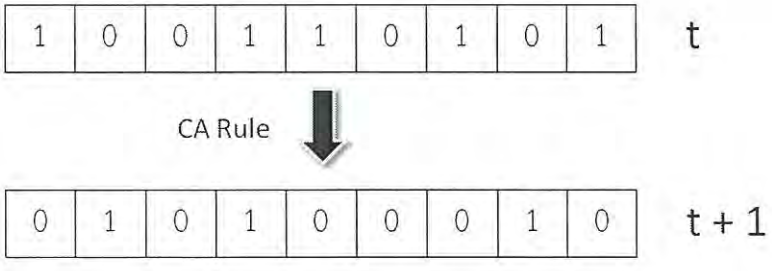
4.1 วิธีการของกลไกการจัดตารางงานบนพื้นฐานของเซลลูลาร์ออโตมาตาสำหรับระบบการประมวลผลแบบกลุ่มเมฆ

เป็นการนำเซลลูลาร์ออโตมาตามาใช้งานร่วมกับเจเนติกอัลกอริทึมสำหรับการค้นหากฎที่มีประสิทธิภาพ เพื่อนำกฎที่หาได้ไปหาวิธีการจัดตารางงานที่สามารถลดเวลาในการประมวลผลหรือเมคสแปน (Makespan) ให้น้อยลง

โดยเซลลูลาร์ออโตมาตาจะถูกนำมาใช้เป็นโมเดลของการจัดตารางงาน และเจเนติกอัลกอริทึมจะทำหน้าที่ในการคัดเลือกเพื่อหากฎ (CA Rule) ที่ดีที่สุดเพื่อนำไปใช้งานต่อไป

4.1.1 การออกแบบโครงสร้างของเซลลูลาร์ออโตมาตา

ในการออกแบบโครงสร้างของเซลลูลาร์ออโตมาตา จะกำหนดให้โครงสร้างเป็นลักษณะแบบไบนารีหนึ่งมิติ (One-Dimensional Binary Cellular Automata) ซึ่งเป็นเซลลูลาร์ออโตมาตาแบบพื้นฐานที่สุด สถานะ (State) ที่เป็นไปได้ของเซลล์จะมีสองสถานะ โดยกำหนดให้เป็น 0 และ 1 และมีการนำกฎมาใช้ในการเปลี่ยนสถานะเมื่อเวลาผ่านไป (t+1) ซึ่งเวลาในการเปลี่ยนสถานะของเซลลูลาร์ออโตมาตานั้นจะเป็นเวลาที่ไม่ต่อเนื่อง (Discrete Time) ดังรูปที่ 4.1



รูปที่ 4.1 เซลลูลาร์ออโตมาตาแบบไบนารีหนึ่งมิติ

กำหนดให้สถานะของเซลล์ถูกแทนด้วยคอมพิวเตอร์เสมือน (Virtual Machine: VM) ในแต่ละเซลล์จะหมายถึงงานที่เข้ามาในระบบ นั่นคือ วิธีการจัดสรรงาน (Task Scheduling) โดยจะถูกแทนด้วยเซลลูลาร์ออโตมาตาตามโครงสร้างที่ได้ออกแบบไว้ ซึ่งความหมายของวิธีการจัดสรรงาน คือ การกำหนดคอมพิวเตอร์เสมือนที่จะนำไปประมวลผลให้งานที่เข้ามาในระบบการประมวลผลแบบกลุ่มเมฆ

เนื่องจากเซลลูลาร์ออโตมาตาแบบไบนารีหนึ่งมิติจะมีสองสถานะคือ 0 และ 1 จึงสามารถกำหนดให้สถานะ 0 คือ คอมพิวเตอร์เสมือนเครื่องที่หนึ่ง (VM 1) และสถานะ 1 คือ คอมพิวเตอร์เสมือนเครื่องที่สอง (VM2) แสดงดังรูป 4.2 ซึ่งจะทำให้เกิดรูปแบบการจัดตารางงานดังรูป 4.3

รูปที่ 4.2 เซลลูลาร์ออโตมาตาแบบไบนารีหนึ่งมิติสำหรับการจัดตารางงานที่ 2 VM

รูปที่ 4.3 การจัดตารางงานตามรูปแบบที่ถูกกำหนดจากเซลลูลาร์ออโตมาตาที่ 2 VM

โดยปกติแล้วจำนวนคอมพิวเตอร์เสมือนที่ถูกใช้งานในระบบจะมีตั้งแต่สองเครื่องขึ้นไป จึงไม่สามารถใช้ 0 และ 1 เป็นค่าในเซลล์ของเซลลูลาร์ออโตมาตาแทนที่คอมพิวเตอร์เสมือนได้ เนื่องจากจะทำให้ระบบถูกจำกัดอยู่แค่เพียงสองเครื่องเท่านั้น แต่ถ้าหากเพิ่มสถานะจากสองให้เป็นตามจำนวนคอมพิวเตอร์เสมือนจะเกิดปัญหาที่ทำให้กฎมีความยาวเพิ่มและมีความซับซ้อนมากขึ้น ซึ่งจะส่งผลต่อการค้นหาคำตอบของเจเนติกอัลกอริทึมดังที่ได้กล่าวไว้ในบทที่ 3

ดังนั้น เพื่อให้เซลลูลาร์ออโตมาตาสามารถนำมาใช้กับระบบที่มีจำนวนคอมพิวเตอร์เสมือนที่มากกว่าสองเครื่อง จึงได้ออกแบบโครงสร้างของเซลลูลาร์ออโตมาตาใหม่โดยการเพิ่มความยาวของเซลลูลาร์ออโตมาตา คือ การเพิ่มจำนวนเซลล์ให้มากขึ้นเพื่อให้สามารถรองรับจำนวนคอมพิวเตอร์เสมือนที่มากกว่าสองเครื่องได้

จำนวนของเซลล์ทั้งหมดที่ใช้ในเซลลูลาร์ออโตมาตา (C) สามารถคำนวณได้ดังสมการที่ 1 โดยกำหนดให้ N คือจำนวนคอมพิวเตอร์เสมือนในระบบการประมวลผลแบบกลุ่มเมฆ และ J คือจำนวนงานทั้งหมดที่เข้ามาในระบบ

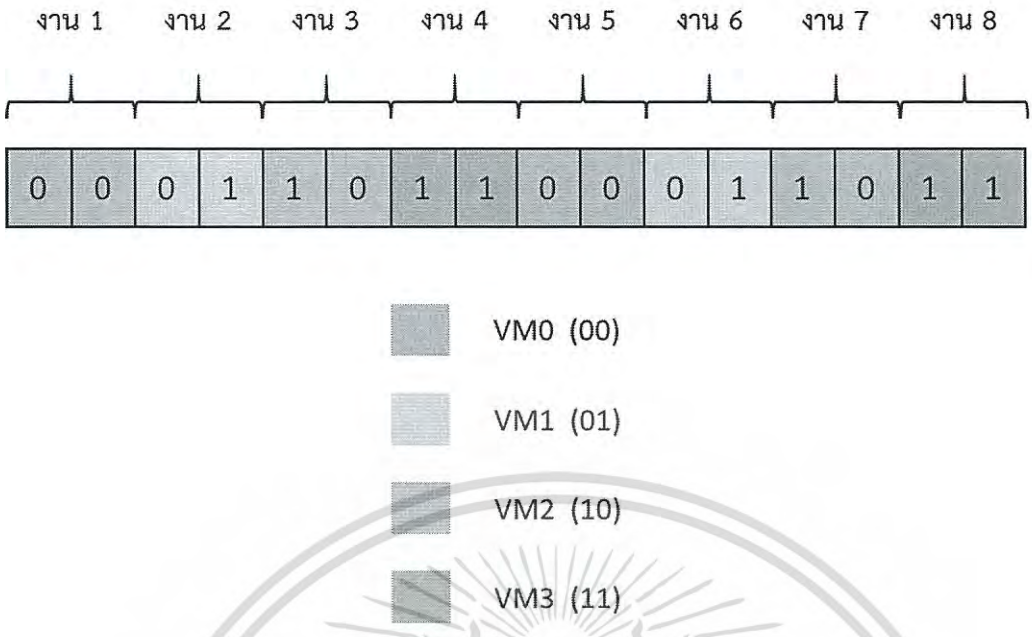
$$C = J(\log_2 N) \qquad \text{---------- (1)}$$

และจำนวนเซลล์ที่ใช้ต่อคอมพิวเตอร์เสมือนหนึ่งเครื่อง (C_{vm}) สามารถคำนวณได้ดังสมการที่ 2

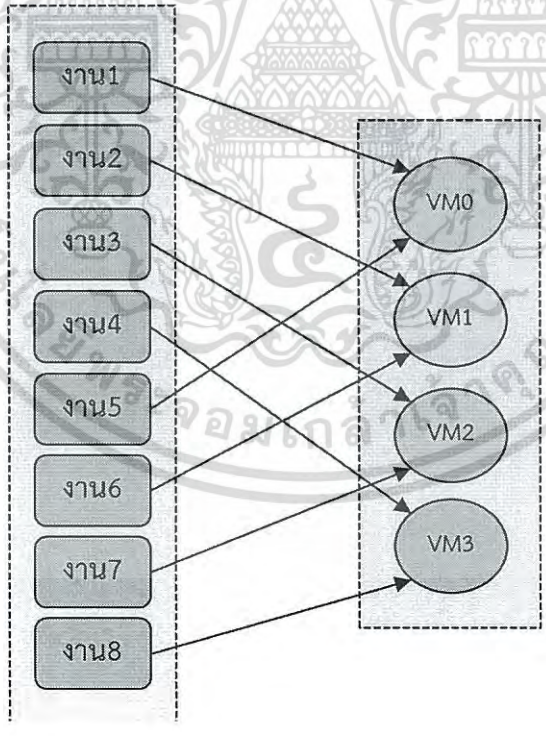
$$C_{vm} = \log_2 N \qquad \text{---------- (2)}$$

สมมติมีงานเข้ามาในระบบประมวลผลทั้งหมด 8 งาน และมีจำนวนคอมพิวเตอร์เสมือน 4 เครื่อง (J = 10, N = 4) จะได้ค่า C = 16 และ C_{vm} = 2 นั่นคือ จำนวนเซลล์ทั้งหมดที่ต้องใช้มีจำนวน 16 เซลล์ และคอมพิวเตอร์เสมือนหนึ่งเครื่องจะถูกแทนค่าด้วยเซลล์ 2 เซลล์

จากรูปที่ 4.4 เป็นตัวอย่างของวิธีการจัดสรรงานที่ถูกแทนด้วยเซลลูลาร์ออโตมาตา เมื่อกำหนดให้มีจำนวนของคอมพิวเตอร์เสมือนในระบบทั้งหมด 4 เครื่อง (00, 01, 10, 11) และจำนวนงานที่เข้ามาในระบบคือ 8 งาน ดังนั้นจำนวนเซลล์ในเซลลูลาร์ออโตมาตาจะมีค่าเท่ากับ 16 เซลล์ และจำนวนเซลล์ที่ใช้แทนงานที่เข้ามาในระบบหนึ่งงานคือ 2 เซลล์ และการจัดสรรงานที่เข้ามาในระบบให้กับคอมพิวเตอร์เสมือนจะเป็นไปดังรูปที่ 4.5 กล่าวคือ งานแรกจะถูกส่งไปยังคอมพิวเตอร์เสมือน VM0 และงานถัดมาจะถูกส่งไปยัง VM1 และจะเป็นเช่นนี้ไปตามลำดับจนถึงงานสุดท้ายซึ่งจะถูกส่งไปยัง VM3



รูปที่ 4.4 เซลลูลาร์ออโตมาตาสำหรับการจัดตารางงานที่ 4 VM



รูปที่ 4.5 การจัดสรรงานในระบบตามสถานะในเซลลูลาร์ออโตมาตา

จะเห็นว่าเซลลูลาร์ออโตมาตาจะมีความสอดคล้องกับงานที่เข้ามาในระบบการประมวลผลแบบกลุ่มเมฆ โดยสถานะของเซลล์ในเซลลูลาร์ออโตมาตาจะพัฒนาไปตามกฎที่ได้ตั้งไว้ สำหรับสถานะเริ่มต้นของเซลลูลาร์ออโตมาตาในเวลา $t = 0$ หมายถึงการจัดสรรงานในเวลาเริ่มต้น (Initial Allocation) สำหรับคอมพิวเตอร์เสมือน ซึ่งสถานะของเซลล์ในตอนเริ่มต้นจะเกิดจากการสุ่มแบบ Uniform Distribution และสถานะของเซลล์จะมีการเปลี่ยนแปลงไปตามกฎที่ได้กำหนดไว้เมื่อเวลาผ่านไป และจะมีการกำหนดรอบของการเปลี่ยนแปลงสถานะของเซลล์ ทำให้การเปลี่ยนแปลงสถานะของเซลลูลาร์ออโตมาตาจะมีผลถึงวิธีการจัดสรรงานในระบบที่เปลี่ยนแปลงตามด้วย

ในการค้นหากฎที่มีประสิทธิภาพและมีความเหมาะสมที่จะสามารถทำให้เซลลูลาร์พัฒนาสถานะของเซลล์จนได้วิธีการจัดสรรงานที่ดีที่สุด จะใช้เจเนติกอัลกอริทึมในการค้นหากฎ โดยกฎที่เจเนติกอัลกอริทึมหามาได้นั้นจะสามารถให้วิธีการจัดสรรงานที่ให้เวลาในการประมวลผลของงานทั้งหมดน้อยที่สุดเป็นเวลา T_{min} โดยสามารถเริ่มที่การจัดสรรงานแบบใดก็ได้ จนได้วิธีการจัดสรรงานตามที่ต้องการ

4.3 การออกแบบการทำงานของวิธีเจเนติกอัลกอริทึม

CATSจะนำวิธีเจเนติกอัลกอริทึมมาใช้ในการค้นหากฎที่ดีที่สุดเพื่อนำไปหาวิธีการจัดสรรงานที่ลดเวลาที่ใช้ในการประมวลผลทั้งหมดได้อย่างมีประสิทธิภาพ จึงมีการกำหนดกระบวนการในแต่ละขั้นตอนของวิธีเจเนติกอัลกอริทึมไว้ดังต่อไปนี้

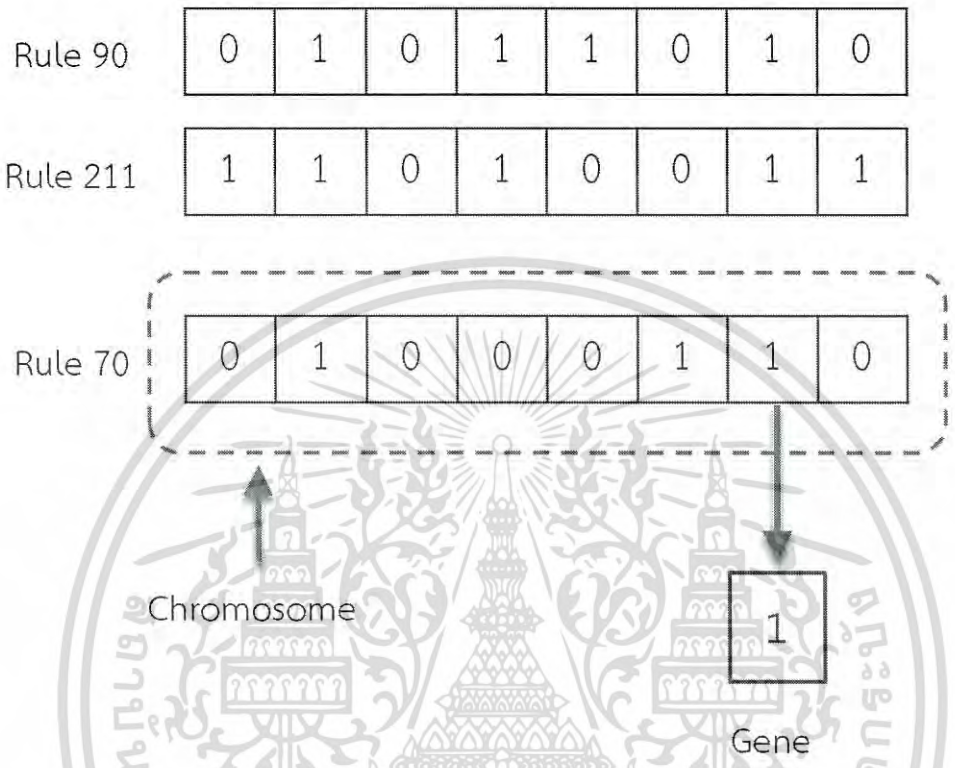
4.3.1 การออกแบบโครโมโซมแทนคำตอบ (Chromosome Encoding)

ประชากรหรือโครโมโซมใช้ในวิธี CATSคือกฎของเซลลูลาร์ออโตมาตาเพื่อใช้กำหนด Output ของเซลล์ตัวเอง $C_i(t)$ และเซลล์เพื่อนบ้าน $C_{i-1}(t)$ และ $C_{i+1}(t)$ ซึ่งเป็นข้อมูลแบบไบนารีที่มีค่า 0 และ 1 ในแต่ละเซลล์ของโครโมโซมหรือเรียกว่ายีน (Gene) ตัวอย่างดังรูป 4.6 เป็นรูปแบบของกฎ 90 ของเซลลูลาร์ออโตมาตา

	000	001	010	011	100	101	110	111
Rule 90	0	1	0	1	1	0	1	0

รูปที่ 4.6 ตัวอย่างกฎ 90 ของเซลลูลาร์ออโตมาตา

เพื่อออกแบบโครโมโซมเพื่อให้สอดคล้องกับปัญหาของวิธี CATS กำหนดให้การออกแบบโครโมโซมเป็นลักษณะ Binary Encoding ซึ่งค่าที่เป็นไปได้ในยีนคือค่า 0 และ 1 เท่านั้น โดยขนาดของโครโมโซมจะมีขนาดเท่ากับ 8 (2^3) ซึ่งจำนวนโครโมโซมที่เป็นไปได้จะมีทั้งหมด 256 รูปแบบ (2^8) ดังตัวอย่างในรูปที่ 4.7



รูปที่ 4.7 ตัวอย่างการออกแบบโครโมโซมสำหรับวิธี CATS

4.3.2 การสร้างประชากรเริ่มต้น (Initial Population)

กำหนดจำนวนประชากรเริ่มต้นโดยการสุ่มเลือกเพื่อสร้างประชากรต้นแบบขึ้นมา และนำไปใช้เป็นจุดเริ่มต้นของขั้นตอนการวิวัฒนาการ ในขั้นตอนนี้จะเป็นขั้นตอนแรกที่เกิดขึ้นก่อนที่จะเฝ้าเข้ากระบวนการของเจเนติกอัลกอริทึม และมีการทดลองเพื่อหาจำนวนประชากรเริ่มต้นที่เหมาะสมสำหรับวิธี CATS โดยมีรายละเอียดในบทที่ 5

4.3.3 การกำหนดฟังก์ชันวัตถุประสงค์ (Fitness Function)

เป็นฟังก์ชั่นที่ใช้ในการประเมินประสิทธิภาพของโครโมโซมเพื่อใช้ในการพิจารณาโครโมโซม เหมาะหรือไม่ที่จะนำมาใช้สืบทอดพันธุกรรมสำหรับสร้างโครโมโซมรุ่นใหม่ โดยวิธีการสำหรับคิดค่าความเหมาะสมนั้นจะใช้สมการที่สอดคล้องกับแต่ละปัญหา ซึ่งค่าที่ได้จะถูกเรียกว่าค่าฟิตเนส (Fitness Value)

สำหรับวิธี CATS นั้นจะกำหนดให้ค่าเฉลี่ยของค่าฟิตเนสเป็นเมคสแปน โดยเมคสแปนคือเวลาที่งานทั้งหมดในระบบใช้ในการประมวลผลตั้งแต่ต้นจนจบ เนื่องจากจุดประสงค์ของวิธี CATS คือการลดเมคสแปนให้น้อยลง ดังนั้นเราจึงใช้ค่าเฉลี่ยของเมคสแปนตัววัดประสิทธิภาพของโครโมโซม

สำหรับค่าฟิตเนส (Fitness) ของวิธี CATS คำนวณได้ดังสมการที่ 1

$$fitness = \frac{1}{T_L - T_S} \qquad \text{---------- (1)}$$

T_S คือ เวลาเริ่มต้นของงานแรกที่ถูกประมวลผล

T_L คือ เวลาสิ้นสุดของงานสุดท้ายที่ถูกประมวลผล

4.3.4 ตัวดำเนินการทางพันธุกรรม (Genetic Operator)

เพื่อให้การเกิดวิวัฒนาการไปสู่คำตอบที่ดีขึ้นจึงได้มีการทำกระบวนการ Genetic Operator และสำหรับวิธี CATS จะแบ่งออกเป็น 3 ขั้นตอน ดังนี้

