อัลกอริทึมสำหรับการจัดสรรโมบายล์เอเจนต์ในระบบการสืบค้นข้อมูล

นายสมชัย แสงทองสกุลเลิศ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมไฟฟ้า จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2544
ISBN 974 – 03 – 1062 – 1

AN ALGORITHM FOR MOBILE AGENT ALLOCATION IN INFORMATION RETRIEVAL SYSTEM

Mr. Somchai Sangthongsakullerd

A Thesis Submitted in Partial Fulfillment of the Requirements

For the Degree of Master of Engineering in Electrical Engineering

Department of Electrical Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2001

ISBN 974 – 03 – 1062 – 1

หัวข้อวิทยานิพนธ์	อัลกอริทึมสำหรับการจัดสรรโมบายล์เอเจนต์ในระบบการสืบค้นข้อมูล
โดย	นายสมชัย แสงทองสกุลเลิศ
สาขาวิชา	วิศวกรรมไฟฟ้า
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.ลัญฉกร วุฒิสิทธิกุลกิจ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สมศักดิ์ ปัญญาแก้ว)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(ศาสตราจารย์ ดร.ประสิทธิ์ ประพิณมงคลการ)

.....อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์ ดร.ลัญฉกร วุฒิสิทธิกุลกิจ)

.....กรรมการ
(รองศาสตราจารย์ ดร.สมชาย จิตะพันธ์กุล)

สมชัย แสงทองสกุลเลิศ : อัลกอริทึมสำหรับการจัดสรรโมบายล์เอเจนต์ในระบบการสืบค้นข้อมูล (AN ALGORITHM FOR MOBILE AGENT ALLOCATION IN NOFORMATION RETREIVAL SYSTEM) อ. ที่ปรึกษา : ผศ.ดร.ลัญฉกร วุฒิสิทธิกุลกิจ, 134 หน้า ISBN 974 – 03 – 1062 – 1

วิทยานิพนธ์ฉบับนี้ได้เสนออัลกอริทึมที่ใช้ในการกำหนดเส้นทางให้กับโมบายล์เอเจนต์ โดยใช้อัลกอริทึม Simulated Annealing และอัลกอริทึม Tabu Search มาประยุกต์ใช้ควบคู่กับอัลกอริทึมฮิวริสติกที่คิดขึ้น และทำการเปรียบเทียบผลลัพธ์ที่ได้กับอัลกอริทึม Brute Force Search (BF) อัลกอริทึม Simulated Annealing (SA) อัลกอริทึม Tabu Search (Tabu) อัลกอริทึม Modified Compact Genetic Algorithm (McGA) (ซึ่งประยุกต์มาจากอัลกอริทึม Compact Genetic Algorithm : cGA) และอัลกอริทึมกำหนดเส้นทางแบบสุ่มที่มีการกระจายแบบยูนิฟอร์ม (Random Uniformly : RU) สำหรับพารามิเตอร์ที่นำมาใช้วิเคราะห์เปรียบเทียบคือค่าต้นทุนที่ต่ำที่สุดของเส้นทางย่อยที่มีค่าสูงที่สุด (Minimum of Maximum Sub-Route Cost : MMSRC) และเวลาที่ใช้ในการประมวลผล ผลลัพธ์ที่ได้จากการวิเคราะห์แสดงให้เห็นว่า กรณีจำนวนโนดในเนตเวิร์กมีค่าต่ำ อัลกอริทึมฮิวริสติกจะให้ค่า MMSRC ที่มีค่ามากกว่าค่า MMSRC ที่ได้จากอัลกอริทึม BF, SA, Tabu และ McGA แต่ให้ค่า MMSRC ที่ต่ำกว่ามากเมื่อเทียบกับอัลกอริทึม RU เมื่อวิเคราะห์กรณีจำนวนโนดในเนตเวิร์กมีค่าสูงอัลกอริทึมฮิวริสติกจะให้ค่า MMSRC ที่ต่ำกว่าค่า MMSRC ที่ได้จากอัลกอริทึม SA และจากการวัดค่าเวลาประมวลผลของแต่ละอัลกอริทึม ผลลัพธ์แสดงให้เห็นว่าอัลกอริทึมฮิวริสติกจะใช้เวลาประมวลผลโดยเฉลี่ยน้อยกว่าอัลกอริทึมกำหนดเส้นทางชนิดอื่น ๆ อย่างเห็นได้ชัด

ในการพัฒนากรรมวิธีการกำหนดเส้นทางให้กับโมบายล์เอเจนต์ของงานวิจัยนี้ อาศัยการประยุกต์แนวคิดมาจากปัญหาการเดินทางของเซลส์แมน (Traveling Salesman Problem : TSP) โดยมีข้อกำหนดว่า เซลส์แมนไม่สามารถเดินย้อนเส้นทางเดิมได้ ซึ่งแตกต่างจากกรณีการกำหนดเส้นทางให้กับโมบายล์เอเจนต์โดยโมบายล์เอเจนต์สามารถเดินย้อนกลับผ่านเส้นทางเดิมได้ ดังนั้นเพื่อศึกษาถึงผลกระทบของข้อจำกัดดังกล่าว วิทยานิพนธ์จึงได้พิจารณาเพิ่มเติมโดยนำรีดิวซ์เมตริกซ์มาประยุกต์ใช้เพื่อช่วยในการวิเคราะห์การกำหนดเส้นทางให้กับโมบายล์เอเจนต์กรณีย้อนกลับเส้นทางเดิมได้ ซึ่งการใช้งานรีดิวซ์เมตริกซ์ในวิทยานิพนธ์ฉบับนี้ได้แบ่งออกเป็น 2 วิธี คือวิธีแทนค่า และวิธีประยุกต์ จากผลลัพธ์แสดงให้เห็นว่า วิธีประยุกต์โดยส่วนใหญ่จะให้ผลลัพธ์ที่ดีกว่าวิธีแทนค่า อย่างไรก็ตามในบางกรณี วิธีประยุกต์อาจให้ค่า MMSRC ที่เลวกว่ากรณีไม่ใช้รีดิวซ์เมตริกซ์ ในขณะที่วิธีแทนค่าจะให้ค่า MMSRC ที่ดีกว่ากรณีไม่ใช้รีดิวซ์เมตริกซ์เสมอ แม้ว่าการนำรีดิวซ์เมตริกซ์มาใช้แม้ว่าจะให้ค่า MMSRC ที่ลดลง แต่ความแตกต่างของค่า MMSRC ที่ลดลงนั้นอยู่ในช่วงร้อยละ 10 ดังนั้นจึงกล่าวได้ว่าการกำหนดเส้นทางให้กับโมบายล์เอเจนต์แบบอนุญาตกลับเส้นทางเดิมได้ ส่งผลกระทบต่อต้นทุนโดยรวมของการกำหนดเส้นทางให้กับโมบายล์เอเจนต์ไม่มากนัก

ภาควิชา	วิศวกรรมไฟฟ้า	ลายมือชื่อนิสิต……
สาขาวิชา	วิศวกรรมไฟฟ้า	ลายมืออาจารย์ที่ปรึกษา…….
ปีการศึกษา	2544	

4270575321 : MAJOR ELECTRICAL ENGINEERING

KEY WORD: INFORMATION RETRIEVAL / MOBILE AGENT / ITINERARY CREATION ALGORITHM

SOMCHAI SANGTHONGSAKULLERD: AN ALGORITHM FOR MOBILE AGENT ALLCATION IN INFORMATION RETRIEVAL SYSTEM. THESIS ASSIST.ADVISOR : LUNCHAKORN WUTTISITTIKULKIJ ,Ph.D. 134 pp. ISBN 974 – 03 – 1062 – 1.

The thesis proposes the mobile agent itinerary creation algorithm by modifying the traditional Simulated Annealing and Tabu Search algorithm in conjunction with proposed heuristic (HR) algorithm and then compares with Brute Force Search algorithm (BF), Simulated Annealing algorithm (SA), Tabu Search Algorithm (Tabu), Modified Compact Genetic Algorithm (McGA), which modified from Compact Genetic Algorithm (cGA), and Random Uniformly Algorithm (RU). To show the pros and cons of each algorithm, the parameters MMSRC (Minimum of Maximum Sub-Route Cost) and Processing Time are used as the key parameters. The results show that in the case of the number of nodes in network is low, HR gives the result slightly worse than MMSRC with respect to BF, SA, Tabu and McGA but greatly better than RU. In the case of the high number of nodes HR gives the better MMSRC with respect to the result from SA. Furthermore, given the processing time parameter HR gives obviously less average processing time with respect to the others.

In this thesis the development of Mobile Agent's itinerary creation algorithm is based on the concept of Traveling Salesman Problem (TSP). For the TSP problem, on his return, the salesman is not allowed to take the same route. This is in contrast to the Mobile Agent's itinerary creation application. With the itinerary creation application, the Mobile Agents are entitled to return to the same node they have previously passed. To study the impact of this discrepancy this thesis extends the investigations by introducing a reduced matrix technique so as to analyze the Mobile Agent's itinerary creation application in the case of allowing the return to the previously passed node. The use of reduced matrix technique is categorized into 2 cases, namely *replacement* and *application* technique. The result shows that in general the *application* technique gives better result than the *replacement*. However in some cases the *application* technique may offer poorer results than the MMSRC that does not use the reduced matrix technique whereas the *replacement* always offers better MMSRC than the MMSRC that does not use reduced matrix technique. Although reduced matrix technique gives better results but the difference is marginal, i.e. within 10%. As a result it is reasonable to conclude that allowing Mobile Agent to return to the previously passed node does not cause significant impact on the overall system costs.

Department Electrical Engineering Student's Signature…..……

Field of Study Electrical Engineering Advisor's Signature…….

Academic Year 2001

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ได้สำเร็จลุล่วงไปได้ด้วยดี ด้วยความช่วยเหลือของผู้ช่วยศาสตราจารย์ ดร.ลัญฉกร วุฒิสิทธิกุลกิจ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้ให้คำแนะนำการวิจัย แนวทางการวิจัย ตลอดจนคำปรึกษาและข้อคิดเห็นต่าง ๆ ในการวิจัยเป็นอย่างดี

ขอขอบคุณนายวสันต์ นส.ณภัทร และพี่บอย ที่ให้คำแนะนำ ช่วยเหลือและให้คำปรึกษาพร้อมทั้งกำลังใจในการแก้ปัญหาต่าง ๆ

ขอขอบคุณ พี่ ๆ เพื่อน ๆ และ น้อง ๆ ที่ห้องปฏิบัติการไฟฟ้าสื่อสารโทรคมนาคม ที่ให้คำแนะนำคำปรึกษาและกำลังใจตลอดช่วงระยะเวลาในการทำวิทยานิพนธ์

สุดท้ายนี้ผู้วิจัยขอกราบขอบพระคุณบิดามารดาข้าพเจ้าที่ได้ให้ความช่วยเหลือด้านทุนทรัพย์ และกำลังใจต่าง ๆ เป็นอย่างดี ตลอดจนพี่ ๆ ทุกคนของข้าพเจ้า

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูป

หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาของปัญหา

ปัจจุบันการติดต่อสื่อสารบนเครือข่ายอินเทอร์เนตมีปริมาณเพิ่มมากขึ้นอย่างรวดเร็วและแพร่หลาย โปรแกรมที่ใช้งานโดยทั่วไปได้แก่ การท่องเว็บบนอินเทอร์เนต (Web-browsing) การรับส่งไปรษณีย์อิเล็กทรอนิกส์ (E-mail) ไอซีคิว (ICQ) การดาวน์โหลดไฟล์ผ่านโปรแกรมเอฟทีพี (FTP) รวมไปถึงการจัดการดูแลเนตเวิร์ก (Network Management) เป็นต้น โปรแกรมเหล่านี้ส่วนใหญ่มีพื้นฐานการติดต่อสื่อสารเป็นแบบไคลเอนต์เซิร์ฟเวอร์ โดยปกติเซิร์ฟเวอร์หนึ่งตัวทำหน้าที่รองรับและให้บริการไคลเอนต์จำนวนหนึ่งได้พร้อมกันยกตัวอย่างเช่นการค้นหาข้อมูลของผู้ใช้บริการจำนวนมากในเว็บไซต์ที่เป็นที่รู้จักกันอย่างแพร่หลาย

เนื่องจากโปรแกรมที่กล่าวมาแล้วนั้นมีโครงสร้างการทำงานแบบรวมศูนย์ (Centralized Communication) จึงทำให้การทำงานแบบไคลเอนต์เซิร์ฟเวอร์มีความเหมาะสมและใช้งานได้ดี แต่อย่างไรก็ตามการทำงานแบบรวมศูนย์ลักษณะเช่นนี้จะทำงานได้อย่างมีประสิทธิภาพก็ต่อเมื่อ ปริมาณข่าวสารที่ถ่ายเทระหว่างไคลเอนต์และเซิร์ฟเวอร์นั้นมีปริมาณไม่มากนัก ถ้ามีปริมาณข่าวสารหรือการใช้งานที่สูงขึ้นจะก่อให้เกิดความคับคั่ง ณ บริเวณที่มีการถ่ายเทข้อมูลข่าวสารได้ ดังตัวอย่างการใช้งานบางประเภท เช่น การจัดการเนตเวิร์ก ตัวที่ทำหน้าที่จัดการควบคุมเนตเวิร์ก (Manager Station) จะทำหน้าที่รวบรวมข้อมูลจากตัวเซิร์ฟเวอร์ต่าง ๆ ทั่วเนตเวิร์ก แล้วนำข้อมูลที่ได้มานั้นมาทำการประมวลผลเพื่อใช้เป็นข้อมูลที่ใช้ในการวิเคราะห์และควบคุมการทำงานของเนตเวิร์ก นอกจากนี้ตัวจัดการเนตเวิร์กยังเป็นเป็นศูนย์กลางการติดต่อของระบบ ดังนั้นถ้าข้อมูลที่ถ่ายเทจากทั่วเนตเวิร์กมาสู่ตัวดูแลระบบนั้นเป็นข้อมูลที่มีปริมาณมาก ย่อมจะทำให้เกิดปัญหาด้านขนาดช่องสัญญาณบนเนตเวิร์ก และเกิดความคับคั่งของการประมวลผลที่ตัวจัดการควบคุมเนตเวิร์ก นอกจากนี้การทำงานที่ใช้แนวคิดไคลเอนต์เซิร์ฟเวอร์นั้น ฟังก์ชันการทำงานต่าง ๆ จะต้องถูกกำหนดตายตัวตอนสร้างระบบ หรือก่อนที่ระบบจะทำงาน เมื่อมีความต้องการที่จะทำการปรับปรุงเปลี่ยนแปลงแก้ไขหรือเพิ่มฟังก์ชันการทำงานจะต้องทำการปรับปรุงและแก้ไขที่ตัวเซิร์ฟเวอร์ทุกตัวที่มีการให้บริการ ดังนั้นจะเห็นได้ว่าการปรับปรุงและแก้ไขเซิร์ฟเวอร์ให้ครบทุกตัวในเนตเวิร์กที่มีขนาดใหญ่อย่างเช่นอินเทอร์เนตได้นั้น เป็นไปได้ค่อนข้างยากลำบาก และจากสภาพของเนตเวิร์กที่มีการเปลี่ยนแปลงค่อนข้างตลอดเวลา ทำให้มีความต้องการด้านความยืดหยุ่น (Flexibility) และการขยายระบบ (Extensibility) เป็นต้น ด้วยเหตุนี้จึงได้มีการพัฒนาแนวคิดรูปแบบใหม่ที่แก้

ปัญหาหรือข้อจำกัดของโครงสร้างการติดต่อแบบรวมศูนย์ โดยอาศัยวิธีการกระจายการติดต่อออกไปในบริเวณกว้างแทนการรวมศูนย์หรือที่เรียกว่าการกระจายการติดต่อ (Distributed Communication) การติดต่อสื่อสารแบบโมบายล์เอเจนต์เป็นวิธีการหนึ่งที่มีหลักการทำงานแบบกระจายการติดต่อสื่อสารที่กำลังได้รับความสนใจอย่างมาก เพราะเป็นวิธีหนึ่งที่สามารถลดหรือแก้ปัญหาที่เกิดขึ้นกับระบบที่ทำงานแบบรวมศูนย์ในปัจจุบัน

สำหรับโครงสร้างการทำงานของโมบายล์เอเจนต์นั้นจำเป็นจะต้องมีการกำหนดลำดับของเซิร์ฟเวอร์หรือเส้นทางที่จะให้โมบายล์เอเจนต์ไปประมวลผล ดังนั้นการกำหนดเส้นทางให้กับโมบายล์เอเจนต์เป็นปัจจัยสำคัญปัจจัยหนึ่งที่มีผลต่อการทำงานของระบบที่ใช้แนวคิดโมบายล์เอเจนต์ โดยลักษณะเส้นทางที่ดีควรจะเป็นเส้นทางที่ให้ผลรวมของเวลาหน่วงในการเคลื่อนย้ายของโมบายล์เอเจนต์ไปแต่ละเซิร์ฟเวอร์มีค่าต่ำที่สุด ดังนั้นวิทยานิพนธ์ฉบับนี้ได้เสนอแนวคิดการกำหนดเส้นทางให้กับโมบายล์เอเจนต์ โดยคิดอัลกอริทึมขึ้นมาใช้เอง และได้ทำการศึกษาวิเคราะห์เปรียบเทียบกับอัลกอริทึมที่มีใช้ในปัจจุบัน ในการเปรียบเทียบสมรรถนะการทำงานของแต่ละอัลกอริทึมนั้นจะใช้ค่าเวลาหน่วง (Delay Time) และเวลาที่ใช้ในการกำหนดเส้นทางของแต่ละอัลกอริทึม หรือเวลาประมวลผลของอัลกอริทึม (Processing Time) โดยจะวิเคราะห์เทียบกับปริมาณโนดในเนตเวิร์กที่เพิ่มขึ้น

สำหรับเนื้อหาในวิทยานิพนธ์ฉบับนี้จะถูกแบ่งออกเป็น 8 บทดังนี้

บทที่ 1 เป็นการกล่าวนำถึงความเป็นมา ที่มาของปัญหา และความสำคัญของวิทยานิพนธ์ พร้อมทั้งกล่าวถึงวัตถุประสงค์และของเขตของการทำวิทยานิพนธ์

บทที่ 2 กล่าวถึงขั้นตอนและหลักการทำงานของโมบายล์เอเจนต์โดยอธิบายเทียบกับหลักการทำงานของไคลเอนต์เซิร์ฟเวอร์

บทที่ 3 กล่าวถึงพื้นฐานของระบบการสืบค้นข้อมูล ซึ่งประกอบไปด้วยหลักการจัดการฐานข้อมูล วิธีการสืบค้นข้อมูล และยังการกล่าวถึงเทคนิคการสืบค้นข้อมูลที่เป็นแนวคิดพื้นฐานตามทฤษฎี และเทคนิคที่กำลังใช้กันอยู่ในปัจจุบัน

บทที่ 4 กล่าวถึงอัลกอริทึมที่ใช้ในการกำหนดเส้นทางเริ่มต้นให้กับโมบายล์เอเจนต์ โดยจะอธิบายถึงหลักการทำงานของอัลกอริทึม Brute Force Search, Simulated Annealing, Tabu Search และ Compact Genetic Algorithm ซึ่งเป็นอัลกอริทึมที่นิยมใช้ในการปรับปรุงคำตอบหรือเส้นทางที่ดีที่สุด

บทที่ 5 กล่าวถึงอัลกอริทึมฮิวริสติกที่คิดขึ้นที่ใช้ในการกำหนดเส้นทางให้กับโมบายล์เอเจนต์ พร้อมทั้งอธิบายแนวทางในการเปรียบเทียบผลลัพธ์ที่ได้จากอัลกอริทึมฮิวริสติกและอัลกอริทึมต่าง ๆ ที่กล่าวไว้ในบทที่ 4 นอกจากนี้ยังมีการคิดอัลกอริทึมประยุกต์คอมแพคเจเนติก (Modified Compact Genetic Algorithm) เพื่อใช้ในการวิเคราะห์เปรียบเทียบถึงสมรรถนะการทำงานของอัลกอริทึมฮิวริสติก ท้ายที่สุดได้มีการเสนอรูปแบบของเนตเวิร์กจำลองประเภทอื่นเพื่อเป็นแนวทางการวิเคราะห์ถึงสมรรถนะการทำงานในสภาวะแวดล้อมของเนตเวิร์กที่แตกต่างกัน (Robustness)

บทที่ 6 เป็นการวิเคราะห์และสรุปผลที่ได้จากการกำหนดเส้นทางเริ่มต้นโดยใช้อัลกอริทึมต่าง ๆ ที่ได้อธิบายไว้ในบทที่ 5

บทที่ 7 เป็นการวิเคราะห์เพิ่มเติมเกี่ยวกับการกำหนดเส้นทางแบบอนุญาตย้อนเส้นทางเดิมได้ โดยจะอธิบายพร้อมแสดงผลการวิเคราะห์ระหว่างค่า MMSRC ที่ได้จากอัลกอริทึมกำหนดเส้นทางแบบไม่ย้อนกลับเส้นทางเดิม และผลที่ได้จากการประยุกต์อัลกอริทึมที่มีให้ทำงานเสมือนอนุญาตย้อนกลับเส้นทางเดิมได้

บทที่ 8 เป็นการวิเคราะห์และสรุปผลที่ได้จากการทำวิทยานิพนธ์พร้อมข้อเสนอแนะเกี่ยวกับการพัฒนาการทำงานการสืบค้น

1.2 วัตถุประสงค์ของการวิจัย

1. เพื่อศึกษาโครงสร้างสถาปัตยกรรมของแนวคิดแบบโมบายล์เอเจนต์ (Mobile Agent) ในเชิงวิเคราะห์เปรียบเทียบกับ แบบไคลเอนต์เซิร์ฟเวอร์ รวมทั้งศึกษาข้อเด่นและข้อด้อยของแต่ละสถาปัตยกรรม
2. เพื่อศึกษาและวิเคราะห์การทำงานของอัลกอริทึมที่มีในปัจจุบันนำมาใช้ในการกำหนดเส้นทางให้กับโมบายล์เอเจนต์
3. เพื่อศึกษาคิดค้นและปรับปรุงอัลกอริทึมกำหนดเส้นทางให้กับโมบายล์เอเจนต์ พร้อมทั้งวิเคราะห์และเปรียบเทียบข้อเด่นข้อด้อยของแต่ละอัลกอริทึมที่คิดขึ้น เทียบกับอัลกอริทึมพื้นฐานที่มีในปัจจุบัน

1.3 ขอบเขตของการวิจัย

เปรียบเทียบสมรรถนะการทำงานของอัลกอริทึมฮิวริสติกที่ได้คิดขึ้น กับอัลกอริทึมที่ใช้ในการปรับปรุงคำตอบที่มีในปัจจุบัน

1.4 ประโยชน์ที่คาดว่าจะได้รับ

จากการคิดค้น ปรับปรุง และวิเคราะห์อัลกอริทึมกำหนดเส้นทางให้กับโมบายล์เอเจนต์ที่คิดขึ้น เทียบกับอัลกอริทึมที่มีในปัจจุบัน ผลลัพธ์ที่ได้สามารถนำไปใช้เป็นในแนวทางในการพัฒนาอัลกอริทึมการกำหนดเส้นทางให้กับโมบายล์เอเจนต์ให้มีประสิทธิภาพดียิ่งขึ้น

บทที่ 2

โมบายล์เอเจนต์

2.1 กล่าวนำ

เนื่องจากการติดต่อแบบรวมศูนย์ทำให้เกิดข้อจำกัดต่าง ๆ ทำให้เกิดแนวคิดที่เป็นลักษณะกระจายการติดต่อ อาทิเช่น Remote Evaluation และ Mobile Agent เป็นต้น ซึ่งต่างมีข้อเด่นข้อด้อยที่แตกต่างกัน ดังนั้นเนื้อหาในบทนี้จะกล่าวถึงหลักการทำงานของแนวคิดแบบการกระจายการติดต่อร่วมกับแนวคิดการทำงานของไคลเอนต์เซิร์ฟเวอร์

2.2 หลักการทำงานของแนวคิดการติดต่อสื่อสารบนเนตเวิร์ก

สำหรับเนื้อหาในหัวข้อนี้จะกล่าวถึงหลักการทำงานอยู่ 3 แบบ คือ

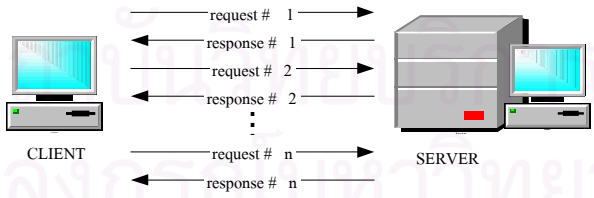
ก.) Client-Sever Operation (CS Operation)

ข.) Remote Evaluation Operation (REV Operation)

ค.) Mobile Agent Operation (MA Operation)

2.2..1 Client-Server Operation (CS Operation)

ปัจจุบันโปรแกรมที่ใช้งานในเนตเวิร์กส่วนใหญ่มีแนวทางการติดต่อแบบรวมศูนย์ ซึ่งมีหลักการทำงานแบบไคลเอนต์เซิร์ฟเวอร์ โดยจะมีการติดต่อที่เป็นชุดคำสั่งแบบการร้องขอและการตอบรับ (Request-Response) ดังรูปที่ 2.1



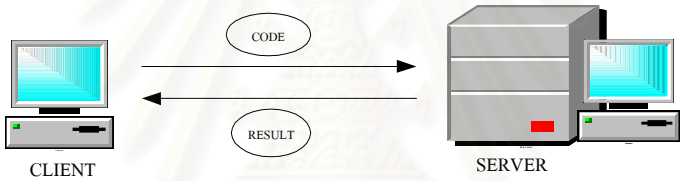
รูปที่ 2.1 CS Operation

ซึ่งขั้นตอนการทำงานที่สำคัญจะเริ่มจากไคลเอนต์ส่งชุดคำสั่งการร้องขอไปยังเซิร์ฟเวอร์ จากนั้นเซิร์ฟเวอร์จะส่งข้อมูลที่อยู่ในรูปของการตอบรับกลับมายังไคลเอนต์ และจะรับส่งกลับไปกลับมาเช่นนี้จนกว่าจะสิ้นสุดการใช้บริการของไคลเอนต์ ดังนั้นจะเห็นได้ว่าในเนตเวิร์กที่มีหลักการทำงานแบบไคลเอนต์เซิร์ฟเวอร์นี้จะเต็มไปด้วยข้อมูลที่ถ่ายเทกันระหว่างไคลเอนต์และเซิร์ฟเวอร์

ซึ่งข้อมูลเหล่านี้จะถูกไปประมวลผลที่ไคลเอนต์อีกครั้งหนึ่ง จะเห็นได้ว่าข้อมูลที่ส่งจากเซิร์ฟเวอร์ไปยังไคลเอนต์ บางส่วนอาจเป็นข้อมูลที่ไคลเอนต์ไม่ต้องการ หรือไม่มีความจำเป็นต่อการประมวลผลที่ไคลเอนต์ จึงทำให้ใช้ทรัพยากรแบนด์วิดท์ที่ไม่มีประสิทธิภาพ นอกจากนี้ขั้นตอนการติดต่อโดยใช้ชุดคำสั่งแบบการร้องขอและการตอบรับนี้จะถูกกำหนดตายตัวที่เซิร์ฟเวอร์ จึงเป็นการยากที่จะทำการปรับปรุงและเปลี่ยนแปลงการทำงานของเซิร์ฟเวอร์ได้ทั้งหมดในเนตเวิร์กขนาดใหญ่ได้

2.2.2 Remote Evaluation Operation (REV Operation)

สำหรับหลักการทำงานแบบ REV จะมีขั้นตอนการทำงานที่แตกต่างไปจากหลักการทำงานแบบไคลเอนต์เซิร์ฟเวอร์ โดยขั้นตอนการทำงานจะไม่มีลักษณะของการส่งชุดคำสั่งการร้องขอและการตอบรับกลับไปกลับมา แต่ REV จะทำการส่งชุดคำสั่งหรือโค้ด (Code) ไปยังเซิร์ฟเวอร์ในทีเดียว จากนั้นจะทำการประมวลผลที่เซิร์ฟเวอร์แล้วส่งผลลัพธ์หรือข้อมูลที่จำเป็นกลับมายังไคลเอนต์ดังรูปที่ 2.2



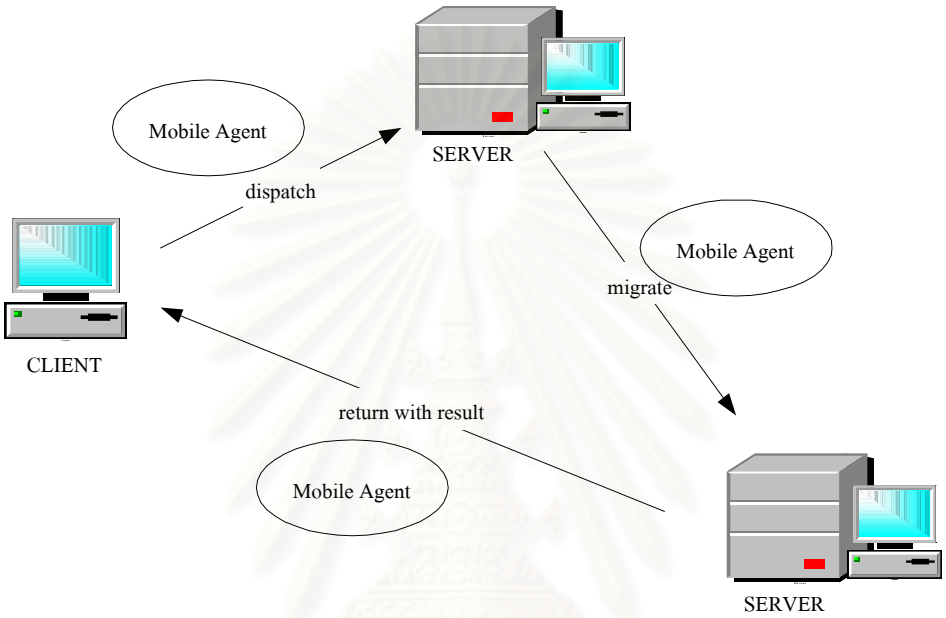
รูปที่ 2.2 REV Operation

ซึ่งการทำงานในลักษณะนี้จะเห็นได้ว่าโค้ดที่ส่งไปประมวลผลที่เซิร์ฟเวอร์นั้นจะต้องมีคุณสมบัติใช้งานได้ทันที (Ready-to-use) นอกจากนี้ขนาดของโค้ดที่ส่งไปประมวลผลที่เซิร์ฟเวอร์นั้นอาจมีขนาดใหญ่กว่าขนาดของชุดคำสั่งการร้องขอที่ใช้ในหลักการแบบไคลเอนต์เซิร์ฟเวอร์ ซึ่งอาจส่งผลให้ค่าใช้จ่ายในการใช้แบนด์วิดท์สูงขึ้นตามมา แต่อย่างไรก็ตามเมื่อเปรียบเทียบปริมาณข้อมูลที่เซิร์ฟเวอร์ส่งกลับมายังไคลเอนต์ในหลักการทำงานแบบ REV แล้ว จะเห็นได้ว่ามีแนวโน้มของการใช้ปริมาณแบนด์วิดท์ที่น้อยกว่า และเนื่องจากโค้ดต้องนำไปประมวลผลที่เซิร์ฟเวอร์ดังนั้นเซิร์ฟเวอร์จำเป็นต้องมีแพลตฟอร์ม (Platform) ที่คล้ายคลึงกัน หรือจะต้องมีคุณสมบัติประมวลผลข้ามแพลตฟอร์มได้ ดังเช่น จาวา (Java) (James and Jenry, 1996) เป็นต้น