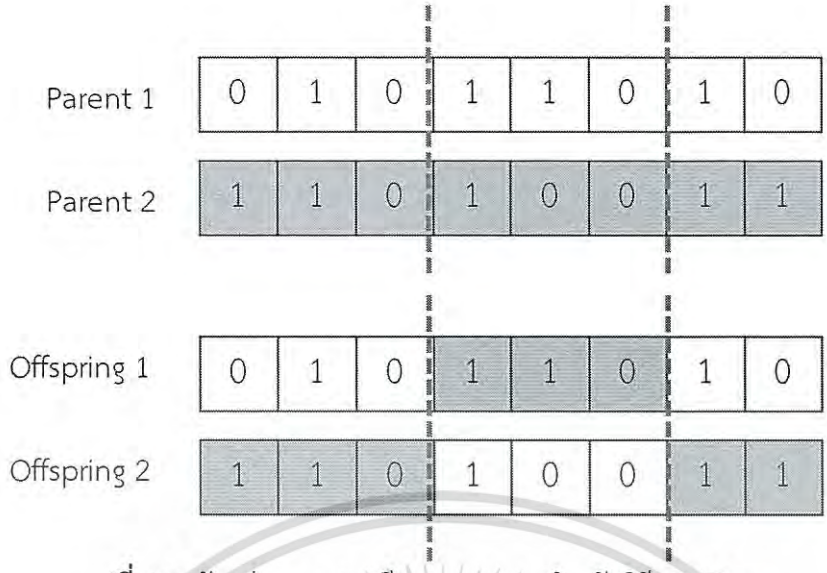
1) การคัดเลือกสายพันธุ์ (Selection)

เป็นการสร้างประชากรใหม่ด้วยการสำเนาซ้ำ (Copy) จากการคัดเลือกประชากรชุดเดิม ซึ่งจะคัดเลือกประชากรที่ดีที่สุดนั่นคือมีค่าเฉลี่ยของเมคสแปนน้อยที่สุดเป็นจำนวน 50% ของประชากรเดิม และประชากรใหม่อีก 50% ที่เหลือจะถูกสร้างจากการ Crossover และ Mutation ในขั้นตอนต่อไป

2) การแลกเปลี่ยนยีนข้ามโครโมโซมหรือครอสโอเวอร์ (Crossover)

เป็นการนำเอาโครโมโซมที่เหลือมาจับคู่และผสมยีนระหว่างกันให้ได้โครโมโซมใหม่เพื่อหาลักษณะทางพันธุกรรมใหม่ที่มีความเหมาะสมดีกว่า โดยมีตัวแปรสำคัญคือ Crossover Probablility (P_c) คือ ค่าความน่าจะเป็นของการเกิด Crossover และในกรณีที่ไม่เกิดการ Crossover เกิดขึ้นจะเป็นการทำสำเนาซ้ำรูปแบบของพันธุกรรมพ่อแม่ไปสู่ลูก (Offspring)

สำหรับวิธีการผสมยีนที่ CATS เลือกใช้จะเป็นการผสมยีนแบบสองจุด (Two-Point Crossover) โดยจะสุ่มตำแหน่งสองตำแหน่งเพื่อใช้ในการ Crossover ซึ่งจะแสดงดังรูป 4.8



รูปที่ 4.8 ตัวอย่างการผสมยีนแบบสองจุดสำหรับวิธี CATS

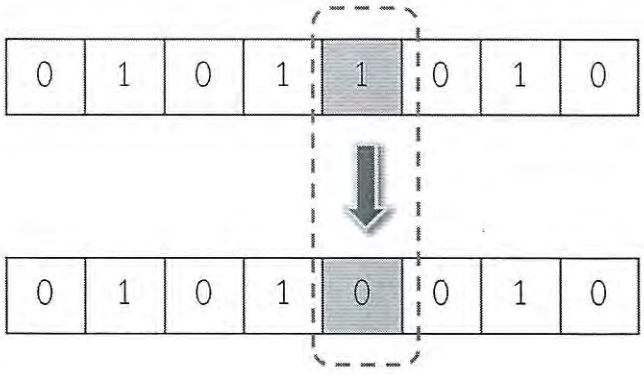
ตัวอย่างการทำ Crossover เช่น ถ้ากำหนดให้ P_c มีค่าเท่ากับ 0.8 นั่นคือมีโอกาส 80% ที่จะยอมให้เกิดการ Crossover ขึ้น ค่า P_c ที่วิธี CATS ใช้จะได้จากการทดลองในบทที่ 5

3) การปรับเปลี่ยนยีนภายในโครโมโซมหรือมิวเทชั่น (Mutation)

กระบวนการ Mutation เป็นการช่วยให้สามารถค้นพบคำตอบที่อาจไม่มีข้อมูลอยู่ในกลุ่มประชากรของโครโมโซมเดิมได้ด้วยการสุ่มเปลี่ยนยีนในโครโมโซมในอัตราความน่าจะเป็นที่ค่อนข้างต่ำ ขั้นตอนการ Mutation จะเกิดขึ้นหลังจากการ Crossover เสร็จสิ้น ซึ่งจะทำการสุ่มประชากรเพื่อเปลี่ยนแปลงผลที่ได้จากการ Crossover หมายความว่า รุ่นลูกที่เกิดจากการ Crossover ระหว่างรุ่นพ่อแม่แล้วจะถูกนำมาดำเนินการ Mutation ต่อไป

ตัวแปรสำหรับการทำ Mutation คือ Mutation Probability (P_m) เป็นค่าความน่าจะเป็นในการเกิด Mutation ในกรณีที่ไม่มีการ Mutation นั่นหมายความว่ามีเพียงการ Crossover เกิดขึ้นเพียงอย่างเดียว แต่ถ้าหากว่า เกิดการ Mutation 100% จะทำให้ทุกตำแหน่งใน Chromosome มีการเปลี่ยนแปลงทั้งหมด

สำหรับวิธีการ CATS นั้น การ Mutation จะใช้รูปแบบของ A Bit-Flip Mutation คือการกลับบิตของข้อมูลจาก 0 เป็น 1 หรือ จาก 1 เป็น 0 ดังตัวอย่างในรูป 4.9



รูปที่ 4.9 ตัวอย่างการ Mutation แบบ Bit Flipped สำหรับวิธี CATS

4.3.5 การทำซ้ำ (Repetition)

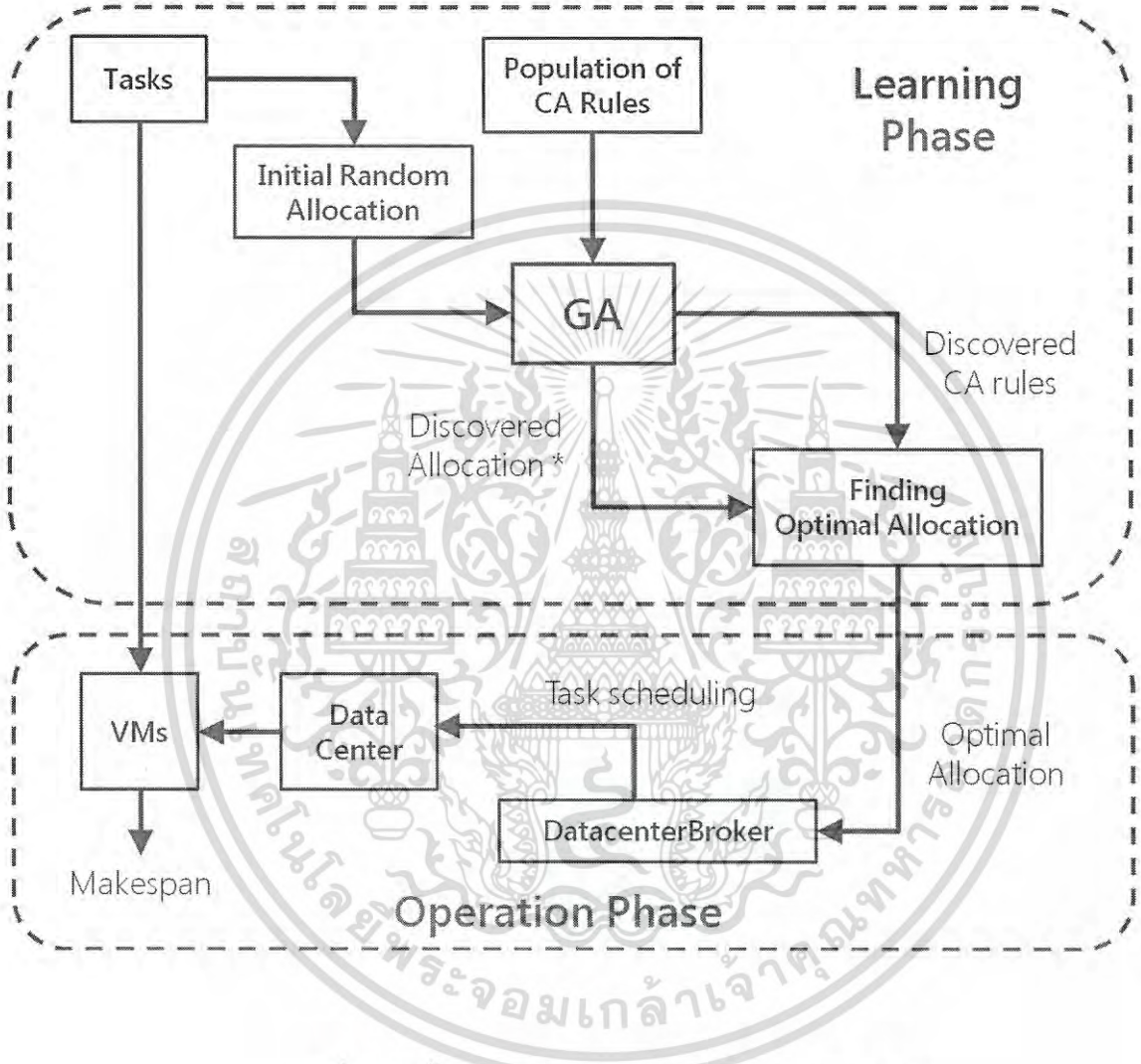
กระบวนการทั้งหมดจะถูกทำซ้ำจนกว่าจะตรงตามเงื่อนไขที่ได้กำหนดไว้

4.3.6 ตรวจสอบเงื่อนไขหยุดการทำงาน

เงื่อนไขการหยุดการทำงาน (Termination Condition) ของวิธี CATS คือการกำหนดจำนวนรุ่น (Generation: G) เมื่อเจเนติกอัลกอริทึมทำจนครบตามจำนวนรุ่นที่ได้กำหนดไว้จะหยุดการทำงานและได้ผลลัพธ์ของประชากรที่เป็นคำตอบที่ Optimal โดยจำนวนของรุ่นที่สามารถลู่เข้าหาคำตอบที่ต้องการจะถูกทดลองในบทที่ 5

4.2 ขั้นตอนการทำงานของCATS

วิธีการCATSจะแบ่งการทำงานออกเป็นสองขั้นตอน ได้แก่ ขั้นตอนการเรียนรู้ (Learning Phase) และขั้นตอนการดำเนินการ (Operation Phase) ดังแสดงในรูปที่ 4.10



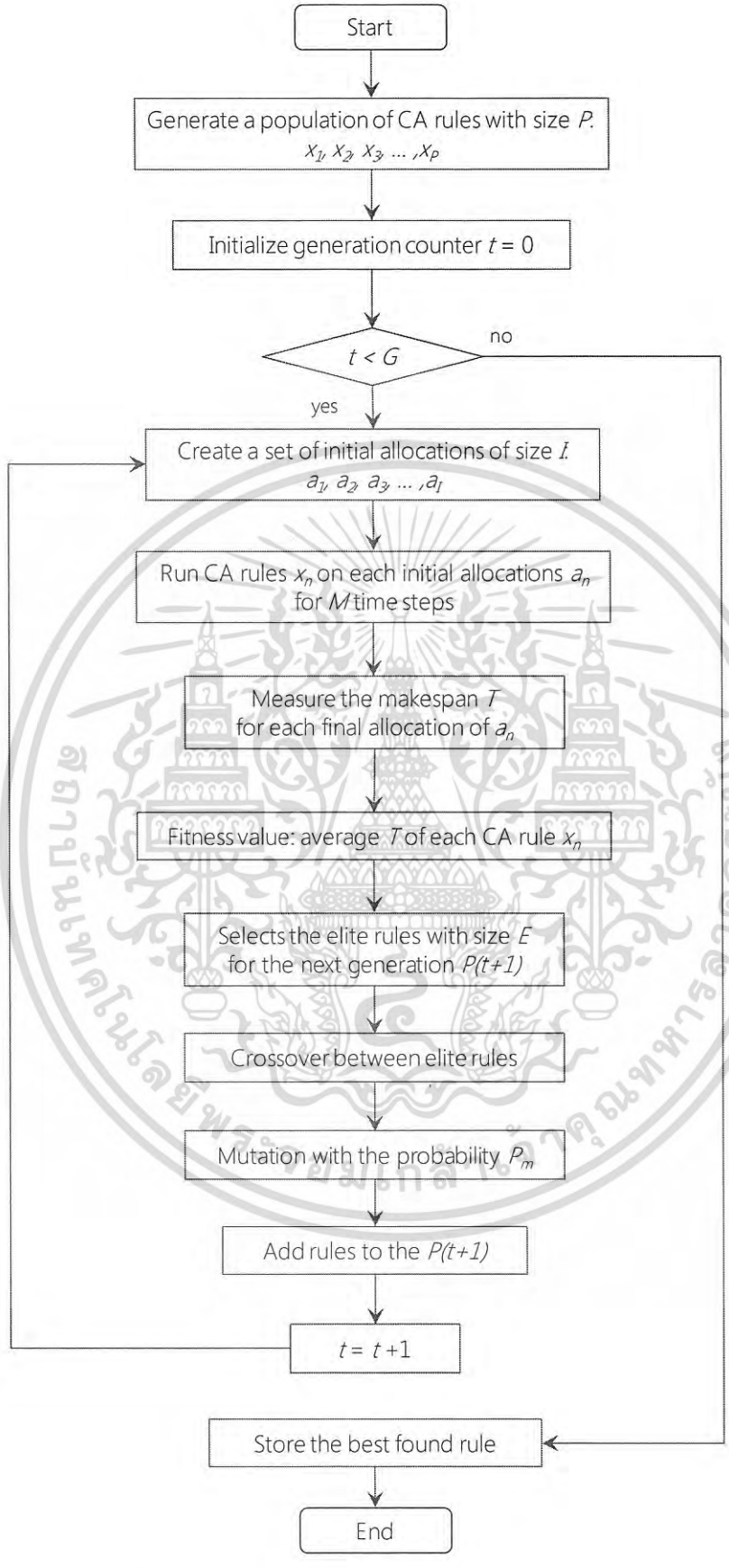
รูปที่ 4.10 ขั้นตอนการเรียนรู้และการดำเนินงานของCATS

4.2.1 ขั้นตอนการเรียนรู้ (Learning Phase)

จุดมุ่งหมายของขั้นตอนการเรียนรู้ คือ เพื่อค้นหากฎที่มีประสิทธิภาพสำหรับการจัดตารางงานและวิธีการจัดตารางที่ดีที่สุด โดยใช้เจเนติกอัลกอริทึมในการค้นหา จากนั้นจะมีการหาวิธีการจัดตารางงานที่ดีที่สุดโดยการนำกฎและวิธีการจัดตารางงานที่หาได้ไปเข้ากระบวนการหาวิธีการจัดตารางงาน และจะถูกนำไปใช้งานต่อใน Operation Phase สำหรับขั้นตอนการทำงานของเจเนติกอัลกอริทึมมีขั้นตอนดังนี้

1) เริ่มต้นด้วยการสร้างประชากรของกฎขึ้นมาโดยการสุ่มเป็นจำนวน *P* กฎ
2) สร้างเซตของเซลลูลาร์ออโตมาตาเพื่อใช้สำหรับการจัดสรรงานในตอนเริ่มต้นเป็นจำนวน *I* โดยวิธีการสุ่มสถานะในแต่ละเซลล์ของเซลลูลาร์ออโตมาตา
3) กฎของเซลลูลาร์ออโตมาตาแต่ละกฎในประชากร *P* จะทำการรันบนแต่ละการจัดสรรงานในตอนเริ่มต้นเป็นเวลา *M* รอบ
4) จากนั้นจึงวัดเวลาที่ใช้ในการประมวลผลของงานเป็นเวลา *T* ของแต่ละการจัดสรรงานในตอนสุดท้าย (ที่รอบ M)
5) หาค่าฟิตเนส (Fitness Value) ของแต่ละกฎที่เป็นสมาชิกในประชากร *P* โดยการหาเวลาเฉลี่ยของ *T* เพื่อใช้ในการคัดเลือกกฎที่มีประสิทธิภาพ
6) เจเนติกอัลกอริทึมจะทำการเลือกกฎที่ดีที่สุดขึ้นมากลุ่มหนึ่ง (Elite Rule) ซึ่งเป็นกฎที่มีค่าฟิตเนสที่ดีที่สุด โดยเลือกกฎมาเป็นจำนวน *E* สำหรับใช้ในการทำสำเนาเพื่อสร้างประชากรรุ่นถัดไปโดยไม่ต้องมีการแก้ไข
7) จำนวนกฎที่เหลืออยู่ในประชากรเป็นจำนวน *P-E* จะถูกนำมาครอสโอเวอร์ (Crossover) ด้วยค่าความน่าจะเป็น P_c เพื่อสร้างประชากรรุ่นใหม่สำหรับรอบถัดไป
8) นำประชากรที่ได้จากการครอสโอเวอร์มาทำการมิวเตชั่น (Mutation) ด้วยความน่าจะเป็นที่ค่า P_m
9) กระบวนการนี้จะถูกดำเนินการไปเรื่อยๆ *G* รอบ (Generation) กฎที่ค้นพบจะถูกนำไปเก็บไว้เพื่อนำไปใช้ในขั้นตอนการดำเนินการหลังจากกระบวนการค้นหากฎเสร็จสิ้น
10) หาวิธีการจัดตารางงานที่ดีที่สุด (Optimal Allocation) โดยการสุ่มประชากรขึ้นมาและนำกฎที่ค้นพบไปหาวิธีการจัดสรรงานที่ให้ค่าเมคสแปนน้อยที่สุด

สามารถเขียนแผนผังแสดงการทำงานอัลกอริทึมสำหรับการค้นหากฎได้ดังรูป 4.11



รูปที่ 4.11 แผนผังของอัลกอริทึมในการค้นหากฎที่มีประสิทธิภาพ

เนื่องจากเซตของการจัดสรรงานที่ใช้ในการทดสอบประสิทธิภาพของกฎจะถูกสร้างขึ้นมาใหม่ทุกครั้งในแต่ละรุ่น ทำให้ค่าฟิตเนสของกฎเปลี่ยนแปลงในทุกๆรอบ จึงเป็นการคัดสรรให้กฎที่มีค่าฟิตเนสที่ดีแบบคงที่สามารถผ่านจากรุ่นไปสู่อีกรุ่นได้ นั่นหมายความว่าผลลัพธ์ที่ได้ในตอนท้ายจะคงเหลือแต่กฎที่สามารถผ่านมาได้ทุกรุ่น ซึ่งเป็นกฎที่มีประสิทธิภาพและให้ผลลัพธ์ที่ดีกว่าแบบอื่น

4.2.2 ขั้นตอนการดำเนินงาน (Operation Phase)

เมื่อได้วิธีการจัดสรรงานจากการค้นพบกฎที่ดีที่สุดในขั้นตอนการเรียนรู้ วิธีการจัดสรรงานจะถูกส่งไปยัง Datacenter Broker เพื่อทำการจัดสรรงานให้กับงานที่เข้ามาในระบบ และ Datacenter Broker จะส่งงานไปยัง Data Center อีกต่อหนึ่งเพื่อให้นำงานไปจัดสรรให้กับคอมพิวเตอร์เสมือนตามที่ได้กำหนดไว้

สำหรับผลการประเมินประสิทธิภาพที่ได้จากการใช้วิธี CATSจะกล่าวถึงในบทถัดไป

บทที่ 5
การประเมินประสิทธิภาพ

ในบทนี้จะกล่าวถึงการนำวิธีมาก่อนได้ก่อน (FCFS) วิธีเจเนติกอัลกอริทึม (GA) วิธีแคทส์ (CATS) และวิธีระบบอาณานิคมมด (ACO) มาจำลองการทำงาน รวมถึงทำการเปรียบเทียบเพื่อประเมินประสิทธิภาพของวิธี CATS กับวิธีอื่น

โดยการประเมินประสิทธิภาพจะทำทั้งหมด 2 ส่วนด้วยกัน ส่วนแรกคือการจำลองการทำงานของวิธี CATS เพื่อค้นหาพารามิเตอร์ที่เหมาะสมสำหรับการนำไปใช้งาน และการจำลองการทำงานเพื่อเปรียบเทียบประสิทธิภาพของวิธี CATS กับวิธี FCFS, GA, และ ACO ในส่วนนี้จะแบ่งออกเป็นการประเมินประสิทธิภาพโดยการกำหนดให้ความสามารถในการประมวลผลของ VM เท่ากันและแตกต่างกัน

สำหรับการจำลองสถานการณ์ของระบบการประมวลผลแบบกลุ่มเมฆจะใช้คลาวด์ซิม (CloudSim) ในการจำลอง โดยจะมีการแก้ไขในส่วนของเอนทิตี (Entity) DatacenterBroker เพื่อนำอัลกอริทึมในการจัดตารางงานไปใช้งานในระบบ

5.1 การจำลองการทำงานของ CATS

5.1.1 การจำลองสถานการณ์เพื่อหาค่า Crossover Probability

Crossover Probability (P_c) คือ ความน่าจะเป็นของการเกิด Crossover ซึ่งมีค่าอยู่ในช่วง 0 - 100 ถ้าไม่เกิดการ Crossover ประชากรรุ่นพ่อแม่และลูกจะมีลักษณะเหมือนกันทุกประการ ซึ่งอาจส่งผลทำให้ไม่เกิดการพัฒนาประชากรเพื่อเข้าหาถึงผลลัพธ์ที่ดีที่สุดได้ แต่ในบางครั้งการที่ Crossover ในทุกครั้งอาจเป็นการสร้างรุ่นลูกที่ด้อยคุณภาพกว่าแทนที่จะใช้ประชากรรุ่นพ่อแม่เป็นประชากรตั้งต้นใน Generation ถัดไป ซึ่งสิ่งที่เกิดขึ้นจะการลดประสิทธิภาพของการเรียนรู้ เนื่องจากจุดประสงค์หลักของวิธีการเจเนติกอัลกอริทึมคือการปรับปรุงประสิทธิภาพของการเรียนรู้ในทุกๆ Generation แต่อย่างไรก็ตามการสร้างประชากรรุ่นลูกอาจไม่จำเป็นในทุกการใช้งาน ดังนั้นจึงต้องมีการทดลองเพื่อหาค่า P_c ที่เหมาะสมที่สุดสำหรับการทำงานของ CATS

ประเภท	พารามิเตอร์	ค่าที่ใช้
ดาต้าเซนเตอร์	จำนวนของดาต้าเซนเตอร์	10 ดาต้าเซนเตอร์
	จำนวนของโฮสต์	2-6 โฮสต์
	ประเภทการจัดการ	Space_shared
		Time_shared

ประเภท	พารามิเตอร์	ค่าที่ใช้
คอมพิวเตอร์เสมือน (VM)	MIPS	250-2000 MIPS
	หน่วยความจำของ VM (RAM)	512-2048 Mbytes
	ประเภทการจัดการ	Space_shared
	จำนวน VM	16 เครื่อง
Task	จำนวนงานในหนึ่งเซต	100-500 งาน
	ความยาวของงาน	5000-15000 MI

ตารางที่ 5.1 พารามิเตอร์ของการจำลองสถานการณ์สำหรับคลาวด์ซิมเพื่อหาค่า P_c

พารามิเตอร์	ค่าที่ใช้
จำนวนรุ่น (G)	20
จำนวนประชากร (P)	20
จำนวนวิธีการจัดสรรงานในตอนเริ่มต้น (I)	20
จำนวนรอบ (M)	20
จำนวนกฎที่ดีที่สุด (E)	10
ความน่าจะเป็นของการ Crossover (P_c)	0.01-0.9
ความน่าจะเป็นของการ Mutation (P_m)	0.02
จำนวนรอบที่ใช้	10

ตารางที่ 5.2 พารามิเตอร์ของการจำลองสถานการณ์สำหรับวิธี CATS เพื่อหาค่า P_c

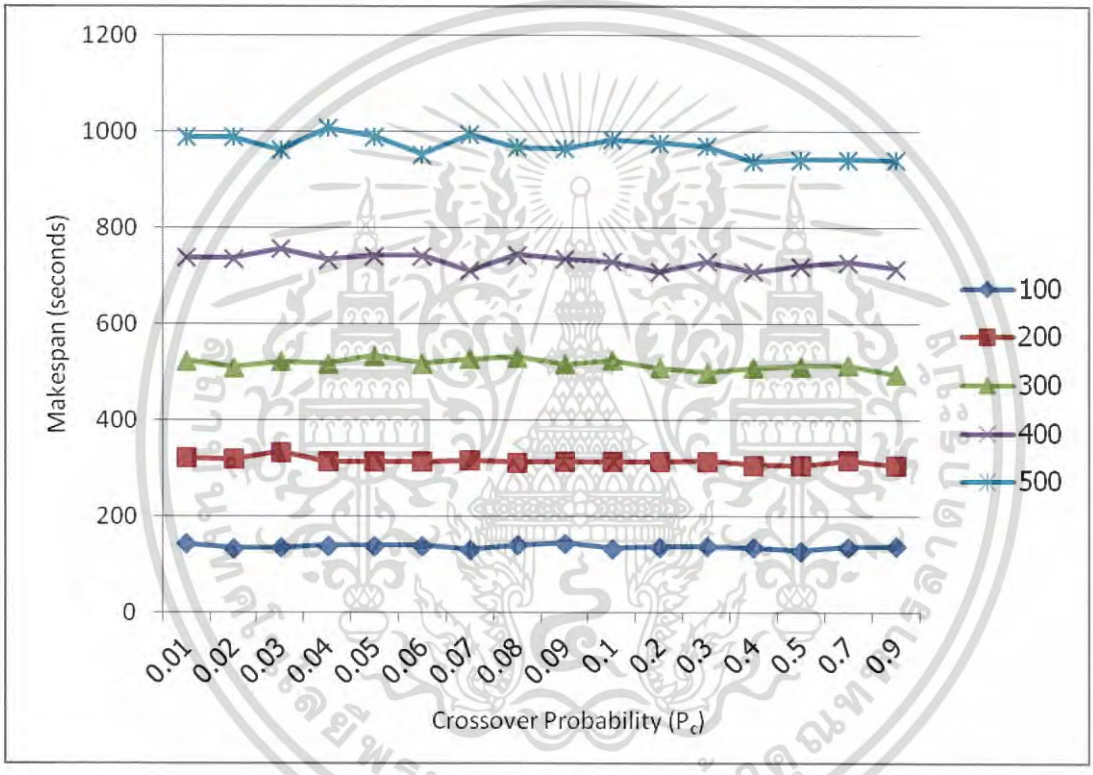
การตั้งค่าพารามิเตอร์ที่ใช้สำหรับวิธี CATS จะให้ทำงานเป็นเวลา 20 รุ่น (Generation: G) มีจำนวนประชากร (Population: P) 20 และกำหนดให้ค่า I = 20, M = 20, E = 10 สำหรับการครอสโอเวอร์และมิวเทชั่น เราได้เลือกใช้วิธี Two-Point Crossover และ A Bit-Flip Mutation โดยกำหนดให้มีค่าความน่าจะเป็นของการ Mutation คือ $P_m = 0.02$

สำหรับการจำลองหาค่า P_c ที่เหมาะสมได้ทำการใช้ค่า P_c ได้แก่ 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.9

กำหนดให้มีจำนวนดาต้าเซนเตอร์ 10 ดาต้าเซนเตอร์ โดยงานที่เข้ามาจะมีจำนวน 100, 200, 300, 400 และ 500 งานตามลำดับ และมีคอมพิวเตอร์เสมือนทั้งหมด 16 เครื่อง ขอบเขตความยาวของงาน (length) จะเริ่มตั้งแต่ 5000 ล้านคำสั่ง (Million Instruction: MI) จนถึง 15000 MI ส่วนความสามารถใน

การประมวลผลของคอมพิวเตอร์เสมือนจะอยู่ระหว่าง 250 ล้านคำสั่งต่อวินาที (Million Instruction Per Second: MIPS) จนถึง 2000 MIPS ทุกๆดาต้าเซนเตอร์จะจัดสรรคอมพิวเตอร์เสมือนบนโฮสต์แบบสุ่มโดยการใช้นโยบายแบบ Time_shared หรือ Space_shared และคอมพิวเตอร์เสมือนทุกเครื่องจะใช้นโยบาย Space_shared เท่านั้น

ในการจำลองสถานการณ์จะดำเนินการจำลองทั้งหมด 10 รอบและหาค่าเฉลี่ย ผลการจำลองสถานการณ์จะได้ดังรูป 5.1 จะเห็นว่าค่า P_c ที่เหมาะสมสำหรับการใช้งานกับวิธี CATS คือ $P_c = 0.9$ เนื่องจากสามารถให้ค่าเฉลี่ยของเมคสแปนที่ดีสำหรับทุกงานที่เข้ามาในระบบและมีความคงที่ไม่แกว่งเหมือนกับเมคสแปนที่ค่า P_c อื่นๆ



รูป 5.1 การจำลองหาค่า P_c กับวิธี CATS

5.1.2 การจำลองสถานการณ์เพื่อหาค่า Mutation Probability

ในเจเนติกอัลกอริทึมค่า Mutation Probability (P_m) คือ ความน่าจะเป็นของการเกิดมิวเทชั่น (Mutation) เป็นการปรับเปลี่ยนยีนภายในโครโมโซม ซึ่งจะถูกกำหนดไว้ในช่วง 0 ถึง 1 การมิวเทชั่นเป็นการเพิ่มโอกาสในการค้นหาคำตอบ (Exploration) โดยการเพิ่มพื้นที่ในการค้นหาค่าคำตอบที่เป็นไปได้ทั้งหมด (Search Space)

แต่การหาค่าที่เหมาะสมนั้นมีความจำเป็นเนื่องจากการที่มีค่า P_m ที่มากเกินไปอาจจะทำให้ไม่สามารถลู่เข้าหาคำตอบได้เนื่องจากการหาคำตอบจะกลายเป็นการสุ่มซึ่งจะไปลดประสิทธิภาพการทำงานของเจเนติก

อัลกอริทึม และในขณะเดียวกันหากกำหนดค่า P_m ที่น้อยเกินไปอาจจะส่งผลให้เกิดปัญหาคำตอบที่เหมาะสมเฉพาะแห่ง (Local Optimum) ซึ่งไม่ใช่คำตอบที่เหมาะสมโดยรวม (Global Optimum) ทำให้ลดโอกาสในการเจอผลลัพธ์ที่ดีกว่า

พารามิเตอร์สำหรับการจำลองการทำงานสำหรับคลาวด์ซิมจะแสดงให้เห็นดังตาราง 5.3 และพารามิเตอร์สำหรับวิธี CATS ดังตาราง 5.4

ประเภท	พารามิเตอร์	ค่าที่ใช้
ดาต้าเซนเตอร์	จำนวนของดาต้าเซนเตอร์	10 ดาต้าเซนเตอร์
	จำนวนของโฮสต์	2-6 โฮสต์
	ประเภทการจัดการ	Space_shared
		Time_shared
คอมพิวเตอร์เสมือน (VM)	MIPS	250-2000 MIPS
	หน่วยความจำของ VM (RAM)	512-2048 Mbytes
	ประเภทการจัดการ	Space_shared
	จำนวน VM	16 เครื่อง
Task	จำนวนงานในหนึ่งเซต	100-500 งาน
	ความยาวของงาน	5000-15000 MI

ตารางที่ 5.3 พารามิเตอร์ของการจำลองสถานการณ์สำหรับคลาวด์ซิมเพื่อหาค่า P_m

พารามิเตอร์	ค่าที่ใช้
จำนวนรุ่น (G)	20
จำนวนประชากร (P)	20
จำนวนวิธีการจัดสรรงานในตอนเริ่มต้น (I)	20
จำนวนรอบ (M)	20
จำนวนกฎที่ดีที่สุด (E)	10
ความน่าจะเป็นของการ Crossover (P_c)	0.9
ความน่าจะเป็นของการ Mutation (P_m)	0.01-0.9
จำนวนรอบที่ใช้	10

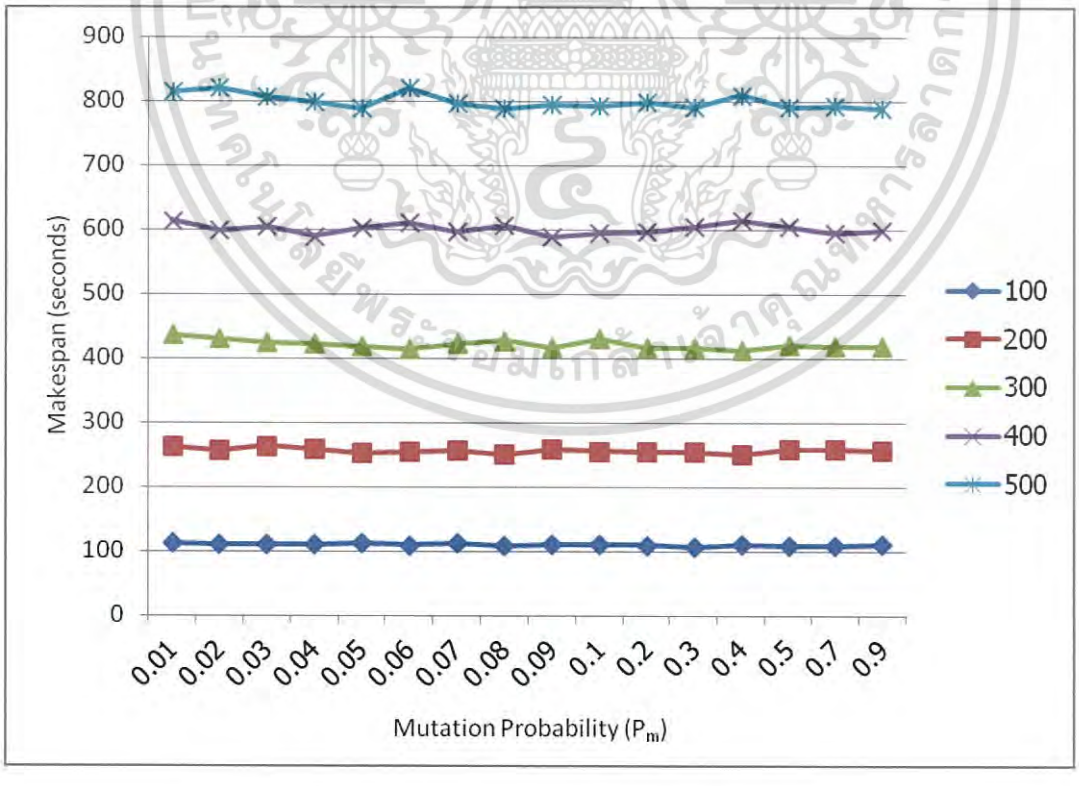
ตารางที่ 5.4 พารามิเตอร์ของการจำลองสถานการณ์สำหรับวิธี CATS เพื่อหาค่า P_m

การตั้งค่าพารามิเตอร์ที่ใช้สำหรับวิธี CATS จะให้ทำงานเป็นเวลา 20 รุ่น (Generation: G) มีจำนวนประชากร (Population: P) 20 และกำหนดให้ค่า I = 20, M = 20, E = 10 สำหรับการครอสโอเวอร์และมิวเทชั่น เราได้เลือกใช้วิธี Two-Point Crossover และ A Bit-Flip Mutation โดยกำหนดให้มีค่าความน่าจะเป็นของการ Crossover คือ P_c = 0.9 ซึ่งเป็นค่าที่ได้จากการทดลองในข้อ 5.1.1

สำหรับการจำลองหาค่า P_m ที่เหมาะสมได้ทำการใช้ค่า P_m ได้แก่ 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.9

กำหนดให้มีจำนวนดาต้าเซนเตอร์ 10 ดาต้าเซนเตอร์ โดยงานที่เข้ามาจะมีจำนวน 100, 200, 300, 400 และ 500 งานตามลำดับ และมีคอมพิวเตอร์เสมือนทั้งหมด 16 เครื่อง ขอบเขตความยาวของงาน (length) จะเริ่มตั้งแต่ 5000 ล้านคำสั่ง (Million Instruction: MI) จนถึง 15000 MI ส่วนความสามารถในการประมวลผลของคอมพิวเตอร์เสมือนจะอยู่ระหว่าง 250 ล้านคำสั่งต่อวินาที (Million Instruction Per Second: MIPS) จนถึง 2000 MIPS ทุกๆดาต้าเซนเตอร์จะจัดสรรคอมพิวเตอร์เสมือนบนโฮสต์แบบสุ่มโดยการใช้นโยบายแบบ Time_shared หรือ Space_shared และคอมพิวเตอร์เสมือนทุกเครื่องจะใช้นโยบาย Space_shared เท่านั้น

ในการทดลองจะใช้ทั้งหมด 10 รอบและหาค่าเฉลี่ย ผลการจำลองสถานการณ์จะได้ดังรูป 5.2 จะเห็นว่าค่า P_m ที่เหมาะสมสำหรับการใช้งานกับวิธี CATS คือ P_m = 0.04



รูปที่ 5.2 การจำลองหาค่า P_m กับวิธี CATS

5.1.3 การจำลองสถานการณ์เพื่อหาจำนวนประชากร

จำนวนประชากร (Population Size: P) จะแสดงให้เห็นถึงจำนวนโครโมโซมทั้งหมดที่เป็นประชากรใน 1 Generation ถ้ามีจำนวนประชากรที่น้อยเกินไปอาจทำให้เจเนติกอัลกอริทึมมีโอกาสน้อยที่จะทำ Crossover และลด Search Space น้อยลง ในทางตรงกันข้าม ถ้ามีจำนวนประชากรที่มากเกินไปจะส่งผลให้เจเนติกอัลกอริทึมทำงานได้ช้าลง การเพิ่มจำนวนประชากรอาจไม่ได้ทำให้ได้ประสิทธิภาพที่ดีขึ้น จึงต้องมีการทดลองเพื่อหาจำนวนประชากรที่เหมาะสมเช่นเดียวกันกับพารามิเตอร์อื่นๆ

พารามิเตอร์สำหรับการจำลองการทำงานสำหรับคลาวด์ซิมจะแสดงให้เห็นดังตาราง 5.5 และพารามิเตอร์สำหรับวิธี CATS ดังตาราง 5.6

ประเภท	พารามิเตอร์	ค่าที่ใช้
ดาต้าเซนเตอร์	จำนวนของดาต้าเซนเตอร์	10 ดาต้าเซนเตอร์
	จำนวนของโฮสต์	2-6 โฮสต์
	ประเภทการจัดการ	Space_shared
		Time_shared
คอมพิวเตอร์เสมือน (VM)	MIPS	250-2000 MIPS
	หน่วยความจำของ VM (RAM)	512-2048 Mbytes
	ประเภทการจัดการ	Space_shared
	จำนวน VM	16 เครื่อง
Task	จำนวนงานในหนึ่งเซต	100-500 งาน
	ความยาวของงาน	5000-15000 MI

ตารางที่ 5.5 พารามิเตอร์ของการจำลองสถานการณ์สำหรับคลาวด์ซิมเพื่อหาค่า P

พารามิเตอร์	ค่าที่ใช้
จำนวนรุ่น (G)	20
จำนวนประชากร (P)	5-100
จำนวนวิธีการจัดสรรงานในตอนเริ่มต้น (I)	20
จำนวนรอบ (M)	20
จำนวนกฎที่ดีที่สุด (E)	10

พารามิเตอร์	ค่าที่ใช้
ความน่าจะเป็นของการ Crossover (P_c)	0.9
ความน่าจะเป็นของการ Mutation (P_m)	0.04
จำนวนรอบที่ใช้	10

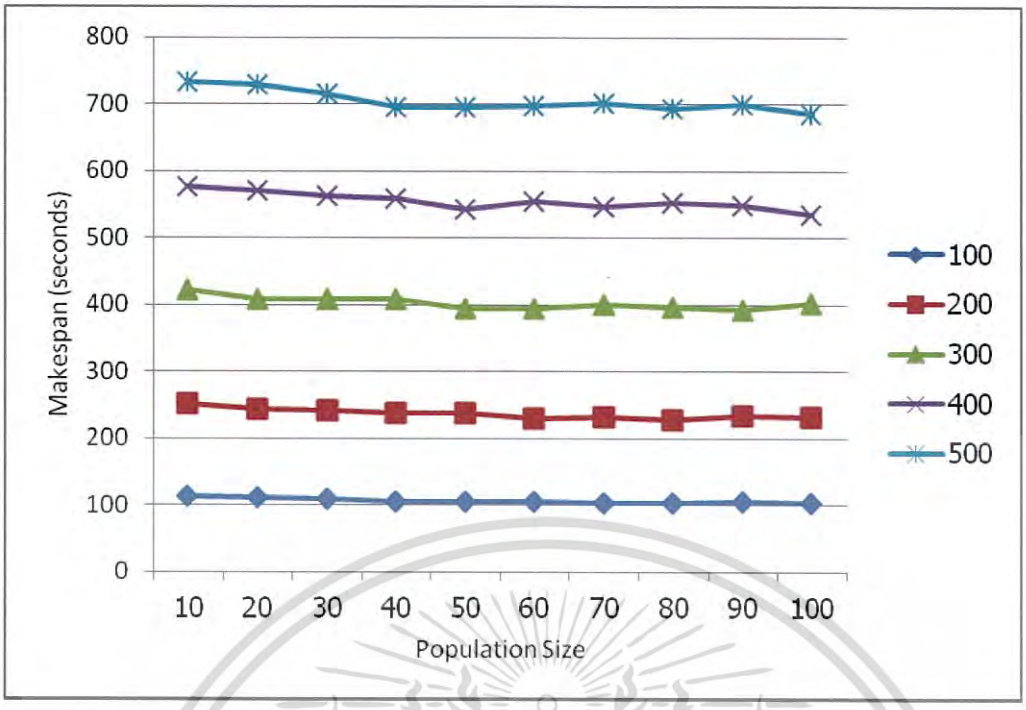
ตารางที่ 5.6 พารามิเตอร์ของการจำลองสถานการณ์สำหรับวิธี CATS เพื่อหาค่า P