2.2..3 Mobile Agent Operation (MA Operation)

หลักการทำงานของโมบายล์เอเจนต์จะมีขั้นตอนการทำงานที่คล้ายคลึงกับหลักการทำงานของ REV กล่าวคือขั้นตอนการทำงานโดยส่งโค้ดจากไคลเอนต์ไปประมวลผลที่เซิร์ฟเวอร์ต่าง ๆ เพื่อให้ได้ข้อมูลที่ต้องการ จากนั้นส่งข้อมูลนั้นกลับไปยังไคลเอนต์ ซึ่งขั้นตอน

การทำงานแบบโมบายล์เอเจนต์จะแตกต่างจากขั้นตอนการทำงานแบบ REV ในแง่ที่ว่า โค้ดที่ถูกส่งไปประมวลผลที่เซิร์ฟเวอร์นั้นไม่ได้ถูกจำกัดอยู่เพียงเซิร์ฟเวอร์เดียว กล่าวคือ เพื่อให้ได้ข้อมูลที่ไคลเอนต์ต้องการที่มากขึ้น โค้ดสามารถไปประมวลผลที่เซิร์ฟเวอร์ได้มากกว่า 1 ตัว จากนั้นส่งผลลัพธ์ที่ได้จากการประมวลผลที่เซิร์ฟเวอร์ต่าง ๆ กลับไปยังไคลเอนต์ สำหรับหลักการทำงานแบบโมบายล์เอเจนต์โค้ดที่ถูกส่งไปประมวลผลที่เซิร์ฟเวอร์จะถูกเรียกว่า โมบายล์เอเจนต์ และขั้นตอนการทำงานอธิบายได้ดังรูปที่ 2.3



รูปที่ 2.3 MA Operation

จากขั้นตอนการทำงานของโมบายล์เอเจนต์ ตัวโมบายล์เอเจนต์เปรียบเสมือนเอเจนต์ตัวหนึ่งที่ทำหน้าที่แทนผู้ใช้บริการ (ไคลเอนต์) ไปประมวลผลที่เซิร์ฟเวอร์ต่าง ๆ เพื่อให้ได้ข้อมูลที่ต้องการแล้วส่งกลับไปยังผู้ใช้บริการ จากที่กล่าวมาแล้วนั้นจะเห็นได้ว่า การติดต่อกันระหว่างผู้ใช้บริการและโมบายล์เอเจนต์นั้นจะมีอยู่เพียง 2 ขั้นตอน คือขั้นตอนที่หนึ่งเป็นช่วงของการปล่อยโมบายล์เอเจนต์ไปประมวลผลที่เซิร์ฟเวอร์ในเนตเวิร์ก และขั้นตอนที่สองเป็นช่วงของการรับข้อมูลจากโมบายล์เอเจนต์ ดังนั้นจะเห็นได้ว่า เมื่อโมบายล์เอเจนต์ถูกปล่อยจากผู้ใช้บริการเข้าสู่เนตเวิร์กแล้วจะไม่มีการส่งข้อมูลหรือการติดต่อใด ๆ กับผู้ใช้บริการอีก ซึ่งทำให้ผู้ใช้บริการสามารถประหยัดค่าใช้จ่ายในการจองช่องสัญญาณที่ติดต่อกับเนตเวิร์กได้ สำหรับรายละเอียดและคุณสมบัติต่าง ๆ ของโมบายล์เอเจนต์จะอธิบายไว้ในหัวข้อต่อไป

2.3 โมบายล์เอเจนต์ (Mobile Agent)

ปัจจุบันเทคโนโลยีโมบายล์เอเจนต์กำลังเป็นที่สนใจของสถานการศึกษา และองค์กรต่าง ๆ จึงมีคำนิยามของโมบายล์เอเจนต์ที่แตกต่างกันออกไป แต่มีส่วนที่คล้ายคลึงกันสามารถสรุปได้ดังนี้ โมบายล์เอเจนต์ คือวัตถุ (Object) ที่ประกอบด้วยโค้ดและข้อมูลที่ถูกสร้างโดยผู้ใช้บริการและส่งเข้าไปในเนตเวิร์กโดยมีความสามารถที่จะเข้าถึงข้อมูลในเซิร์ฟเวอร์ต่าง ๆ บนเนตเวิร์กที่มีความแตกต่างกันทางสถาปัตยกรรม (Heterogeneous Network) เพื่อทำการประมวลผลโดยใช้ทรัพยากรต่าง ๆ (Resource) บนเซิร์ฟเวอร์นั้น ๆ (Remote-Resource Access) รวมไปถึงความสามารถที่จะย้ายตัวเอง (โค้ด) ไปยังเซิร์ฟเวอร์อื่น ๆ (อาจเนื่องมาจากสภาวะแวดล้อมของเนตเวิร์กบริเวณนั้น หรือเนื่องจากโมบายล์เอเจนต์ตัวอื่น เป็นต้น) เพื่อประมวลผลเพิ่มเติมให้ได้ผลลัพธ์ตามความต้องการของผู้ใช้บริการ จากนั้นส่งผลลัพธ์ที่ได้กลับไปยังผู้ใช้บริการ (Akhil and Christine,1998; Alfonso, Gian and Giovanni, 1998; Anselm, Oswald and Johann, 1996; Magedanz and Eckardt, 1996 ; Wen-Shyen, Lin and Yao-Nan, 1999)

จากความหมายของโมบายล์เอเจนต์จะเห็นได้ว่าขั้นตอนการทำงานของโมบายล์เอเจนต์จะมีความแตกต่างจากขั้นตอนการทำงานแบบไคลเอนต์เซิร์ฟเวอร์ โดยการทำงานของโมบายล์เอเจนต์จะเน้นการประมวลผลให้ใกล้กับทรัพยากรของระบบ หรือการประมวลผลที่เซิร์ฟเวอร์นั่นเอง แทนที่จะส่งข้อมูลจากเซิร์ฟเวอร์กลับมาประมวลผลที่ไคลเอนต์ นอกจากนี้ยังมีคุณลักษณะเฉพาะของการทำงานของโมบายล์เอเจนต์โดยจะอธิบายรายละเอียดต่าง ๆ ในหัวข้อต่อไปนี้

ก.) คุณลักษณะของโมบายล์เอเจนต์ (Mobile Agent's Properties and Characteristics)
ข.) โครงสร้างการทำงานของโมบายล์เอเจนต์ (Mobile Agent's Infrastructure)
ค.) ขั้นตอนการทำงานของโมบายล์เอเจนต์ (Mobile Agent's Operation)

2.3..1 คุณลักษณะของโมบายล์เอเจนต์ (Mobile Agent's Properties and Characteristics)

จากความสามารถการประมวลผลที่ตัวเซิร์ฟเวอร์ ตัวโมบายล์เอเจนต์เปรียบเสมือนโปรแกรมที่ถูกตั้งค่าไว้ให้ไปประมวลผลที่เซิร์ฟเวอร์โดยอัตโนมัติแทนผู้ใช้บริการ ดังนั้นคุณสมบัติแรกที่โมบายล์เอเจนต์จำเป็นต้องมีคือความฉลาด (Intelligent) ซึ่งคุณสมบัตินี้จะมีตั้งแต่การถูกโปรแกรมไว้ล่วงหน้า รวมไปจนถึงความสามารถที่จะเรียนรู้ (Artificial Intelligent) สิ่งต่าง ๆ เองได้ ซึ่งทั้งนี้ขึ้นอยู่กับลักษณะงานที่จะใช้ด้วย นอกจากนี้โมบายล์เอเจนต์ยังเปรียบเสมือนตัวแทนของผู้ใช้บริการที่ไปประมวลผลยังเซิร์ฟเวอร์ต่าง ๆ แล้วส่งผลลัพธ์กลับมายังผู้ใช้บริการ ดังนั้นตัวโมบายล์เอเจนต์นอกจากจะต้องมีการติดต่อกับผู้ใช้บริการแล้ว โมบายล์เอเจนต์จะต้องมีการ

ติดต่อกับ โมบายล์เอเจนต์เซิร์ฟเวอร์ และเอเจนต์บริการ (Service/Resource Agent) ยิ่งไปกว่านั้น โมบายล์เอเจนต์อาจมีการติดต่อกับโมบายล์เอเจนต์ด้วยกันเองเพื่อบรรลุจุดประสงค์ของงานแทนที่จะทำงานโดยใช้โมบายล์เอเจนต์เพียงตัวเดียว จากที่กล่าวมาสามารถสรุปคุณลักษณะของโมบายล์เอเจนต์ได้ดังนี้

(ก) ***Agent Intelligent*** : เป็นคุณสมบัติที่สำคัญที่โมบายล์เอเจนต์ต้องมี โดยคุณสมบัตินี้จะแตกต่างกันตามลักษณะของงานที่ใช้ และเป็นหน้าที่ของโปรแกรมเมอร์ที่จะกำหนดคุณสมบัติต่าง ๆ เหล่านี้ให้กับโมบายล์เอเจนต์

(ข) ***Asynchronous Operation*** : การทำงานของโมบายล์เอเจนต์สามารถทำได้โดยอิสระจากการติดต่อผู้ใช้บริการ โดยเมื่อโมบายล์เอเจนต์ถูกปล่อยเข้าสู่เนตเวิร์กเพื่อประมวลผลที่เซิร์ฟเวอร์แล้ว โมบายล์เอเจนต์สามารถที่จะทำงานโดยปราศจากผู้ใช้บริการได้ และสามารถถูกกระตุ้น (Triggered) ให้ทำงานเฉพาะอย่างได้โดยเหตุการณ์จำเพาะ (Event) หรือโดยการตั้งเวลาไว้ได้ (Time of Day) โดยไม่จำเป็นต้องพึ่งการติดต่อหรือจองช่องสัญญาณเพื่อติดต่อกับผู้ใช้บริการตลอดเวลา

(ค) ***Agent Communication*** : ในการประมวลผลที่เซิร์ฟเวอร์ต่าง ๆ เพื่อให้ได้ข้อมูลที่ผู้ใช้ต้องการนั้น ตัวโมบายล์เอเจนต์จำต้องมีการติดต่อกับส่วนอื่นๆ ของระบบเช่น ตัวผู้ใช้บริการเอง (User Interface Agent) หรือ ทรัพยากร (Resource) ของระบบ ซึ่งทรัพยากรของระบบสามารถมองได้ 2 แบบ คือ Remote และ Local ดังนั้นจะมีความหลากหลายของทรัพยากรระบบที่โมบายล์เอเจนต์จะต้องเข้าถึง เช่น ฐานข้อมูล (Database) และ Information Server เป็นต้น

(ง) ***Agent Cooperation*** : นอกจากทรัพยากรของระบบและผู้ใช้บริการที่โมบายล์เอเจนต์จะต้องติดต่อแล้ว โมบายล์เอเจนต์อาจจะต้องติดต่อกับโมบายล์เอเจนต์ด้วยกันเอง ซึ่งอาจหมายถึงการแลกเปลี่ยนข้อมูลของโมบายล์เอเจนต์ของผู้ใช้บริการคนอื่น หรือการติดต่อกับโมบายล์เอเจนต์ที่อยู่กลุ่มเดียวกัน สำหรับงานที่ใช้โมบายล์เอเจนต์มากกว่า 1 ตัว (Multi-Mobile Agent)

(จ) ***Agent Mobility*** : การประมวลผลที่เซิร์ฟเวอร์ของโมบายล์เอเจนต์นั้น บางครั้งเซิร์ฟเวอร์ไม่สามารถรองรับการทำงานของโมบายล์เอเจนต์ของผู้ใช้บริการที่มีในขณะนั้นได้อีก ดังนั้นตัวโมบายล์เอเจนต์จะต้องย้ายการประมวลผลไปยังเซิร์ฟเวอร์อื่นที่เหมาะสม โดยโมบายล์เอเจนต์อาจหยุดการประมวลผลขณะนั้นไว้แล้วไปประมวลผลต่อที่เซิร์ฟเวอร์อื่น (Suspend and Resume) แต่อย่างไรก็ตามในปัจจุบันภาษาที่ใช้ในการเขียนโมบายล์เอเจนต์นั้นไม่ทุกภาษาที่จะสามารถให้โมบายล์เอเจนต์หยุดการประมวลผลขณะนั้น แล้วเริ่มการประมวลผลต่อจากจุดที่หยุดไว้นั้นที่เซิร์ฟเวอร์อื่นได้ ดังเช่นภาษาจาวาเป็นต้น ซึ่งภาษาจาวาจะไม่อนุญาตให้โปรแกรมเมอร์เขียนโปรแกรม

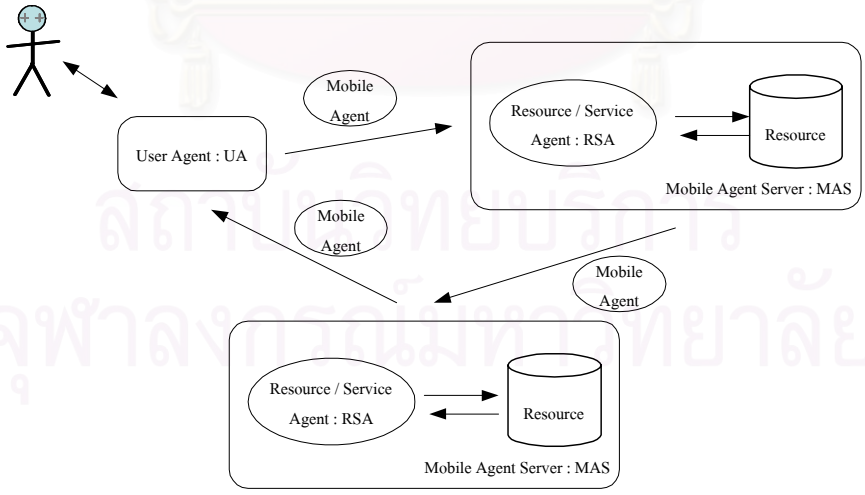
ที่เข้าถึงหน่วยความจำได้โดยตรง ดังนั้นคุณสมบัติข้อนี้จะถูกแบ่งออกเป็น 2 ประเภท ขึ้นตามลักษณะของภาษาที่ใช้ในการเขียนโมบายล์เอเจนต์

(จ1) ***Remote Execution*** เป็นการส่งโมบายล์เอเจนต์ไปยังเซิร์ฟเวอร์ปลายทาง เมื่อถึงจุดหมายปลายทางแล้วจะถูกกระตุ้นให้ประมวลผลตั้งแต่ต้นของชุดคำสั่งที่มีตามที่ได้ถูกกำหนดไว้ ดังนั้นโมบายล์เอเจนต์ที่มีคุณสมบัติของ Remote Execution จะไม่สามารถทำการ Suspended and Resume ได้ หากจะต้องเปลี่ยนการประมวลผลที่เซิร์ฟเวอร์อื่น ตัวโมบายล์เอเจนต์จะต้องเริ่มการประมวลผลใหม่ตั้งแต่ต้น

(จ2) ***Migration*** ในขณะที่โมบายล์เอเจนต์มีการประมวลผลอยู่นั้น หากโมบายล์เอเจนต์ไม่สามารถประมวลผลต่อที่เซิร์ฟเวอร์นั้นได้อีก (ไม่ว่าเหตุผลใดก็ตาม ยกเว้นเซิร์ฟเวอร์ดาวน์) ตัวโมบายล์เอเจนต์จะหยุดการประมวลผลขณะนั้นไว้แล้วส่งตัวเองไปยังเซิร์ฟเวอร์ที่เหมาะสม (หรืออาจถูกเซิร์ฟเวอร์ขณะนั้นส่งไปให้เซิร์ฟเวอร์อื่นแทน) โดยจะถูกส่งไปพร้อมกับ Registry Stack แล้วเริ่มการประมวลผลจากจุดที่หยุดไว้นั้นที่เซิร์ฟเวอร์ใหม่โดยใช้ข้อมูลจาก Registry Stack

2.3.2 โครงสร้างการทำงานโดยใช้โมบายล์เอเจนต์ (Mobile Agent's Infrastructure)

สำหรับโครงสร้างการทำงานของโมบายล์เอเจนต์ โดยพื้นฐานสามารถแบ่งออกได้เป็น 4 ส่วน ดังรูปที่ 2.4



รูปที่ 2.4 โครงสร้างการทำงานของโมบายล์เอเจนต์

(ก) ***User Agent (UA)*** : UA จะทำหน้าที่เป็นตัวกลางในการติดต่อระหว่างผู้ใช้บริการกับตัวโมบายล์เอเจนต์ โดยเริ่มต้นการทำงานผู้ใช้บริการจะกำหนดลักษณะของข้อมูลที่

ต้องการ และสิ่งต่าง ๆ ที่ต้องการให้กับ UA จากนั้น UA จะสร้างตัวโมบายล์เอเจนต์ขึ้นมาเพื่อให้สอดคล้องกับสิ่งที่ผู้ใช้บริการต้องการ แล้วส่งโมบายล์เอเจนต์ไปประมวลผลยังเซิร์ฟเวอร์ต่าง ๆ เพื่อให้ได้ข้อมูลที่ต้องการ เมื่อเสร็จสิ้นการประมวลผลที่เซิร์ฟเวอร์ต่าง ๆ แล้ว โมบายล์เอเจนต์จะกลับมายัง UA โดย UA จะแปลงข้อมูลที่ได้จากโมบายล์เอเจนต์ให้เป็นรูปแบบของข้อมูลที่ผู้ใช้บริการต้องการอีกทีหนึ่ง ดังนั้นหน้าที่สำคัญของ UA คือการกำหนดโค้ดที่เหมาะสมให้กับโมบายล์เอเจนต์เพื่อสอดคล้องกับความต้องการของผู้ใช้บริการให้มากที่สุด รวมไปถึงการนำเสนอรูปแบบของข้อมูลที่เหมาะสมให้กับผู้ใช้บริการ

(ข) ***Mobile Agent Server (MAS)*** : MAS ทำหน้าที่เป็นตัวกลางในการติดต่อระหว่างโมบายล์เอเจนต์และ RSA (Service/Resource Agent) โดยหน้าที่ที่สำคัญคือการกำหนดค่าสภาวะแวดล้อมต่าง ๆ ที่เหมาะสมสำหรับการประมวลผลให้กับโมบายล์เอเจนต์ หรือที่เรียกว่า AEE (Agent Execution Environment) เพื่อให้โมบายล์เอเจนต์ได้ใช้ทรัพยากรที่มีได้อย่างคุ้มค่า และเพื่อให้สามารถทำงานร่วมกับ RSA ได้อย่างมีประสิทธิภาพ

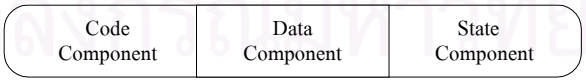
(ค) ***Service/Resource Agent (RSA)*** : เป็นส่วนที่ทำหน้าที่ในการเข้าถึงข้อมูลที่ต้องการในการประมวลผลของโมบายล์เอเจนต์ ซึ่งส่วนใหญ่คือฐานข้อมูลนั่นเอง โดยจะทำงานควบคู่กับ MAS

(ง) ***Mobile Agent (MA)*** : โมบายล์เอเจนต์เป็นโค้ดที่ถูกสร้างขึ้นโดยผู้ใช้บริการ (โดยผ่าน UA) เพื่อไปประมวลผลที่เซิร์ฟเวอร์ (โดยผ่าน MAS) แล้วส่งข้อมูลที่ได้กลับไปยังผู้ใช้บริการ โดยส่วนประกอบที่สำคัญของโมบายล์เอเจนต์จะมีอยู่ 3 ส่วนดังนี้ (รูปที่ 2.5)

(ง1) Code Component

(ง2) Data Component

(ง3) State Component



Mobile Agent's Context

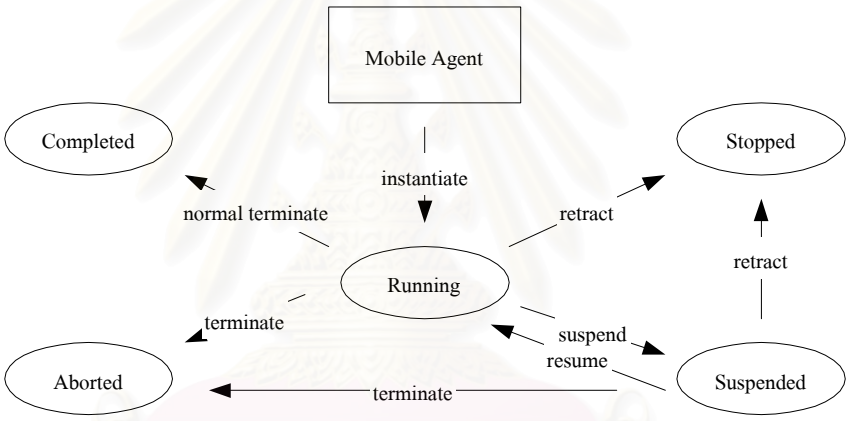
รูปที่ 2.5 ส่วนประกอบโมบายล์เอเจนต์

โดย Code Component ทำหน้าที่เก็บชุดคำสั่งที่ใช้ในการประมวลผล Data Component ทำหน้าที่เก็บข้อมูลที่จะถูกส่งกลับไปยังผู้ใช้บริการ และ State

Component ทำหน้าที่เก็บค่าสถานะขณะนั้นของโมบายล์เอเจนต์ ซึ่งสถานะของโมบายล์เอเจนต์จะสามารถแบ่งได้ 5 สถานะ คือ

- ***Running*** เป็นสถานะที่บ่งบอกว่าโมบายล์เอเจนต์กำลังทำการประมวลผลอยู่
- ***Suspended*** เป็นสถานะที่บ่งบอกว่าโมบายล์เอเจนต์หยุดการประมวลไว้ชั่วคราว เพื่อรอการเริ่มประมวลผลต่อจากจุดที่หยุดไว้ ณ เซิร์ฟเวอร์อื่น
- ***Stopped*** เป็นสถานะที่บ่งบอกว่าโมบายล์เอเจนต์ถูกกำหนดให้หยุดการประมวลผลอย่างถาวร และส่งข้อมูลที่มีอยู่ในขณะนั้นกลับไปยังผู้ใช้บริการ
- ***Aborted*** เป็นสถานะที่บ่งบอกว่าโมบายล์เอเจนต์ถูกยกเลิกการทำงาน
- **Completed** เป็นสถานะที่บ่งบอกว่าโมบายล์เอเจนต์เสร็จสิ้นการประมวลผลตามที่ได้ถูกกำหนดไว้ และส่งผลลัพธ์กลับไปยังผู้ใช้บริการ

สำหรับความสัมพันธ์ของสถานะต่าง ๆ สามารถอธิบายได้ดังรูปที่ 2.6



รูปที่ 2.6 วัฏจักรสถานะของโมบายล์เอเจนต์

2.3.3 ขั้นตอนการทำงานของโมบายล์เอเจนต์ (Mobile Agent's Operation)

สำหรับขั้นตอนการทำงานของโมบายล์เอเจนต์พิจารณาได้จากรูปที่ 2.4 ขั้นตอนการทำงานของโมบายล์เอเจนต์จะเริ่มต้นโดยผู้ใช้บริการกำหนดสิ่งที่ต้องการ ซึ่งอาจจะเป็นข้อมูลรูปภาพ หรือสภาพของปริมาณทราฟฟิกของเนตเวิร์กขณะนั้น (ซึ่งจะแตกต่างกันไปตามลักษณะของงานที่ใช้) ให้กับ UA จากนั้น UA จะแปลงความต้องการของผู้ใช้บริการให้เป็นโค้ดหรือตัวโมบายล์เอเจนต์ แล้วจะกำหนดเส้นทางเริ่มต้นให้กับโมบายล์เอเจนต์ โดยจะกำหนดลำดับก่อนหลังของเซิร์ฟเวอร์ที่โมบายล์เอเจนต์จะต้องไปประมวลผล (สำหรับวิธีการกำหนดเส้นทางเริ่มต้นอาจแตกต่างกันขึ้นกับระบบการใช้งาน) จากนั้นโมบายล์เอเจนต์จะไปประมวลผลที่เซิร์ฟเวอร์ต่าง ๆ ที่ได้กำหนดไว้ เมื่อโมบายล์เอเจนต์มาถึง MAS โมบายล์เอเจนต์จะถูกตรวจสอบตามขั้นตอนต่าง ๆ ของ MAS จากนั้น MAS จะสร้างสภาวะแวดล้อม AEE ที่จำเป็นต่อการประมวลผลให้กับ

โมบายล์เอเจนต์ เมื่อโมบายล์เอเจนต์เสร็จสิ้นการประมวลผลครบทุก MAS แล้ว โมบายล์เอเจนต์จะกลับไปยัง UA จากนั้น UA จะแปลงข้อมูลที่ได้จากโมบายล์เอเจนต์ให้อยู่ในรูปแบบที่เหมาะสมในการนำเสนอต่อผู้ใช้บริการ

2.4 เอกสารและงานวิจัยที่เกี่ยวข้อง

แนวคิดของการทำ Remote Execution มีมานานแล้วในอดีต ไม่ว่าจะเป็น “remote (batch) job”, function shipping” หรือ “remote evaluation” แต่เนื่องจากในอดีตที่ผ่านมานั้น ความสามารถที่จะทำการประมวลผลใกล้ทรัพยากร (locally to resource) ยังมีไม่เพียงพอ ประกอบกับปัญหาเรื่อง resource sharing และ load balancing (Magedanz and Eckardt, 1996) จึงทำให้แนวคิดการทำ Remote Execution ไม่เป็นที่นิยมเท่าที่ควร ปัจจุบันมีองค์กรต่าง ๆ ที่พยามยามจะจัดตั้งมาตรฐานของโมบายล์เอเจนต์ขึ้นมา ซึ่งมี 2 องค์กรคือ OMG (Object Management Group) และ FIPA (Foundation for Intelligent Physical Agents) นอกจากนี้ยังมีสถานบันการศึกษารวมไปถึงบริษัทเอกชนต่าง ๆ ให้ความสนใจกับเทคโนโลยีโมบายล์เอเจนต์เป็นอย่างมาก โดยมีการผลิตระบบโมบายเอเจนต์ที่เป็นของตัวเองขึ้นมา เช่น Ambit, Ara, D’Agent, General Magic, IBM Aglets, MARS, Mole, PageSpace, TucSoN, Agent Tcl, Concordia, Odyssey, Tacoma, Voyager และ Grasshopper เป็นต้น โดยภาษาที่ใช้เขียน (Programming Language) ส่วนใหญ่จะเป็น ทิคเคิ่ล (Tcl) , ซี (C) และ จาวา (Java) เนื่องจากภาษาจาวาไม่สามารถให้ผู้ใช้เข้าถึงส่วนของหน่วยความจำได้โดยตรง ดังนั้นการพัฒนาระบบโมบายล์เอเจนต์โดยใช้ภาษาจาวา จะไม่สามารถทำ Mobility แบบ Migration ได้ สามารถทำได้เพียง Remote Execution การประยุกต์ใช้โมบายล์เอเจนต์เข้ากับงานต่าง ๆ นั้นสามารถแบ่งได้เป็น 2 ประเภทคือ Asynchronous Information Exchange และ Pre-Establishment of real-time information exchange (Magedanz, Rothermel and Krause, 1996) ในการใช้งานประเภทแรกจะใช้โมบายล์เอเจนต์ในการถ่ายเทข้อมูลที่ต้องการบนเนตเวิร์ก ส่วนการใช้งานประเภทหลังจะใช้โมบายล์เอเจนต์ในการทำ signaling และ configuration สำหรับตัวอย่างการใช้งานของโมบายล์เอเจนต์ได้แก่ Network Management, Information Retrieval, QoS, Service Provisioning เป็นต้น ในการทำ Network Management (NM) (Uyless, 1995) มีการให้ความสนใจการใช้งานในด้านนี้มากอันเนื่องมาจาก การใช้งานด้าน NM สามารถใช้ประโยชน์จากโมบายล์เอเจนต์ได้ค่อนข้างเต็มที่ โดยจะใช้โมบายล์เอเจนต์พัฒนาการใช้งานของโพรโทคอลเอสเอ็นเอ็มพี (Simple Network Management Protocol : SNMP) และ ซีเอ็มไอพี (Common Management Information Protocol : CMIP) โดยงานวิจัยในปัจจุบันมีการนำเสนอตัวอย่างโครงสร้างการทำงานของ NM โดยใช้แนวคิดของโมบายล์เอเจนต์ (Akhil and Christine, 1998; Damianos et al., 1999; Gatot et al., 1998; Marcelo and Otto, 1999) มีการเปรียบเทียบโครงสร้างการทำงาน NM ที่มีในปัจจุบันเปรียบเทียบกับ ตัวอย่างโครงสร้างที่ใช้แนวคิดโมบายล์เอเจนต์ (Magedanz, Rothermel and