การตั้งค่าพารามิเตอร์ที่ใช้สำหรับวิธี CATS จะให้ทำงานเป็นเวลา 20 รุ่น (Generation: G) กำหนดให้ค่า I = 20, M = 20, E = 10 สำหรับการครอสโอเวอร์และมิวเทชั่น เราได้เลือกใช้วิธี Two-Point Crossover และ A Bit-Flip Mutation โดยกำหนดให้มีค่าความน่าจะเป็นของการ Crossover คือ $P_c = 0.9$ และค่าความน่าจะเป็นของการ Mutation คือ $P_m = 0.04$

สำหรับการจำลองหาค่า P ที่เหมาะสมได้ทำการใช้ค่า P ได้แก่ 5, 10, 15, 20, 30, 50, 80, 100

กำหนดให้มีจำนวนดาต้าเซนเตอร์ 10 ดาต้าเซนเตอร์ โดยงานที่เข้ามาจะมีจำนวน 100, 200, 300, 400 และ 500 งานตามลำดับ และมีคอมพิวเตอร์เสมือนทั้งหมด 16 เครื่อง ขอบเขตความยาวของงาน (length) จะเริ่มตั้งแต่ 5000 ล้านคำสั่ง (Million Instruction: MI) จนถึง 15000 MI ส่วนความสามารถในการประมวลผลของคอมพิวเตอร์เสมือนจะอยู่ระหว่าง 250 ล้านคำสั่งต่อวินาที (Million Instruction Per Second: MIPS) จนถึง 2000 MIPS ทุกๆดาต้าเซนเตอร์จะจัดสรรคอมพิวเตอร์เสมือนบนโฮสต์แบบสุ่มโดยการใช้นโยบายแบบ Time_shared หรือ Space_shared และคอมพิวเตอร์เสมือนทุกเครื่องจะใช้นโยบาย Space_shared เท่านั้น

ในการทดลองจะใช้ทั้งหมด 10 รอบและหาค่าเฉลี่ย ผลการจำลองสถานการณ์จะได้ดังรูป 5.3 ผลลัพธ์ที่ได้คือ CATS จะให้คำตอบที่เริ่ม Optimal เมื่อใช้จำนวนประชากรตั้งแต่ 50 เป็นต้นไป



รูปที่ 5.3 การจำลองหาค่า P กับวิธี CATS

5.1.3 การจำลองสถานการณ์เพื่อหาค่า Generation

เป็นการจำลองเพื่อค้นหาช่วง Generation ที่ค่าฟิตเนสลู่เข้าหาคำตอบที่ดีที่สุด เพื่อให้สามารถประเมินจำนวนรอบการทำงานที่จำเป็นต้องใช้ในเจเนติกอัลกอริทึมสำหรับ CATS ได้ ในการจำลองสถานการณ์จะใช้พารามิเตอร์ดังตาราง 5.7 และ 5.8 โดยทดสอบกับงานที่เข้ามาในระบบจำนวน 300 งาน ใช้คอมพิวเตอร์เสมือนจำนวน 16 เครื่อง และกำหนดจำนวนรอบของ Generation (G) ทั้งหมด 200 รอบ และมีการจำลองทั้งหมด 10 รอบ เพื่อนำไปหาค่าเฉลี่ยต่อไป

ประเภท	พารามิเตอร์	ค่าที่ใช้
ดาต้าเซนเตอร์	จำนวนของดาต้าเซนเตอร์	10 ดาต้าเซนเตอร์
	จำนวนของโฮสต์	2-6 โฮสต์
	ประเภทการจัดการ	Space_shared
		Time_shared
คอมพิวเตอร์เสมือน (VM)	MIPS	250-2000 MIPS
	หน่วยความจำของ VM (RAM)	512-2048 Mbytes
	ประเภทการจัดการ	Space_shared
	จำนวน VM	16 เครื่อง

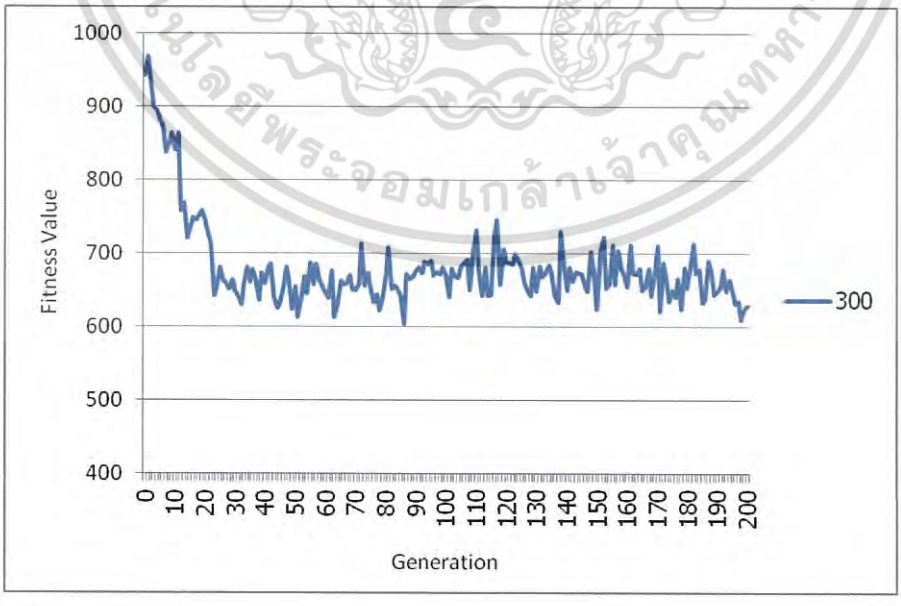
ประเภท	พารามิเตอร์	ค่าที่ใช้
Task	จำนวนงาน	300 งาน
	ความยาวของงาน	5000-15000 MI

ตารางที่ 5.7 พารามิเตอร์ของการจำลองสถานการณ์สำหรับคลาวด์ซิมเพื่อหาค่า G

พารามิเตอร์	ค่าที่ใช้
จำนวนรุ่น (G)	100
จำนวนประชากร (P)	20
จำนวนวิธีการจัดสรรงานในตอนเริ่มต้น (I)	20
จำนวนรอบ (M)	20
จำนวนกฎที่ดีที่สุด (E)	10
ความน่าจะเป็นของการ Crossover (P_c)	0.9
ความน่าจะเป็นของการ Mutation (P_m)	0.04
จำนวนรอบที่ใช้	10

ตารางที่ 5.8 พารามิเตอร์ของการจำลองสถานการณ์สำหรับวิธี CATS เพื่อหาค่า G

ผลลัพธ์จะได้ดังรูปที่ 5.4 แสดงให้เห็นการลู่เข้าหาค่า Fitness Value ที่ดีที่สุดจะเริ่มที่ประมาณรอบที่ 30 ดังนั้นค่า Generation ที่เหมาะสมในนำไปใช้กับวิธี CATS คือ 30 Generation



รูปที่ 5.4 การจำลองหาค่า Generation กับวิธี CATS

5.2 การเปรียบเทียบประสิทธิภาพของ CATS กับวิธีการอื่น

เปรียบเทียบประสิทธิภาพโดยการเปรียบเทียบเมคสแปน เพื่อหาระบบที่ให้เมคสแปนที่น้อยที่สุด สำหรับระบบที่นำมาเปรียบเทียบประสิทธิภาพกับวิธี CATS มีดังนี้

1) วิธีแบบมาก่อนได้ก่อน (First-Come-First-Served: FCFS)

2) วิธีเจเนติกอัลกอริทึม (Genetic Algorithm: GA)

3) วิธีระบบอาณานิคมมด (Ant Colony Optimization: ACO)

พารามิเตอร์ที่ใช้ในการจำลองสถานการณ์เพื่อเปรียบเทียบประสิทธิภาพจะใช้พารามิเตอร์ดังตารางที่ 5.9, 5.10, 5.11 และ 5.12

โดยในตารางที่ 5.9 จะมีการสุ่มค่าพารามิเตอร์บางค่าให้มีความแตกต่างกันตามช่วงที่กำหนด ได้แก่ จำนวนของโฮสต์ 2-6 โฮสต์ ความสามารถในการประมวลผลของ VM 250-2000 MIPS ความยาวของงาน 5000-15000 MI

ประเภท	พารามิเตอร์	ค่าที่ใช้
ดาต้าเซนเตอร์	จำนวนของดาต้าเซนเตอร์	10 ดาต้าเซนเตอร์
	จำนวนของโฮสต์	2-6 โฮสต์
	ประเภทการจัดการ	Space_shared
		Time_shared
คอมพิวเตอร์เสมือน (VM)	MIPS	250-2000 MIPS
	หน่วยความจำของ VM (RAM)	512-2048 Mbytes
	ประเภทการจัดการ	Space_shared
	จำนวน VM	2-32 VM
Task	จำนวนงานในหนึ่งเซต	100-500 งาน
	ความยาวของงาน	5000-15000 MI

ตารางที่ 5.9 พารามิเตอร์ของการจำลองสถานการณ์สำหรับคลาวด์ซิมเพื่อเปรียบเทียบกับระบบอื่น

พารามิเตอร์	ค่าที่ใช้
จำนวนรุ่น (G)	30
จำนวนประชากร (P)	50
จำนวนวิธีการจัดสรรงานในตอนเริ่มต้น (I)	20
จำนวนรอบ (M)	20
จำนวนกฎที่ดีที่สุด (E)	20
ความน่าจะเป็นของการ Crossover (P_c)	0.9
ความน่าจะเป็นของการ Mutation (P_m)	0.04
จำนวนรอบที่ใช้	10

ตารางที่ 5.10 พารามิเตอร์ของการจำลองสถานการณ์สำหรับวิธี CATS เพื่อเปรียบเทียบกับระบบอื่น

พารามิเตอร์	ค่าที่ใช้
จำนวนรุ่น (G)	30
จำนวนประชากร (P)	50
จำนวนกฎที่ดีที่สุด (E)	20
ความน่าจะเป็นของการ Crossover (P_c)	0.9
ความน่าจะเป็นของการ Mutation (P_m)	0.04
จำนวนรอบที่ใช้	10

ตารางที่ 5.11 พารามิเตอร์ของการจำลองสถานการณ์สำหรับวิธีเจเนติกอัลกอริทึม

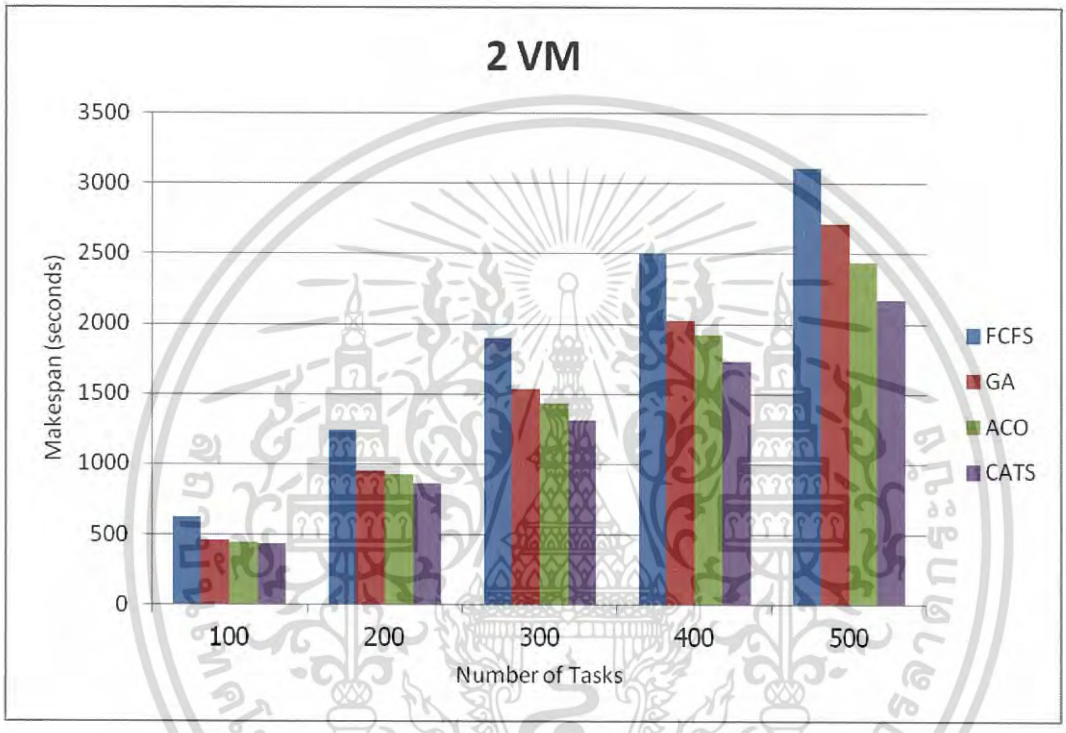
พารามิเตอร์	ค่าที่ใช้
จำนวนรอบของมด	50
จำนวนมด (m)	10
α	0.3
β	1
ρ	0.4
Q	100
จำนวนรอบที่ใช้	10

ตารางที่ 5.12 พารามิเตอร์ของการจำลองสถานการณ์สำหรับวิธีระบบอาณานิคมมด

5.2.1 การเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นที่ 2, 4, 8, 16, 32 VM

ในการเปรียบเทียบเมคสแปนจะเทียบวิธี CATS กับวิธีอื่นเมื่อมีงานเข้ามาในระบบ 100-500 งาน โดยจะประเมินประสิทธิภาพที่จำนวน VM เป็น 2, 4, 8, 16, 32 โดยใช้พารามิเตอร์ต่างๆตามที่ได้กล่าวข้างต้น และเปรียบเทียบเมคสแปนในรูปแบบของกราฟ ดังแสดงในรูปที่ 5.5 ถึง 5.10

1) การเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นที่ 2 VM

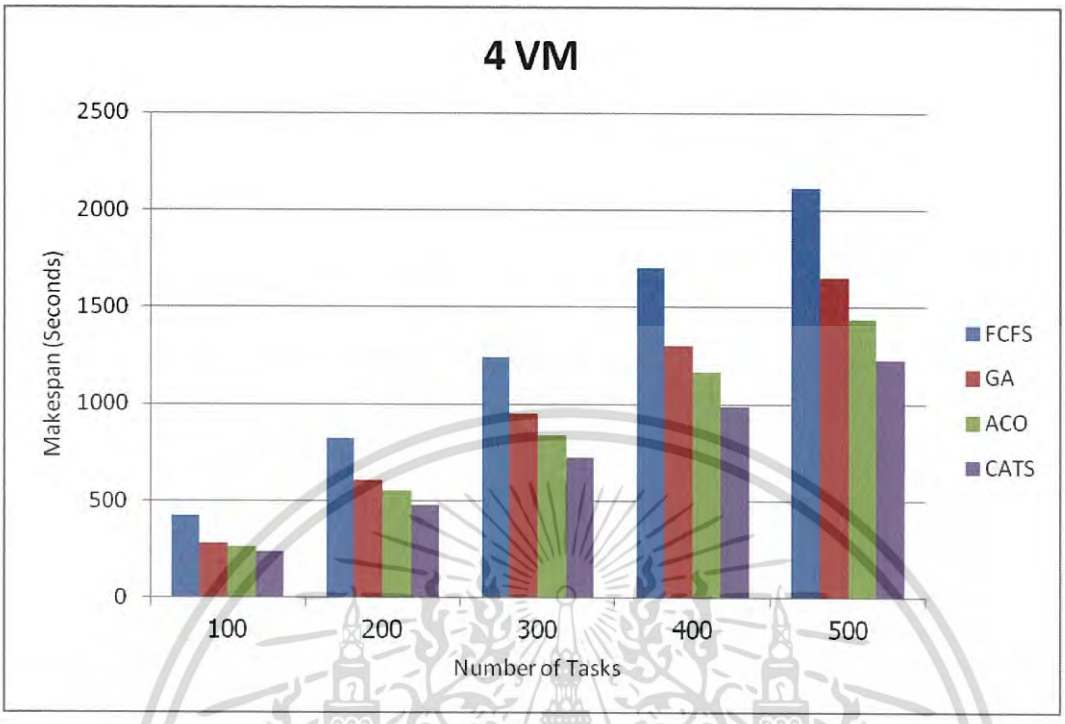


รูปที่ 5.5 ผลการเปรียบเทียบเมคสแปนของวิธี CATS กับวิธีอื่นที่ 2 VM

จากรูปที่ 5.5 แสดงค่าเมคสแปนกับจำนวนงานที่เพิ่มขึ้นในระบบประมวลผล โดยจากกราฟจะเรียงลำดับประสิทธิภาพในการลดเมคสแปนของแต่ละวิธีจากมากที่สุดไปยังน้อยที่สุดได้ดังนี้ CATS, ACO, GA และ FCFS ตามลำดับ

สังเกตได้ว่าวิธี CATS มีประสิทธิภาพในการลดเมคสแปนที่เหนือกว่าวิธี FCFS, GA, ACO ส่งผลทำให้วิธีของ CATS ช่วยลดระยะเวลาของการประมวลผลของงานที่เข้ามาในระบบที่มีจำนวน 2 VM ตั้งแต่ 100, 200, 300, 400 และ 500 งาน ได้มากที่สุดเมื่อเทียบกับวิธีอื่น

2) การเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นที่ 4 VM

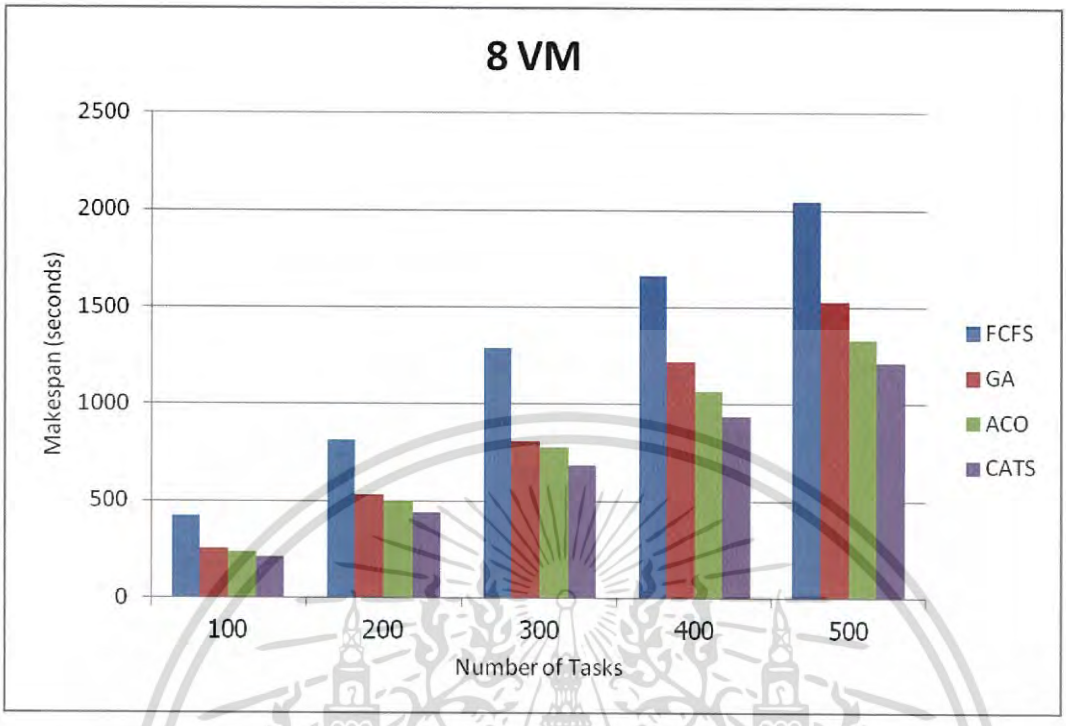


รูปที่ 5.6 ผลการเปรียบเทียบเมคสแปนของวิธี CATS กับวิธีอื่นที่ 4 VM

จากรูปที่ 5.6 แสดงค่าเมคสแปนกับจำนวนงานที่เพิ่มขึ้นในระบบประมวลผล โดยจากกราฟจะเรียงลำดับประสิทธิภาพในการลดเมคสแปนของแต่ละวิธีจากมากที่สุดไปยังน้อยที่สุดได้ดังนี้ CATS, ACO, GA และ FCFS ตามลำดับ

สังเกตได้ว่าวิธี CATS มีประสิทธิภาพในการลดเมคสแปนที่เหนือกว่าวิธี FCFS, GA, ACO ส่งผลทำให้วิธีของ CATS ช่วยลดระยะเวลาของการประมวลผลของงานที่เข้ามาในระบบที่มีจำนวน 4 VM ตั้งแต่ 100, 200, 300, 400 และ 500 งาน ได้มากที่สุดเมื่อเทียบกับวิธีอื่น

3) การเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นที่ 8 VM

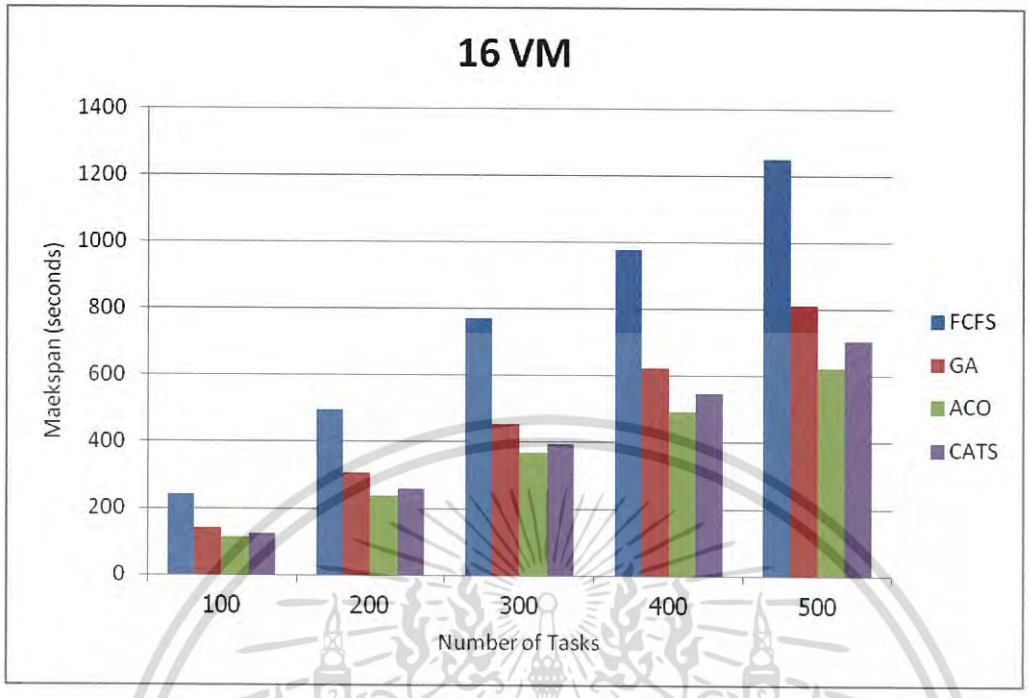


รูปที่ 5.7 ผลการเปรียบเทียบเมคสแปนของวิธี CATS กับวิธีอื่นที่ 8 VM

จากรูปที่ 5.7 แสดงค่าเมคสแปนกับจำนวนงานที่เพิ่มขึ้นในระบบประมวลผล โดยจากกราฟจะเรียงลำดับประสิทธิภาพในการลดเมคสแปนของแต่ละวิธีจากมากที่สุดไปยังน้อยที่สุดได้ดังนี้ CATS, ACO, GA และ FCFS ตามลำดับ

สังเกตได้ว่าวิธี CATS มีประสิทธิภาพในการลดเมคสแปนที่เหนือกว่าวิธี FCFS, GA, ACO ส่งผลทำให้วิธีของ CATS ช่วยลดระยะเวลาของการประมวลผลของงานที่เข้ามาในระบบที่มีจำนวน 8 VM ตั้งแต่ 100, 200, 300, 400 และ 500 งาน ได้มากที่สุดเมื่อเทียบกับวิธีอื่น

4) การเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นที่ 16 VM

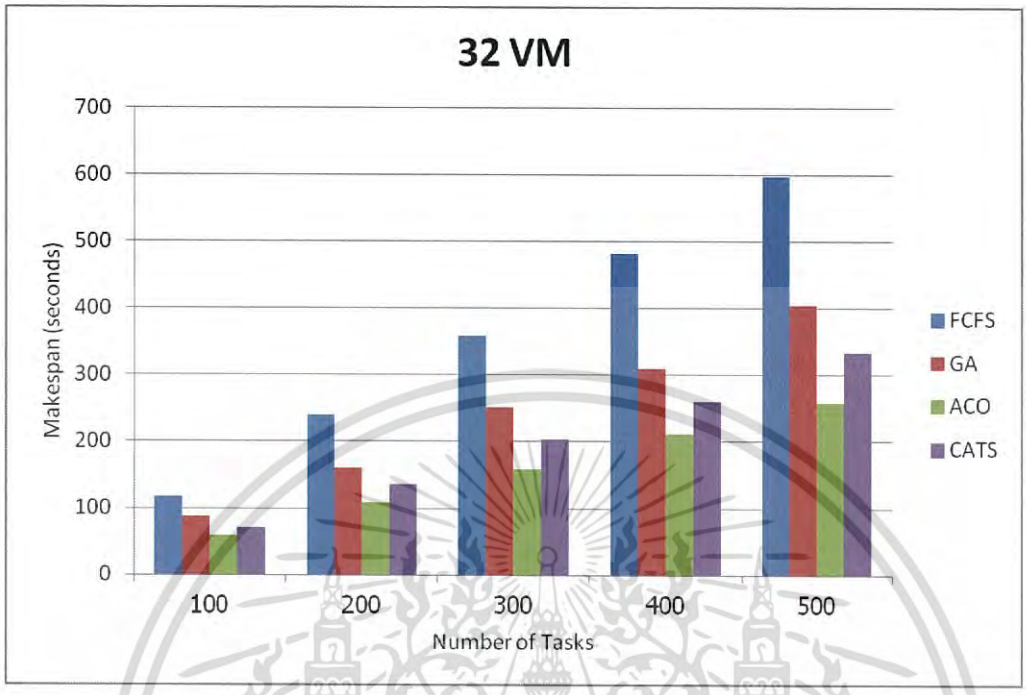


รูปที่ 5.8 ผลการเปรียบเทียบเมคสแปนของวิธี CATS กับวิธีอื่นที่ 16 VM

จากรูปที่ 5.8 แสดงค่าเมคสแปนกับจำนวนงานที่เพิ่มขึ้นในระบบประมวลผล โดยจากกราฟจะเรียงลำดับประสิทธิภาพในการลดเมคสแปนของแต่ละวิธีจากมากที่สุดไปยังน้อยที่สุดได้ดังนี้ ACO, CATS, GA และ FCFS ตามลำดับ

สังเกตได้ว่าวิธี CATS มีประสิทธิภาพในการลดเมคสแปนที่เหนือกว่าวิธี FCFS, GA แต่ยังมีประสิทธิภาพที่ด้อยกว่าวิธี ACO เมื่อเริ่มมีจำนวนของ VM ที่เพิ่มขึ้นเป็น 16 VM ส่งผลทำให้วิธีของ ACO ช่วยลดระยะเวลาของการประมวลผลของงานที่เข้ามาในระบบที่มีจำนวน 16 VM ตั้งแต่ 100, 200, 300, 400 และ 500 งาน ได้มากที่สุด

5) การเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นที่ 32 VM

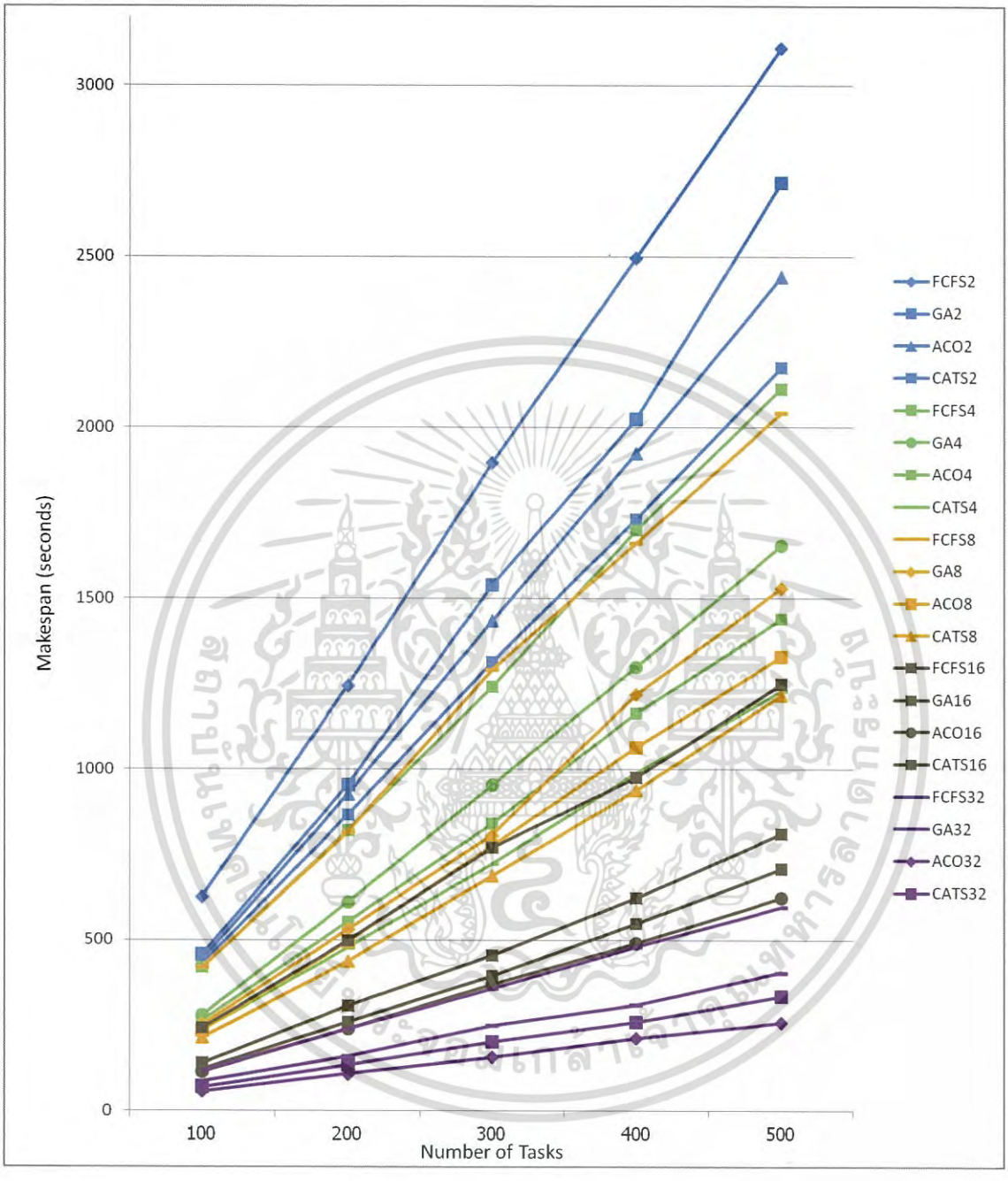


รูปที่ 5.9 ผลการเปรียบเทียบเมคสแปนของวิธี CATS กับวิธีอื่นที่ 32 VM

จากรูปที่ 5.9 แสดงค่าเมคสแปนกับจำนวนงานที่เพิ่มขึ้นในระบบประมวลผล โดยจากกราฟจะเรียงลำดับประสิทธิภาพในการลดเมคสแปนของแต่ละวิธีจากมากที่สุดไปยังน้อยที่สุดได้ดังนี้ ACO, CATS, GA และ FCFS ตามลำดับ

สังเกตได้ว่าวิธี CATS มีประสิทธิภาพในการลดเมคสแปนที่เหนือกว่าวิธี FCFS, GA แต่ยังมีประสิทธิภาพที่ด้อยกว่าวิธี ACO เมื่อเริ่มมีจำนวนของ VM ที่เพิ่มขึ้นเป็น 32 VM ส่งผลทำให้วิธีของ ACO ช่วยลดระยะเวลาของการประมวลผลของงานที่เข้ามาในระบบที่มีจำนวน 32 VM ตั้งแต่ 100, 200, 300, 400 และ 500 งาน ได้มากที่สุด

6) การเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นตั้งแต่ 2, 4, 8, 16, 32 VM



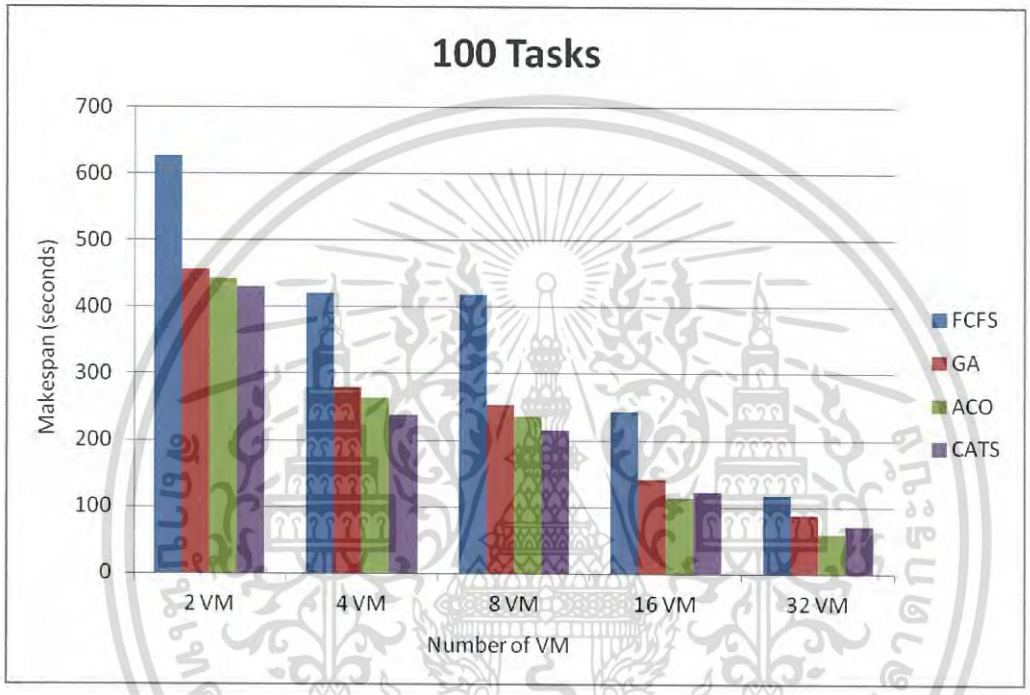
รูปที่ 5.10 ภาพรวมของผลการเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นในแกนจำนวนงาน

รูปที่ 5.10 จะแสดงสรุปภาพรวมของการเปรียบเทียบเมคสแปนของวิธี CATS และวิธีอื่นๆในแกนจำนวนงานที่ 100, 200, 300, 400, 500 เมื่อมีจำนวน VM เป็น 2, 4, 8, 16, 32

5.2.2 การเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นสำหรับ 100, 200, 300, 400, 500 งาน

ในการเปรียบเทียบเมคสแปนจะเทียบวิธี CATS กับวิธีอื่นเมื่อมีจำนวน VM เป็น 2, 4, 8, 16, 32 โดยจะประเมินประสิทธิภาพที่จำนวนงานเป็น 100, 200, 300, 400, 500 งาน โดยใช้พารามิเตอร์ต่างๆตามที่ได้กล่าวข้างต้น และเปรียบเทียบเมคสแปนในรูปแบบของกราฟ ดังแสดงในรูปที่ 5.11 ถึง 5.16

1) การเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นที่ 100 งาน

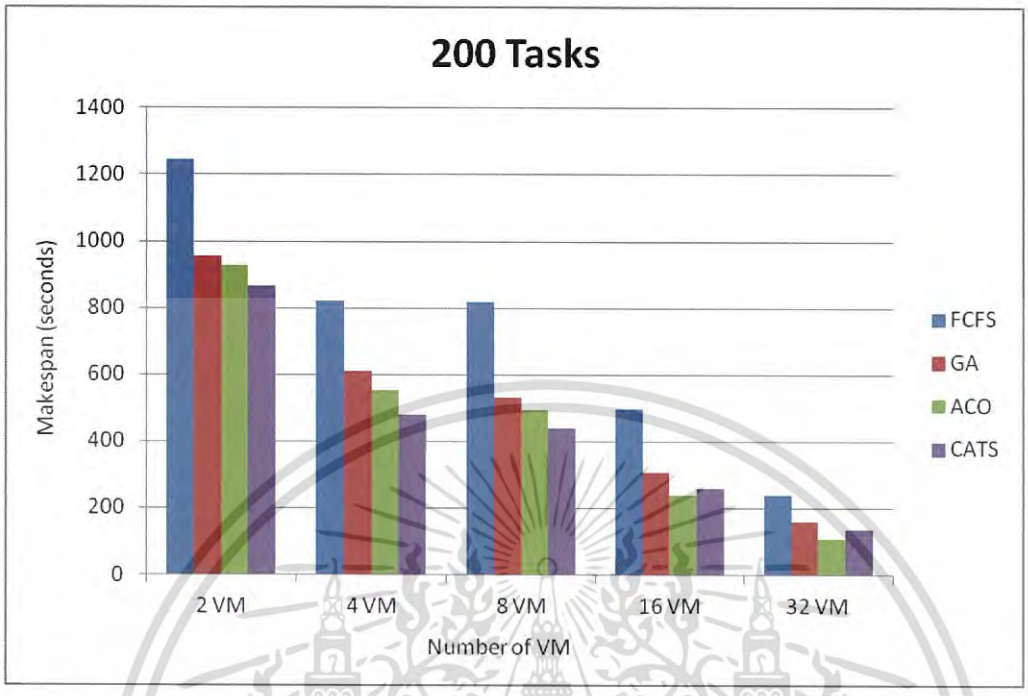


รูปที่ 5.11 ผลการเปรียบเทียบเมคสแปนของวิธี CATS กับวิธีอื่นที่ 100 งาน

จากรูปที่ 5.11 แสดงค่าเมคสแปนกับจำนวน VM ที่เพิ่มขึ้นในระบบประมวลผล โดยเมื่อมีจำนวน VM เพิ่มขึ้นจะช่วยลดระยะเวลาในการประมวลผลให้น้อยลงที่จำนวนงานเท่ากันซึ่งมีจำนวนงานเป็น 100 งาน จากกราฟจะเรียงลำดับประสิทธิภาพในการลดเมคสแปนของแต่ละวิธีจากมากที่สุดไปยังน้อยที่สุดเมื่อมีจำนวน VM เป็น 2, 4, 8 ได้ดังนี้ CATS, ACO, GA, และ FCFS ตามลำดับ และจำนวน VM ที่ 16, 32 จะให้เรียงลำดับประสิทธิภาพในการลดเมคสแปนจากมากไปน้อยเป็น ACO, CATS, GA, และ FCFS ตามลำดับ

แสดงให้เห็นว่าวิธี CATS มีประสิทธิภาพในการลดเมคสแปนที่เหนือกว่าวิธี FCFS, GA เมื่อมีจำนวน VM เป็น 2, 4, 8 แต่ยังมีประสิทธิภาพที่ด้อยกว่าวิธี ACO เมื่อเริ่มมีจำนวนของ VM ที่เพิ่มขึ้นเป็น 16, 32

2) การเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นที่ 200 งาน

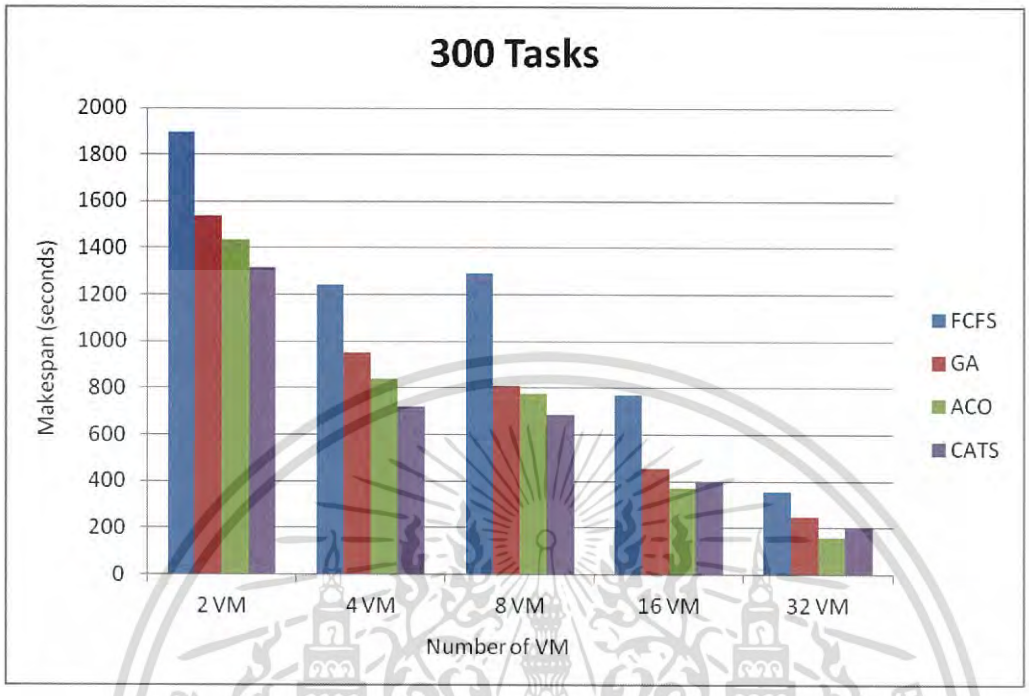


รูปที่ 5.12 ผลการเปรียบเทียบเมคสแปนของวิธี CATS กับวิธีอื่นที่ 200 งาน

จากรูปที่ 5.12 แสดงค่าเมคสแปนกับจำนวน VM ที่เพิ่มขึ้นในระบบประมวลผล โดยเมื่อมีจำนวน VM เพิ่มขึ้นจะช่วยลดระยะเวลาในการประมวลผลให้น้อยลงที่จำนวนงานเท่ากันซึ่งมีจำนวนงานเป็น 200 งาน จากกราฟจะเรียงลำดับประสิทธิภาพในการลดเมคสแปนของแต่ละวิธีจากมากที่สุดไปยังน้อยที่สุดเมื่อมีจำนวน VM เป็น 2, 4, 8 ได้ดังนี้ CATS, ACO, GA, และ FCFS ตามลำดับ และจำนวน VM ที่ 16, 32 จะให้เรียงลำดับประสิทธิภาพในการลดเมคสแปนจากมากไปน้อยเป็น ACO, CATS, GA, และ FCFS ตามลำดับ

แสดงให้เห็นว่าวิธี CATS มีประสิทธิภาพในการลดเมคสแปนที่เหนือกว่าวิธี FCFS, GA เมื่อมีจำนวน VM เป็น 2, 4, 8 แต่ยังมีประสิทธิภาพที่ด้อยกว่าวิธี ACO เมื่อเริ่มมีจำนวนของ VM ที่เพิ่มขึ้นเป็น 16, 32

3) การเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นที่ 300 งาน

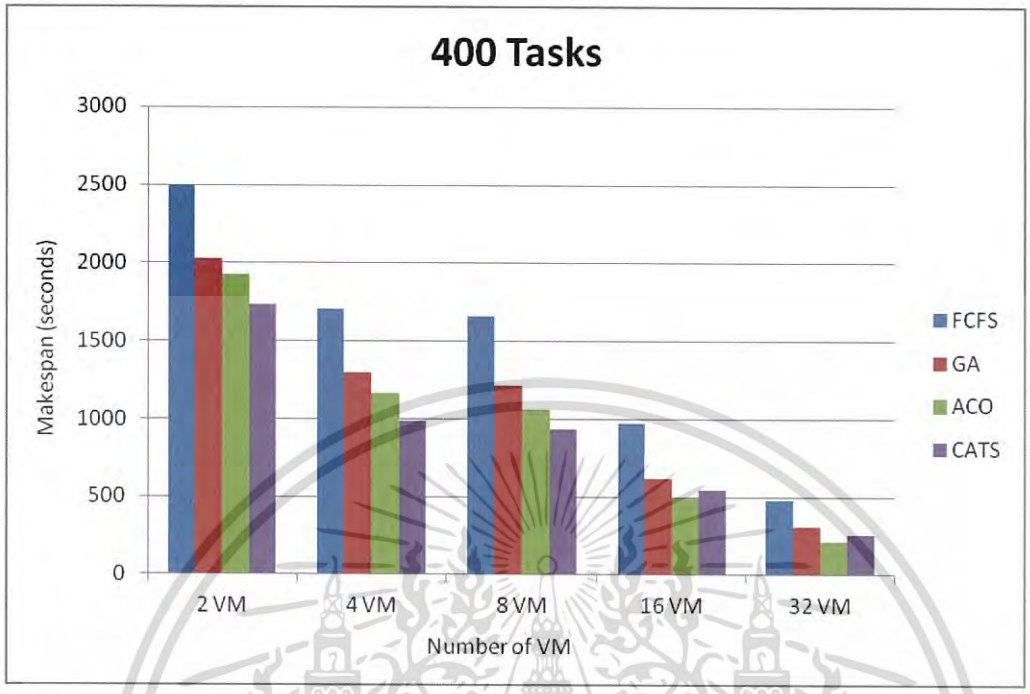


รูปที่ 5.13 ผลการเปรียบเทียบเมคสแปนของวิธี CATS กับวิธีอื่นที่ 300 งาน

จากรูปที่ 5.13 แสดงค่าเมคสแปนกับจำนวน VM ที่เพิ่มขึ้นในระบบประมวลผล โดยเมื่อมีจำนวน VM เพิ่มขึ้นจะช่วยลดระยะเวลาในการประมวลผลให้น้อยลงที่จำนวนงานเท่ากันซึ่งมีจำนวนงานเป็น 300 งาน จากกราฟจะเรียงลำดับประสิทธิภาพในการลดเมคสแปนของแต่ละวิธีจากมากที่สุดไปยังน้อยที่สุดเมื่อมีจำนวน VM เป็น 2, 4, 8 ได้ดังนี้ CATS, ACO, GA, และ FCFS ตามลำดับ และจำนวน VM ที่ 16, 32 จะให้เรียงลำดับประสิทธิภาพในการลดเมคสแปนจากมากไปน้อยเป็น ACO, CATS, GA, และ FCFS ตามลำดับ

แสดงให้เห็นว่าวิธี CATS มีประสิทธิภาพในการลดเมคสแปนที่เหนือกว่าวิธี FCFS, GA เมื่อมีจำนวน VM เป็น 2, 4, 8 แต่ยังมีประสิทธิภาพที่ด้อยกว่าวิธี ACO เมื่อเริ่มมีจำนวนของ VM ที่เพิ่มขึ้นเป็น 16, 32

4) การเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นที่ 400 งาน

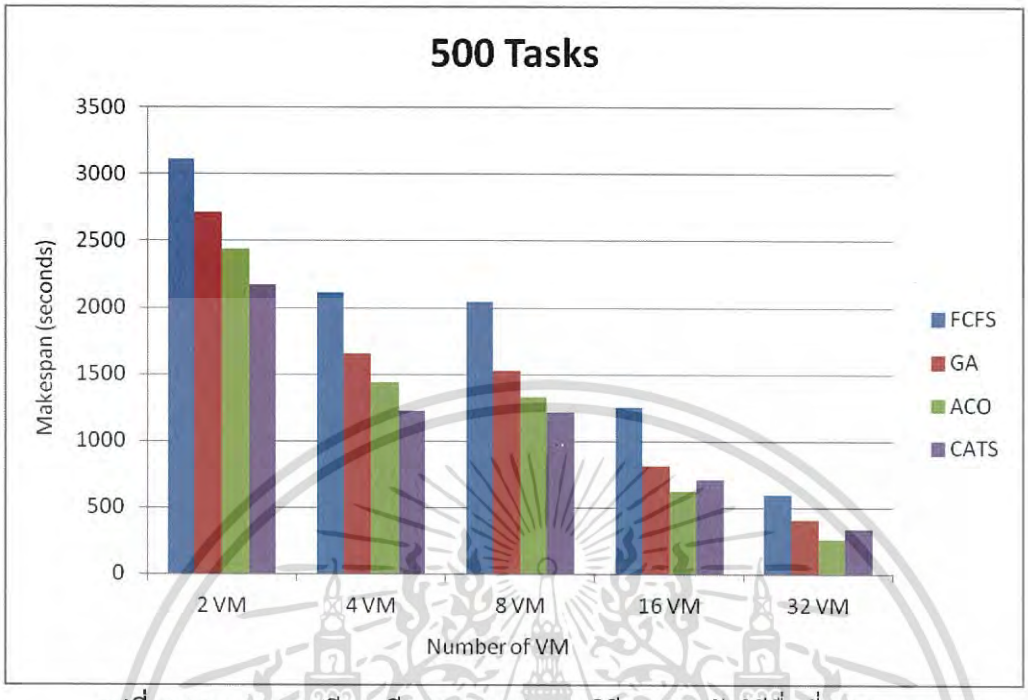


รูปที่ 5.14 ผลการเปรียบเทียบเมคสแปนของวิธี CATS กับวิธีอื่นที่ 400 งาน

จากรูปที่ 5.14 แสดงค่าเมคสแปนกับจำนวน VM ที่เพิ่มขึ้นในระบบประมวลผล โดยเมื่อมีจำนวน VM เพิ่มขึ้นจะช่วยลดระยะเวลาในการประมวลผลให้น้อยลงที่จำนวนงานเท่ากันซึ่งมีจำนวนงานเป็น 400 งาน จากกราฟจะเรียงลำดับประสิทธิภาพในการลดเมคสแปนของแต่ละวิธีจากมากที่สุดไปยังน้อยที่สุดเมื่อมีจำนวน VM เป็น 2, 4, 8 ได้ดังนี้ CATS, ACO, GA, และ FCFS ตามลำดับ และจำนวน VM ที่ 16, 32 จะให้เรียงลำดับประสิทธิภาพในการลดเมคสแปนจากมากไปน้อยเป็น ACO, CATS, GA, และ FCFS ตามลำดับ

แสดงให้เห็นว่าวิธี CATS มีประสิทธิภาพในการลดเมคสแปนที่เหนือกว่าวิธี FCFS, GA เมื่อมีจำนวน VM เป็น 2, 4, 8 แต่ยังมีประสิทธิภาพที่ด้อยกว่าวิธี ACO เมื่อเริ่มมีจำนวนของ VM ที่เพิ่มขึ้นเป็น 16, 32

5) การเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นที่ 500 งาน

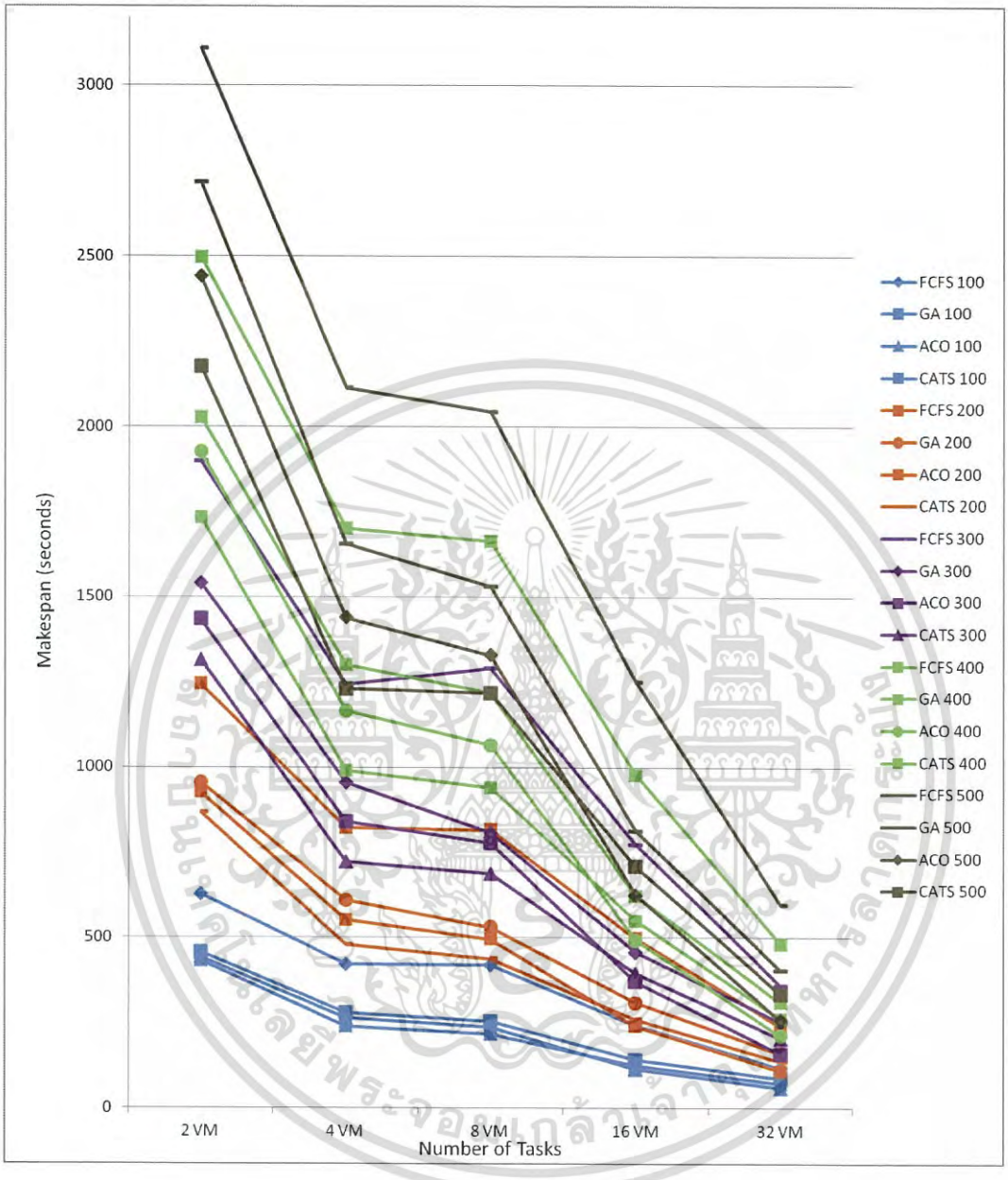


รูปที่ 5.15 ผลการเปรียบเทียบเมคสแปนของวิธี CATS กับวิธีอื่นที่ 500 งาน

จากรูปที่ 5.15 แสดงค่าเมคสแปนกับจำนวน VM ที่เพิ่มขึ้นในระบบประมวลผล โดยเมื่อมีจำนวน VM เพิ่มขึ้นจะช่วยลดระยะเวลาในการประมวลผลให้น้อยลงที่จำนวนงานเท่ากันซึ่งมีจำนวนงานเป็น 500 งาน จากกราฟจะเรียงลำดับประสิทธิภาพในการลดเมคสแปนของแต่ละวิธีจากมากที่สุดไปยังน้อยที่สุดเมื่อมีจำนวน VM เป็น 2, 4, 8 ได้ดังนี้ CATS, ACO, GA, และ FCFS ตามลำดับ และจำนวน VM ที่ 16, 32 จะให้เรียงลำดับประสิทธิภาพในการลดเมคสแปนจากมากไปน้อยเป็น ACO, CATS, GA, และ FCFS ตามลำดับ

แสดงให้เห็นว่าวิธี CATS มีประสิทธิภาพในการลดเมคสแปนที่เหนือกว่าวิธี FCFS, GA เมื่อมีจำนวน VM เป็น 2, 4, 8 แต่ยังมีประสิทธิภาพที่ด้อยกว่าวิธี ACO เมื่อเริ่มมีจำนวนของ VM ที่เพิ่มขึ้นเป็น 16, 32

6) การเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นตั้งแต่ 100, 200, 300, 400, 500 งาน



รูปที่ 5.16 ภาพรวมของผลการเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นในแกนจำนวน VM

รูปที่ 5.16 จะแสดงสรุปภาพรวมของการเปรียบเทียบเมคสแปนของวิธี CATS และวิธีอื่นๆในแกนจำนวน VM ที่ 2, 4, 8, 16, 32 เมื่อมีจำนวน VM เป็น 100, 200, 300, 400, 500

จากผลการประเมินประสิทธิภาพในข้อ 5.2.1 และ 5.2.2 แสดงให้เห็นว่าวิธี CATS ใช้เมคสแปนน้อยที่สุดเมื่อเทียบกับวิธีอื่น เมื่อมีจำนวน VM น้อย เป็นจำนวน 2, 4, 8 เครื่อง โดยเรียงลำดับเป็น CATS, ACO, GA และ FCFS ตามลำดับ และเมื่อจำนวน VM มากขึ้นเป็น 16, 32 เครื่อง วิธี CATS จะให้ผลลัพธ์ที่ด้อยกว่าวิธี ACO แต่ยังสามารถลดเมคสแปนได้น้อยกว่าวิธี GA ซึ่งการจำลองผลการประเมินประสิทธิภาพจะเป็นการกำหนดให้ความสามารถในการประมวลผลของ VM ไม่เท่ากัน

ดังนั้น จึงได้มีการเพิ่มการประเมินประสิทธิภาพเมื่อกำหนดให้ความสามารถในการประมวลผลของ VM (MIPS) เท่ากันทั้งหมด รายละเอียดการประเมินประสิทธิภาพจะอธิบายดังข้อที่ 5.2.3

5.2.3 การเปรียบเทียบประสิทธิภาพวิธี CATS กับวิธีอื่นเมื่อกำหนดให้ MIPS คงที่

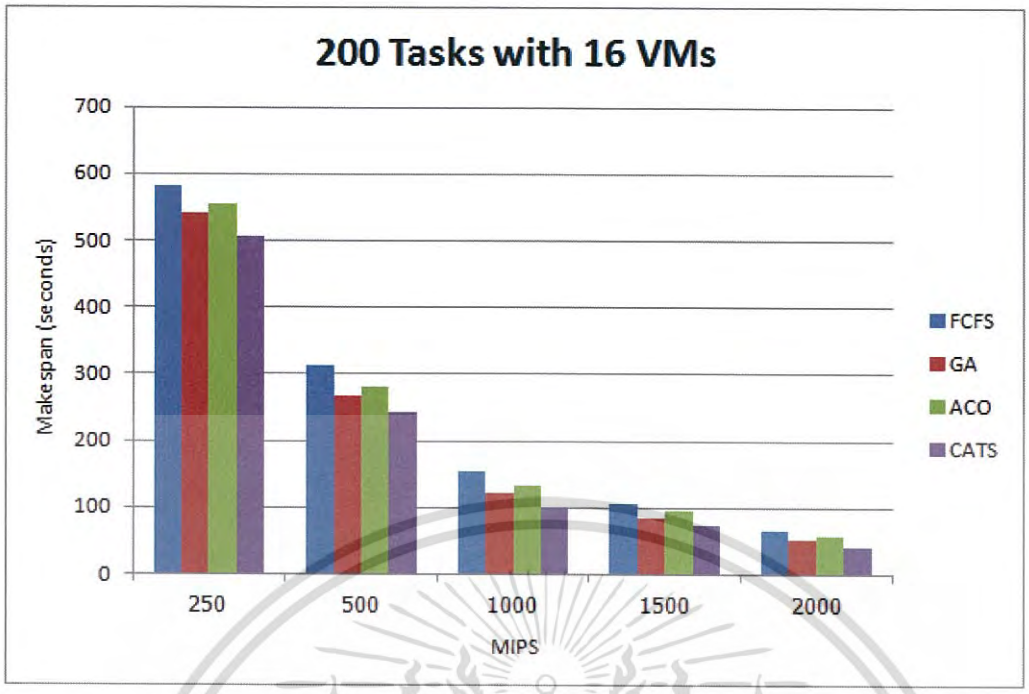
กำหนดให้มีความสามารถในการประมวลผลของคอมพิวเตอร์เสมือน (MIPS) มีค่าคงที่ โดยในการประเมินประสิทธิภาพจะเปรียบเทียบที่ 250, 500, 1000, 1500 และ 2000 MIPS ตามลำดับ ที่จำนวน 16 VM มีงานเข้ามาในระบบ 200 งานและ 500 งาน

พารามิเตอร์อื่นที่ใช้งานจะแสดงให้เห็นดังตารางที่ 5.13 โดยจะมีการสุ่มค่าพารามิเตอร์บางค่าตามช่วงที่กำหนด ได้แก่ จำนวนของโฮสต์ 2-6 โฮสต์ ความยาวของงาน 5000-15000 MI

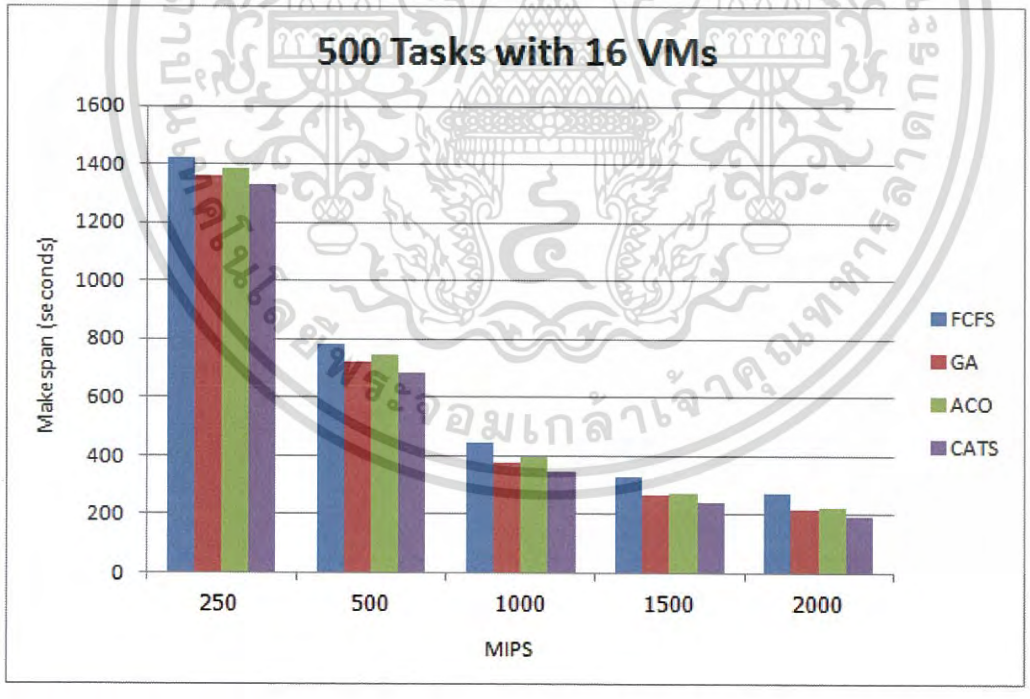
สำหรับพารามิเตอร์ของวิธี CATS, GA, ACO จะใช้ค่าเดียวกับการประเมินประสิทธิภาพในหัวข้อ 5.2.2 โดยผลที่ได้จะแสดงดังรูป 5.17 และ 5.18