Krause, 1996; Magedanz and Eckardt, 1996; Wen-Shyen et al., 1999; Yu et al., 1999) นอกจากนี้มีการนำ CORBA หรือ Mobile Agent Platform ต่าง ๆ รวมไปถึง JAIN และ JINI (Sun Microsystems, 2000) (ซึ่งพัฒนามาจากจาวา) นำมาประยุกต์ใช้กับ NM (Akhil and Christine, 1998; Hongsong and Sun, 1999) QoS เป็นอีกเรื่องหนึ่งที่มีการใช้โมบายล์เอเจนต์ประยุกต์การใช้งานโดยลักษณะการใช้งานจะนำโมบายล์เอเจนต์ทำ QoS Negotiation เพื่อรองรับการติดต่อให้มีคุณภาพมากยิ่งขึ้น (Guedes et al., 1997; Mamadou and Tatsuo, 1998) สำหรับงานด้าน Information Retrieval (Dik, 1997; George et al., 1994; Jihoon, 1998; Kevin and Raman, 1999; Micheal and Merray, 1999; Micheal, Zlatko and Elizabeth, 1999; Mohan, 1994; Raymie, 1997; Tak and Hector, 1994; Van, 1999) เป็นการใช้งานอีกประเภทหนึ่งที่สามารถใช้ประโยชน์จากโมบายล์เอเจนต์ค่อนข้างจะเห็นได้อย่างชัดเจน โดยมีการนำ Pattern Recognition มาประยุกต์ใช้กับโมบายล์เอเจนต์เพื่อเพิ่มความสามารถในด้าน AI ให้กับโมบายล์เอเจนต์ (Hans-Gunter and Michael, 1999; Jihoon, 1999) รวมไปถึงการพัฒนา Search Engine โดยใช้โมบายล์เอเจนต์ (Robert et al., 1998) เป็นต้น นอกจากที่กล่าวมา มีการพัฒนาและใช้งานโมบายล์เอเจนต์ในด้านต่าง ๆ เช่น การส่งข้อมูลมัลติมีเดีย (Multimedia) โดยใช้โมบายล์เอเจนต์ (Karmouch and Horlait, 1998) การประยุกต์ใช้กับ Internet Telephony SIP (Handley et al., 1999) และ H.323 (DataBeam Corporation, 1998) โดยใช้โมบายล์เอเจนต์ในการพัฒนาเกตเวย์ (Gateway) เพื่อรองรับการบริการต่าง ๆ ของ Internet Telephony (Wenping and Radu, 2000) การทำ Active Network โดยใช้โมบายล์เอเจนต์เป็นโครงสร้างพื้นฐานในการรองรับการใช้งาน และเพิ่มโพรโทคอลใหม่ ๆ โดยการตกลงของผู้ใช้บริการแทนที่จะทำการเปลี่ยนแปลงที่ตัวระบบทั้งหมด นอกจากนี้มีการเสนอแนวทางการติดต่อกันระหว่างโมบายล์เอเจนต์ด้วยกันโดยเสนอแนวทางการติดต่อแบบต่าง ๆ ที่มีข้อดีข้อเสียแตกต่างกันไปตามสภาพแวดล้อมการใช้งาน (Giacomo, Letizia and Franco, 2000) หรือการติดต่อระหว่างโมบายล์เอเจนต์ และ โมบายล์เอเจนต์เซิร์ฟเวอร์ (Joachim et al., 1998) เป็นต้น

นอกจากนี้การใช้งานการสืบค้นข้อมูล มีการนำโมบายล์เอเจนต์มาประยุกต์ใช้ในงานด้านการสืบค้นข้อมูลภาพถ่ายทางการแพทย์ (Kevin and Raman, 1999) ซึ่งนับเป็นแนวคิดที่กำลังเป็นที่น่าสนใจในปัจจุบัน

สำหรับการทำงานของโมบายล์เอเจนต์จำเป็นจะต้องมีลิสต์เริ่มต้นหรือตารางลำดับของเซิร์ฟเวอร์ที่โมบายล์เอเจนต์จะต้องไปประมวลผล ซึ่งมีการเสนอผลงานอัลกอริทึมในการกำหนดเส้นทางเริ่มต้นให้กับโมบายล์เอเจนต์เหล่านี้ (Brian, 1999; Sun Microsystems, 2000) โดยใช้อัลกอริทึมมาตรฐาน (Ronald, 1998) ที่มีในปัจจุบันมาประยุกต์ใช้เพื่อให้สอดคล้องกับการทำงานของโมบายล์เอเจนต์

บทที่ 3

ระบบการสืบค้นข้อมูล

3.1 กล่าวนำ

ปัจจุบันข้อมูลที่สามารถสืบค้นได้ในเนตเวิร์กนั้นมีปริมาณเพิ่มขึ้นมากในแต่ละวัน แต่ละปีจะมีเอกสารและบทความทางวิชาการ รวมไปถึงหนังสือต่าง ๆ ที่ถูกตีพิมพ์ออกมานับล้านฉบับ และมีแนวโน้มที่จะมากขึ้นในอนาคต ดังนั้นเพื่อให้ผู้ใช้บริการสามารถเข้าถึงข้อมูลที่มีในฐานข้อมูลได้นั้น ข้อมูลจำเป็นจะต้องถูกจัดเก็บในฐานข้อมูลอย่างมีระบบเสียก่อน โดยข้อมูล เช่น เอกสารวิชาการ หนังสือ นิตยสาร จะต้องถูกนำมาแปลงให้อยู่ในรูปดิจิตอล (Digital) แล้วนำมาทำดัชนี (Index) เพื่อใช้สะดวกในการทำการสืบค้นข้อมูล ดังนั้นจะเห็นได้ว่าในระบบการสืบค้นข้อมูลจะประกอบไปด้วยส่วนสำคัญ 2 ส่วนคือการจัดเก็บฐานข้อมูล และการสืบค้นข้อมูล โดยในการจัดเก็บข้อมูล ข้อมูลที่ถูกนำมาใช้ทำเป็นดัชนีได้แก่ ชื่อเรื่อง ชื่อผู้แต่ง บทคัดย่อ คำสำคัญ ตัวแบ่งหัวเรื่อง (Subject Classification) เป็นต้น ในการทำดัชนีของฐานข้อมูลจะกระทำเพียงครั้งเดียวตอนเริ่มต้น หรือเมื่อมีการเปลี่ยนแปลงฐานข้อมูล จากนั้นเมื่อผู้ใช้บริการต้องการสืบค้นข้อมูล ผู้ใช้บริการจะป้อนคีย์เวิร์ด (Keyword) ซึ่งอาจจะเป็นคำ ข้อความ หรือประโยค แล้วระบบจะทำการแปลงคีย์เวิร์ดที่ได้นี้ให้อยู่ในรูปแบบที่เหมาะสมเพื่อใช้ในการสืบค้นข้อมูลเทียบกับดัชนีที่มีในฐานข้อมูลได้ (Michael et al., 1999) ดังนั้นเนื้อหาในบทนี้จะกล่าวถึงระบบการสืบค้นข้อมูล ที่ประกอบไปด้วยการจัดเก็บฐานข้อมูล และการสืบค้นข้อมูล ดังนี้

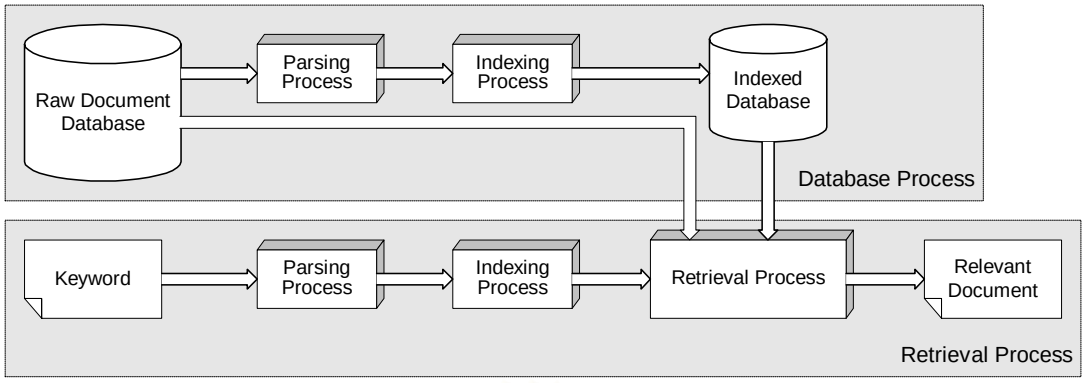
3.2 หลักการทำงานระบบการสืบค้นข้อมูล

กระบวนการทำงานการสืบค้นข้อมูลสามารถแบ่งได้ 2 ส่วนคือ

ก.) การจัดเก็บฐานข้อมูล

ข.) การสืบค้นข้อมูล

โดยการทำงานของทั้ง 2 ส่วนสามารถอธิบายได้ในรูปที่ 3.1



รูปที่ 3.1 ขั้นตอนการสืบค้นข้อมูล

การจัดเก็บฐานข้อมูลเป็นการแปลงข้อมูลที่มีอยู่ให้อยู่ในรูปแบบเฉพาะที่เหมาะสมสำหรับการสืบค้นข้อมูล ซึ่งกระบวนการนี้จะต้องผ่านขั้นตอนที่สำคัญ 2 ขั้นตอน คือ Parsing Process และ Indexing Process และเช่นเดียวกันกับกระบวนการสืบค้นข้อมูล คีย์เวิร์ดที่ผู้ใช้บริการป้อนเข้ามานั้นจะถูกแปลงให้อยู่ในรูปแบบเฉพาะที่ใช้สำหรับการสืบค้นข้อมูล ซึ่งจะต้องผ่านขั้นตอน Parsing Process และ Indexing Process เช่นเดียวกัน เมื่อได้ข้อมูลที่ถูกแปลงให้อยู่ในรูปแบบเฉพาะสำหรับการสืบค้นข้อมูลแล้ว คีย์เวิร์ดที่ถูกแปลงรูปแล้วนี้จะถูกนำไปประมวลผลผ่านขั้นตอน Retrieval Process เพื่อเทียบกับข้อมูลที่มีในฐานข้อมูล จากรายละเอียดขั้นต้น ขั้นตอนการทำงานที่สำคัญของการสืบค้นข้อมูลประกอบไปด้วยส่วนสำคัญ 3 ส่วนดังนี้

ก.) Parsing Process

ข.) Indexing Process

ค.) Retrieval Process

3.2.1 Parsing Process

ในขั้นตอนการสืบค้นข้อมูล ข้อมูล (Document) จะถูกมองเป็นกลุ่มของคำหรือเทอม (Terms) ที่มีทั้งเทอมที่มีความสำคัญ และไม่สำคัญ ดังนั้นก่อนที่จะนำข้อมูลไปผ่านกระบวนการอื่นนั้นจะต้องทำการตัดคำหรือเทอมที่ไม่สำคัญทิ้งไปเสียก่อน หรือที่เรียกว่า Parsing สำหรับเทอมม่สำคัญจะแบ่งออกเป็น 2 ชนิดคือ

(ก) ***Stop Word*** : เทอมที่เป็น Stop Word คือเทอมไม่ได้ใช้ในการทำการสืบค้นข้อมูล ซึ่งอาจเป็นเทอมพื้น ๆ เกินไป หรือเทอมที่ไม่ให้ความหมายอะไร ตัวอย่างในภาษาอังกฤษ เช่น a, an, the, about, and etc. เป็นต้น ในการทำงาน ตัวโปรแกรมจะทำการตรวจสอบข้อมูลเป็นเทอมต่อเทอมแล้วเปรียบเทียบว่าเทอมที่กำลังตรวจสอบอยู่นั้นว่า

อยู่ในลิสต์ของ Stop Word (Stop List) หรือไม่ ถ้าอยู่ในลิสต์ก็จะทำการตัดคำ ๆ นั้นออก โดยสำหรับตัวอย่างของ Stop List สามารถดูได้ในภาคผนวก ก.

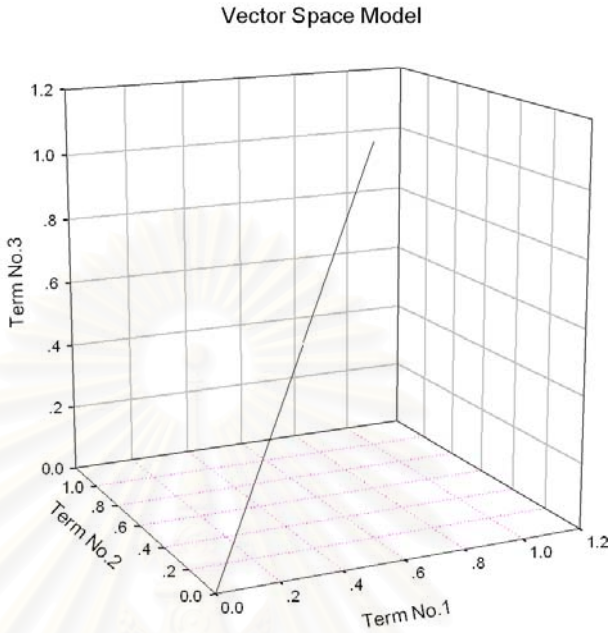
(ข) ***Word Stemming*** : ในข้อมูลนอกจากจะมีเทอมเป็น Stop Word แล้ว เทอมบางเทอมสามารถให้ความหมายเหมือนกันได้ กล่าวคือเทอมบางเทอม มาจากรากศัพท์คำเดียวกัน หรือเป็นเทอมที่ถูกใส่ prefix หรือ suffix เพิ่มเข้าไป ดังนั้นแทนที่จะเรามองเป็นเทอมที่แตกต่างกัน แต่จะถูกมองเป็นเทอม เดียวกัน ตัวอย่างเช่น “play” และ “playing” จะถูกมองเป็นเทอมเดียวกันคือคำว่า “play” หรือ “classify” และ “classification” จะถูกมองเป็นคำว่า “classify” เป็นต้น แต่อย่างไรก็ตามเทอมบางเทอมไม่จำเป็นจะต้องสอดคล้องตามกฎเหล่านี้เสมอไป ตัวอย่างเช่นคำว่า “equal” บางโปรแกรมที่เขียนอัลกอริทึมที่ไม่รอบคอบอาจมอง “ual” ใน “eaual” เป็น suffix แล้วทำการตัดทิ้งเหลือเพียง “eq” ซึ่งไม่มีความหมายอะไร หรือ “confidential” กับ “confident” ซึ่งให้ความหมายที่แตกต่างกัน ดังนั้นสิ่งเหล่านี้เป็นหน้าที่ของโปรแกรมเมอร์ที่จะต้องเขียนโปรแกรมให้รัดกุมกับข้อยกเว้นต่าง ๆ เหล่านี้

นอกเหนือจากที่กล่าวมาแล้วในเรื่องของ Stop Word และ Word Stemming ยังมีข้อยกเว้นอีกหลายประการเช่น เทอมบางเทอมสามารถตีความได้หลายความหมาย อย่างเช่นคำว่า “bank” ซึ่งอาจหมายถึงเทอมที่อยู่ในหมวดของ “computer memory” หรือเทอมที่อยู่ในหมวดการเงิน “financial” เป็นต้น ซึ่งเรียกเทอมเหล่านี้ว่า *polysemy* หรืออีกตัวอย่างหนึ่งเช่นเทอมบางเทอมถึงแม้ไม่ได้มากจากรากศัพท์เดียวกันแต่มีความหมายเหมือนกันเช่น “myocardial infarctions” กับ “heart attack” ซึ่งมีความหมายเหมือนกันแต่ระดับของการใช้ภาษาแตกต่างกัน คำเหล่านี้เรียกว่า *synonymy* จากที่กล่าวมาแล้วนั้นเป็นหน้าที่ของโปรแกรมเมอร์ที่จะต้องเขียนอัลกอริทึมที่ชัดเจนและรัดกุมเพื่อจัดการกับข้อยกเว้นเหล่านี้ (Brian et al, 1999)

3.2.2 Indexing Process

เมื่อข้อมูลผ่านกระบวนการ Parsing Process แล้ว ข้อมูลจะถูกมองเป็นกลุ่มของเทอมที่มีเทอมสำคัญ (ที่ไม่มี Stop Word) สำหรับ Indexing Process จะนำข้อมูลมาผ่านกระบวนการเพื่อทำดัชนีให้กับแต่ละข้อมูลนั้น ๆ เพื่อนำไปใช้ใน Retrieval Process

โดยพื้นฐานการทำดัชนีข้อมูลคือการกำหนดค่าน้ำหนัก (Weight) ให้กับแต่ละเทอมในข้อมูลนั่นเอง ดังนั้นข้อมูลจะประกอบไปด้วยชุดของเทอมกับค่าน้ำหนักของเทอมนั้น ๆ และจะสามารถมองเป็นเวกเตอร์ (Vector) ได้ โดยมองเป็นเวกเตอร์ที่มีมิติเท่ากับจำนวนเทอมที่มีทั้งหมด เรียกว่า Vector Space Model (VSM) ดังรูปที่ 3.2



รูปที่ 3.2 Vector Space Model

แต่ในการใช้งานจริงเวกเตอร์จะถูกทำเป็นเมตริกซ์ (matrix) ซึ่งมีจำนวนแถวเท่ากับจำนวนเทอม และจำนวนคอลัมน์เท่ากับจำนวนเวกเตอร์ เรียกว่า Term-by-Document Matrix ดังนั้นถ้าข้อมูลไม่มีเทอมใด สมาชิกในเมตริกซ์ที่ตำแหน่งนั้นจะเท่ากับ 0 และถ้ามีเทอมใดค่าสมาชิกในตำแหน่งนั้นจะเป็นค่าน้ำหนักของเทอมนั้น ๆ ดังตัวอย่างเช่น สมมติว่ามีข้อมูล D1, D2, D3 และกำหนดให้ข้อมูล D1 มีเทอมสำคัญ a, b, c, d, e ข้อมูล D2 มีเทอมสำคัญ a, b และข้อมูล D3 มีเทอมสำคัญ c, e, f, g, h, i , j ดังนี้

$$D1 = ((a,0.46),(b,0.14),(c,0.17),(d,0.62),(e,0.59))$$
$$D2 = ((a,0.95),(b,0.30))$$
$$D3 = ((c,0.14),(e,0.49),(f,0.17),(g,0.42),(h,0.11),(i,0.10),(j,0.72))$$

จะเห็นได้ว่า D1 ที่เทอม a, b, c, d และ e จะมีค่าตัวเลขซึ่งก็คือค่าน้ำหนักของแต่ละเทอม และเช่นเดียวกับข้อมูล D2 และ D3 ดังนั้นจะสามารถเขียนเป็นเมตริกซ์ได้ดังนี้

$$\begin{bmatrix} 0.46 & 0.95 & 0 \\ 0.14 & 0.30 & 0 \\ 0.17 & 0 & 0.14 \\ 0.62 & 0 & 0 \\ 0.59 & 0 & 0.49 \\ 0 & 0 & 0.17 \\ 0 & 0 & 0.42 \\ 0 & 0 & 0.11 \\ 0 & 0 & 0.10 \\ 0 & 0 & 0.72 \end{bmatrix}$$

จากเมตริกซ์จะเห็นได้ว่าจะมีจำนวนแถวเท่ากับจำนวนเทอม (a ถึง j) และมีจำนวนคอลัมน์เท่ากับจำนวนข้อมูล (D1 ถึง D3) โดย D1 มีแต่เทอม a, b, c, d, e ดังนั้นค่าสมาชิกในเมตริกซ์จึงเป็นค่าน้ำหนักของแต่ละเทอม และเนื่องจากไม่มีเทอม f, g, h, i และ j ดังนั้นสมาชิกในเมตริกซ์จึงเป็น 0 ดังจะเห็นได้จากสมาชิกในคอลัมน์ที่ 1

สำหรับวัตถุประสงค์ของการใช้ค่าน้ำหนักนั่น เพื่อให้เกิดประสิทธิภาพในการสืบค้นข้อมูลที่มากขึ้น โดยจะกำหนดให้ค่าสมาชิกในเมตริกซ์ (ค่าน้ำหนัก) เป็นผลคูณของแปร 3 ตัว คือ

$$a_{ij} = l_{ij} \cdot g_i \cdot d_j$$

โดย l_{ij} (Local Weight) เป็นค่าน้ำหนักของเทอมที่ i ปรากฏในข้อมูลที่ j

g_i (Global Weight) เป็นค่าน้ำหนักของเทอมที่ i ที่ปรากฏในข้อมูลที่มีทั้งหมด

d_j (Document Normalization) เป็นตัวกำหนดลักษณะการทำ normalize ข้อมูลที่มีอยู่ในเมตริกซ์

ซึ่งค่า l_{ij}, g_i, d_j จะมีค่าต่าง ๆ ดังตารางต่อไปนี้ (ตารางที่ 3.1-3.3)

ตารางที่ 3.1 ค่า l_{ij}

Symbol	Name	Formula
b	Binary	$X(f_{ij})$
l	Logarithmic	$\log(1+f_{ij})$
n	Augmented Normalized Term Frequency	$\left(X(f_{ij})+\left(f_{ij}/\max_k f_{kj}\right)\right)/2$
t	Term Frequency	f_{ij}

เมื่อ f_{ij} คือจำนวนครั้งที่เทอมปรากฏในข้อมูลชุดที่ j

$$X(r) = \begin{cases} 1 & if\ r > 0 \\ 0 & if\ r = 0 \end{cases}$$

ตารางที่ 3.2 ค่า g_i

Symbol	Name	Formula
x	None	1
e	Entropy	$1+\left(\sum_j \left(p_{ij} \log(p_{ij}) \Big/ \log n \right)\right)$
f	Inverse Document Frequency (idf)	$\log\left(n \Big/ \sum_j X(f_{ij}) \right)$
g	GfIdf	$\left(\sum_j (f_{ij})\right) \Big/ \sum_j X(f_{ij})$
n	Normal	$1 \Big/ \sqrt{\sum_j f_{ij}^2}$
p	Probabilitic Inverse	$\log\left(\left(n - \sum_j X(f_{ij}) \right) \Big/ \sum_j X(f_{ij}) \right)$

เมื่อ $p_{ij} = f_{ij} \Big/ \sum_j f_{ij}$

ตารางที่ 3.3 ค่า d_j

Symbol	Name	Formula
x	None	1
c	Cosine	$\left(\sum_j (g_i l_{ij})^2\right)^{-1/2}$

ซึ่งค่า l_{ij}, g_i, d_j เราจะเลือกใช้ค่าใดนั้นขึ้นอยู่กับลักษณะของข้อมูลตัวอย่างเช่น ข้อมูลประเภท Technical Report หรือ journal articles จะใช้รูปแบบ $a_{ij} = n \cdot x \cdot x$ จะใช้ค่าน้ำหนักเป็น $a_{ij} = \left(X(f_{ij}) + \left(f_{ij} \Big/ \max_k f_{kj} \right) \right) \Big/ 2$ เป็นต้น (Michael and Merray, 1999)

3.2.3 Retrieval Process

ในกระบวนการจัดเก็บฐานข้อมูล เมื่อข้อมูลผ่าน Parsing Process และ Indexing Process แล้ว ฐานข้อมูลจะประกอบไปด้วยดัชนีของข้อมูลที่อยู่ในรูปเมตริกซ์ที่มีสมาชิกเป็นค่าน้ำหนักต่าง ๆ และเช่นเดียวกันในกระบวนการสืบค้นข้อมูล เมื่อผู้ใช้บริการส่งคีย์เวิร์ดมา ระบบจะนำคีย์เวิร์ดมาผ่าน Parsing Process และ Indexing Process จะได้เมตริกซ์ที่มี 1 คอลัมน์ (มองคีย์เวิร์ดเสมือนข้อมูล 1 ชุด หรืออาจจะเรียกว่า Query Vector) แล้วนำเมตริกซ์ของคีย์ที่ได้ไปผ่าน similarity function เทียบกับ Term-by-Document Matrix เพื่อให้ได้ค่าข้อมูลเพื่อใช้ในการเรียกข้อมูลที่ผู้ใช้ต้องการ สำหรับ similarity function ก็คือการผ่าน cosine function นั่นเอง ดังสมการที่ 3.1 (สำหรับขั้นตอนการทำงานจะอธิบายในตัวอย่างหัวข้อ ตัวอย่างการทำงาน)

$$\cos\theta_j = \frac{a_j^T q}{\|d_j\|_2 \ \|q\|_2} = \frac{\sum_{i=1}^{m} a_{ij} q_i}{\sqrt{\sum_{i=1}^{m} a_{ij}^2}\sqrt{\sum_{i=1}^{m} q_i^2}}$$

เมื่อ a_{ij} คือค่าสมาชิกใน Term-by-Document Matrix

และ q_i คือค่าสมาชิกใน Keyword Matrix

3.3 ตัวอย่างการทำงานโดยใช้ Vector Space Model

กำหนดให้ข้อมูลมี 7 ชุดดังนี้

ตารางที่ 3.4 ตัวอย่างชุดข้อมูล

No.	Title
1.	*Infant & Toddler First Aid*
2.	*Babies & Children's Room (for Your Home)*
3.	*Chile Safety at Home*
4.	*Your Baby's Health and Safety : From Infant to Toddler*
5.	*Baby Proofing Basics*
6.	*Your Guide to Easy Rust Proofing*
7.	*Beanie Babies Collector's Guide*

เมื่อนำข้อมูลไปผ่าน Parsing Process (Stop Word) จะได้ข้อมูลดังนี้

ตารางที่ 3.5 ตัวอย่างข้อมูลที่ผ่าน Parsing Process (Stop Word)

D1: Infant, Toddler
D2: Babies, Children's
D3: Child, Safety
D4: Baby's, Health, Safety, Toddler
D5: Baby, Proofing
D6: Guide, Proofing
D7: Babies, Guide

จากนั้นเมื่อนำข้อมูลไปผ่าน Parsing Process (Word Stemming) จะได้ข้อมูลดังนี้

ตารางที่ 3.6 ตัวอย่างข้อมูลที่ผ่าน Parsing Process (Word Stemming)

T1 : Bab(y,ies,y's)
T2: Child(ren's)
T3: Guide
T4: Health
T5: Home
T6: Infant
T7: Proofing
T8: Safety
T9: Toddler

เมื่อนำผลลัพธ์ที่ได้จาก Parsing Process ไปผ่าน Indexing Process จะได้ Term-by-Document Matrix ดังนี้ (เมื่อกำหนดให้ค่าน้ำหนักมีค่าเป็น 1)

$$\tilde{A} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

และเมื่อทำเป็น unit column (normalize) จะได้ unit column matrix ดังนี้

$$\mathbf{A} = \begin{bmatrix} 0 & 0.5774 & 0 & 0.4472 & 0.7071 & 0 & 0.7071 \\ 0 & 0.5774 & 0.5774 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.7071 & 0.7071 \\ 0 & 0 & 0 & 0.4472 & 0 & 0 & 0 \\ 0 & 0.5774 & 0.5774 & 0 & 0 & 0 & 0 \\ 0.7071 & 0 & 0 & 0.4472 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.7071 & 0.7071 & 0 \\ 0 & 0 & 0.5774 & 0.4472 & 0 & 0 & 0 \\ 0.7071 & 0 & 0 & 0.4472 & 0 & 0 & 0 \end{bmatrix}$$

สมมติว่าผู้ใช้บริการส่งคีย์เวิร์ดและนำมาผ่าน Parsing Process และ Indexing Process ได้ query vector ดังนี้

query : “Child Proofing”

query vector : $q = (0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0)^T$

เมื่อนำไปผ่าน similarity function จะได้ค่า $\cos\theta_j$ ดังนี้

$$\cos\theta_1 = 0.000$$
$$\cos\theta_2 = 0.408$$
$$\cos\theta_3 = 0.408$$
$$\cos\theta_4 = 0.000$$
$$\cos\theta_5 = 0.500$$
$$\cos\theta_6 = 0.500$$
$$\cos\theta_7 = 0.000$$

ดังนั้นจะเห็นได้ว่าข้อมูลที่ 5 และ 6 ให้ค่า similarity function มากที่สุด ดังนั้นในฐานข้อมูลที่มีอยู่ ข้อมูลที่สอดคล้องกับความต้องการของผู้ใช้มากที่สุดคือข้อมูลชุดที่ 5 และ 6

จากรายละเอียดข้างต้น การคำนวณ similarity function เทียบกับ Term-by-Document Matrix จะเห็นได้ว่า ถ้าเมตริกซ์มีขนาด n x m จะต้องมีการคำนวณทั้งหมด n x m ครั้ง และนอกจากนี้ ในทางปฏิบัติ จำนวนข้อมูลมีหน่วยเป็นหลักล้าน และเทอมมีหน่วยเป็นหลักหมื่น หรือกรณีที่ฐานข้อมูลมีขนาดใหญ่ ซึ่งมีผลทำให้การทำ VSM ไม่สามารถทำได้ในทางปฏิบัติ เนื่องจากจำเป็นต้องใช้หน่วยความจำมหาศาล ดังนั้นจึงมีวิธีการแปลง Term-by-Document Matrix ให้เป็น Inverted List แล้วนำ Inverted List มาใช้แทน VSM ดังหัวข้อต่อไป

3.4 ตัวอย่างการทำงานโดยใช้ Inverted List

Inverted List จะมีหลักการการทำงานโดยสร้างลิสต์ขึ้นมาอีกชุดหนึ่งจาก Term-by-Document Matrix โดยในลิสต์จะประกอบไปด้วยลิสต์ของเทอมต่าง ๆ ซึ่งแต่ละเทอมจะมีค่า Document ID ที่มีเทอมนั้นอยู่กับค่าน้ำหนักของเทอมในข้อมูลนั้น ๆ ดังนั้นเมื่อได้ query vector จากคีย์เวิร์ดจากผู้ใช้บริการ โปรแกรมจะทำการตรวจสอบแต่ละเทอมใน query vector เทียบกับ Inverted List ว่าเทอมที่มีใน query vector นั้นมีอยู่ในข้อมูลใดบ้าง แล้วจึงคำนวณค่า similarity function เฉพาะข้อมูลนั้น ๆ ดังตัวอย่าง

กำหนดให้มี ข้อมูล D1-D3 และมีเทอม a ถึง j (จากตัวอย่างในหัวข้อ 3.2.2) ดังนี้

$$D1 = ((a,0.46),(b,0.14),(c,0.17),(d,0.62),(e,0.59))$$
$$D2 = ((a,0.95),(b,0.30))$$
$$D3 = ((c,0.14),(e,0.49),(f,0.17),(g,0.42),(h,0.11),(i,0.10),(j,0.72))$$

เมื่อทำ VSM จะได้เมตริกซ์ดังนี้

$$\begin{bmatrix} 0.46 & 0.95 & 0 \\ 0.14 & 0.30 & 0 \\ 0.17 & 0 & 0.14 \\ 0.62 & 0 & 0 \\ 0.59 & 0 & 0.49 \\ 0 & 0 & 0.17 \\ 0 & 0 & 0.42 \\ 0 & 0 & 0.11 \\ 0 & 0 & 0.10 \\ 0 & 0 & 0.72 \end{bmatrix}$$

เมื่อนำแต่ละเทอมมาสร้าง Inverted List จะได้ผลลัพธ์ดังนี้

a	(D1,0.46) (D2,0.14)
b	(D1,0.14) (D2,0.30)
c	(D1,0.17) (D3,0.14)
d	(D1,0.62)
e	(D1,0.59) (D3,0.49)
f	(D3,0.17)
g	(D3,0.42)
h	(D3,0.11)
i	(D3,0.10)
l	(D3,0.72)

เมื่อกำหนดให้ query vector มีค่าดังนี้

$$q = ((b,0.15),(d,0.32),(f,0.21),(h,0.14),(j,0.90))$$

จากนั้นโปรแกรมจะทำการตรวจสอบกับ Inverted List และดูที่เทอม b, d, f, h และ j ดังนั้นการประมวลผลที่โปรแกรมจะทำจริงมี 6 ขั้นตอนคือ

1. (b,0.15) x (D1,0.14)
2. (b,0.15) x (D2,0.30)
3. (d,0.32) x (D1,0.62)
4. (f,0.21) x (D3,0.17)
5. (h,0.14) x (D3,0.11)
6. (j,0.90) x (D3,0.72)

เมื่อเปรียบเทียบกับ VSM จะต้องทำการคูณทั้งหมด 30 ครั้งซึ่งมีความแตกต่างกันอย่างเห็นได้ชัดเจน ดังนั้นจึงทำให้ในปัจจุบันการทำงานการสืบค้นข้อมูลส่วนใหญ่จะมีลักษณะการทำงานแบบ Inverted List เป็นหลัก สำหรับอัลกอริทึมการกำหนดเส้นทางให้กับโมบายล์เอเจนต์จะกล่าวถึงในบทต่อไป

บทที่ 4

อัลกอริทึมการกำหนดเส้นทาง

4.1 กล่าวนำ

ในการทำงานโดยใช้โมบายล์เอเจนต์ ปัจจัยหนึ่งที่มีผลต่อสมรรถนะการทำงานและประสิทธิภาพการทำงานของระบบคือ การกำหนดเส้นทางให้กับโมบายล์เอเจนต์ เนื่องจากโมบายล์เอเจนต์สามารถเคลื่อนย้ายไปประมวลผลที่เซิร์ฟเวอร์ใด ๆ ก็ได้ในเนตเวิร์ก ทำให้ในตอนเริ่มต้นของการทำงาน จำเป็นจะต้องกำหนดเส้นทาง หรือลำดับของเซิร์ฟเวอร์ให้โมบายล์เอเจนต์เสียก่อน ดังนั้นวิธีการกำหนดเส้นทางให้กับโมบายล์เอเจนต์คือ วิธีการจัดลำดับของโนด (เซิร์ฟเวอร์) ในเนตเวิร์กที่โมบายล์เอเจนต์ใช้ในการประมวลผลโดยมีค่าเวลาหน่วง (Delay Time) เป็นค่าต้นทุน (Cost) ที่ใช้ในฟังก์ชันวัตถุประสงค์ (Objective Function) นั่นเอง ซึ่งการกำหนดเส้นทางนั้น ถ้าเป็นเส้นทางที่ใช้โมบายล์เอเจนต์เพียงตัวเดียว อาจนำวิธี *"ปัญหาการเดินทางของเซลส์แมน"* (Traveling Salesman Problem : TSP) นำมาใช้กำหนดเส้นทางได้ แต่เนื่องจากการประมวลผลที่เซิร์ฟเวอร์ต่าง ๆ ของโมบายล์เอเจนต์สามารถใช้โมบายล์เอเจนต์ได้มากกว่าหนึ่งตัว(Mulit-Mobile Agent) เพื่อให้เกิดการประมวลที่ทั่วถึง ดังนั้นกรณีที่ใช้โมบายล์เอเจนต์มากกว่าหนึ่งตัวจะไม่สามารถใช้วิธีการของ TSP ได้เลย เพราะว่า TSP คือปัญหาการกำหนดเส้นทางที่เคลื่อนที่ผ่านโนดต่าง ๆ โดยใช้เส้นทางเดียว (Single Path) และไม่มีการย้อนกลับ (Non-Returning Characteristic) หรือผ่านโนดที่เคยเคลื่อนที่ผ่านไปแล้ว แต่ปัญหาการกำหนดเส้นทางของโมบายล์เอเจนต์ไม่จำเป็นต้องเป็นเส้นทางเดียว สามารถเกิดขึ้นได้หลายเส้นทาง และเนื่องจากสามารถใช้โมบายล์เอเจนต์ได้มากกว่าหนึ่งตัว แต่มีจุดเริ่มต้นที่จุดเดียวกันคือโนดที่เริ่มต้นปล่อยตัวโมบายล์เอเจนต์ ดังนั้นเพื่อที่จะสามารถกำหนดเส้นทางให้กับโมบายล์เอเจนต์ได้จึงต้องนำวิธี TSP มาดัดแปลงประยุกต์ใช้ โดยเนื้อหาในบทนี้จะอธิบายการนำ TSP มาประยุกต์ใช้ในการกำหนดเส้นทางให้กับโมบายล์เอเจนต์ โดยใช้อัลกอริทึม Brute Force Search, Simulated Annealing, Tabu Search และ Modified Compact Genetic Algorithm

เนื่องจากการประยุกต์ TSP มาใช้ในการกำหนดเส้นทางให้กับโมบายล์เอเจนต์มีได้หลายวิธี ดังนั้นเนื้อหาในบทนี้จะเสนอวิธีนำอัลกอริทึมต่าง ๆ มาประยุกต์ใช้ในการกำหนดเส้นทางให้กับโมบายล์เอเจนต์ และเนื่องจากวิทยานิพนธ์ฉบับนี้ได้แสดงการเปรียบเทียบข้อเด่นข้อด้อยของแต่ละวิธี ดังนั้นเนื้อหาในบทนี้จะเริ่มต้นด้วยการกล่าวถึงรายละเอียดการกำหนดค่าต้นทุนที่ใช้ใน

การกำหนดค่าลิงก์ที่เชื่อมต่อโนดต่าง ๆ ในเนตเวิร์กเพื่อใช้ในการเปรียบเทียบวิธีการต่าง ๆ ที่ใช้กำหนดเส้นทาง ต่อจากนั้นจะอธิบายถึงวิธีการต่าง ๆ ที่นำมาใช้กำหนดเส้นทาง

4.2 การกำหนดค่าเมตริกซ์ต้นทุน

ในการทดสอบและเปรียบเทียบอัลกอริทึมการกำหนดเส้นทางต่าง ๆ นั้น จำเป็นจะต้องมีค่าต้นทุนเพื่อเป็นเกณฑ์สำหรับการเปรียบเทียบ ดังนั้นเนื้อหาในหัวข้อนี้จะกล่าวถึงวิธีการกำหนดค่าต้นทุนของเนตเวิร์กจำลอง

ในเนตเวิร์กจำลองจะประกอบไปด้วยโนดต่าง ๆ โดยในที่นี้จะกำหนดให้มีโนดเซิร์ฟเวอร์สำหรับการประมวลผลของโมบายล์เอเจนต์จำนวน n โนด และโนดของผู้ใช้บริการที่ทำหน้าที่ปล่อยตัวโมบายล์เอเจนต์ไปประมวลผลยังโนดเซิร์ฟเวอร์ต่าง ๆ เป็นโนดที่ศูนย์ รวมเป็น n+1 โนด และแต่ละโนดจะถูกเชื่อมโยงกันด้วยลิงก์ (Link) โดยจะมีลักษณะเชื่อมต่อกันหมด (Full Inter-Connected) และแต่ละลิงก์จะมีค่าต้นทุนเป็นค่าประจำแต่ละลิงก์นั้น ๆ ดังนั้นสมมติว่า ถ้าจะกำหนดให้โนดใดไม่ต่อกัน ค่าต้นทุนของลิงก์ที่เชื่อมต่อโนดคู่นั้นจะถูกกำหนดให้มีค่ามาก ๆ

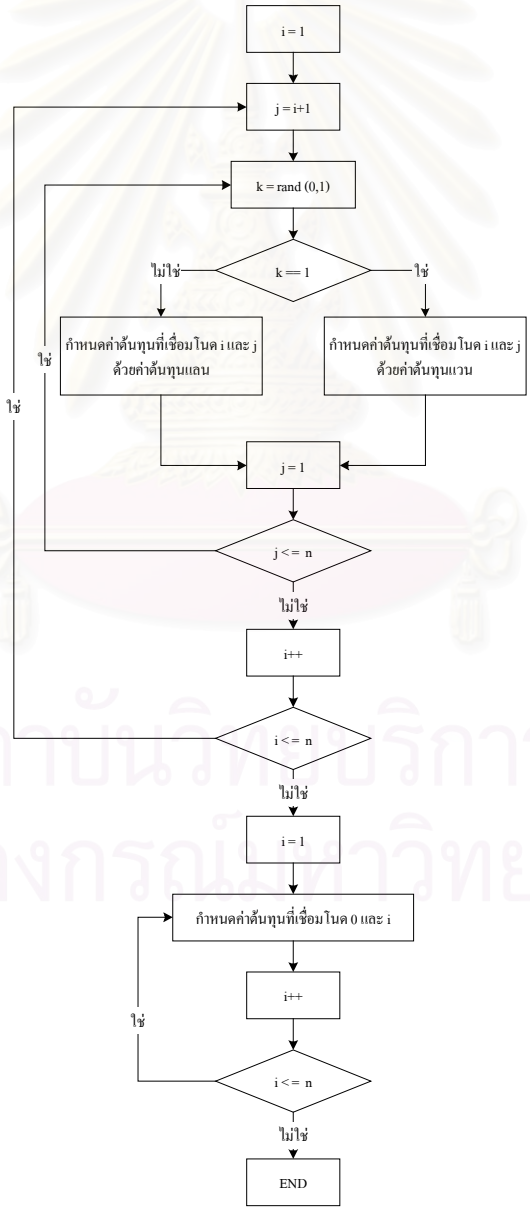
สำหรับค่าต้นทุนของแต่ละลิงก์จะถูกมองเป็นเมตริกซ์ที่มีขนาด (n+1) x (n+1) เมื่อกำหนดให้ n คือจำนวนโนด และกำหนดให้โนดที่ศูนย์เป็นโนดเริ่มต้นในการปล่อยตัวโมบายล์เอเจนต์ดังนั้นค่าต้นทุนที่เชื่อมต่อโนดที่ *i* และ *j* จะมีค่าเท่ากับค่าในเมตริกซ์ในแถวที่ *i* คอลัมน์ที่ *j* (เมื่อกำหนดให้แถวและคอลัมน์เริ่มต้นที่ศูนย์) ดังตัวอย่างรูปที่ 4.1

$$\begin{bmatrix}
0 & 231 & 221 & 226 & 216 & 198 & 248 & 195 & 154 & 158 & 213 \\
231 & 0 & 12 & 37 & 114 & 173 & 14 & 9 & 161 & 11 & 9 \\
221 & 12 & 0 & 83 & 4 & 13 & 195 & 58 & 14 & 10 & 14 \\
226 & 37 & 83 & 0 & 6 & 185 & 160 & 14 & 86 & 82 & 188 \\
216 & 114 & 4 & 6 & 0 & 63 & 8 & 14 & 6 & 47 & 5 \\
198 & 173 & 13 & 185 & 63 & 0 & 51 & 131 & 6 & 7 & 8 \\
248 & 14 & 195 & 160 & 8 & 51 & 0 & 12 & 96 & 14 & 11 \\
195 & 9 & 58 & 14 & 14 & 131 & 12 & 0 & 11 & 14 & 12 \\
154 & 161 & 14 & 86 & 6 & 6 & 96 & 11 & 0 & 9 & 4 \\
158 & 11 & 10 & 82 & 47 & 7 & 14 & 14 & 9 & 0 & 84 \\
213 & 9 & 14 & 188 & 5 & 8 & 11 & 12 & 4 & 84 & 0
\end{bmatrix}$$

รูปที่ 4.1 ค่าเมตริกซ์ต้นทุนกรณี n = 10

ในเนตเวิร์กจำลอง แต่ละโนด (โนดของเซิร์ฟเวอร์) จะถูกเชื่อมโยงด้วยลิงก์ที่มีค่าอยู่ 2 แบบ คือ ลิงก์ที่เป็นต้นทุนแลน (Local Area Network : LAN) และลิงก์ที่เป็นต้นทุนแวน

(Wide Area Netowrk : WAN) ค่าต้นทุนในที่นี้จะใช้เวลาประวิงระหว่างโนดเป็นค่าต้นทุนซึ่งมีค่าเป็นมิลลิวินาที (millisecond) โดยค่าต้นทุนแลนมีค่าอยู่ในช่วง 3-15 มิลลิวินาที ค่าต้นทุนแวนมีค่าอยู่ในช่วง 16-250 มิลลิวินาที และสำหรับค่าต้นทุนระหว่างโนดใด ๆ กับโนดเริ่มต้น (โนดที่ศูนย์) จะมีค่าอยู่ในช่วง 3 – 300 มิลลิวินาที (Jin-Wook et al., 1999) ดังนั้นในการกำหนดค่าให้เมตริกซ์ต้นทุนจะประกอบไปด้วย 2 ขั้นตอนใหญ่ ๆ คือ ขั้นตอนที่ 1 เป็นการกำหนดค่าต้นทุนให้กับลิงก์ที่เชื่อมโยงคู่โนดใด ๆ ที่ไม่ใช่โนดเริ่มต้น ขั้นตอนที่ 2 เป็นการกำหนดค่าต้นทุนให้กับลิงก์ที่เชื่อมโยงคู่โนดใด ๆ กับโนดเริ่มต้น ในขั้นตอนที่ 1 นั้นแต่ละค่าในเมตริกซ์จะผ่านกระบวนการสุ่ม 2 ครั้ง คือ ครั้งแรกเป็นการสุ่มเพื่อกำหนดว่าจะเป็นค่าต้นทุนแลน หรือต้นทุนแวน และครั้งที่ 2 เป็นการกำหนค่าให้กับค่าต้นทุนแลน หรือแวนนั้น ๆ สำหรับขั้นตอนการกำหนดค่าเมตริกซ์ต้นทุนแสดงไว้ดัง รูปที่ 4.2



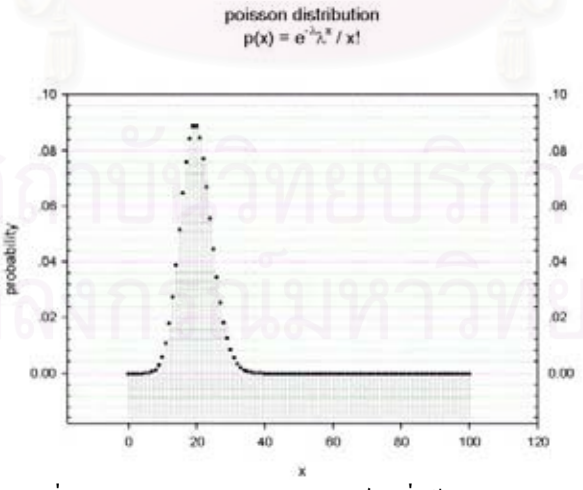
รูปที่ 4.2 ผังงานการกำหนดค่าเมตริกซ์ต้นทุน (แบบอิสระ)

จากค่าเมตริกซ์ต้นทุนในรูปที่ 4.1 นั้นเป็นการกำหนดที่เรียกว่าเป็นการกำหนดค่าต้นทุนให้แต่ละลิงก์อย่างอิสระกล่าวคือ แต่ละลิงก์ที่เชื่อมคู่โนดนั้นสามารถจะเป็นค่าต้นทุนแลนหรือแวนก็ได้ และเป็นแบบสมมาตร (Symmetric) (หมายความว่าต้นทุนลิงก์ที่เชื่อมต่อกับคู่โนดใด ๆ จะมีค่าเท่ากันทั้งขาไปและขากลับ) แต่อย่างไรก็ตามวิธีการกำหนดค่าอย่างอิสระวิธีนี้จะมีปัญหาคือ จะสามารถเกิดกรณี (เส้นทาง) ที่สามารถเชื่อมโยงทุกโนดด้วยค่าต้นทุนแลนได้ ซึ่งไม่เป็นจริงในทางปฏิบัติ ยกตัวอย่างเช่น จากเมตริกซ์ต้นทุนในรูปที่ 4.1 สามารถจัดเส้นทางที่แต่ละโนดสามารถเชื่อมโยงกันโดยลิงก์ต้นทุนแลนเท่านั้นได้ดังรูปที่ 4.3

ตัวอย่างเส้นทางที่ผ่าน 5 โนด	[4 – 6 – 7 – 8 – 10]
ตัวอย่างเส้นทางที่ผ่าน 6 โนด	[2 – 4 – 5 – 8 – 9 – 10]
ตัวอย่างเส้นทางที่ผ่าน 10 โนด	[2 – 1 – 10 – 4 – 3 – 4 – 6 – 9 – 8 – 5]

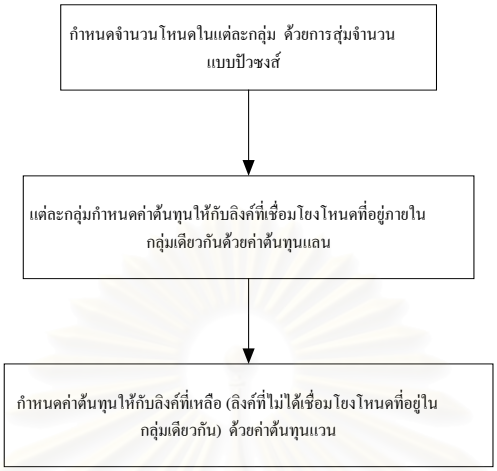
รูปที่ 4.3 ตัวอย่างเส้นทางที่แต่ละโนดเชื่อมโยงกันด้วยลิงก์ที่เป็นค่าต้นทุนแลน

จากรูปที่ 4.3 จะเห็นได้ว่าค่าเมตริกซ์ต้นทุนที่กำหนดแบบอิสระสามารถก่อให้เกิดเส้นทางที่เชื่อมต่อทุกโนดในเนตเวิร์กด้วยค่าต้นทุนแลนได้ ซึ่งไม่เป็นจริงในทางปฏิบัติ ดังนั้นเพื่อแก้ไขปัญหาที่เกิดขึ้น การกำหนดค่าต้นทุนให้กับลิงก์ต่าง ๆ นั้นจะมีลักษณะเป็นลำดับชั้น (Hierarchy) ซึ่งจะมองเนตเวิร์กประกอบไปด้วยกลุ่มของโนด ดังนั้นในการกำหนดค่าต้นทุนให้กับลิงก์จะต้องมีการกำหนดค่าเริ่มต้นของเนตเวิร์กเสียก่อนว่า แต่ละกลุ่มของโนดควรจะประกอบไปด้วยจำนวนโนดเท่าใด สำหรับในการกำหนดจำนวนโนดที่ใช้ในเนตเวิร์กจำลองนี้ จะใช้การสุ่มแบบปัวส์ซงที่มีค่าเฉลี่ยอยู่ที่ 10 % ของจำนวนโนดดังรูปที่ 4.4



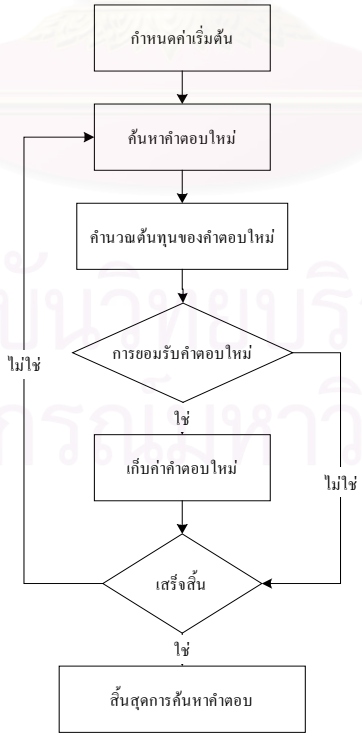
รูปที่ 4.4 กราฟการกระจายแบบปัวส์ซงที่ใช้ในการกำหนด
จำนวนโนดของแต่ละกลุ่มโนดในเนตเวิร์กจำลอง

เมื่อกำหนดค่าเริ่มต้นจำนวนโนดในแต่ละกลุ่มแล้ว จากนั้นจึงกำหนดค่าต้นทุนให้แก่ลิงก์ที่เชื่อมแต่ละคู่โนดในแต่ละกลุ่มให้เป็นค่าต้นทุนแลน แล้วกำหนดค่าลิงก์ที่เหลือด้วยค่าต้นทุนแวน ดังรูปที่ 4.5



รูปที่ 4.5 ผังงานการกำหนดค่าเมตริกซ์ต้นทุนแบบลำดับชั้น (Hierarchy)

สำหรับอัลกอริทึมที่ใช้ในการกำหนดเส้นทางให้กับโมบายล์เอเจนต์ ในหัวข้อวิทยานิพนธ์นี้จะเสนออัลกอริทึมที่สำคัญ 6 อัลกอริทึมคือ Brute Force Search, Simulated Annealing, Tabu Search, Modified Compact Genetic, Random Uniform และ Heuristic ซึ่งอัลกอริทึมทั้ง 6 แบบนี้มีหลักการที่คล้ายคลึงกัน แต่จะแตกต่างกันที่วิธีการทำงาน ซึ่งสามารถสรุปหลักการทำงานของอัลกอริทึมทั้ง 6 กรณีได้ดังรูป (สำหรับการทำงานของฮิวริสติกจะอธิบายไว้ในบทที่ 6)



รูปที่ 4.6 ผังงานการกำหนดเส้นทางให้กับโมบายล์เอเจนต์

ก่อนที่จะกล่าวถึงอัลกอริทึมต่าง ๆ จำเป็นจะต้องเข้าใจลักษณะของเส้นทาง และการคำนวณค่าต้นทุนเสียก่อน รวมไปถึงความหมายของแต่ละขั้นตอนในผังงาน ดังนี้

การกำหนดค่าเริ่มต้น : เป็นขั้นตอนการกำหนดค่าเริ่มต้นให้กับค่าพารามิเตอร์ต่าง ๆ รวมไปถึงการกำหนดค่าคำตอบเริ่มต้นที่ใช้ในอัลกอริทึม ซึ่งคำตอบเริ่มต้นคือแบบของเส้นทางที่เป็นไปได้ที่จะกำหนดให้กับโมบายล์เอเจนต์ ตัวอย่างของเส้นทางของโมบายล์เอเจนต์ที่มีจำนวนโนดเท่ากับ 10 จำนวนโมบายล์เอเจนต์เท่ากับ 3 เช่น

เส้นทางย่อยสำหรับโมบายล์เอเจนต์ที่ 1	[1 – 3 – 5 – 4]
เส้นทางย่อยสำหรับโมบายล์เอเจนต์ที่ 2	[2 – 10]
เส้นทางย่อยสำหรับโมบายล์เอเจนต์ที่ 3	[9 – 7 – 8 – 6]
รูปแบบของเส้นทาง [1 – 3 – 5 – 4] [2 – 10] [9 – 7 – 8 – 6]	

ตัวอย่างเส้นทางที่กำหนดให้โมบายล์เอเจนต์ เมื่อ n = 10 จำนวนโมบายล์เอเจนต์ = 3

ซึ่งหมายความว่า การกำหนดให้โมบายล์เอเจนต์ตัวที่หนึ่งเริ่มต้นจากโนดเริ่มต้น (โนดศูนย์) แล้วไปยังโนดที่ 1, 3, 5 และ 4 ตามลำดับ จากนั้นกลับมายังที่โนดศูนย์ ซึ่งเป็นโนดเริ่มต้น และเช่นเดียวกันกับโมบายล์เอเจนต์ตัวที่ 2 และ 3 เริ่มต้นจากโนดที่ 0 แล้วไปยังโนดที่ 2, 10 และ 9, 7, 8, 6 แล้วกลับมายังโนดที่ 0 ตามลำดับ

การค้นหาคำตอบใหม่ : การค้นหาคำตอบใหม่ก็คือการค้นหารูปแบบของเส้นทางที่เป็นไปได้ที่ไม่ซ้ำกับคำตอบเดิม (รูปแบบของเส้นทาง) ที่ผ่านมา ซึ่งวิธีการกำหนดขั้นตอนใหม่จะแตกต่างกันไปตามแต่ละอัลกอริทึม

การคำนวณค่าต้นทุนของคำตอบใหม่ : สำหรับการคำนวณต้นทุนของคำตอบ คือการกำหนดเวลาหน่วงที่เกิดขึ้นอันเนื่องมาจากคำตอบ (รูปแบบของเส้นทาง) นั้น ๆ ดังนั้นในกรณีของรูปแบบเส้นทางที่ใช้โมบายล์เอเจนต์ตัวเดียว ค่าต้นทุนของคำตอบคือ ผลรวมของเวลาหน่วงระหว่างลิงก์ต่าง ๆ ที่อยู่บนเส้นทางนั้น สำหรับกรณีการกำหนดรูปแบบเส้นทางที่ใช้โมบายล์เอเจนต์มากกว่าหนึ่งตัว ลักษณะของรูปแบบเส้นทางจะประกอบไปด้วยเส้นทางย่อย ๆ ที่กำหนดให้โมบายล์เอเจนต์แต่ละตัว ดังเช่นตัวอย่างในหัวข้อย่อย *การกำหนดค่าเริ่มต้น* ดังนั้นเมื่อยึดตามหลักการใช้งานจริง เวลาหน่วงของรูปแบบเส้นทางที่กำหนดให้กับโมบายล์เอเจนต์มากกว่าหนึ่งตัวจะมีค่าเท่ากับเวลาหน่วงของเส้นทางย่อยที่มีค่ามากที่สุดนั่นเอง เพราะในการใช้งานจริงจะมีการปล่อยโมบายล์เอเจนต์เข้าไปในเนตเวิร์กพร้อม ๆ กัน ดังนั้นการคำนวณค่าต้นทุนของคำตอบใหม่คือการคำนวณค่าเวลาหน่วงของรูปแบบเส้นทางที่เกิดขึ้นใหม่ เพื่อความเข้าใจจะยกตัวอย่างการคำนวณค่าต้นทุนของ

รูปแบบเส้นทางทั้งกรณีที่ใช้โมบายล์เอเจนต์ตัวเดียว และกรณีที่ใช้โมบายล์เอเจนต์มากกว่าหนึ่งตัวดังนี้

ตัวอย่างที่ 4.1 *การคำนวณต้นทุนของรูปแบบเส้นทางที่ใช้โมบายล์เอเจนต์หนึ่งตัว*

กำหนดให้จำนวนโนดเท่ากับ 5

กำหนดให้รูปแบบของเส้นทางคือ [1 – 3 – 4 – 2 – 5]

และกำหนดให้ค่าเวลาหน่วงที่เชื่อมโยงโนด i และ j ใด ๆ เป็น L(i,j)

ดังนั้นค่าต้นทุนของคำตอบเท่ากับ

$$L(0,1) + L(1,3) + L(3,4) + L(4,2) + L(2,5) + L(5,0)$$

ตัวอย่างที่ 4.2 *การคำนวณต้นทุนรูปแบบของเส้นทางที่ใช้โมบายล์เอเจนต์มากกว่าหนึ่งตัว*

กำหนดให้จำนวนโนดเท่ากับ 10

กำหนดให้จำนวนโมบายล์เอเจนต์เท่ากับ 3

กำหนดให้รูปแบบของเส้นทางย่อยที่กำหนดให้กับโมบายล์เอเจนต์ตัวที่ 1 คือ

[1 – 3 – 5 – 4]

กำหนดให้รูปแบบของเส้นทางย่อยที่กำหนดให้กับโมบายล์เอเจนต์ตัวที่ 2 คือ

[2 – 10]

กำหนดให้รูปแบบของเส้นทางย่อยที่กำหนดให้กับโมบายล์เอเจนต์ตัวที่ 3 คือ

[9 – 7 – 8 – 6]

ดังนั้นค่าต้นทุนของเส้นทางย่อยที่ 1 เท่ากับ

$$L_1 = L(0,1) + L(1,3) + L(3,5) + L(5,4) + L(4,0)$$

ดังนั้นค่าต้นทุนของเส้นทางย่อยที่ 2 เท่ากับ

$$L_2 = L(0,2) + L(2,10) + L(10,0)$$

ดังนั้นค่าต้นทุนของเส้นทางย่อยที่ 3 เท่ากับ

$$L_3 = L(0,9) + L(9,7) + L(7,8) + L(8,6) + L(6,0)$$

จะได้ค่าต้นทุนของคำตอบเท่ากับ $L = \max \{L_1, L_2, L_3\}$

ดังนั้นในการหาคำตอบที่ดีที่สุดคือการหารูปแบบเส้นทางที่ให้ *ค่าต้นทุนที่ต่ำที่สุดของค่าต้นทุนของเส้นทางย่อยที่มีค่าสูงที่สุด (Minimum of Maximum Sub-Route Cost : MMSRC)* ซึ่งต่อไปจะเรียกว่า MMSRC

การยอมรับคำตอบใหม่ : การยอมรับคำตอบใหม่คือกรณีที่ค่าคำตอบใหม่ (เส้นทาง) มีค่าต้นทุนที่คำนวณได้ มีค่าดีกว่า (น้อยกว่า) ค่าต้นทุนของคำตอบเดิม แต่อย่างไรก็ตามบางอั

ลกอริทึมสามารถยอมรับคำตอบใหม่ที่มีค่าต้นทุนด้อยกว่า (มากกว่า) เดิมได้ (ดังเช่นอัลกอริทึม Simulated Annealing) แต่จะยอมรับด้วยค่าความน่าจะเป็นค่าหนึ่งซึ่งจะกล่าวรายละเอียดในแต่ละอัลกอริทึม

การเก็บค่าคำตอบใหม่ : การเก็บค่าคำตอบใหม่คือการเก็บค่าคำตอบที่เป็นคำตอบที่ดีที่สุด ณ ขณะนั้น ซึ่งส่วนใหญ่จะเก็บค่าคำตอบเมื่อค่าต้นทุนของคำตอบนั้นมีค่าต่ำที่สุด ณ ขณะนั้น

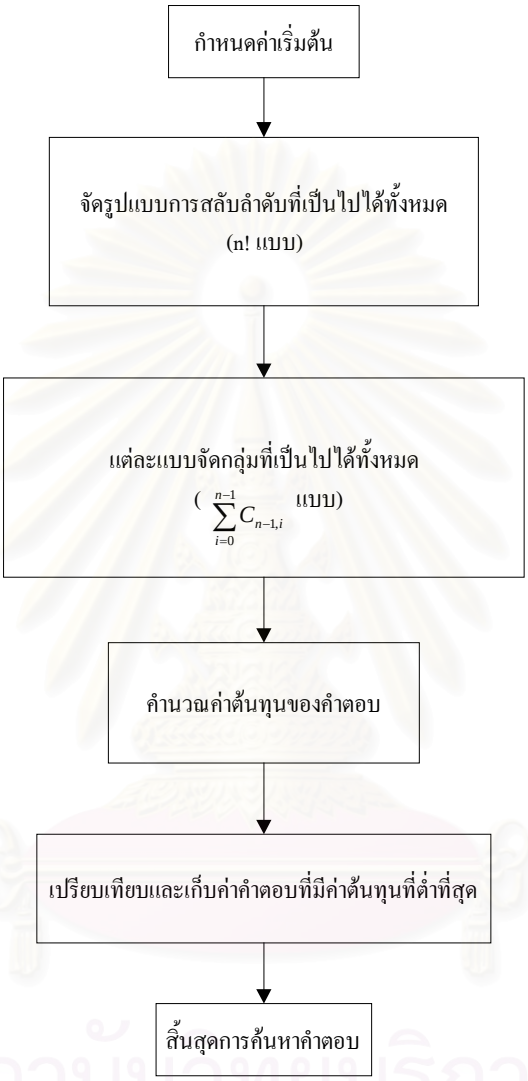
จากขั้นตอนที่กล่าวมาแล้วนั้น บางขั้นตอนจะมีความแตกต่างกันในแต่ละอัลกอริทึม บางขั้นตอนก็คล้ายกัน ซึ่งจะอธิบายรายละเอียดของแต่ละอัลกอริทึมดังหัวข้อต่อไป

4.3 การค้นหาคำตอบโดยใช้อัลกอริทึม Brute Force Search

การค้นหาคำตอบโดยใช้อัลกอริทึม Brute Force Search เป็นการค้นหาแบบทุกกรณีโดยจะทำการค้นหาคำตอบที่ดีที่สุดจากกรณีที่เป็นไปได้ทั้งหมด ดังนั้นหลักการคือ การค้นหารูปแบบของคำตอบที่เกิดขึ้นได้ทุกกรณี และคำนวณค่าต้นทุนของคำตอบทุกกรณีมาเปรียบเทียบเพื่อหาค่าที่ต่ำที่สุด แต่เนื่องจากลักษณะของคำตอบที่เกิดขึ้นได้มีความหลากหลาย ดังนั้นขั้นตอนการค้นหาคำตอบที่ดีที่สุดจะต้องมีการแบ่งลักษณะของคำตอบเป็นกลุ่มย่อย ๆ แล้วจึงนำผลลัพธ์แต่ละกลุ่มย่อยมาเปรียบเทียบกันอีกครั้งหนึ่ง โดยลักษณะของกลุ่มย่อยจะเป็นกลุ่มของเส้นทางที่ใช้จำนวนโมบายล์เอเจนต์เท่ากัน ตัวอย่างเช่น กรณีจำนวนโนดเท่ากับ 5 ลักษณะของคำตอบจะเป็นเป็นได้ 5 กลุ่มคือ กลุ่มของคำตอบที่ใช้โมบายล์เอเจนต์ตัวเดียว กลุ่มของคำตอบที่ใช้โมบายล์เอเจนต์ 2 ตัว จนถึงกลุ่มที่ใช้โมบายล์เอเจนต์ 5 ตัว ซึ่งจะเห็นได้ว่า ถ้ามี n โนด จะแบ่งกลุ่มของคำตอบได้ n กลุ่ม จากนั้นจะคำนวณค่าต้นทุนของคำตอบที่เกิดขึ้นได้ทั้งหมดในแต่ละกลุ่ม แล้วนำค่าที่ดีที่สุดของแต่ละกลุ่มมาเปรียบเทียบกันอีกครั้งหนึ่งเพื่อให้ได้คำตอบที่ดีที่สุด สำหรับค่าคำตอบที่ดีที่สุดของแต่ละกลุ่ม ก็คือค่า MMSRC ในแต่ละกลุ่มนั่นเอง

สำหรับขั้นตอนที่สำคัญของการค้นหารูปแบบของคำตอบที่เกิดขึ้นได้ทุกกรณีมีอยู่ด้วยกัน 2 ส่วนคือ การสลับลำดับ (Permutation) และ การจัดกลุ่ม (Grouping) การสลับลำดับคือการจัดเรียงแบบต่าง ๆ ที่เป็นไปได้ ตัวอย่างเช่น การจัดเรียงเลข “1”, “2”, “3” จะสามารถสลับลำดับให้แตกต่างกันได้ 6 แบบคือ “123”, “132”, “213”, “231”, “312” และ “321” นั่นคือจำนวนวิธีที่เป็นไปได้ทั้งหมดจะเท่ากับ 3! (3 factorial) สำหรับการจัดกลุ่มคือการแบ่งสิ่งของที่มีออกเป็นกลุ่มอย่างไรก็ได้ ตัวอย่างเช่น “123” จะถูกจัดกลุ่มออกมาได้ทั้งหมด 4 แบบ ($C_{2,0}+C_{2,1}+C_{2,2}$) คือ [123], [1][23], [12][3] และ [1][2][3] โดยจะเห็นได้ว่า [123] ก็คือเส้นทางที่ใช้โมบายล์เอเจนต์เพียงตัวเดียวผ่านโนดที่ต้องการทั้งหมดซึ่งก็คือ 1, 2, 3 และ [1][23] ก็คือเส้นทางที่ใช้โมบายล์เอ

เจนต์ 2 ตัวในการผ่านโนดที่ต้องการทั้งหมดโดยตัวแรกผ่านโนด 1 ตัวที่สองผ่านโนด 2 และ 3 จากขั้นตอนทั้ง 2 จะทำให้เกิดกรณีเส้นทางที่เป็นไปได้ทั้งหมด จากนั้นจึงนำเส้นทางแต่ละแบบมาคำนวณค่าต้นทุน แล้วเปรียบเทียบและเก็บค่าที่ดีที่สุด สำหรับขั้นตอนการทำงานของกระบวนการค้นหาแบบทุกกรณีอธิบายได้ดังรูปที่ 4.7



รูปที่ 4.7 ขั้นตอนการทำงานของกระบวนการค้นหาแบบทุกกรณี (Brute Force Search)

สำหรับผลลัพธ์และการวิเคราะห์จะกล่าวโดยละเอียดในเนื้อหาในบทที่ 6 การวิเคราะห์อัลกอริทึมฮิวริสติก

4.4 การค้นหาคำตอบโดยใช้อัลกอริทึม Simulated Annealing

จากหลักการและขั้นตอนการค้นหาคำตอบโดยใช้อัลกอริทึม Brute Force Search จะเห็นได้ว่า แนวคิดการทำงานของอัลกอริทึมนั้นจะตรงไปตรงมา คือการค้นหาคำตอบที่ดีที่สุดจากคำตอบที่มีทั้งหมด แต่ในการทำงานจริงนั้นเวลาที่ใช้ในการค้นหาคำตอบที่เป็นไปได้ทั้งหมดแทบจะเป็นไปไม่ได้ในกรณีที่จำนวนโนดมีค่ามาก ๆ (การวิเคราะห์จะอธิบายในบทที่ 6) ดังนั้นจึงเกิดวิธีการต่าง ๆ ที่จะสุ่มเลือกเฉพาะคำตอบที่มีแนวโน้มจะมีค่าที่ดีที่สุดในกลุ่มจำนวนหนึ่งออกมาเพื่อเป็นตัวแทนของคำตอบที่เป็นไปได้ทั้งหมด มาคำนวณหาค่าต้นทุนของคำตอบที่ดีที่สุด โดยวิทยานิพนธ์ฉบับนี้จะศึกษาอัลกอริทึม Simulated Annealing, Tabu Search และ Modified Compact Genetic พร้อมทั้งคิดวิธีนำอัลกอริทึมไปประยุกต์ใช้กับอัลกอริทึมฮิวริสติก

หลักการทำงานของการค้นหาคำตอบโดยใช้อัลกอริทึม Simulated Annealing จำลองมาจากกระบวนการให้ความร้อนแก่โลหะเพื่อให้โลหะอ่อนตัว เป็นวิธีการค้นหาคำตอบได้ดีในปัญหาที่ซับซ้อนและแก้ปัญหาที่การค้นหามักติดอยู่ใน Local Optimum คือ การที่คำตอบที่ค้นพบถูกจำกัดอยู่ในวงแคบ ซึ่งเกิดจากการที่กระบวนการค้นหาคำตอบ ยอมรับเฉพาะคำตอบที่ดีขึ้นเท่านั้น ทำให้โอกาสที่จะค้นหาคำตอบแบบอื่น ๆ น้อยลง คำตอบสุดท้ายที่ได้จึงเป็นคำตอบที่ต่ำที่สุดเฉพาะในกลุ่มของคำตอบเพียงกลุ่มเดียวเท่านั้น

เนื่องจากอัลกอริทึม Simulated Annealing มีการยอมรับคำตอบที่มีค่าต้นทุนที่ด้อยกว่าเดิมได้ ดังนั้นลักษณะคำตอบที่ใช้ในอัลกอริทึม Simulated Annealing จะแบ่งได้ 2 แบบคือ คำตอบที่ใช้ในการค้นหาคำตอบใหม่ (จะเรียกว่า *คำตอบปัจจุบัน*) กับคำตอบที่มีค่าต้นทุนที่ดีที่สุด ณ ขณะนั้น (จะเรียกว่า *คำตอบที่ดีที่สุด*) โดยคำตอบปัจจุบันคือคำตอบใหม่ที่มีค่าต้นทุนที่ดีขึ้น หรือคำตอบใหม่ที่มีค่าต้นทุนที่ด้อยลงแต่ได้รับการยอมรับด้วยค่าความน่าจะเป็น p ซึ่งคำตอบปัจจุบันจะถูกใช้เป็นแนวทางในการหาคำตอบใหม่ สำหรับคำตอบที่ดีที่สุดคือคำตอบใหม่ที่มีค่าต้นทุนต่ำที่สุดที่เคยเกิดขึ้น

ในอัลกอริทึม Simulated Annealing การแก้ปัญหา Local Optimum จะแก้ปัญหาโดยการยอมรับคำตอบที่ยืดหยุ่นกว่า คือจะไม่ยอมรับแต่คำตอบที่ดีขึ้นเท่านั้น แต่จะมีการยอมรับคำตอบที่ด้อยลงด้วยค่าความน่าจะเป็นค่าหนึ่ง (p) ทำให้โอกาสที่จะค้นหาคำตอบที่ดีซึ่งอาจจะอยู่ในบริเวณอื่นเป็นไปได้มากยิ่งขึ้น ค่าความน่าจะเป็นที่จะยอมรับคำตอบที่ด้อยลงดังกล่าวมีค่าไม่เท่ากันในแต่ละรอบของการค้นหาคำตอบ ในรอบแรก ๆ จะมีค่าสูง (ใกล้เคียง 1) เมื่อกระบวนการค้นหาคำตอบดำเนินต่อไประยะหนึ่งจึงค่อย ๆ ลดลงด้วยค่า factor ค่าหนึ่งจนกระทั่งในที่สุดมีค่าเกือบเป็นศูนย์ ดังสมการที่ 4.1

$$p = \exp\left(-\frac{|Cost_{i+1} - Cost_i|}{t_i}\right) \quad (4.1)$$

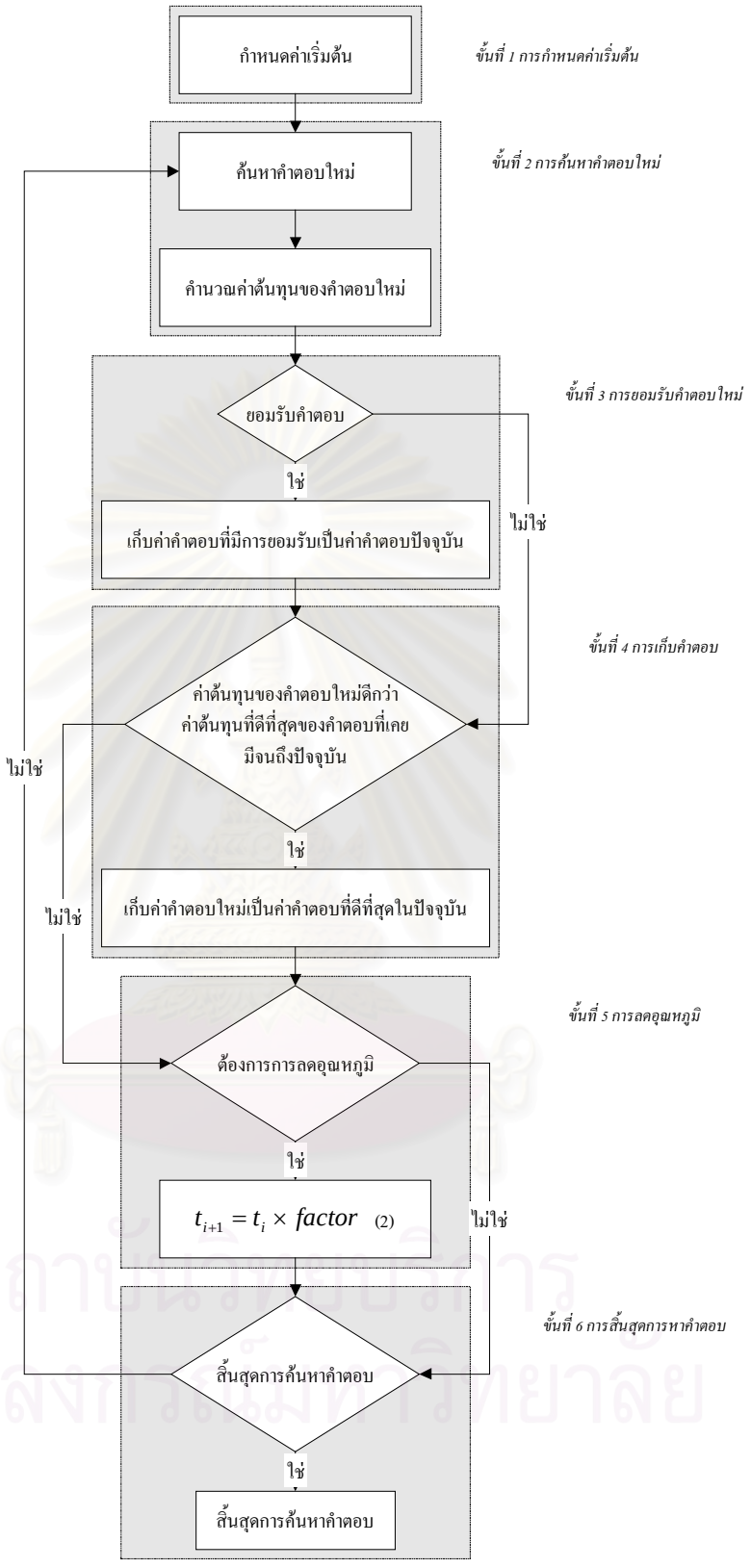
เมื่อกำหนดให้ค่า $Cost_{i+1}$ คือค่าต้นทุนของรอบ (iteration) ถัดไป $Cost_i$ คือค่าต้นทุนของรอบปัจจุบัน สำหรับค่า t_i คือค่าอุณหภูมิ ณ รอบปัจจุบัน จากสมการที่ 4.1 จะเห็นได้ว่าค่า p จะถูกกำหนดโดยค่าความแตกต่างต้นทุน $|Cost_{i+1}-Cost_i|$ และค่าอุณหภูมิควบคุม (Temperature Control) t_i ซึ่งมีการกำหนดค่าดังสมการที่ 4.2

$$t_{i+1} = factor\ X\ t_i \quad (4.2)$$

จากสมการที่ 4.2 ค่า *factor* โดยปกติจะมีค่าอยู่ในช่วง 0 ถึง 1 และค่า t_i จะถูกคูณด้วยค่า factor ทุก ๆ ช่วงของรอบการค้นหาจำนวนหนึ่ง (ในที่นี้จะเรียกว่า iteration reduction หรือค่า *r*) ดังนั้นจะเห็นได้ว่าถ้ามีการกำหนดค่าเริ่มต้น t_0 ให้มีค่าสูงมาก ๆ จะมีผลทำให้ค่า p ในสมการที่ 4.1 มีค่าใกล้เคียง 1 และเช่นเดียวกันเมื่อจำนวนรอบของกระบวนการการค้นหาคำตอบผ่านไปมากเท่าไร จะมีผลทำให้ค่า t_i มีค่าลดลงมากจนใกล้เคียงศูนย์เท่านั้น มีผลทำให้ค่า p ในสมการที่ 4.1 มีค่าใกล้เคียงศูนย์ ซึ่งหมายความว่าจะไม่มีการยอมรับคำตอบที่ด้อยลงเมื่อจำนวนรอบของกระบวนการค้นหาคำตอบมีค่าสูง ๆ นั่นเอง

จากที่กล่าวมาจะเห็นได้ว่าค่าพารามิเตอร์ที่มีผลต่อการค้นหาคำตอบโดยใช้อัลกอริทึม Simulated Annealing จะมีอยู่ด้วยกัน 3 ตัวคือ ค่าอุณหภูมิเริ่มต้น (t_0) ค่าการทอนอุณหภูมิ (*factor*) และค่าจำนวนรอบที่ทำให้เกิดการลดทอนอุณหภูมิ (*r*) ซึ่งค่าพารามิเตอร์เหล่านี้จะมีค่าแตกต่างกันไปขึ้นกับลักษณะการใช้งาน ดังนั้นการค้นหาคำตอบโดยใช้อัลกอริทึม Simulated Annealing จะต้องมีกระบวนการค้นหาค่าพารามิเตอร์ที่เหมาะสมเสียก่อน (สำหรับวิธีหาพารามิเตอร์และผลลัพธ์ที่ได้จะกล่าวโดยละเอียดในเนื้อหาบทที่ 6)

ดังนั้นจากแนวคิดการค้นหาคำตอบโดยใช้อัลกอริทึม Simulated Annealing ข้างต้น ขั้นตอนการทำงานจะแตกต่างไปจากผังงานการค้นหาคำตอบที่ได้อธิบายไว้ในรูปที่ 4.6 ไปบ้าง โดยจะอธิบายได้ดังรูปที่ 4.8



รูปที่ 4.8 ผังงานขั้นตอนการทำงานของการค้นหาคำตอบโดย
ใช้อัลกอริทึม Simulated Annealing

จากผังงานรูปที่ 4.8 สามารถอธิบายขั้นตอนการทำงานได้ดังนี้

ขั้นตอนที่ 1: การกำหนดค่าเริ่มต้น เป็นการเลือกคำตอบเริ่มต้นและคำนวณค่าต้นทุนของคำตอบเริ่มต้นนั้น ค่าต้นทุนที่ได้จะกำหนดให้เป็นค่าต้นทุนที่ยอมรับเริ่มต้นและเป็นค่าต้นทุนที่ดีที่สุดเริ่มต้น และกำหนดค่าพารามิเตอร์ต่าง ๆ ที่จะต้องใช้ในอัลกอริทึม

ขั้นตอนที่ 2: การค้นหาคำตอบใหม่ เป็นการหาคำตอบใหม่โดยการเลือกเส้นทางใหม่ขึ้นมาที่ไม่ซ้ำกับเส้นทางที่ผ่านมาแล้ว

ขั้นตอนที่ 3: การยอมรับคำตอบใหม่ หากมีการพัฒนาคำตอบ (ค่าต้นทุน-MMSRC ของคำตอบใหม่มีค่าน้อยกว่าของเดิม) ให้ยอมรับเส้นทางใหม่นี้เป็นคำตอบปัจจุบัน แต่ถ้าไม่มีการพัฒนาคำตอบ จะยอมรับคำตอบใหม่ให้เป็นคำตอบปัจจุบันด้วยค่าความน่าจะเป็น p จากสมการที่ (4.1) หากมีการยอมรับคำตอบที่เกิดขึ้นครบ r ครั้ง ให้ทำการลดอุณหภูมิควบคุมตามสมการที่ (4.2)

ขั้นตอนที่ 4: การเก็บคำตอบ จากคำตอบที่มีการยอมรับ หากคำตอบที่ได้มีค่าต้นทุนที่ดีกว่าค่าต้นทุนของคำตอบที่ดีที่สุด ให้เก็บค่าคำตอบนี้ไว้ และเก็บค่าคำตอบเป็นค่าคำตอบที่ดีที่สุดแทน

ขั้นตอนที่ 5: การลดอุณหภูมิ เป็นการตรวจสอบว่ารอบที่มีการค้นหาคำตอบใหม่ครบช่วงของการลดค่าพารามิเตอร์อุณหภูมิหรือไม่ โดยจะทำการลดค่าอุณหภูมิควบคุมตามสมการที่ (4.2) ทุก ๆ ช่วงรอบค่า r

ขั้นตอนที่ 6: การสิ้นสุดการหาคำตอบ หลังจากทำขั้นที่ 4 แล้วให้ทำวนซ้ำตั้งแต่ขั้นที่ 2 จนกว่าจะไม่มีคำตอบที่พัฒนาขึ้นอีก หรือครบจำนวนรอบที่ตั้งไว้ หรือไม่มีคำตอบใหม่ให้เลือกอีก

เพื่อเป็นการทำความเข้าใจถึงกระบวนการค้นหาคำตอบโดยใช้อัลกอริทึม Simulated Annealing จะยกตัวอย่างการทำงานดังต่อไปนี้

ตัวอย่างที่ 4.3 *การทำงานของอัลกอริทึม Simulated Annealing*

กำหนดให้จำนวนโนดเท่ากับ 10
กำหนดให้จำนวนโมบายล์เอเจนต์เท่ากับ 3
กำหนดให้คำตอบเริ่มต้นคือ [1 – 2 – 3] [4 – 5 – 6] [7 – 8 – 9 – 10]
และกำหนดให้จำนวนรอบที่ใช้เท่ากับ 100

ตารางที่ 4.1 ตัวอย่างการทำงานการค้นหาคำตอบโดยใช้อัลกอริทึม Simulated Annealing

รอบที่	คำตอบในรอบปัจจุบัน	ค่าต้นทุนของคำตอบ	การยอมรับ
0	[1 – 2 – 3] [4 – 5 – 6] [7 – 8 – 9 – 10]	460	
1	[1 – 2 – 3 – **4**] [5 – 6] [7 – 8 – 9 – 10]	435	ยอมรับ
2	[1 – 2 – 3 – 4] [5 – 6] [7 – **9** – **8** – 10]	420	ยอมรับ

3	[1 – 2 – 3 – 4] [5 – **7** – 6] [9 – 8 – 10]	433	ไม่ยอมรับ
4	[1 – 2 – 3 – 4] [5 – 6 – **10**] [7 – **9** – **8**]	402	ยอมรับ
5	[1 – 2 – 4] [5 – 6 – **10**] [**3** – 7 – **9** – **8**]	398	ยอมรับ
6	[1 – **6** – 4] [5 – **2** – **10**] [3 – 7 – **9** – **8**]	402	ยอมรับ
.	.	.	.
.	.	.	.
.	.	.	.
98	[8 – 4 - 3] [2 – 5] [7 – **1** – 10 – 9 – 6]	235	ยอมรับ
99	[8 – **6** - 3] [2 – 5] [7 – 1 – 10 – 9 – **4**]	238	ไม่ยอมรับ
100	[8 – 4 - 3] [2 – 5] [7 – **6** – 10 – 9 – **1**]	234	ยอมรับ

จากตัวอย่างการทำงานจะเห็นได้ว่าในรอบที่ 3 นั้นค่าคำตอบที่ได้มีค่าต้นทุนเท่ากับ 433 ซึ่งแย่กว่าเดิม และไม่ได้การยอมรับจากขั้นตอนการยอมรับคำตอบดังนั้นค่าคำตอบปัจจุบันจะเป็นคำตอบที่ได้จากรอบที่ 2 จากนั้นค่าคำตอบจากรอบที่ 4 จะได้จากการนำค่าคำตอบในรอบที่ 2 มาใช้ จะไม่นำค่าคำตอบของรอบที่ 3 มาใช้ เช่นเดียวกัน ในรอบที่ 6 นั้นค่าต้นทุนของคำตอบได้เท่ากับ 402 ซึ่งแย่กว่าค่าต้นทุนของคำตอบในรอบที่ 5 แต่เนื่องจากผ่านการยอมรับในขั้นตอนการยอมรับคำตอบ ดังนั้นคำตอบที่ได้จากรอบที่ 6 จะเป็นคำตอบปัจจุบัน จากนั้นการหาคำตอบในรอบถัดไป (รอบที่ 7) จะนำคำตอบในรอบที่ 6 ไปใช้ และจะทำเช่นนี้จนกว่าจะได้จำนวนรอบครบตามที่กำหนดไว้ ซึ่งในที่นี้กำหนดไว้ที่ 100 รอบ

4.5 การค้นหาคำตอบโดยใช้อัลกอริทึม Tabu Search

สำหรับการค้นหาคำตอบโดยใช้อัลกอริทึม Tabu Search จะมีหลักการที่สำคัญอยู่ที่การเลือกหาคำตอบใหม่ จากชื่อของอัลกอริทึม คำว่า “tabu” จะแปลว่า “forbidden” หรือ “*ห้าม*” โดยหลักการของการเลือกคำตอบใหม่จะมีลักษณะการห้ามหรือคงที่ของตำแหน่งเดิมบางตำแหน่งในคำตอบปัจจุบัน โดยจะมีลิสต์ของตำแหน่งที่ถูกห้ามไว้ซึ่งเรียกว่า ลิสต์ทาบู (Tabu List) โดยตำแหน่งที่ห้ามจะถูกคงที่ไว้เป็นช่วงระยะหนึ่งของการค้นหาคำตอบจากนั้นจึงจะสามารถถูกเปลี่ยนตำแหน่งได้ โดยหลักการทำงานของอัลกอริทึม Tabu Search สามารถอธิบายเป็นขั้นตอนได้ดังนี้

ขั้นตอนที่ 1: การกำหนดค่าเริ่มต้น เป็นการเลือกคำตอบเริ่มต้นแล้วหาค่าต้นทุนของคำตอบเริ่มต้นนั้น ค่าคำตอบที่ได้จะกำหนดให้เป็นค่าคำตอบปัจจุบันเริ่มต้นและเป็นค่าคำตอบที่ดีที่สุดเริ่มต้น และกำหนดค่าพารามิเตอร์ต่าง ๆ ที่จะต้องใช้ในอัลกอริทึม

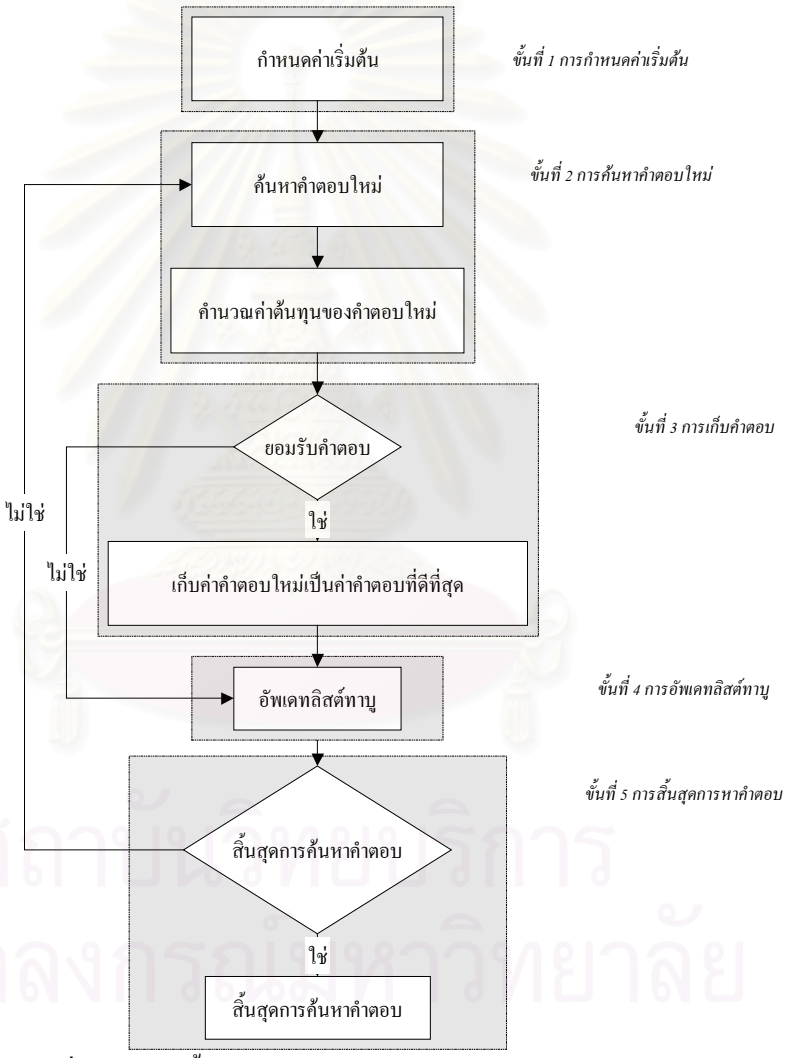
ขั้นตอนที่ 2: การค้นหาคำตอบใหม่ เป็นการหาคำตอบใหม่โดยการเลือกเส้นทางใหม่ขึ้นมาที่ไม่ซ้ำกับเส้นทางที่ผ่านมาแล้ว โดยใช้หลักการของอัลกอริทึมTabu Search

ขั้นตอนที่ 3: การเก็บคำตอบ เมื่อค่าต้นทุนของคำตอบใหม่ที่ได้มีค่าดีกว่าคำตอบที่ดีที่สุด จะทำการเก็บค่าคำตอบนี้ไว้ และถือว่าเป็นคำตอบที่ดีที่สุด

ขั้นตอนที่ 4: การอัพเดทลิสต์ทาบู เมื่อมีการค้นหาคำตอบใหม่จะต้องมีการอัพเดทลิสต์ทาบูเพื่อใช้ในขั้นตอนของการค้นหาคำตอบใหม่

ขั้นตอนที่ 5: การสิ้นสุดการหาคำตอบ เมื่อผ่านขั้นที่ 4 แล้วจะวนซ้ำทำตั้งแต่ขั้นที่ 2 ใหม่จนกว่าจะครบรอบที่ตั้งไว้ หรือไม่มีคำตอบใหม่เหลือให้เลือกอีก

จากขั้นตอนที่กล่าวมาทั้งหมดสามารถแสดงด้วยผังงานดังรูปที่ 4.9



รูปที่ 4.9 ผังงานขั้นตอนการทำงานของการค้นหาคำตอบ
โดยใช้อัลกอริทึม Tabu Search

สังเกตว่า อัลกอริทึม Tabu Search จะไม่มีการยอมรับคำตอบที่ด้อยกว่าเดิมเมื่อนอัลกอริทึม Simulated Annealing ดังนั้นคำตอบปัจจุบัน และคำตอบที่ดีที่สุดจะเป็นคำตอบเดียวกัน

เพื่อเป็นการทำความเข้าใจถึงกระบวนการค้นหาคำตอบโดยใช้อัลกอริทึม Tabu Search จะยกตัวอย่างการทำงานดังต่อไปนี้

ตัวอย่างที่ 4.4 *การทำงานของอัลกอริทึม Tabu Search*

กำหนดให้จำนวนโนดเท่ากับ 5

กำหนดให้คำตอบเริ่มต้นคือ [1 – 2 – 3 – 4 – 5]

กำหนดให้ลิสต์ทาบูเริ่มต้นคือ [0 – 0 – 0 – 0 – 0] (สังเกตได้ว่าขนาดของลิสต์ทาบูจะเท่ากับขนาดของคำตอบ)

และกำหนดให้ตำแหน่งที่ถูกคงที่จะถูกคงที่ด้วยจำนวนรอบเท่ากับ 3

ตารางที่ 4.2 ตัวอย่างขั้นตอนการทำงานการค้นหาคำตอบโดยใช้อัลกอริทึม Tabu Search

รอบที่	ตำแหน่งที่มีการสลับที่	คำตอบใหม่	ลิสต์ทาบู
1	1 แทนที่ 3	[3 – 2 – **1** – 4 – 5]	[0 – 0 – **3** – 0 – 0]
2	2 แทนที่ 4	[3 – 4 – **1** – **2** – 5]	[0 – 0 – **2** – **3** – 0]
3	1 แทนที่ 5	[5 – 4 – **1** – **2** – **3**]	[0 – 0 – **1** – **2** – **3**]
4	2 แทนที่ 1	[**4** – 5 – 1 – **2** – **3**]	[**3** – 0 – 0 – **1** – **2**]
5	3 แทนที่ 2	[**4** – **1** – 5 – 2 – **3**]	[**2** – **3** – 0 – 0 – **1**]
:	:	:	:

จากตารางที่ 4.2 จะเห็นได้ว่าแต่ละตำแหน่งที่ถูกขีดเส้นใต้ไว้จะถูกคงที่เป็นจำนวนรอบอย่างน้อยเท่ากับ 3 รอบ โดยเริ่มต้นตั้งแต่รอบที่ 1 เมื่อมีการสลับที่ระหว่างตำแหน่งที่ 1 และ 3 ลิสต์ทาบูจาก [0 – 0 – 0 – 0 – 0] จะถูกอัพเดทเป็น [0 – 0 – 3 – 0 – 0] ซึ่งหมายความว่าที่ตำแหน่งที่ 3 จะถูกคงที่ไว้เป็นระยะเวลาเท่ากับ 3 รอบ เมื่อเริ่มรอบที่ 2 มีการสลับที่ระหว่างตำแหน่งที่ 2 และ 4 ลิสต์ทาบูจาก [0 – 0 – 3 – 0 – 0] จะถูกอัพเดทเป็น [0 – 0 – 2 – 3 – 0] ซึ่งหมายความว่า ที่ตำแหน่งที่ 4 จะถูกคงที่ไว้เป็นระยะเวลาเท่ากับ 3 รอบ แต่ที่ตำแหน่งที่ 3 จะถูกลดค่าลงมาเหลือแค่ 2 จะมีความหมายว่าที่ตำแหน่งที่ 3 นี้จะถูกคงที่อีก 2 รอบ โดยความหมายของคำว่าคงที่หมายถึง จะไม่มีการสลับที่กับตำแหน่งนั้นเด็ดขาด เพราะฉะนั้นสมมติว่ารอบที่ 2 แทนที่จะกำหนดให้เกิดการสลับที่ระหว่างตำแหน่งที่ 2 และ 4 แต่เปลี่ยนเป็นตำแหน่งที่ 2 และ 3 แทน ซึ่งตำแหน่งที่ 3 ในลิสต์ทาบูมีค่าเท่ากับ 3 ดังนั้นการสลับตำแหน่งที่ 2 และ 3 ในลักษณะนี้ไม่สามารถทำได้ ดังนั้นจะเห็นได้ว่าพารามิเตอร์ที่สำคัญต่ออัลกอริทึม Tabu Search คือ ค่าจำนวนรอบที่ถูกคงที่ไว้ของแต่

ละตำแหน่ง โดยในที่นี้จะขอเรียกว่า *iteration tabu* : r_t สำหรับวิธีกำหนดพารามิเตอร์และผลลัพธ์ที่ได้จะกล่าวโดยละเอียดในเนื้อหาบทที่ 6

4.6 การค้นหาคำตอบโดยใช้อัลกอริทึม Modified Compact Genetic Algorithm (McGA)

อัลกอริทึมประยุกต์คอมแพคเจเนติก (Modified Compact Genetic Algorithm : McGA) เป็นการนำเอาอัลกอริทึมมาตรฐานคอทมแพคเจเนติก (Compact Genetic Algorithm : cGA) (George, 1999) มาประยุกต์ใช้ โดยนำมาทำการดัดแปลงให้สอดคล้องกับการทำงาน ดังนั้นเนื้อหาในบทนี้จะแบ่งออกเป็น 2 ส่วนคือ

ก.) หลักการทำงานของอัลกอริทึมคอมแพคเจเนติก (Compact Genetic Algorithm : cGA)

ข.) การประยุกต์อัลกอริทึมคอมแพคเจเนติกเพื่อใช้ในการกำหนดเส้นทางให้กับโนด (Modified Compact Genetic Algorithm : McGA)

4.6.1 หลักการทำงานของอัลกอริทึมคอมแพคเจเนติก (Compact Genetic Algorithm : cGA)

อัลกอริทึมคอมแพคเจเนติก (Compact Genetic Algorithm : cGA) เป็นอัลกอริทึมที่มีการพัฒนามาจากอัลกอริทึมเจเนติก (Genetic Algorithm : GA) อีกทีหนึ่ง โดยจะเน้นคำตอบที่อยู่ในรูปไบนารี (Binary) ที่มีการ crossover แบบยูนิฟอร์ม (Uniform Crossover) โดยเนื้อหาในบทนี้จะอธิบายหลักการทำงานของอัลกอริทึม cGA สำหรับรายละเอียดหลักการทำงานของอัลกอริทึม GA ศึกษาต่อได้ใน (Ronald, 1998) และรายการอ้างอิงประเภท web pages การทำงาน (Pseudo Code) ของ cGA อธิบายได้ดังรูปที่ 4.10

```
1) initialize probability vector
   for i := 1 to l do p[i] := 0.5;

2) generate two individuals from the vector
   a := generate(p);
   b := generate(p);

3) let them compete
   winner,loser := compete(a,b);

4) update the probability vector towards
   for i := 1 to l do
         if winner[i] ≠ loser[i] then
         if winner[i] = 1    then p[i] := p[i] + 1/n
                             else p[i] := p[i] – 1/n;

5) check if the vector has converged
   for i := 1 to l do
         if p[i] > 0 and p[i] < 1 then
               return to step 2;

6) p represents the final solution

compact GA parameters:
   n: population size.
   l: chromosome length.

         The Figure from (George, Fernando and David, 1999:289)
```

รูปที่ 4.10 หลักการทำงานของอัลกอริทึม cGA

จากรูปที่ 4.10 การทำงานของ cGA แบ่งออกเป็น 6 ขั้นตอนคือ

ขั้นตอนที่ 1: เป็นการกำหนดค่าเวกเตอร์ความน่าจะเป็น ที่มีมิติเท่ากับ *l* และแต่ละค่าในแต่ละมิติมีค่าความน่าจะเป็นเริ่มต้นเท่ากับ 0.5

ขั้นตอนที่ 2: เป็นการสร้างชุดคำตอบขึ้นมา 2 ชุดจากเวกเตอร์ความน่าจะเป็น โดยค่าความน่าจะเป็นในแต่ละมิติเป็นค่าความน่าจะเป็นของการสุ่มตัวแปรแล้วได้ค่าเป็น 1 ดังนั้นชุดคำตอบแต่ละชุดจะประกอบไปด้วย “0” และ “1” ซึ่งมีจำนวนรวมกันทั้งหมดเท่ากับมิติของเวกเตอร์ความน่าจะเป็น

ขั้นตอนที่ 3: นำชุดคำตอบที่ได้ (ที่เป็นไบนารี) มาทำการเปรียบเทียบโดยใช้ objective function ที่ขึ้นกับงานที่ใช้ ทำการเปรียบเทียบหาชุดคำตอบที่ได้ค่า objective function ที่ดีกว่า และเรียกชุดคำตอบที่ดีกว่าว่า “*winner*” เรียกชุดคำตอบที่ด้อยกว่าว่า “*loser*”

ขั้นตอนที่ 4: เป็นการปรับปรุงค่าเวกเตอร์ความน่าจะเป็น โดยเปรียบเทียบชุดคำตอบทีละมิติของ *winner* และ *loser* ถ้ามิติที่ *i* ใด ๆ ของ *winner* มีค่าเป็น 1 ให้ปรับปรุงเพิ่มค่าความน่าจะเป็นที่ตำแหน่งมิติที่ *i* ในเวกเตอร์ความน่าจะเป็นด้วย 1/n ในทางตรงข้ามถ้ามิติที่ *i* ใด ๆ ของ *winner* มีค่าเป็น 0 ให้ปรับปรุงลดค่าความน่าจะเป็นที่ตำแหน่งมิติที่ *i* ในเวกเตอร์ความน่าจะเป็นด้วย 1/n เมื่อ n คือขนาดของประชากร

ขั้นตอนที่ 5: เป็นการตรวจสอบการข้อกำหนดของการสิ้นสุดการทำงาน โดยจะสิ้นสุดการทำงานของอัลกอริทึมเมื่อค่าความน่าจะเป็นในมิติใด ๆ ในเวกเตอร์ความน่าจะเป็นมีค่าไม่อยู่ในช่วง [0,1]

ขั้นตอนที่ 6: เวกเตอร์ความน่าจะเป็นที่ได้จะเป็นคำตอบที่ต้องการ

แต่เนื่องจากการทำงานของอัลกอริทึมคอมแพคเจเนติก มีลักษณะเหมาะสมกับงานที่มีลักษณะการทำงานเกี่ยวข้องกับไบนารี ไม่เหมาะสมสำหรับงานกำหนดเส้นทางให้กับโมบายล์เอเจนต์โดยตรง ดังนั้นจึงต้องมีการประยุกต์อัลกอริทึม เพื่อให้เหมาะสมกับงานกำหนดเส้นทางโดยใช้อัลกอริทึมประยุกต์คอมแพคเจเนติก

4.6.2 อัลกอริทึมประยุกต์คอมแพคเจเนติก (Modified Compact Genetic Algorithm : McGA)

สำหรับเนื้อหาในหัวข้อนี้จะเป็นการอธิบายถึงหลักการทำงานของ McGA โดยขั้นตอนการทำงานของอัลกอริทึมสามารถอธิบายได้เป็นขั้นตอนย่อยดังต่อไปนี้

ขั้นตอนที่ 1: *การกำหนดเวกเตอร์ความน่าจะเป็นให้กับแต่ละโนดในเนตเวิร์ก (Probabilistic Vector)* โดยแต่ละโนดจะมีค่าเวกเตอร์ความน่าจะเป็นของตัวมันเอง ซึ่งเวกเตอร์ความน่าจะเป็นนี้จะมีมิติ (Dimension) เท่ากับจำนวนเส้นทาง (Route Amount : *r*) ที่จะกำหนดให้กับโมบายล์เอเจนต์ กล่าวคือ ถ้ากำหนดให้มีการใช้เส้นทางเท่ากับ 3 ดังนั้นเวกเตอร์ความน่าจะเป็นจะมีมิติเท่ากับ 3 (ซึ่งตรงกับค่า *l* ในอัลกอริทึม cGA)และสำหรับค่าเริ่มต้นของความน่าจะเป็นในแต่ละมิติจะมีค่าเท่ากับ ส่วนกลับของจำนวนเส้นทางนั่นเอง (1/r) ดังนั้นถ้าในเนตเวิร์กมีจำนวนโนดเท่ากับ n จะต้องมีเวกเตอร์ความน่าจะเป็นทั้งหมด n ชุดนั่นเอง

ขั้นตอนที่ 2: *การกำหนดอันดับของเส้นทางให้กับแต่ละโนดโดยใช้เวกเตอร์ความน่าจะเป็น* จากขั้นตอนที่ 1 แต่ละโนดจะมีเวกเตอร์ความน่าจะเป็นที่มีมิติเท่ากับจำนวนเส้นทางที่ใช้ในเนตเวิร์ก ดังนั้นเมื่อพิจารณาแต่ละเวกเตอร์ความน่าจะเป็นจะเห็นได้ว่า ผลรวมของความน่าจะเป็นในแต่ละมิติจะมีค่าเป็น 1 ดังสมการที่ 4.3

กำหนดให้จำนวนเส้นทางที่ใช้เท่ากับ r

ดังนั้นความน่าจะเป็นแต่ละมิติของเวกเตอร์ความน่าจะเป็นเท่ากับ 1/r

ดังนั้นจะได้เวกเตอร์ความน่าจะเป็นของโนดที่ *I* เป็น

$$p_i = \left(1/r_1, 1/r_2, 1/r_3, 1/r_4, ..., 1/r_r \right) \qquad (4.3)$$

เมื่อ $1/r_j$ คือความน่าจะเป็นของมิติที่ j โดยมีค่าเริ่มต้นเท่ากับ $1/r$

ดังนั้น $\sum_{j=1}^{r} 1/r_j = r \bullet 1/r = 1$

ดังนั้นการกำหนดลำดับของเส้นทางจากเวกเตอร์จะกำหนดโดยการสร้างตัวแปรสุ่มขึ้นมาตัวหนึ่งที่มีช่วงอยู่ระหว่าง 0 ถึง 1 โดยผลลัพธ์การสุ่มจะนำมาเปรียบเทียบกับความน่าจะเป็นสะสมของแต่ละมิติในแต่ละเวกเตอร์ โดยผลลัพธ์ที่ได้จะอยู่เส้นทางที่ *j* เมื่อตัวแปรสุ่มอยู่ในช่วง (cp_{j-1}, cp_j) เมื่อกำหนดให้ cp_j คือความน่าจะเป็นสะสมของมิติที่ *j* ดังสมการที่ 4.4

$$cp_j = \sum_{i=1}^{i=j} 1/r_i \qquad (4.4)$$

ดังนั้นผลลัพธ์ที่ได้จากแต่ละเวกเตอร์ความน่าจะเป็นจะเป็นอันดับของเส้นทางที่โนดนั้นจะเป็นสมาชิก ดังนั้นผลลัพธ์ที่ได้จากขั้นตอนนี้จะทำให้รู้ได้ว่า โนดใดจะต้องอยู่ในเส้นทางใด สำหรับในการเริ่มต้นใช้งานของอัลกอริทึม McGA จะต้องเริ่มต้นกำหนดเส้นทางไว้ 2 ชุดโดยจะเรียกว่าเส้นทางชุดที่ 0 และเส้นทางชุดที่ 1 เพื่อใช้ในขั้นตอนที่ 4

ขั้นตอนที่ 3: *การนำเส้นทางที่ได้ไปผ่านการจัดลำดับเส้นทางโดยใช้อัลกอริทึม Simulated Annealing* หลังจากที่ได้เส้นทางที่ประกอบไปด้วยโนดที่แน่นอนแล้วขั้นตอนต่อไปคือนำแต่ละเส้นทางมาจัดลำดับของโนดเพื่อให้ได้เส้นทางที่ใช้ค่าต้นทุนที่น้อยที่สุด โดยในที่นี้จะใช้อัลกอริทึม Simulated Annealing มาใช้ในการจัดลำดับ

ขั้นตอนที่ 4: *การปรับปรุงเวกเตอร์ความน่าจะเป็น* จากขั้นตอนที่ 2 ได้มีการกำหนดเส้นทางไว้ 2 ชุดคือเส้นทางชุดที่ 0 และเส้นทางชุดที่ 1 ดังนั้นหลักการขั้นตอนที่ 4 คือการเปรียบเทียบว่า เส้นทางที่กำหนดใหม่ (เส้นทางชุดที่ 1) มีการพัฒนาคำตอบ (MMSRC) ที่ดีขึ้นเมื่อเทียบกับเส้นทางเดิม (เส้นทางชุดที่ 0) หรือไม่ โดยถ้ามีการพัฒนาของคำตอบ (ค่า MMSRC ที่ได้มีค่าน้อยลง) จะทำการปรับค่าความน่าจะเป็นในแต่ละมิติของเวกเตอร์ความน่าจะเป็นของแต่ละโนด โดยมีหลักการคือ ถ้าเส้นทางที่ได้ใหม่มีการพัฒนาของคำตอบ จะมีการตรวจสอบแต่ละเส้นทางย่อยในเส้นทางชุดใหม่ว่าประกอบด้วยโนดใดบ้าง โดยจะเพิ่มค่าความน่าจะเป็นของมิติ (สำหรับมิติที่มีค่าอันดับเท่ากับอันดับของเส้นทาง) ในเวกเตอร์ของแต่ละโนดด้วยค่าเท่ากับ $(r-1)/r^2$ และจะลดค่าความน่าจะเป็นของมิติ (สำหรับมิติที่มีค่าอันดับไม่เท่ากับอันดับของเส้นทาง) ในเวกเตอร์ของแต่ละโนดด้วยค่าเท่ากับ $1/r^2$ โดยจะอธิบายดังตัวอย่างที่ 4.1

ตัวอย่างที่ 4.1 *การปรับปรุงเวกเตอร์ความน่าจะเป็น*

กำหนดให้จำนวนโนดเท่ากับ 7

กำหนดให้มีการใช้เส้นทาง (*r*) เท่ากับ 3

ดังนั้น $1/r = 0.3333$ และ $1/r^2$ เท่ากับ 0.1111

กำหนดให้เส้นทาง (เส้นทางชุดที่ 0) ที่ผ่านขั้นตอนที่ 3 แล้วเป็น

[1 – 2 – 3] [4 – 5] [6 – 7]

และกำหนดให้เวกเตอร์ความน่าจะเป็นของแต่ละโนดมีค่าดังนี้

เวกเตอร์โนดที่ 1 : (0.3333, 0.3333, 0.3333)

เวกเตอร์โนดที่ 2 : (0.3333, 0.3333, 0.3333)

เวกเตอร์โนดที่ 3 : (0.3333, 0.3333, 0.3333)

เวกเตอร์โนดที่ 4 : (0.3333, 0.3333, 0.3333)

เวกเตอร์โนดที่ 5 : (0.3333, 0.3333, 0.3333)

เวกเตอร์โนดที่ 6 : (0.3333, 0.3333, 0.3333)

เวกเตอร์โนดที่ 7 : (0.3333, 0.3333, 0.3333)

กำหนดให้เส้นทางที่กำหนดขึ้นใหม่ (เส้นทางชุดที่ 1) ที่ผ่านขั้นตอนที่ 3 และมีการพัฒนาของค่า MMSRC เป็น [4 – 1 – 6] [7 – 3] [2 – 5]

ดังนั้นจะมีการปรับปรุงค่าความน่าจะเป็นในแต่ละมิติในแต่ละเวกเตอร์ของแต่ละโนดเป็น

เวกเตอร์โนดที่ 1 : (**0.5555**, 0.2222, 0.2222)

เวกเตอร์โนดที่ 2 : (0.2222, 0.2222, **0.5555**)

เวกเตอร์โนดที่ 3 : (0.2222, **0.5555**, 0.2222)

เวกเตอร์โนดที่ 4 : (**0.5555**, 0.2222, 0.2222)

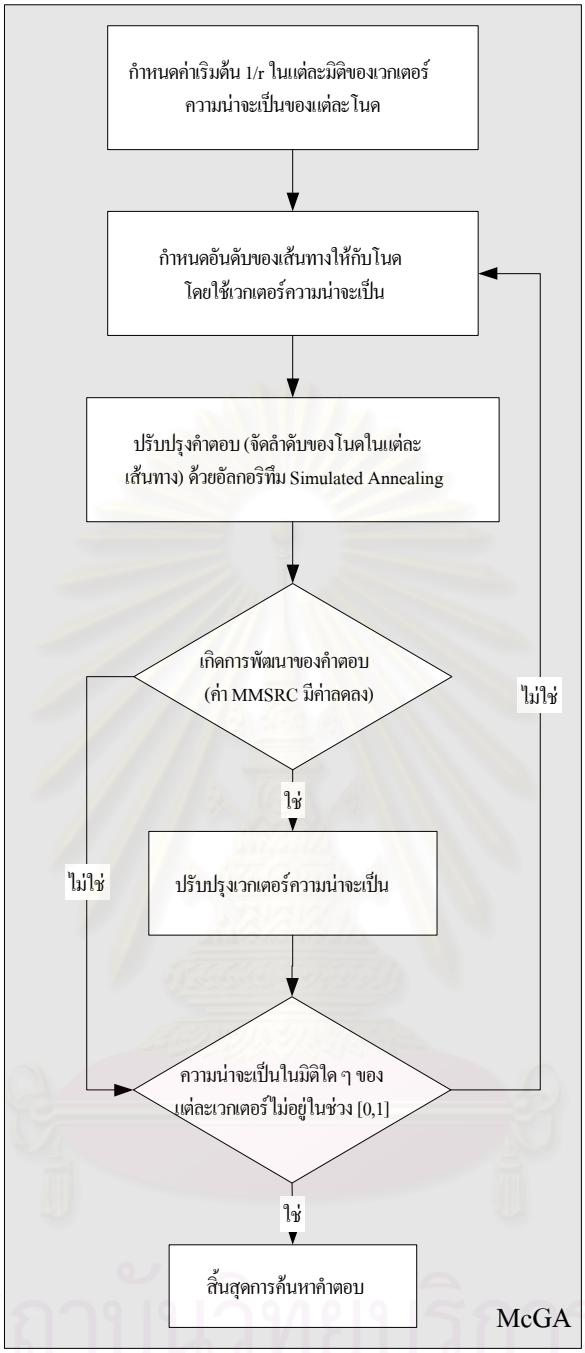
เวกเตอร์โนดที่ 5 : (0.2222, 0.2222, **0.5555**)

เวกเตอร์โนดที่ 6 : (**0.5555**, 0.2222, 0.2222)

เวกเตอร์โนดที่ 7 : (0.2222, **0.5555**, 0.2222)

ขั้นตอนที่ 5: *การหยุดการทำงาน* สำหรับการหยุดการทำงานจะหยุดเมื่อค่าความน่าจะเป็นในมิติใด ๆ ในเวกเตอร์ความน่าจะเป็นของโนดใด ๆ ก็ตามที่มีค่ามากกว่าหนึ่ง และน้อยกว่าศูนย์จะถือเป็นการยุติการทำงานในทันที และเวกเตอร์ความน่าจะเป็นของแต่ละโนดที่ได้จะเป็นคำตอบ

จากขั้นตอนที่กล่าวมาทั้งหมดสามารถเขียนผังงานได้ดังรูปที่ 4.11



รูปที่ 4.11 ผังงานการทำงานของอัลกอริทึม McGA

สำหรับการผลลัพธ์และการวิเคราะห์จะกล่าวไว้ในบทที่ 6 การวิเคราะห์อัลกอริทึมฮิวริสติก

บทที่ 5

วิธีกำหนดเส้นทางให้กับโมบายล์เอเจนต์

5.1 กล่าวนำ

สำหรับเนื้อหาในบทนี้จะเสนอวิธีการกำหนดเส้นทางให้กับโมบายล์เอเจนต์ โดยนำอัลกอริทึมมาตรฐานคือ Brute Force Search (BF), Simulated Annealing (SA) และ Tabu Search (Tabu) มาประยุกต์ใช้ในอัลกอริทึมฮิวริสติกที่คิดขึ้น และมีการเสนออัลกอริทึมที่คิดขึ้นเพิ่มเติมอีก 2 ชนิดคือ อัลกอริทึมประยุกต์คอมแพคเจเนติก (Modified Compact Genetic Algorithm : McGA) ซึ่งได้อธิบายไว้แล้วในบทที่ 4 และอัลกอริทึมกำหนดเส้นทางอิสระแบบยูนิฟอร์ม (Random Uniformly Algorithm : RU) เพื่อนำมาใช้วิเคราะห์ในเชิงเปรียบเทียบกับผลลัพธ์ที่ได้จากอัลกอริทึมฮิวริสติก

วิทยานิพนธ์ฉบับนี้นอกจากจะเสนอการวิเคราะห์ในเชิงเปรียบเทียบกับอัลกอริทึมต่างๆ แล้ว ยังได้เสนอการวิเคราะห์เชิงเปรียบเทียบด้านประสิทธิภาพและสมรรถนะของอัลกอริทึมต่าง ๆ (Robustness) โดยการนำไปประมวลผลบนเนตเวิร์กจำลองที่มีสภาวะแวดล้อมที่แตกต่างกัน ดังนั้นเนื้อหาในบทนี้จะกล่าวถึงรายละเอียดของอัลกอริทึมฮิวริสติก อัลกอริทึม RU และวิธีกำหนดเนตเวิร์กจำลองที่มีสภาวะแวดล้อมแตกต่างกัน โดยจะแบ่งเนื้อหาออกเป็น 3 ส่วนคือ

ก.) ขั้นตอนการทำงานของอัลกอริทึมฮิวริสติก
ข.) ขั้นตอนการทำงานของอัลกอริทึม RU
ค.) วิธีกำหนดเนตเวิร์กจำลองที่มีสภาวะแวดล้อมแตกต่างกัน

5.2 ขั้นตอนการทำงานของอัลกอริทึมฮิวริสติก

อัลกอริทึมฮิวริสติกที่คิดขึ้นมาใช้ในวิทยานิพนธ์นี้จะแบ่งออกเป็น 3 แบบคือ
ก.) ฮิวริสติกแบบที่ 1 (HR-1)
ข.) ฮิวริสติกแบบที่ 2 (HR-2)
ค.) ฮิวริสติกแบบที่ 3 (HR-3)

5.2.1 อัลกอริทึมฮิวริสติกแบบที่ 1 (HR-1)

ขั้นตอนการทำงานของอัลกอริทึมฮิวริสติกแบบที่ 1 จะมีอยู่ 2 ขั้นตอนดังนี้ (รูปที่ 5.1)

รูปที่ 5.1 ผังงานขั้นตอนการทำงานของอัลกอริทึมฮิวริสติกแบบที่ 1

ขั้นตอนที่ 1: การกำหนดเส้นทางโดยใช้อัลกอริทึมฮิวริสติก –1 การทำงานของอัลกอริทึมฮิวริสติก-1 มีแนวคิดว่า ในการส่งโมบายล์เอเจนต์ไปยังเซิร์ฟเวอร์ต่าง ๆ ที่ต้องการนั้น โมบายล์เอเจนต์แต่ละตัวควรเลือกไปยังเซิร์ฟเวอร์ที่อยู่ใกล้กัน กล่าวคือถ้ามองเนตเวิร์กคือกลุ่มของเซิร์ฟเวอร์ ดังนั้นจำนวนโมบายล์เอเจนต์ที่ใช้ในการประมวลผลที่เซิร์ฟเวอร์ควรจะมีจำนวนเท่ากับจำนวนของกลุ่มของเซิร์ฟเวอร์ที่อยู่ใกล้กันในเนตเวิร์กนั่นเอง ดังนั้นส่วนที่สำคัญสำหรับขั้นตอนนี้คือ การกำหนดเซิร์ฟเวอร์ที่อยู่ใกล้กันให้อยู่ในกลุ่มของเส้นทางเดียวกัน จากที่ได้เคยกล่าวไว้ในเรื่องการกำหนดค่าต้นทุนในบทที่ 4 ค่าต้นทุนที่กำหนดให้ลิงค์จะมี 2 แบบคือ ค่าต้นทุนแลน และค่าต้นทุนแวน ดังนั้นหลักการกำหนดเส้นทางคือการนำเซิร์ฟเวอร์ที่เชื่อมโยงกันด้วยลิงค์ต้นทุนแลนให้อยู่ในกลุ่มเดียวกัน หรือกล่าวคือเป็นการจัดกลุ่มของเซิร์ฟเวอร์ เมื่อได้กลุ่มของเซิร์ฟเวอร์แล้ว ขั้นตอนต่อไปคือการจัดลำดับก่อนหลังของเซิร์ฟเวอร์ในแต่ละกลุ่ม โดยนำอัลกอริทึม SA หรืออัลกอริทึม Tabu มาทำการจัดลำดับ หรือการปรับปรุงคำตอบนั่นเอง

สำหรับการทำงานของอัลกอริทึมฮิวริสติก-1 สามารถอธิบายได้ดังอัลกอริทึมต่อไปนี้

อัลกอริทึมฮิวริสติก-1 : Pseudo Code

```
route_amount = 0;
for i = 1 to n
      unmarked(node[i]);

for i = 1 to n {
      if node[i] is not marked {
            mark(node[i]);
            route_amount++;
            Join_Route(route_amount,node_i);
            for j = i+1 to n {
                  if node[j] is not marked {
                        if Link(node_i,node_j) is LAN cost {
                              Join_Route(route_amount,node_j);
                              mark(node[j]);
                        }}}}}
```

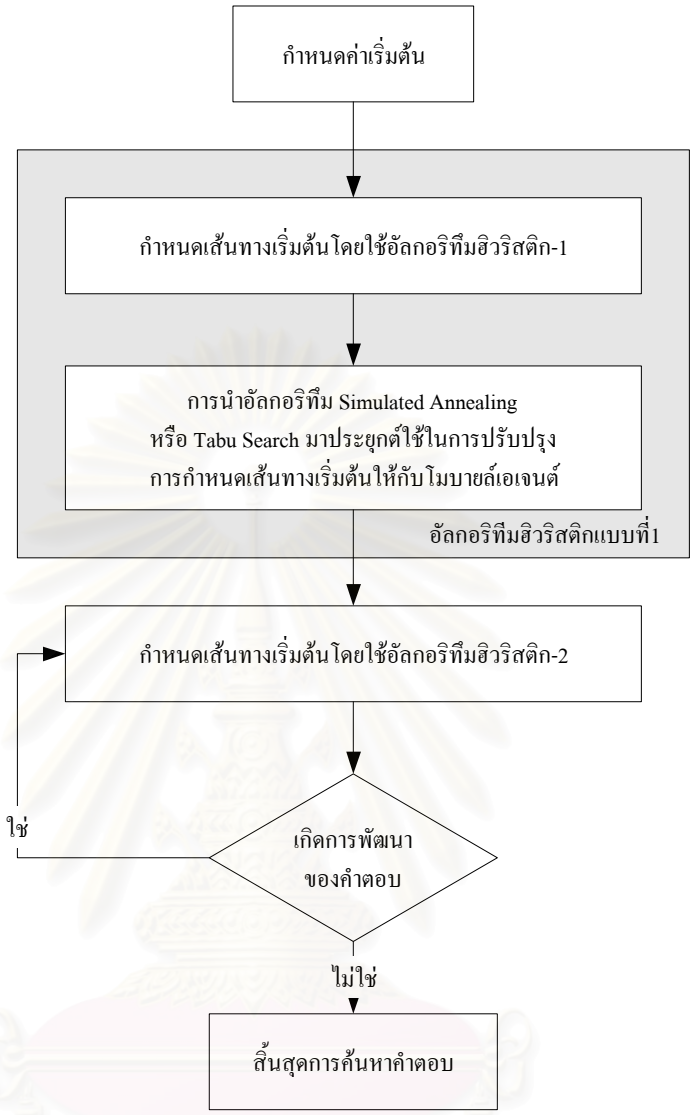
เมื่อ `Join_Route(x,node_y)` คือการรวม $node_y$ ไว้ในเส้นทางย่อยที่ x

`Link(node_x,node_y)` คือค่าต้นทุนลิงก์ที่เชื่อมต่อ $node_x$ และ $node_y$

ขั้นตอนที่ 2: การนำ SA และ Tabu มาประยุกต์ในการปรับปรุงการกำหนดเส้นทางให้กับโมบายล์เอเจนต์ จากอัลกอริทึมฮิวริสติก-1 จะได้ชุดของเส้นทางที่จะกำหนดให้กับโมบายล์เอเจนต์ โดยจะนำแต่ละเส้นทางย่อยมาทำการจัดลำดับโดยใช้อัลกอริทึม SA หรืออัลกอริทึม Tabu และนำค่าต้นทุนของแต่ละเส้นทางย่อยที่ได้นำมาหาค่า MMSRC สำหรับอัลกอริทึมฮิวริสติกแบบที่ 1 ที่ใช้อัลกอริทึม SA จะเรียกว่า HRS-1 และอัลกอริทึมฮิวริสติกแบบที่ 1 ที่ใช้อัลกอริทึม Tabu จะเรียกว่า HRT-1

5.2.2 อัลกอริทึมฮิวริสติกแบบที่ 2 (HR-2)

ขั้นตอนการทำงานของอัลกอริทึมฮิวริสติกแบบที่ 2 จะมีอยู่ 3 ขั้นตอน โดย 2 ขั้นตอนแรกจะเหมือนกับอัลกอริทึมฮิวริสติกแบบที่ 1 ทุกประการ แต่จะเพิ่มขั้นตอนที่ 3 ซึ่งเป็นวิธีการปรับปรุงคำตอบโดยใช้อัลกอริทึมฮิวริสติก-2 แสดงไว้ดังรูปที่ 5.2



รูปที่ 5.2 ผังงานขั้นตอนการทำงานของอัลกอริทึมฮิวริสติกแบบที่ 2

จากรูปที่ 5.2 ขั้นตอนที่เพิ่มขึ้นมาคือขั้นตอนของการปรับปรุงโดยใช้อัลกอริทึมฮิวริสติก-2 ซึ่งมีแนวคิดที่จะนำกลุ่มของเซิร์ฟเวอร์ที่ได้จากการผ่านอัลกอริทึมฮิวริสติกแบบที่ 1 มารวมกัน โดยมองว่า กลุ่มของเซิร์ฟเวอร์ที่ใกล้กันควรจะนำมารวมให้อยู่ในเส้นทางเดียวกัน สำหรับขั้นตอนการรวมกลุ่มของเซิร์ฟเวอร์ที่อยู่ใกล้กันให้อยู่ในเส้นทางเดียวกัน จะมีอัลกอริทึมดังนี้

อัลกอริทึมฮิวริสติก-2 (HR-2) : Pseudo Code ในส่วนของการรวมกลุ่ม

ขั้นตอนการกำหนดค่าเริ่มต้น (ผลลัพธ์จากอัลกอริทึมฮิวริสติก-1)

กำหนดให้เส้นทางย่อยสำหรับโมบายล์เอเจนต์ตัวที่ i (MA_i) เป็น (s_{i1},s_{i2},…,s_{in})

กำหนดให้ผลลัพธ์ที่ได้จากอัลกอริทึมฮิวริสติก-1 มีโมบายล์เอเจนต์ n ตัว

(MA_i : i = 1 ถึง n)

กำหนดให้ค่าต้นทุนของโมบายล์เอเจนต์ที่ i เป็น *Cost* (MA_i)

และกำหนดให้ค่า MMSRC ณ ขณะนั้น เท่ากับ Z

ขั้นตอนการรวมกลุ่มของเนตเวิร์ก

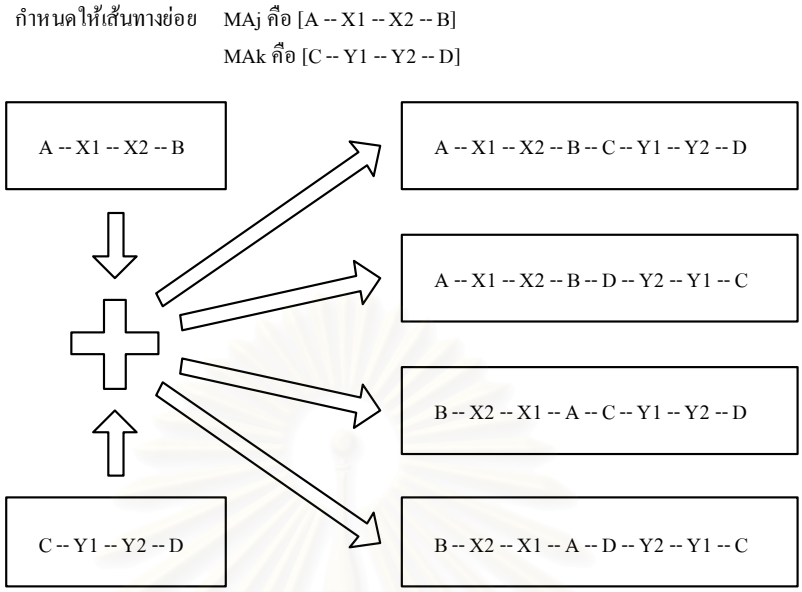
```
for i = 1 to n {
      unmarked MAi;}

j = 0;
for i = 1 to n {
      if (MAi is not marked) {
            j++;
            MAj = MAi;
            MAi is marked;
            for k = i+1 to n {
                  if (MAk is not marked) {
                        if (Cost (Join (MAj,MAk)) <= Z) {
                              MAj = Join (MAj,MAk);
                              MAk is marked;
}}}}}

calculate MMSRC of new itinerary
```

เมื่อ `Join (MAj,MAk)` คือการรวมเส้นทาง MA_j และ MA_k เข้าด้วยกัน

`Cost (Join (MAj,MAk)` คือค่าต้นทุนของการรวมเส้นทางเข้าด้วยกัน

รูปที่ 5.3 Pseudo Code HR-2 การรวมกลุ่มของเซิร์ฟเวอร์

สำหรับการรวมกลุ่มที่เกิดขึ้นในอัลกอริทึมฮิวริสติก-2 (`Join(MAj,MAk)`) จะสามารถเกิดขึ้นได้ 4 กรณี ดังนั้นในการรวมกลุ่มจะต้องเลือกรูปแบบของเส้นทางที่ให้ค่าต้นทุนที่ดีที่สุด (ค่าต้นทุนที่ต่ำที่สุด) โดยการคำนวณการรวมกลุ่มทั้ง 4 แบบแล้วเลือกแบบที่ดีที่สุด ดังรูปที่ 5.4

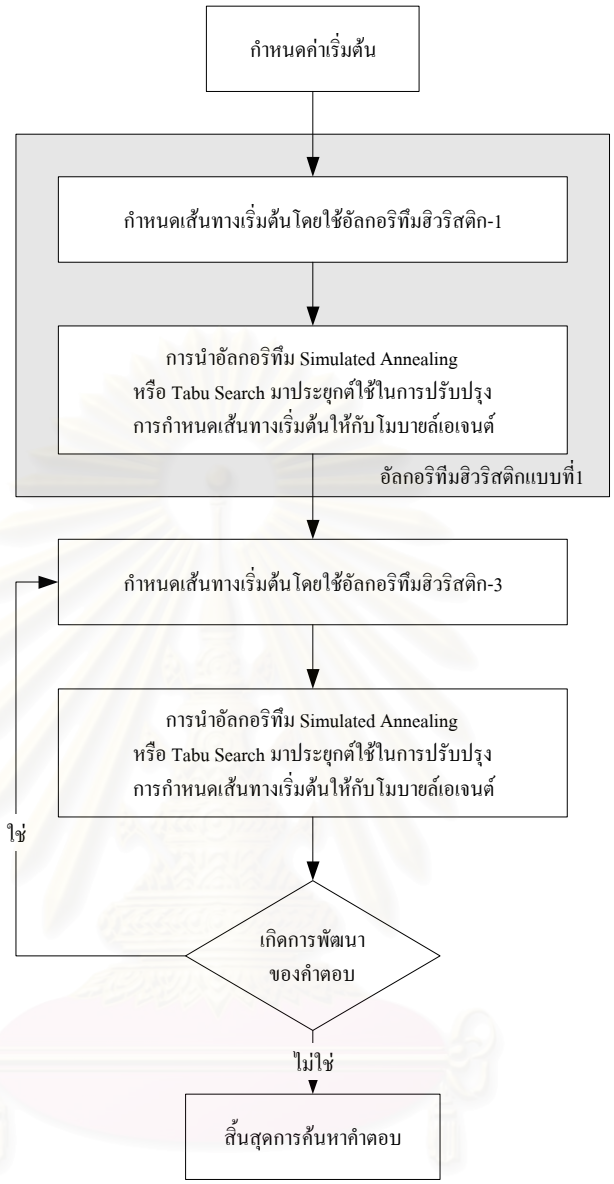


รูปที่ 5.4 ตัวอย่างการรวมเส้นทาง 2 เส้นทางเข้าด้วยกันของ $\texttt{Join}(MA_j, MA_k)$

สำหรับ อัลกอริทึมฮิวริสติกแบบที่ 2 ที่ใช้อัลกอริทึม SA จะเรียกว่า HRS-2 และอัลกอริทึมที่ใช้อัลกอริทึม Tabu จะเรียกว่า HRT-2

5.2.3 อัลกอริทึมฮิวริสติกแบบที่ 3 (HR-3)

ขั้นตอนการทำงานของอัลกอริทึมฮิวริสติกแบบที่ 3 จะนำผลลัพธ์ที่ได้จากการผ่านอัลกอริทึมฮิวริสติกแบบที่ 1 มาทำการปรับปรุง ซึ่งที่มาของแนวคิดของอัลกอริทึมฮิวริสติกแบบที่ 3 จะเหมือนกับอัลกอริทึมฮิวริสติกแบบที่ 2 คือนำกลุ่มของเซิร์ฟเวอร์ที่ใกล้กันให้อยู่ในเส้นทางเดียวกัน แต่วิธีการรวมกลุ่มของอัลกอริทึมแบบที่ 3 จะแตกต่างจากแบบที่ 2 โดยการทำงานแบบที่ 2 ในขั้นตอนของฮิวริสติก-2 จะไม่มีการนำอัลกอริทึม SA หรือ Tabu มาร่วมใช้ในการกำหนดเส้นทางการรวมกลุ่มของเซิร์ฟเวอร์ แต่ในอัลกอริทึมฮิวริสติกแบบที่ 3 จะนำอัลกอริทึม SA หรือ Tabu มาร่วมใช้ในการกำหนดเส้นทางการรวมกลุ่มของเซิร์ฟเวอร์ดังรูปที่ 5.5



รูปที่ 5.5 ผังงานขั้นตอนการทำงานของอัลกอริทึมฮิวริสติกแบบที่ 3

สำหรับวิธีการรวมกลุ่มเซิร์ฟเวอร์เข้าด้วยกันของอัลกอริทึมฮิวริสติก-3 จะแตกต่างจากอัลกอริทึมฮิวริสติก-2 โดยในอัลกอริทึมฮิวริสติก-3 จะมีการทำลิสต์ขึ้นมาชุดหนึ่งที่จับคู่ของกลุ่มของเซิร์ฟเวอร์ที่อยู่ใกล้กันที่สุด จากนั้นจึงนำแต่ละคู่มารวมให้อยู่ในเส้นทางเดียวกัน เมื่อทำครบทุกคู่แล้วจะได้ชุดของเส้นทางใหม่ แล้วนำแต่ละเส้นทางย่อยในชุดของเส้นทางใหม่นี้ไปทำการจัดลำดับของเซิร์ฟเวอร์ใหม่อีกครั้งหนึ่งด้วยอัลกอริทึม SA หรือ Tabu แล้วพิจารณาค่า MMSRC ของเส้นทางใหม่ที่ได้ว่ามีการพัฒนาดีขึ้นกว่าเดิมหรือไม่ ถ้าเกิดการพัฒนาของค่า MMSRC จะยอมรับการรวมกลุ่ม และจะทำการรวมกลุ่มวนซ้ำจนกว่าไม่เกิดการพัฒนาของค่า MMSRC ดังรูปที่ 5.5

5.3 ขั้นตอนการทำงานของอัลกอริทึม RU (Random Uniformly Algorithm)

เพื่อแสดงให้เห็นถึงการเปรียบเทียบประสิทธิภาพการทำงานของอัลกอริทึมฮิวริสติก และอัลกอริทึมอื่น ๆ จึงนำวิธีการกำหนดเส้นทางแบบอิสระ เพื่อใช้เป็นแนวทางในการวิเคราะห์ผลลัพธ์ที่ได้จากแต่ละอัลกอริทึม โดยขั้นตอนการทำงานสามารถอธิบายได้ดังนี้

```
RU Algorithm : Pseudo Code

for i = 1 to n {
      for route = 1 to i {
            while (iteration <= iteration_max) {
                  generate_random_itinerary ();
                  calculate_itinerary_cost ();
                  if (improve ())
                        store_itinerary ();
            }
            if (MMSRC_improve ())
                  store_best_itinerary ();
      }
}

n = the number of nodes
route = the amount of route assigned for itinerary
```

รูปที่ 5.6 Pseudo Code RU Algorithm

การทำงานของอัลกอริทึมจะเริ่มจากในแต่ละค่าจำนวนโนด จะทำการแบ่งค่าจำนวนเส้นทางที่ใช้ซึ่งมีค่าเริ่มตั้งแต่ 1 เส้นทางไปจนถึงค่าเท่ากับจำนวนโนดที่กำลังพิจารณาอยู่ จากนั้นที่แต่ละค่าจำนวนเส้นทางที่ใช้จะทำการกำหนดอันดับของโนดให้กับแต่ละเส้นทางแบบอิสระแล้วคำนวณค่าต้นทุน และทำวนซ้ำจนกว่าจะครบจำนวนรอบที่ตั้งไว้ โดยแต่ละรอบจะทำการเปรียบเทียบค่าต้นทุนของเส้นทางที่ได้เปรียบเทียบกับเส้นทางก่อนหน้าเพื่อตรวจสอบการพัฒนาของคำตอบ ถ้ามีการพัฒนาของคำตอบจะเก็บค่าคำตอบนั้นไว้ แต่ถ้าไม่มีการพัฒนาจะปล่อยทิ้งไว้ไม่นำมาพิจารณา เมื่อได้ค่าคำตอบที่ดีที่สุดของแต่ละค่าจำนวนเส้นทางที่ใช้แล้ว จะนำค่าคำตอบที่ดีที่สุดของแต่ละค่าจำนวนเส้นทางที่ใช้มาเปรียบเทียบกัน เพื่อหาค่าจำนวนเส้นทางที่ใช้ที่ดีที่สุดและเก็บคำตอบที่ได้เป็นค่าคำตอบที่ดีสุด

สำหรับฟังก์ชั่นในรูปที่ 5.6 อธิบายความหมายการใช้งานได้ดังนี้

generate_random_itinerary() คือฟังก์ชั่นที่ทำหน้าที่ในการกำหนดอันดับของโนดให้แก่แต่ละเส้นทาง โดยมีการกำหนดโดยใช้ตัวแปรสุ่มแบบ Discrete Uniform

`calculate_itinerary_cost()` คือฟังก์ชั่นที่ทำหน้าที่คำนวณต้นทุนของเส้นทางใหม่ที่กำหนดขึ้น

`improve()` คือฟังก์ชั่นที่ทำหน้าที่ตรวจสอบค่าต้นทุนของเส้นทางใหม่ ว่ามีค่าดีขึ้นกว่าเดิมหรือไม่

`store_itinerary()` คือฟังก์ชั่นที่ทำหน้าที่เก็บชุดคำตอบ (เส้นทาง) ที่ให้ค่าต้นทุนที่ดีขึ้น

`MMSRC_improve()` คือฟังก์ชั่นที่ทำหน้าที่ตรวจสอบค่าต้นทุนที่ดีที่สุดในแต่ละค่าจำนวนเส้นทางที่ใช้ในเนตเวิร์กนำมาเปรียบเทียบว่ามีค่าที่ดีขึ้นกว่าเดิมหรือไม่

`store_best_itinerary()` คือฟังก์ชั่นที่ทำหน้าที่เก็บชุดคำตอบ (เส้นทาง) ที่ดีที่สุดที่ค่า route ขณะนั้นเอาไว้

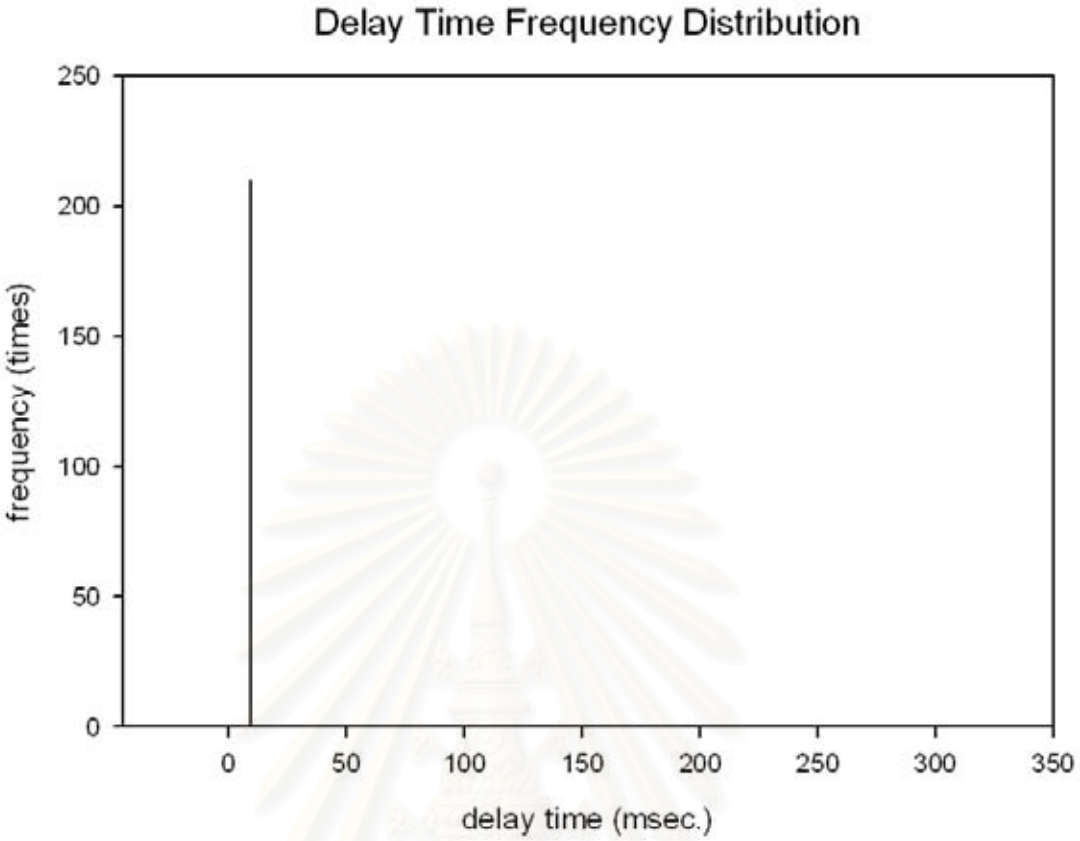
เนื่องจากวัตถุประสงค์ของการนำอัลกอริทึม RU มาใช้เพื่อเป็นเกณฑ์ในการเปรียบเทียบในกรณีกำหนดเส้นทางแบบอิสระโดยไม่ใช้อัลกอริทึมมาตรฐานใดเลย ดังนั้นผลลัพธ์ที่ได้จากอัลกอริทึม RU จะถูกประมวลผลขึ้นมา 2 ชุดคืออัลกอริทึม RU ที่ใช้จำนวนรอบในการประมวลผลเท่ากับจำนวนรอบในการประมวลผลของอัลกอริทึม SA (RU-SA) และ HRS-1 (RU-HR) ซึ่งผลที่ได้และการวิเคราะห์อธิบายไว้ในบทที่ 6

5.4 วิธีกำหนดเนตเวิร์กจำลองที่มีสภาวะแวดล้อมแตกต่างกัน

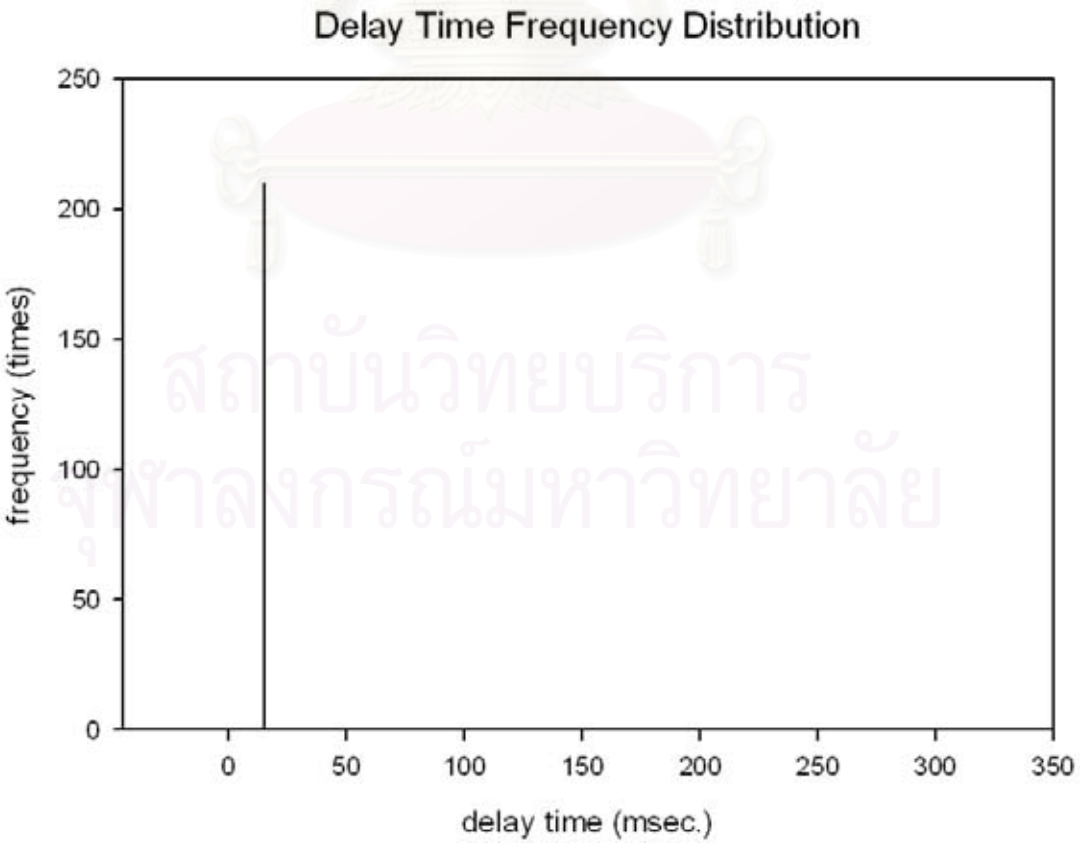
สำหรับเนตเวิร์กจำลองที่จะนำมาใช้ในการประมวลผลเพื่อเปรียบเทียบผลลัพธ์ จะเป็นเนตเวิร์กจำลองที่มีการกำหนดค่าต้นทุนแบบระดับชั้น (Hierarchy) ซึ่งได้อธิบายไว้ในบทที่ 4 แต่เพื่อเป็นการวิเคราะห์เชิงเปรียบเทียบในแง่ Robustness ของแต่ละอัลกอริทึมจึงได้เสนอวิธีกำหนดเนตเวิร์กจำลองขึ้นมาอีก 6 แบบดังนี้

5.4.1 การกำหนดเนตเวิร์กจำลองโดยใช้ค่าต้นทุนค่าเดียว

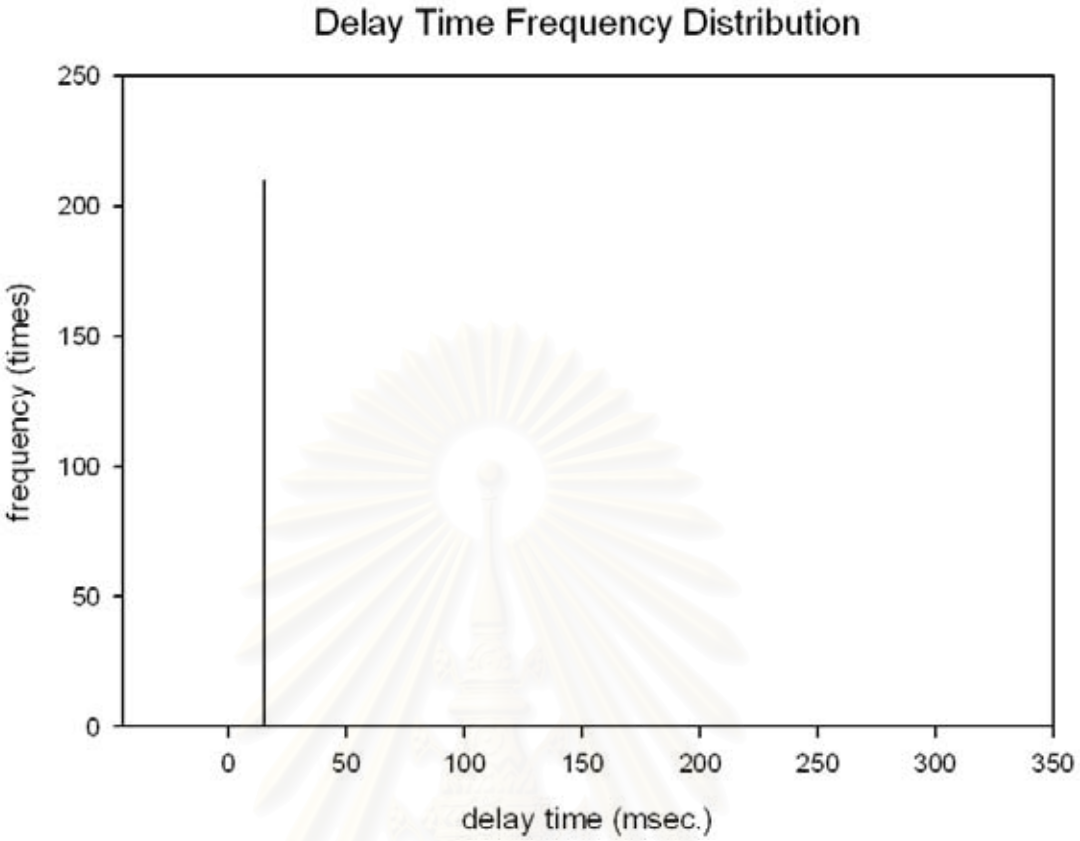
ในการกำหนดเนตเวิร์กจำลองโดยใช้ค่าต้นทุนค่าเดียวเป็นการกำหนดค่าต้นทุนของลิงก์ที่เชื่อมต่อระหว่างโนดในเนตเวิร์กด้วยค่าต้นทุนค่าเดียวกันตลอด สำหรับค่าต้นทุนที่จะใช้กำหนดให้กับเนตเวิร์กจำลองจะใช้ 3 ค่าคือ ค่าเฉลี่ยของค่าต้นทุนแลน (9 msec.) ค่าเฉลี่ยของค่าต้นทุนแวน (158 msec.) ค่าสูงสุดของค่าต้นทุนแลน (15 msec.) และค่าต่ำสุดของค่าต้นทุนแวน (16 msec.) โดยจะแสดงการกระจายของค่าต้นทุนในแต่ละเนตเวิร์กจำลองดังรูปที่ 5.7 – รูปที่ 5.10



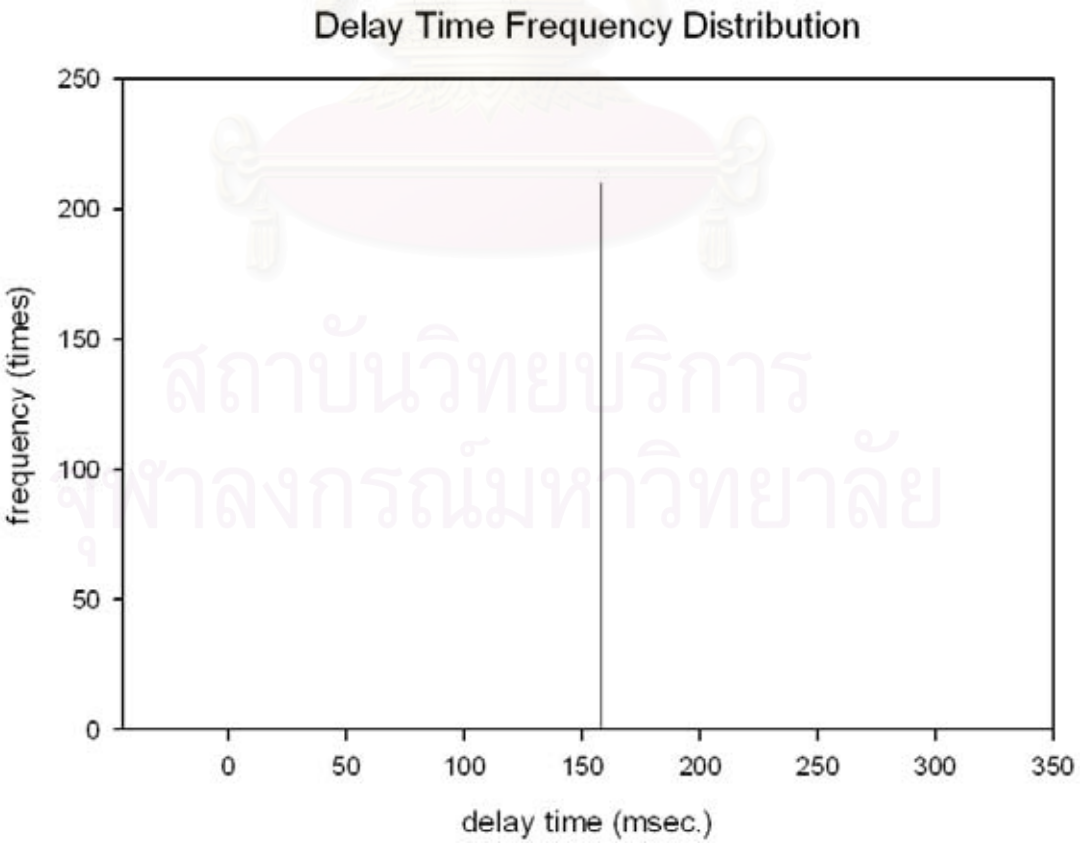
รูปที่ 5.7 การกระจายค่าต้นทุนของเนตเวิร์กจำลอง กรณี 9 msec



รูปที่ 5.8 การกระจายค่าต้นทุนของเนตเวิร์กจำลอง กรณี 15 msec



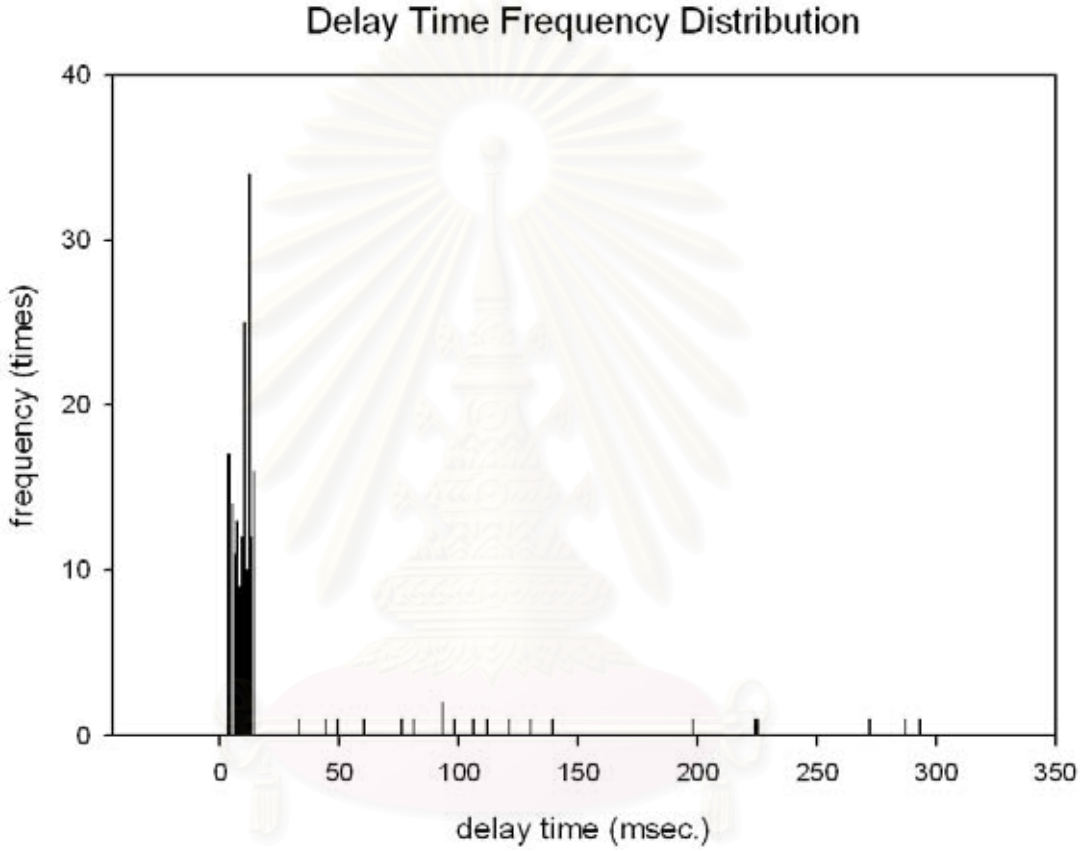
รูปที่ 5.9 การกระจายค่าต้นทุนของเนตเวิร์กจำลอง กรณี 16 msec



รูปที่ 5.10 การกระจายค่าต้นทุนของเนตเวิร์กจำลอง กรณี 158 msec

5.4.2 การกำหนดเนตเวิร์กจำลองแบบ LAN Neighboring (LNI)

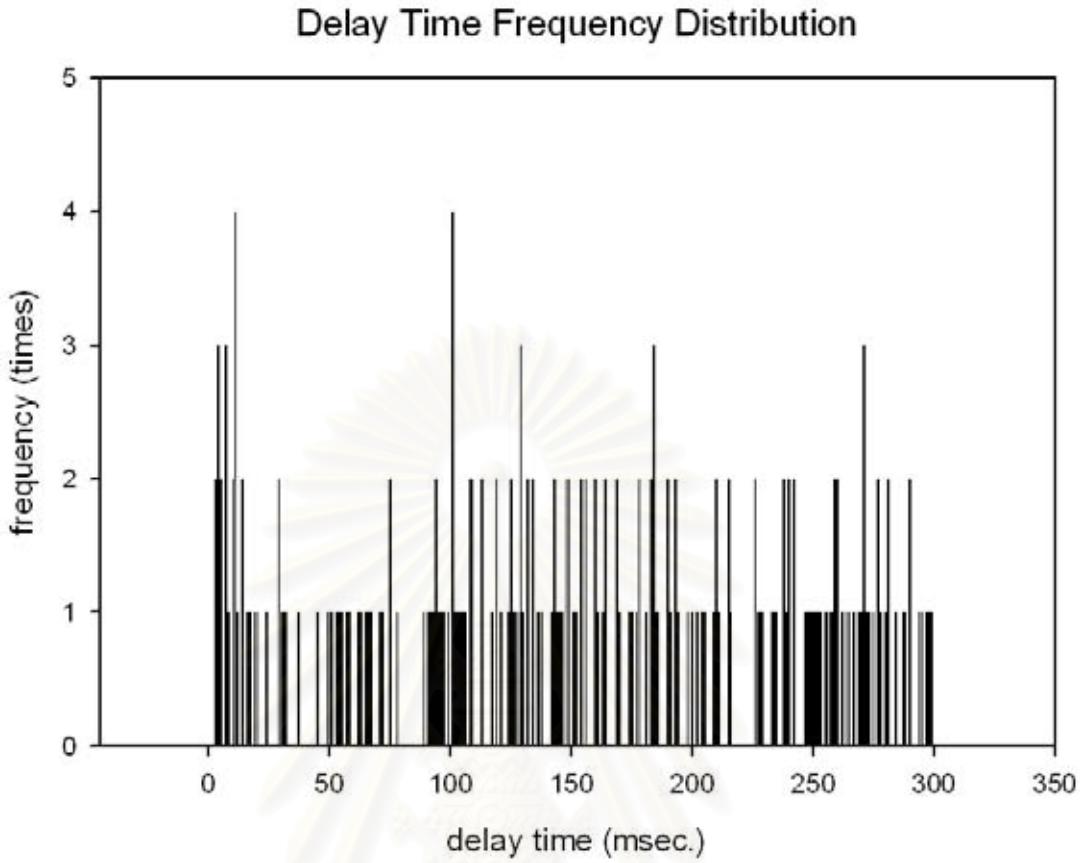
การกำหนดเนตเวิร์กจำลองแบบ LNI เป็นการกำหนดค่าต้นทุนให้แก่ลิงก์ที่เชื่อมต่อระหว่างโนดใด ๆ ที่ไม่ใช่โนดเริ่มต้นด้วยค่าต้นทุนแลนเท่านั้น ส่วนลิงก์ที่เชื่อมโยงระหว่างโนดเริ่มต้นกับโนดใด ๆ จะถูกกำหนดด้วยค่าต้นทุนแวน โดยตัวอย่างการกระจายของค่าต้นทุนในเนตเวิร์กจำลองแบบ LNI กรณี n = 20 แสดงไว้ในรูปที่ 5.11



รูปที่ 5.11 ตัวอย่างการกระจายค่าต้นทุนของเนตเวิร์กจำลอง LNI

5.4.3 การกำหนดเนตเวิร์กจำลองแบบ WAN Neighboring (WNI)

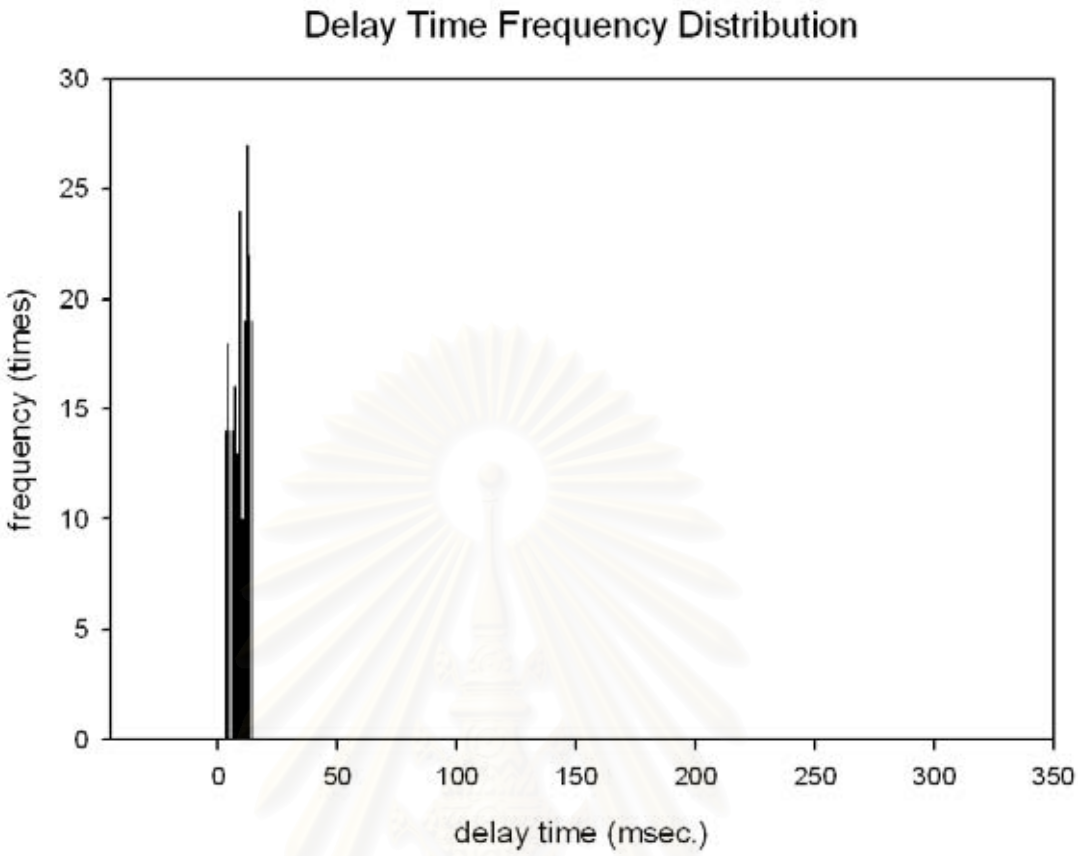
การกำหนดเนตเวิร์กจำลองแบบ WNI เป็นการกำหนดให้ค่าลิงก์ที่เชื่อมต่อระหว่างโนดใด ๆ ที่ไม่ใช่โนดเริ่มต้นด้วยค่าต้นทุนแวนเท่านั้น ส่วนลิงก์ที่เชื่อมโยงระหว่างโนดเริ่มต้นกับโนดใด ๆ จะถูกกำหนดด้วยค่าต้นทุนแลน ซึ่งตัวอย่างการกระจายของค่าต้นทุนในเนตเวิร์กจำลองแบบ WNI กรณี n = 20 แสดงไว้ในรูปที่ 5.12



รูปที่ 5.12 ตัวอย่างการกระจายค่าต้นทุนของเนตเวิร์กจำลอง WNI

5.4.4 การกำหนดเนตเวิร์กจำลองแบบ LAN Network (L-Net)

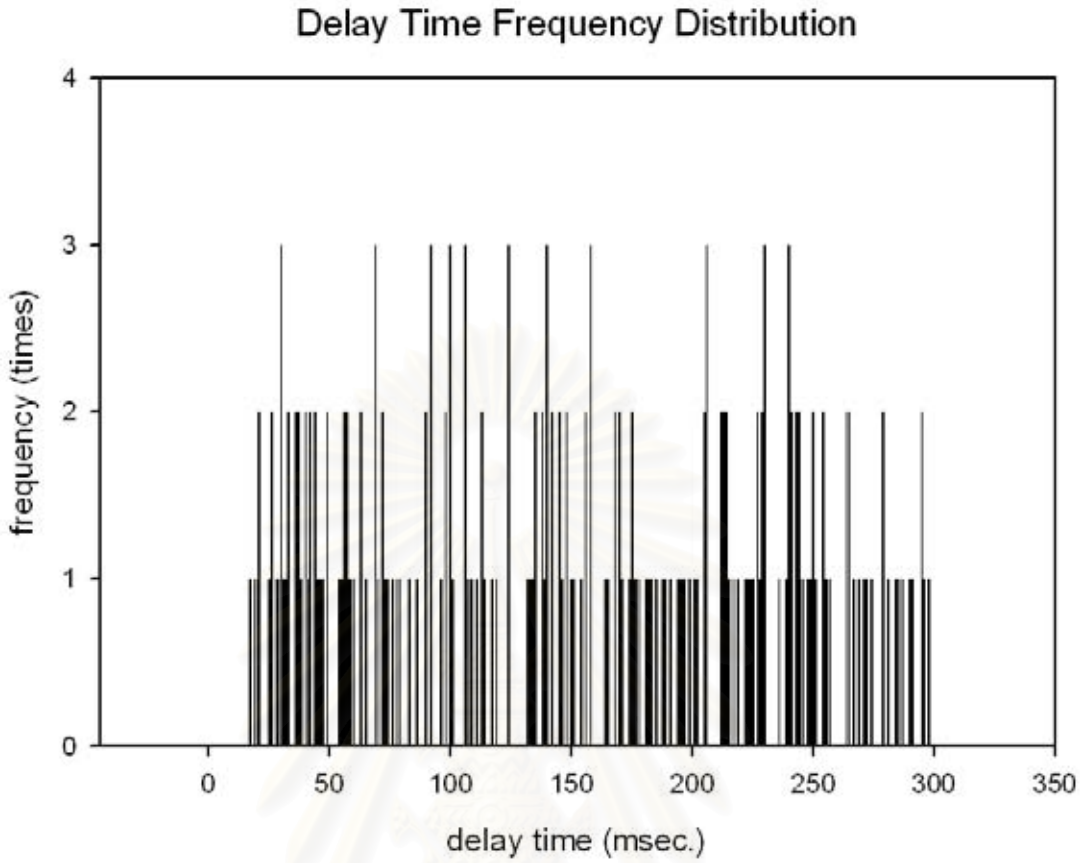
การกำหนดเนตเวิร์กจำลองแบบ L-Net เป็นการกำหนดให้ค่าลิงก์ที่เชื่อมต่อระหว่างโนดใด ๆ ที่รวมทั้งโนดเริ่มต้นด้วยค่าต้นทุนแลนเหมือนกันหมด ซึ่งตัวอย่างการกระจายของค่าต้นทุนในเนตเวิร์กจำลองแบบ WNI กรณี n = 20 แสดงไว้ในรูปที่ 5.13



รูปที่ 5.13 ตัวอย่างการกระจายค่าต้นทุนของเนตเวิร์กจำลอง L-Net

5.4.5 การกำหนดเนตเวิร์กจำลองแบบ WAN Network (W-Net)

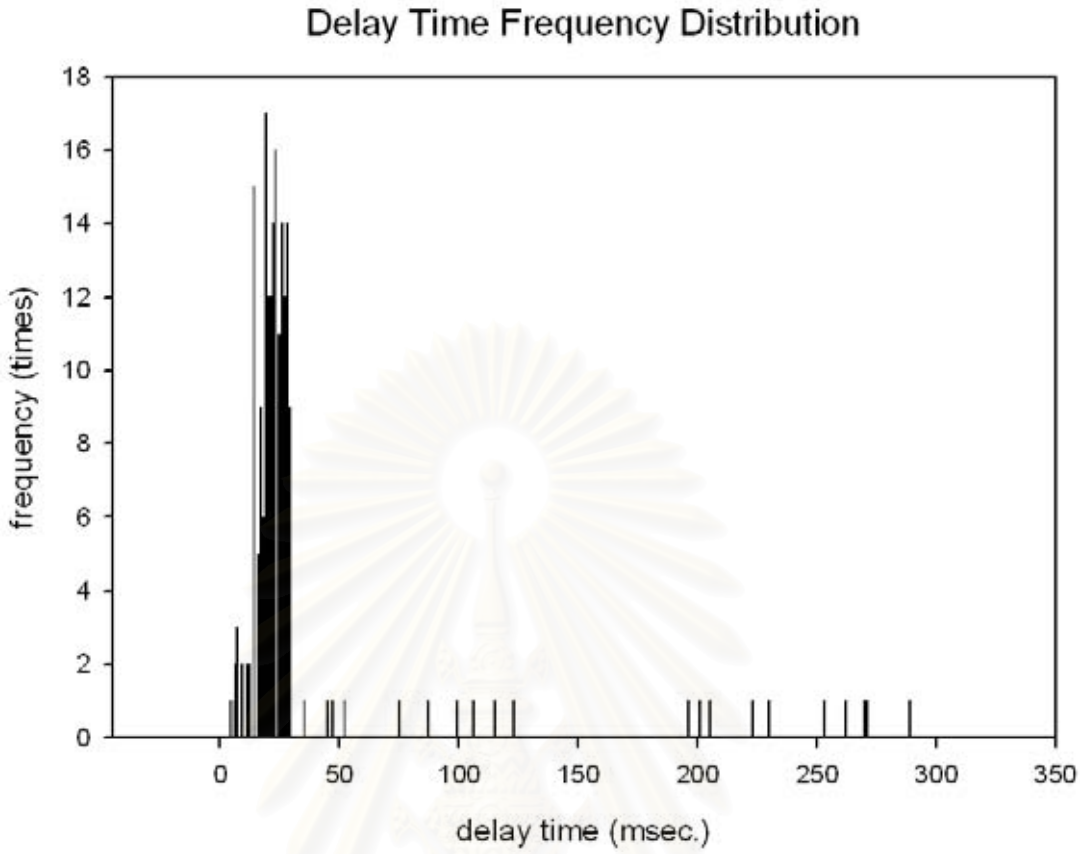
การกำหนดเนตเวิร์กจำลองแบบ W-Net เป็นการกำหนดให้ค่าลิงก์ที่เชื่อมต่อระหว่างโนดใด ๆ ที่รวมทั้งโนดเริ่มต้นด้วยค่าต้นทุนแวนเหมือนกันหมด ซึ่งตัวอย่างการกระจายของค่าต้นทุนในเนตเวิร์กจำลองแบบ WNI กรณี n = 20 แสดงไว้ในรูปที่ 5.14



รูปที่ 5.14 ตัวอย่างการกระจายค่าต้นทุนของเนตเวิร์กจำลอง W-Net

5.4.6 การกำหนดเนตเวิร์กจำลองแบบ Hierarchy LAN Neighboring (HLNI)

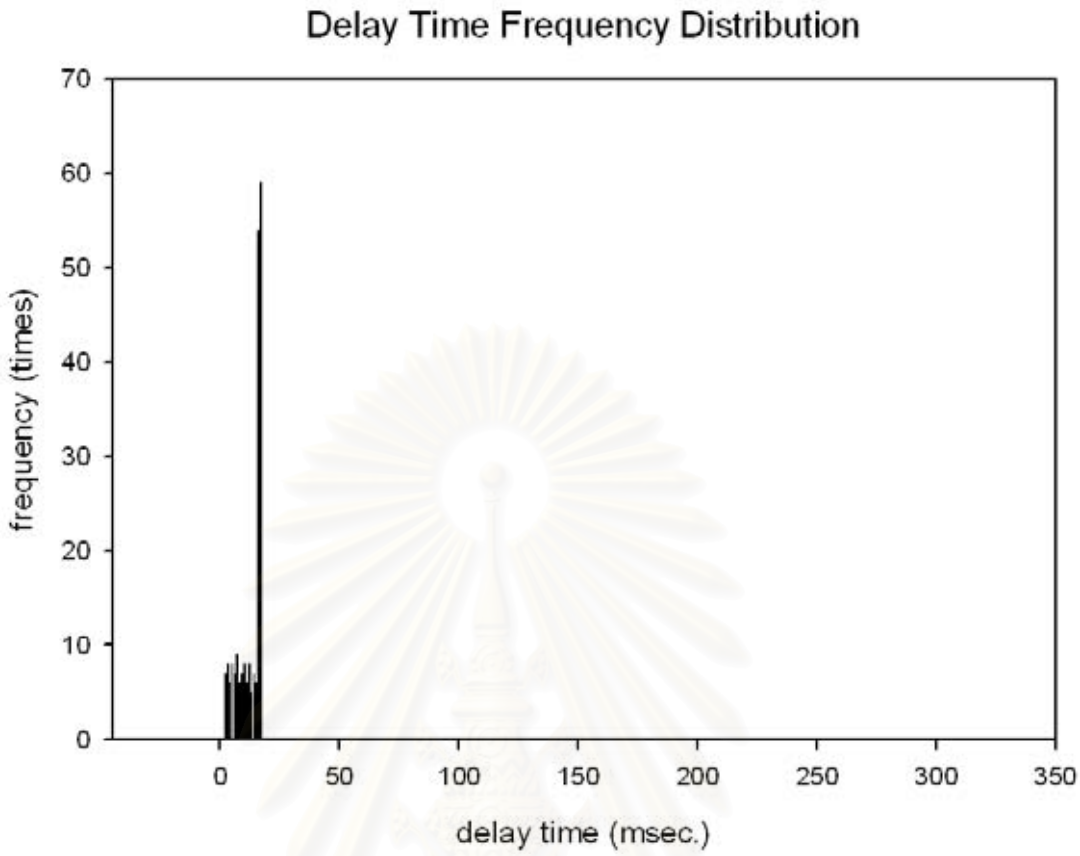
เนื่องจากการกำหนดแบบ LNI และ L-Net เป็นการกำหนดค่าต้นทุนให้แก่ลิงก์แบบอิสระ ซึ่งจะทำให้เกิดรูปแบบลักษณะเนตเวิร์กที่ประกอบไปด้วยโนดที่อยู่ติดกันหมด (อธิบายไว้ในหัวข้อการกำหนดเมตริกซ์ต้นทุนในบทที่ 4) และไม่เป็นจริงในทางปฏิบัติ ดังนั้นจึงกำหนดเนตเวิร์กเพิ่มขึ้นมาอีก 2 ชุดคือ Hierarchy LAN Neighboring (HLNI) และ Hierarchy LAN Network (HL-Net) โดยจะมีลักษณะการกำหนดเหมือนกับเมตริกซ์ต้นทุนโดยการกำหนดกลุ่มของโนดเสียก่อน แล้วจึงกำหนดค่าต้นทุนให้แก่ลิงก์ที่เชื่อมต่อแต่ละโนดในแต่ละกลุ่มด้วยค่าต้นทุนแลน ค่าต้นทุนลิงก์ที่เชื่อมต่อระหว่างโนดระหว่างกลุ่มด้วยค่าต่ำสุดของค่าต้นทุนแวน (16 มิลลิวินาที) และกำหนดค่าต้นทุนลิงก์ที่เชื่อมต่อระหว่างโนดใด ๆ กับโนดเริ่มต้นด้วยค่าต้นทุนแวน ซึ่งตัวอย่างการกระจายของค่าต้นทุนในเนตเวิร์กจำลองแบบ WNI กรณี n = 20 แสดงไว้ในรูปที่ 5.15



รูปที่ 5.15 ตัวอย่างการกระจายค่าต้นทุนของเนตเวิร์กจำลอง HLNI

5.4.7 การกำหนดเนตเวิร์กจำลองแบบ Hierarchy LAN Network (HL-Net)

การกำหนดเนตเวิร์กจำลองแบบ HL-Net เป็นการกำหนดกำหนดค่าต้นทุนให้แก่ลิงก์ที่เชื่อมต่อแต่ละโนดในแต่ละกลุ่มด้วยค่าต้นทุนแลน ค่าต้นทุนลิงก์ที่เชื่อมต่อระหว่างโนดระหว่างกลุ่มด้วยค่าต่ำสุดของค่าต้นทุนแวน (16 มิลลิวินาที) และกำหนดค่าต้นทุนลิงก์ที่เชื่อมต่อระหว่างโนดใด ๆ กับโนดเริ่มต้นด้วยค่าต้นทุนแลนซึ่งตัวอย่างการกระจายของค่าต้นทุนในเนตเวิร์กจำลองแบบ WNI กรณี n = 20 แสดงไว้ในรูปที่ 5.16



รูปที่ 5.16 ตัวอย่างการกระจายค่าต้นทุนของเนตเวิร์กจำลอง HL-Net

โดยผลลัพธ์ที่ได้จากการประมวลผลโดยใช้อัลกอริทึมต่าง ๆ จะอธิบายและวิเคราะห์ไว้ในบทที่ 6 ต่อไป

บทที่ 6

การวิเคราะห์อัลกอริทึมกำหนดเส้นทางให้กับโมบายล์เอเจนต์

6.1 กล่าวนำ

เนื้อหาในบทนี้จะกล่าวถึงการนำอัลกอริทึมต่าง ๆ ที่อธิบายไว้ในบทที่ 4 และบทที่ 5 มาทำการวิเคราะห์และเปรียบเทียบถึงประสิทธิภาพในการทำงาน โดยจะเน้นการวิเคราะห์ที่เวลาที่ใช้ในการประมวลผล และค่า MMSRC ที่ได้จากแต่ละอัลกอริทึม รวมไปถึงการวิเคราะห์ในเรื่อง Robustness โดยนำอัลกอริทึมเหล่านี้มาประมวลผลบนเนตเวิร์กจำลองที่มีลักษณะการกำหนดค่าต้นทุนให้แก่ลิงก์ที่แตกต่างกัน

อย่างไรก็ตามก่อนที่จะนำอัลกอริทึมต่าง ๆ ที่กล่าวไว้แล้วนั้นมาใช้งานได้ บางอัลกอริทึมจำเป็นที่จะต้องกำหนดค่าพารามิเตอร์ให้เหมาะสมกับงานที่ใช้เสียก่อน โดยอัลกอริทึมที่จำเป็นจะต้องกำหนดค่าพารามิเตอร์ก่อนใช้งานคือ อัลกอริทึม SA และอัลกอริทึม Tabu ดังนั้นเนื้อหาในบทนี