ประเภท	พารามิเตอร์	ค่าที่ใช้
ดาต้าเซนเตอร์	จำนวนของดาต้าเซนเตอร์	10 ดาต้าเซนเตอร์
	จำนวนของโฮสต์	2-6 โฮสต์
	ประเภทการจัดการ	Space_shared
		Time_shared
คอมพิวเตอร์เสมือน (VM)	MIPS	250-2000 MIPS
	หน่วยความจำของ VM (RAM)	512-2048 Mbytes
	ประเภทการจัดการ	Space_shared
	จำนวน VM	8-16 VM
Task	จำนวนงานในหนึ่งเซต	200, 500 งาน
	ความยาวของงาน	5000-15000 MI

ตารางที่ 5.13 พารามิเตอร์ของการจำลองสถานการณ์สำหรับคลาวด์ซิมเมื่อ MIPS คงที่



รูปที่ 5.17 ผลการเปรียบเทียบเมคสแปนของวิธี CATS กับวิธีอื่นที่ 200 งาน เมื่อ MIPS คงที่



รูปที่ 5.18 ผลการเปรียบเทียบเมคสแปนของวิธี CATS กับวิธีอื่นที่ 500 งาน เมื่อ MIPS คงที่

จากรูปที่ 5.17 และ 5.18 แสดงให้เห็นว่าวิธี CATS จะทำให้ได้ค่าเมคสแปนน้อยที่สุด ตามด้วย GA, ACO, FCFS ตามลำดับเมื่อมีการกำหนดให้มีความสามารถในการประมวลผลของ VM เท่ากันทั้งหมด นั่นคือ วิธี CATS จะทำงานได้ดีที่สุดในกรณีที่มีความสามารถในการประมวลผลของ VM เท่ากัน

ในกรณีที่ความสามารถในการประมวลผลของ VM ไม่เท่ากัน จะทำให้วิธี CATS ให้เมคสแปนที่น้อยกว่าวิธีอื่นในกรณีที่มีจำนวน VM น้อยที่เป็น 2, 4, 8 เครื่อง และเมื่อจำนวน VM เพิ่มเป็น 16, 32 วิธี ACO จะให้เมคสแปนที่น้อยที่สุด ตามด้วย CATS, GA, FCFS ตามลำดับ แสดงให้เห็นว่าวิธี CATS และ GA จะหาคำตอบได้ดีกว่าวิธี ACO เมื่อความแตกต่างของความสามารถในการประมวลผลของ VM น้อยกว่า ดังผลการประเมินประสิทธิภาพ

เมื่อพิจารณาผลการประเมินประสิทธิภาพทั้งหมดจะเห็นว่าวิธี CATS จะให้ผลลัพธ์ที่ดีกว่า GA ในทุกๆกรณี โดยนำ CA มาใช้ร่วมกับ GA ในวิธี CATS จะสามารถทำให้ค้นหาคำตอบได้ดีกว่าการใช้ GA เพียงอย่างเดียว เนื่องจากความสามารถในการจัดการตนเอง (Self-Organization) ของ CA ที่เกิดจากการเปลี่ยนสถานะของเซลล์ใน CA จากกฎที่ดีที่สุดที่ค้นหาได้จาก GA และยังเป็นการใช้ประโยชน์จากการ Mutation ที่ช่วยหลีกเลี่ยงปัญหาการลู่เข้าหาคำตอบที่เร็วเกินไป (Premature Convergence) ที่อาจเกิดขึ้นกับการใช้ GA เพียงอย่างเดียว ดังนั้นการนำ CA มาใช้ร่วมกับ GA ในวิธี CATS จะเป็นการเพิ่มโอกาสในการพบคำตอบที่ดีกว่า

บทที่ 6
สรุปผลการวิจัยและข้อเสนอแนะ

ในบทนี้เรากล่าวสรุปผลงานวิจัยที่ได้นำเสนอพร้อมทั้งปัญหาต่างๆที่พบระหว่างการจำลองการทำงานและแนวทางการปรับปรุงงานวิจัยในอนาคต

6.1 สรุปผลการวิจัยที่นำเสนอ

จากผลการประเมินประสิทธิภาพในบทที่ 5 พบว่าทั้งวิธี GA, ACO, CATS สามารถลดเมคสแปนของงานที่เข้ามาในระบบในทุกกรณีเมื่อเทียบกับวิธีแบบดั้งเดิมที่ใช้ในระบบการประมวลผลแบบกลุ่มเมฆ นั่นคือวิธี FCFS

วิธี CATS ที่ได้พัฒนามาจากวิธีเจเนติกอัลกอริทึมสามารถทำงานได้ดีกว่าวิธีเจเนติกอัลกอริทึมในทุกๆกรณี ตั้งแต่จำนวน VM ที่ 2, 4, 8, 16, 32 และเมื่อมีงานเข้ามาในระบบ 100, 200, 300, 400, 500 งาน ซึ่งแสดงให้เห็นว่าการนำเซลลูลาร์ออโตมาตามาใช้ร่วมกับเจเนติกอัลกอริทึมจะช่วยเพิ่มประสิทธิภาพของเจเนติกอัลกอริทึมให้สามารถค้นหาคำตอบของปัญหาที่ดีกว่าซึ่งผลลัพธ์ที่ได้จากการประเมินประสิทธิภาพจะแตกต่างกันอย่างมีนัยสำคัญ

ในกรณีที่ความสามารถในการประมวลผลของ VM แตกต่างกัน เมื่อเทียบประสิทธิภาพของวิธี CATS และวิธี ACO พบว่าวิธี CATS จะทำงานได้ดีกว่าเมื่อมีจำนวนทรัพยากรหรือ VM เป็นจำนวนน้อย โดยในการประเมินประสิทธิภาพจะแสดงให้เห็นว่าวิธี CATS ทำงานได้ดีกว่าวิธี ACO เมื่อมีจำนวน VM เป็น 2, 4, 8 เครื่อง ในทุกๆจำนวนงานที่เข้ามาในระบบ และจะด้อยกว่าวิธี ACO เมื่อจำนวน VM เป็น 16 และ 32 เป็นต้นไป

สำหรับกรณีที่ความสามารถในการประมวลผลของ VM เท่ากัน จะพบว่าวิธี CATS สามารถให้ผลลัพธ์ที่ดีกว่า GA, ACO, FCFS ในทุกๆกรณี เมื่อกำหนดให้จำนวน VM เป็น 16 และงานที่เข้ามาในระบบเป็น 200 และ 500 งาน

ดังนั้นจึงสรุปได้ว่าการนำเซลลูลาร์ออโตมาตามาพัฒนาร่วมกับเจเนติกอัลกอริทึมสามารถนำมาประยุกต์ใช้งานกับการตารางงานบนระบบการประมวลผลแบบกลุ่มเมฆได้ และให้ผลลัพธ์ที่ดีขึ้นเมื่อวัดประสิทธิภาพเทียบกับวิธีเจเนติกอัลกอริทึมแบบดั้งเดิม

นอกจากนี้วิธี CATS ยังให้ผลลัพธ์ที่ดีที่สุดเมื่อเทียบกับวิธีอื่นเมื่อมีจำนวน VM น้อยและมีความสามารถในการประมวลผลของ VM ต่างกัน จึงเหมาะกับระบบประมวลผลแบบกลุ่มเมฆที่มีขนาดเล็กและมีทรัพยากรจำกัด ซึ่งจะช่วยลดเมคสแปนของงานที่เข้ามาในระบบได้อย่างมีประสิทธิภาพ แต่ถ้าระบบมีความสามารถในการประมวลผลของ VM เท่ากัน CATS จะให้ผลลัพธ์ที่ดีที่สุดในทุกกรณี

6.2 ปัญหาและอุปสรรค

ปัญหาที่พบ คือ เมื่อมีการเปลี่ยนเวอร์ชั่นของคลาวด์ซิมจากเวอร์ชั่น 3.0.1 เป็น 3.0.3 เนื่องจากมีการปรับปรุงความเร็วของการใช้งานเพื่อให้ทำงานได้เร็วขึ้น แต่ฟังก์ชั่นในการทำงานหลายส่วนมีการเปลี่ยนแปลงเพื่อให้สามารถใช้งานได้ตามปกติ จึงต้องทำการปรับโปรแกรมที่เขียนขึ้นมาใหม่ในส่วนที่เกี่ยวข้องกับคลาวด์ซิมเกือบทั้งหมดเนื่อง อีกทั้งต้องทดสอบความถูกต้องของการทำงานในทุกส่วน ทำให้เป็นการเพิ่มระยะเวลาในการทำงานเพิ่มขึ้นมาก

6.3 แนวทางการปรับปรุงงานวิจัยในอนาคต

งานวิจัยนี้ได้แสดงให้เห็นว่าวิธี CATS สามารถนำมาใช้งานร่วมกับเจเนติกอัลกอริทึมและประยุกต์ใช้งานบนระบบการประมวลผลแบบกลุ่มเมฆได้ โดยปรับปรุงจากการใช้งานเพียงสองเครื่องในระบบมัลติโพรเซสเซอร์ให้รองรับการทำงานที่มากกว่าสองเครื่องได้ แต่จำนวน VM ที่ใช้จะต้องเป็นจำนวนที่เท่ากันกับ 2^n โดย n คือจำนวนเซลล์ในเซลลูลาร์ออโตมาตาที่ใช้แสดงถึง VM หนึ่งเครื่อง หากสามารถพัฒนาวิธีการ CATS เพื่อให้รองรับจำนวน VM ได้ที่จำนวนใดๆจะเป็นการเพิ่มความยืดหยุ่นในการใช้วิธี CAT ได้มากยิ่งขึ้น

เอกสารอ้างอิง

[1] M. D. Dikaiakos, "Cloud Computing: Distributed Internet Computing for IT and Scientific Research," Internet Computing, IEEE , vol.13, no.5, pp.10,13, Sept.-Oct. 2009.

[2] R. Buyya, J. Broberg , and A. M. Goscinski, Cloud Computing: Principles and Paradigms (Wiley Series on Parallel and Distributed Computing). Wiley, 2011.

[3] F. Chang and R. Viswanathan, "Optimal Resource Allocation in Clouds," Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on , pp.418,425, 5-10 July 2010.

[4] A. Swiecicka and F. Seredynski, "Applying cellular automata in multiprocessor scheduling," Parallel Computing in Electrical Engineering, 2002. PARELEC '02. Proceedings International Conference on , pp.177,182, 2002.

[5] F. Seredynski and P. Bouvry, "Multiprocessor scheduling algorithms based on cellular automata training," Intelligent Engineering Systems Through Artificial Neural Networks, vol. 13, pp. 255-260, 2003

[6] D. Rajarshi , M. Melanie, and J. P. Crutchfield, "A Genetic Algorithm Discovers Particle-Based Computation in Cellular Automata," Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature, p.344-353, October 09-14, 1994.

[7] S. Wolfram, Cellular Automata And Complexity: Collected Papers. Westview Press, 1994.

[8] J. L. Schiff, Cellular Automata: A Discrete View of the World. Wiley-Interscience, 2008.

[9] D. E. Goldberg, Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Professional, 1989.

[10] R. N. Calheiros, R. Ranjan, and R. Buyya, "CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services," Technical Report, GRIDS-TR-2009-1, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, 2009.

[11] R. N. Calheiros, R. Ranjan, A. Beloglazov, and R. Buyya, "CloudSim: A Toolkit for the Modeling and Simulation of Cloud Resource Management and Application Provisioning Techniques," Journal Software—Practice & Experience archive, vol. 41, pp. 23-50, 2011.

[12] Hou, E. S H; Hong, R.; Ansari, N., "Efficient multiprocessor scheduling based on genetic algorithms," Industrial Electronics Society, 1990. IECON '90., 16th Annual Conference of IEEE , vol., no., pp.1239,1243 vol.2, 27-30 Nov 1990.

[13] Di Martino, V.; Mililotti, M., "Scheduling in a grid computing environment using genetic algorithms," Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM , vol., no., pp.5 pp,, 15-19 April 2001.

[14] Tawfeek, M.A; El-Sisi, A; Keshk, AE.; Torkey, F.A, "Cloud task scheduling based on ant colony optimization," Computer Engineering & Systems (ICCES), 2013 8th International Conference on , vol., no., pp.64,69, 26-28 Nov. 2013.

[15] Aggarwal, M.; Kent, R.D.; Ngom, A, "Genetic algorithm based scheduler for computational grids," High Performance Computing Systems and Applications, 2005. HPCS 2005. 19th International Symposium on , vol., no., pp.209,215, 15-18 May 2005.

[16] Sung-Ho Woo; Sung-Bong Yang; Shin-Dug Kim; Tack-Don Han, "Task scheduling in distributed computing systems with a genetic algorithm," High Performance Computing on the Information Superhighway, 1997. HPC Asia '97 , vol., no., pp.301,305, 28 Apr-2 May 1997.

[17] Monnier, Y.; BEAUVAIS, Jean-Pime; Deplanche, Anne-Marie, "A genetic algorithm for scheduling tasks in a real-time distributed system," Euromicro Conference, 1998. Proceedings. 24th , vol.2, no., pp.708,714 vol.2, 25-27 Aug 1998.

[18] Pinal Salot, "A Survey of Various Scheduling Algorithm in Cloud Computing Environment", IJRET: International Journal of Research in Engineering and Technology, vol.2, Feb 2013.

[19] Blum, Christian, and Andrea Roli. "Metaheuristics in combinatorial optimization: Overview and conceptual comparison." ACM Computing Surveys (CSUR) 35.3 (2003): 268-308.

[20] Dorigo, Marco, Vittorio Maniezzo, and Alberto Colorni. "Ant system: optimization by a colony of cooperating agents." IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26.1 (1996): 29-41.

[21] Yang, Xin-She. Nature-inspired metaheuristic algorithms. Luniver press, 2010.

[22] Shiffman, Daniel, Shannon Fry, and Zannah Marsh. The nature of code. D. Shiffman, 2012.

[23] Ganguly, Niloy, et al. "A survey on cellular automata." (2003).

[24] Wolfram, Stephen. Cellular automata and complexity: collected papers. Vol. 1. Reading, MA: Addison-Wesley, 1994.

[25] Berlekamp, Elwyn R., John Horton Conway, and Richard K. Guy. Winning ways for your mathematical plays. Vol. 3. Natick: AK Peters, 2003.

[26] Umeo, Hiroshi. "Firing squad synchronization problem in cellular automata." Encyclopedia of Complexity and Systems Science. Springer New York, 2009. 3537-3574.

[27] Beckers, Andreas, and Thomas Worsch. "A perimeter–time CA for the queen bee problem." Parallel Computing 27.5 (2001): 555-569.

[28] Schweitzer, Frank, Laxmidhar Behera, and Heinz Mühlenbein. "Evolution of cooperation in a spatial prisoner's dilemma." Advances in Complex systems 5.02n03 (2002): 269-299.

[29] Toffoli, Tommaso. "Cellular automata as an alternative to (rather than an approximation of) differential equations in modeling physics." Physica D: Nonlinear Phenomena 10.1-2 (1984): 117-127.

[30] Chopard, Bastien, Michael Droz, and Max Kolb. "Cellular automata approach to non-equilibrium diffusion and gradient percolation." Journal of Physics A: Mathematical and General 22.10 (1989): 1609.

[31] Packard, Norman H. "Lattice models for solidification and aggregation." First International Symosium for Science on Form (1986).

[32] De Boer, Rob J., Pauline Hogeweg, and Alan S. Perelson. "Growth and recruitment in the immune network." Theoretical and Experimental Insights into Immunology. Springer Berlin Heidelberg, 1992. 223-247.

[33] Dos Santos, Rita Maria Zorzenon, and Sérgio Coutinho. "Dynamics of HIV infection: A cellular automata approach." Physical review letters 87.16 (2001): 168102.

[34] Moreira, Joana, and Andreas Deutsch. "Cellular automaton models of tumor development: a critical review." Advances in Complex Systems 5.02n03 (2002): 247-267.

[35] Moore, J. H., and L. W. Hahn. "A cellular automata-based pattern recognition approach to identifying gene-gene and gene-environment interactions." American Journal of Human Genetics. Vol. 67. No. 4. 5720 SOUTH WOODLAWN AVE, CHICAGO, IL 60637-1603 USA: UNIV CHICAGO PRESS, 2000.

[36] Schönfisch, Birgitt, and Michael Kinder. "A fish migration model." International Conference on Cellular Automata. Springer Berlin Heidelberg, 2002.

[37] H. Baltzer, W. P. Braun, and W. Kohler. "Cellular Automata Model for Vegetable Dynamics. Ecological Modelling", 107:113–125, 1998

[38] Resnick, Mitchel. Turtles, termites, and traffic jams: Explorations in massively parallel microworlds. Mit Press, 1997.

[39] Sakoda, James M. "The checkerboard model of social interaction." The Journal of Mathematical Sociology 1.1 (1971): 119-132.

[40] Keenan, Donald C., and Mike J. O'Brien. "Competition, collusion, and chaos." Journal of Economic Dynamics and Control 17.3 (1993): 327-353.

[41] Axelrod, Robert M. The evolution of cooperation. Basic books, 2006.

[42] S. Chattopadhyay, S. Adhikari, S. Sengupta, and M. Pal. "Highly Regular, Modular, and Cascadable Design of Cellular Automata-based Pattern Classifier." IEEE Transaction on VLSI Systems, 8(6):724–735, December 2000

[43] S. Chattopadhyay and P. Pal Chaudhuri. "Efficient Signatures of Boolean Functions for Rapid Matching in Antifuse based FPGA Technology Mapping." In International Conference on Computer Systems and Education, Bangalore, India, June 1994

[44] Nandi, S., B. K. Kar, and P. Pal Chaudhuri. "Theory and applications of cellular automata in cryptography." IEEE Transactions on computers 43.12 (1994): 1346-1357.

[45] Seredynski, Franciszek, Pascal Bouvry, and Albert Y. Zomaya. "Cellular automata computations and secret key cryptography." Parallel Computing 30.5 (2004): 753-766.

[46] Wolfram, Stephen. "Random sequence generation by cellular automata." Advances in applied mathematics 7.2 (1986): 123-169.

[47] Meier, Willi, and Othmar Staffelbach. "Analysis of pseudo random sequences generated by cellular automata." Workshop on the Theory and Application of of Cryptographic Techniques. Springer Berlin Heidelberg, 1991.

[48] S. Wolfram. Universality and Complexity in Cellular Automata. Physica D, 10:1–35, 1984

[49] P. D. Hortencius, R. D McLeod, W. Pries, D. M. Miller, and H. C. Card. "Cellular Automata based Pseudo-random Number Generators for Built-in Self-Test." IEEE Trans. on CAD, 8(8):842–859, August 1989.

[50] Popovici, Adriana, and Dan Popovici. "Cellular automata in image processing." Fifteenth International Symposium on Mathematical Theory of Networks and Systems. Vol. 1. 2002.

[51] Rosin, Paul L. "Training cellular automata for image processing." IEEE transactions on image processing 15.7 (2006): 2076-2087.

[52] Cunha, Renan O., et al. "Simulating large wireless sensor networks using cellular automata." Proceedings of the 38th annual Symposium on Simulation. IEEE Computer Society, 2005.

[53] Banerjee, Indrajit, et al. "Effective fault detection and routing scheme for wireless sensor networks." Computers & Electrical Engineering 40.2 (2014): 291-306.

[54] Schurgers, Curt, and Mani B. Srivastava. "Energy efficient routing in wireless sensor networks." Military communications conference, 2001. MILCOM 2001. Communications for network-centric operations: Creating the information force. IEEE. Vol. 1. IEEE, 2001.

[55] Tonguz, Ozan K., Nawaporn Wisitpongphan, and Fan Bai. "DV-CAST: A distributed vehicular broadcast protocol for vehicular ad hoc networks." IEEE Wireless Communications 17.2 (2010).

[56] Subrata, Riky, and Albert Y. Zomaya. "Evolving cellular automata for location management in mobile computing networks." IEEE Transactions on parallel and distributed systems 14.1 (2003): 13-26.

[57] Bandini, Stefania, Giancarlo Mauri, and Roberto Serra. "Cellular automata: From a theoretical parallel computational model to its application to complex systems." Parallel Computing 27.5 (2001): 539-553.

[58] Cannataro, Mario, et al. "A parallel cellular automata environment on multicomputers for computational science." Parallel Computing 21.5 (1995): 803-823.

ภาคผนวก

ผลงานวิจัยที่ได้รับการตีพิมพ์

International Conference on Engineering, Applied Sciences, and Technology
August 21-24, 2013, The Sukosol, Bangkok, Thailand

Evolving of Cellular Automata for Task Scheduling in Cloud Computing Systems

Ingkwan Vashirashudej
Department of Computer Engineering,
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
s5611513@kmitl.ac.th

Sakchai Thipchaksurat
Department of Computer Engineering,
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
ktsakcha@kmitl.ac.th

Abstract— **Task scheduling is one of the major challenging issue in cloud computing system, which aims to assign tasks to virtual machines (VMs) and minimizes the total execution time of tasks. In this paper, we propose the evolving of cellular automata to the task scheduling (CATS) in cloud computing system. In CATS scheme, the cellular automata (CA) is used for setting the task scheduling rules. The effective rule can be discovered from genetic algorithm. We evaluate the effectiveness of CATS scheme by means of the simulation. The performance metric is compared in the terms of makespan with the traditional First-Come-First-Served (FCFS) scheme. The simulation result shows that the CATS scheme can provide the lower makespan than that of FCFS scheme.**

Keywords-task scheduling; cloud computing system; virtual machines; cellular automata; genetic algorithms;

I. Introduction

Cloud computing system is now very popular and promising system in information technology (IT) that moves computing and data away from computers into a large data centers. It refers to applications which are delivered as services over the Internet and the actual cloud infrastructure. Cloud-service clients can be done some functions by themselves such as adding more capacity at peak demand, experiment with new services, and removing unneeded capacity [1]. The benefits of cloud computing is that the cost for deploying new business or technology capabilities and increasing the economic efficiency by developing the expensive infrastructure can be reduced. Cloud computing deployment model comes in three forms: public clouds, private clouds, and hybrid clouds. A public cloud is one in which the services and infrastructure are provided off-site over the Internet whereas the services and infrastructure of a private cloud are maintained in a private network and it is designed for restricted access to a single enterprise (or extended enterprise). Therefore, it provides a higher level of security and control. A hybrid cloud is a composition of two or more private and public clouds to perform distinct functions within the same organization. According to the abstraction level of the capability provided and the service model of providers, the services of cloud computing system can be divided into three classes such as the Infrastructure-as-a-Service (IaaS), the Platform-as-a-Service (Paas), and the Software-as-a-Service (Saas) [2].

The characteristics of cloud computing are the virtualization, distribution and dynamically scalability. Virtualization is the main characteristic, with the support of virtualization technology in cloud computing, a cloud is built up of numerous physical machines which can run multiple virtual machines (VMs) at the same time. This is presented to the application layer of the system e.g. as Amazon Elastic Compute Cloud (EC2) [3]. There are several issues in cloud computing and task scheduling is one of the major issues. In cloud computing platforms, the traditional task scheduling algorithm is First-Come-First-Served (FCFS) scheme but it is not optimal and cannot utilize resources in parallel.

However, it is very difficult to manually assign tasks to computing resources in clouds. Therefore, we need an efficient algorithm for task scheduling in order to match suitable resources for tasks better. The scheduling problem can be classified as a NP-hard optimization problem.

We have studied the several task scheduling techniques. We find that cellular automata (CA) can be applied for task scheduling. The authors in [5] and [6] applied CA for multiprocessor scheduling. However, CA has not been applied for task scheduling in cloud computing system. This encourages us to apply CA for task scheduling in cloud computing.

In this paper, we present the evolving of cellular automata for task scheduling in cloud computing system. Our proposed is calls CATS scheme. In CATS scheme, we use cellular automata for setting the task scheduling rules. Then, the genetic algorithm is used for discovering the effective task scheduling rules. The main objective of this paper is to minimize the total execution time.

The rest of this paper is organized as following: Section II reviews related works, and then in Section III introduces the CloudSim toolkit. Section IV presents our proposed scheduling algorithm in denils and how it works. Section V

presents the simulation results. Finally, Section VI concludes this paper.

II. Related Works

The authors in [4] and [5] introduced a cellular automata-based scheduler to find an allocation which minimizes the total makespan of parallel tasks in the two-processor system. Cellular automata (CA) are interesting because complex global behavior arises from simple local interactions and CA can be applied for solving the scheduling problem according to some scheduling rules which must be found. Effective rules for CA are discovered by a genetic algorithm (GA).

CA is a discrete-time system that can be used for modeling several phenomena in the physical systems or natural systems. It is formed by a collection of cells also called lattice and each cell has its own state. The number of possible states in a cell depends on the value of k. For example, if k is equal to 2, the possible value for each cell can has the value of 0 or 1. The state of the cell i at time $t+1$ depends on the states of its neighborhood with radius r and itself at time t. Over time, the cells can change from state to state and the CA's rules will determine how the states change [6] [7].

The simplest CA is the one-dimensional binary cellular automata which the state of each cell i in a lattice can be either 0 or 1 ($k = 2$). For example, assuming that $r = 1$ with the transition rule 30 (00011110). There are two neighborhoods for each cell and it can be expressed in 8 (2^3) different states [8]. All possible neighborhood states, transition function, and the evolution of CA are illustrated in Fig.1.

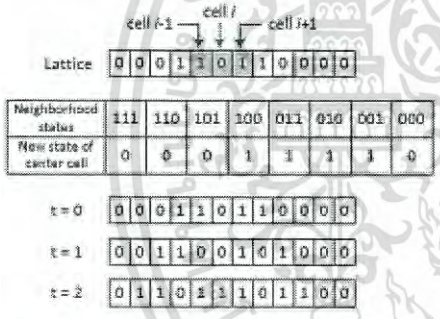


Figure 1. Example of rule 30 of one-dimensional binary cellular automata with $r = 1$

In cloud computing systems, the traditional scheme for task scheduling is First-Come-First Served (FCFS). FCFS scheme provides a simple process scheduling algorithm that saves valuable CPU resources. It uses nonpreemptive scheduling which a task is automatically queued and processed according to the arrival of task without other biases or preferences. But FCFS scheme is not an optimal solution for task scheduling. The authors in [4] and [5] used CA for task scheduling in two-processor system since the state of a cell of CA can represent as an allocation of a task to the specific processor and used GA for discovering the effective rules of CA which give the optimal or suboptimal makespan for task scheduling. This CA-based scheduler is divided into two phases: learning phase and operation phase. In learning phase, GA is applied to discover optimal rules of CA suitable for solving instances of a scheduling problem. In operation phase, discovered rules of CA are able to find an optimal solution of the scheduling problem for any initial allocation of a task set in two-processor system. Since there is no implementation of CA for task scheduling in cloud computing systems, this motivates us to apply CA in our work.

However, there are a large number of VMs in cloud computing systems and consequently, it can cause substantial complexity gain from the length of the rule of CA with increasing number of processor k and neighborhood radius r. When considering scheduling problem in the case of more than two processors, the length of rule grows rapidly with k and r. It is calculated as k^{2r+1} as shown in Table I with different k when $r = 1$ so it is difficult for GA to search in large spaces. This problem also becomes a subject of our work to make it works efficiency in cloud computing systems.

TABLE I. Lengths of rules for different k

k	The length of a rule
2	8
3	27
4	64
5	125
10	1000
20	8000
30	27000
50	125000

III. CloudSim Toolkit

A. Characteristic and Modeling

CloudSim is a toolkit for simulation and experimentation of cloud computing scenarios and it also supports modeling of large scale cloud computing infrastructures. CloudSim provides basic classes for describing data centers, virtual machines, service brokers, applications, users, computational resources, and policies for management (e.g., scheduling and allocation) [9].

The Datacenter is the core hardware infrastructure services related to the clouds. It is composed of a set of hosts, which are responsible for managing VMs by assigning a pre-configured processing capability, memory, storage, and a

provisioning policy for allocating processing cores to VMs. A host represents a physical computing node which has one or more VMs inside. There are two steps in allocation of cloud resources: VM provisioning and Application provisioning. The allocation policy can be managed by assigning specific CPU cores to specific VMs called space-shared policy or to dynamically distribute the capacity of a core among VMs called time-shared policy, and to assign cores to VMs on demand, or to specify other policies [10] [11].

B. *Design and Implementation of CloudSim*

CloudSim does not satisfy all the requests. CloudSim users are required to extend the specific classes (or entities) at user code level [12]. The main classes in CloudSim related to our work are listed as follows:

1) Cloud Information Service (CIS) class: The CIS is the core of simulate scheduling since it maps user or broker requests to suitable cloud providers. It also provides resource registration such as datacenter and broker.

2) Datacenter class: It is composed of a set of hosts which is resposible for managing VMs. Datacenter behaves like an IaaS provider by receiving requests for VMs from brokers and creating the VMs in hosts.

3) DatacenterBroker class: This class represents a broker acting on behalf of a user. It modifies two machanisms: the machanism for submitting VM provisioning requests to data centers and the mechanism for submitting the tasks to VMs. The CloudSim users have to extend this class for conducting experiments with their own policies.

4) VirtualMachine class: It represents a software implementation of a machine that executes applications called virtual machine (VM) which works like a physical machine. Each virtual machine divides the resources received from the host among tasks running on it.

5) Cloudlet class: A cloudlet class is also known as a task. CloudSim represents the complexity of an application in terms of its computational requirements. This class is managed by the scheduling policy which is implemented in DatacenterBroker Class.

To develop and evaluate our scheduling scheme, we extend and implement CATS scheme in DatacenterBroker class since it is responsible for managing task scheduling and resource allocation in cloud computing systems. The CloudSim simulation data flow is shown in Fig.2. When the simulation starts, each datacenter registers itself with the CIS and handle requests from a broker, all VMs are created afterwards. When a new task set arrives into the system, DatacenterBroker sends requests to all datacenters for scheduling the arrival tasks as the scheduling policy implemented in DatacenterBroker [13].

IV. Our Proposed Scheme

In this section, we describe our task scheduling scheme called the evolving of cellular automata for task scheduling in cloud computing systems or CATS scheme. To implement it for a large number of virtual machines (VMs) in cloud computing, we design our scheduler working with multiple sets of VMs and it divides a task set and distribute to the sets of VMs.

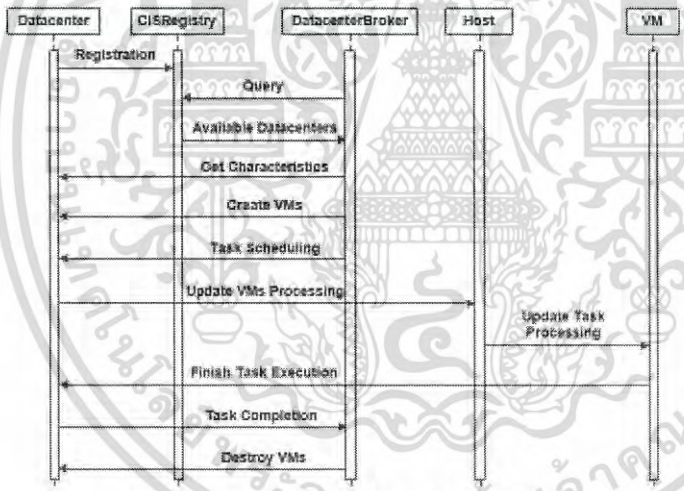


Figure 2. The flow diagram of communication among core entities in CloudSim

For example, there are 10 VMs with a task set of 200 tasks, all VMs are divided into 10 sets with two VMs in each set and the number of tasks in each VM set is 20 tasks. The scheduler has to work for 10 times to find the optimal allocation of a task set.

In addition, we assume that a nonlinear structure of a task set is approximated by a one-dimensional CA which each cell has two possible states 0 or 1. Our automaton is a binary one ($k = 2$). The state 0 or 1 of a cell means that a corresponding task is allocated either in the VM0 or VM1 in a VM set. With each task of a task set, an elementary cell is associated. The CA has null boundary conditions. We assume that "absent cells" are always in state 0.

The CA corresponding to a task set evolves according to its rule. Initial states of CA is an initial allocation of tasks for two VMs. Changing states of CA results in changing an allocation of tasks in the system. Therefore, we use GA to discover the best rule of CA that provides the minimize makespan *T*.

CAT scheme can be divided into two phases: the learning phase and the operation phase as shown in Fig. 3. The purpose of the learning phase is to discover rules for task scheduling by using GA. The GA begins with randomly generating a population of rules with size *P* and randomly choosing *I* initial allocations which correspond to initial allocations of a task set. CA rules will run on each initial allocation for *M* time steps then measure the makespan *T* for each final allocation. A fitness value for each rule in the population is the average *T*. The GA selects the best rules ("elite") with number *E* to the next generation without modification. The remaining *P-E* rules are generated by crossover between elite rules and mutated with probability P_m. This process is continued a predefined number G of generations and discovered rules will be stored after completion. Searching rules is conducted with use of GA in the following way:

```
BEGIN
  Initialize t to zero
  Initialize the population P() of CA rules of a size P
  Initialize a set of makespan T of a size P
  Initialize the number of generations G
  Initialize the number of time steps M
  Initialize the number of elite rules E
  Initialize the mutation probability Pm
  FOR G iterations
    Create a set of a size I of initial allocations
    FOR I iterations
      FOR P iterations
        FOR M iteration
          Determine the next state of CA cells
      ENDFOR
    ENDFOR
    Calculate the average T of P
    Move E of the best rules from P(t) to P(t+1)
    Crossover with the selected rules from E
    Mutation with probability Pm
    Add rules to P (t+1)
    t = t + 1
  ENDFOR
  Store the best rules from P()
END
```

In operation phase, when a task set is randomly allocated, the best found rule of CA is able to find an optimal scheduling in a finite number of time steps by selecting the allocation which gives the minimum makespan. The simulation result of the performance evaluation is presented in the next section.

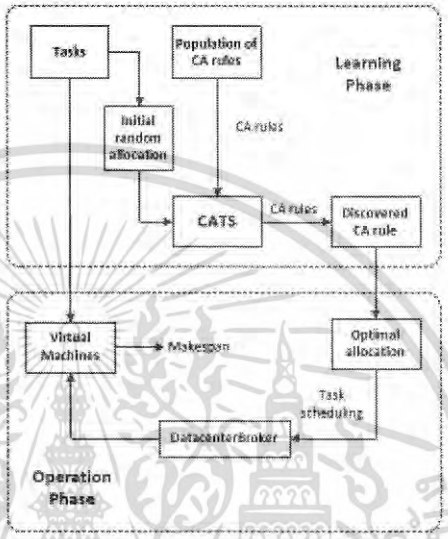


Figure 3. Learning phase and operation phase of CATS scheme

V. Simulation and Results

A. Assumptions

The performance of our proposed scheme or CATS scheme is evaluated by means of the simulation using CloudSim [11]. We assume that for a task set there is no precedence constraints among tasks that means each task is independent. They also cannot be interrupted during their execution.

B. Simulation Parameters for CloudSim

We assume 10 datacenters with 100, 200, 300, 400 and 500 tasks and 50 VMs. The range of the length of each task is from 5000 Million Instruction (MI) to 15000 MI. Computing performance of VMs is vary between 250 (Million Instruction Per Second) MIPS to 2000 MIPS. All datacenters randomly use allocation policy for VM allocation with Time_shared or

Space_shared and all VMs use only Space_shared policy. The simulation parameters are shown in Table 2.

C. Simulation Parameters for CATS Scheme

The parameters setting for CATS scheme is shown in Table 3. The radius r of CA is 1. We use 100 generations with the number of population is 50, also $I = 50$, $M = 50$, and $E = 25$. For crossover and mutation, we use a two-point crossover and a bit-flip mutation with probability (P_m) = 0.03 in GA.

TABLE II. PARAMETERS SETTING OF CLOUD SIMULATOR

Type	Parameter	Value
Datacenter	Number of datacenters	10 datacenters
	Number of hosts	2-6 hosts
	Type of manager	Space_shared
		Time_shared
Virtual Machine (VM)	MIPS	250-2000 MIPS
	VM memory (RAM)	512-2048 Mbytes
	Type of manager	Space_shared
Task	Total number of task in a task set	100-500 tasks
	Length of task	5000-15000 MI

TABLE III. PARAMETERS SETTING OF CAs-BASED SCHEDULER

Parameter	Value
Number of generation (G)	100
Number of population (P)	50
Number of initial allocation (I)	50
Number of Time step (M)	50
Number of Elite rule (E)	25
Mutation probability (P_m)	0.03
CA's radius	1

D. Simulation Results

We compare the performance of our proposed scheme (CATS scheme) with that of the First Come First Served (FCFS). The aim of FCFS scheme is to find the earliest completion time of each task individually and the aim of CA-based scheduler is to find an allocation which minimizes the makespan of a given task set. The total execution of a task set (makespan) of two algorithms are compared. Figure 4 shows that the average makespan of CATS with 100-500 tasks is obviously lower than FCFS. Figure 5 shows a run of CATS with the best found rule after 100 generations, starting with a random initial allocation of 100 tasks. It presents a value of T in 50 time steps. The GA can discover the CA rule at the average T equal 79 at $M = 9$.

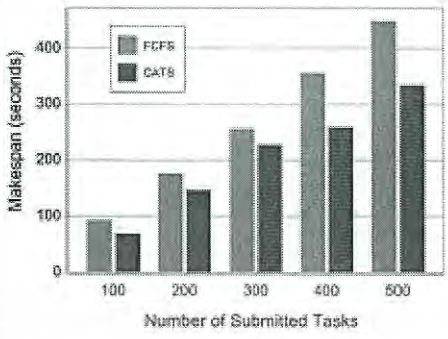


Figure 4. The average makespan of 100-500 tasks set

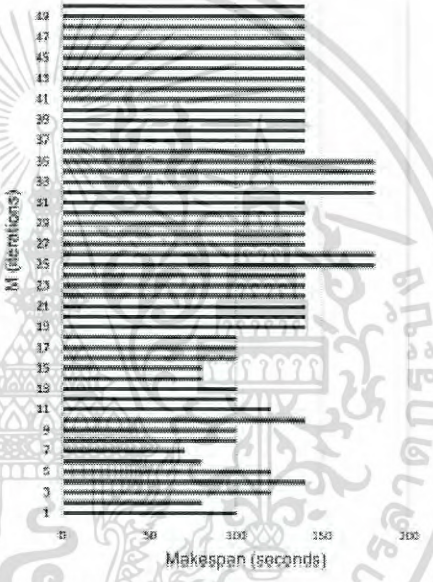


Figure 5. Time diagram of the best found rule for 100 tasks discovered by CATS scheme

VI. CONCLUSIONS

In this paper, we have proposed the application of cellular automata and genetic algorithm to task scheduling in cloud computing systems call CATS scheme. The cellular automata

(CA) have been considered for setting the task scheduling rules. The effective rule has been discovered by using genetic algorithm (GA). The effectiveness of CATS scheme has been evaluated by means of simulation. The simulation result has shown that CATS scheme can provide the lower makespan comparing with that of the traditional First-Come-First-Serve (FCFS) scheme.

REFERENCES

[1] M. D. Dikaiakos, "Cloud Computing: Distributed Internet Computing for IT and Scientific Research," Internet Computing, IEEE , vol.13, no.5, pp.10,13, Sept.-Oct. 2009.

[2] R. Buyya, J. Broberg , and A. M. Goscinski, Cloud Computing: Principles and Paradigms (Wiley Series on Parallel and Distributed Computing). Wiley, 2011.

[3] F. Chang and R. Viswanathan, "Optimal Resource Allocation in Clouds," Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on , pp.418,425, 5-10 July 2010.

[4] A. Swiecicka and F. Seredynski, "Applying cellular automata in multiprocessor scheduling," Parallel Computing in Electrical Engineering, 2002. PARELEC '02. Proceedings International Conference on , pp.177,182, 2002.

[5] F. Seredynski and P. Bouvry, "Multiprocessor scheduling algorithms based on cellular automata training," Intelligent Engineering Systems Through Artificial Neural Networks, vol. 13, pp. 255-260, 2003

[6] D. Rajarshi , M. Melanie, and J. P. Crutchfield, "A Genetic Algorithm Discovers Particle-Based Computation in Cellular Automata," Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature, p344-353, October 09-14, 1994.

[7] S. Wolfram, Cellular Automata And Complexity: Collected Papers. Westview Press, 1994.

[8] J. L. Schiff, Cellular Automata: A Discrete View of the World. Wiley-Interscience, 2008.

[9] D. E. Goldberg, Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Professional, 1989.

[10] R. N. Calheiros, R. Ranjan, and R. Buyya, "CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services," Technical Report, GRIDS-TR-2009-1, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, 2009.

[11] R. N. Calheiros, R. Ranjan, A. Beloglazov, and R. Buyya, "CloudSim: A Toolkit for the Modeling and Simulation of Cloud Resource Management and Application Provisioning Techniques," Journal Software—Practice & Experience archive, vol. 41, pp. 23-50, 2011.

[12] B. Ghalem , Z. T. Fatima, and Z. Wieme, "Approaches to Improve the Resources Management in the Simulator CloudSim," ICICA 2010, LNCS 6377, DOI: 10.1007/978-3-642-16167-4_25, pp. 189-196, 2010.

[13] R. Buyya, R. Ranjan , R. N. Calheiros, "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities," Proceedings of the 7th High Performance Computing and Simulation (HPCS 2009) Conference, Leipzig, Germany, DOI: 10.1109/HPCSIM.2009.5192685, 2009.

ประวัติผู้เขียน

ชื่อ-นามสกุล	นางสาวอิงขวัญ วชิระชูเดช
วัน เดือน ปีเกิด	1 มีนาคม 2533 กรุงเทพฯ
ที่อยู่	36/121 สาทรเฮอริเทจเรสซิเดนท์ ถ.นราธิวาสราชนครินทร์ แขวงยานนาวา เขตสาทร กทม. 10120
	โทร. 080-099-9301
ประวัติการศึกษา	สำเร็จการศึกษา วิศวกรรมศาสตรบัณฑิต ในปีการศึกษา 2554
	คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมคอมพิวเตอร์
	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง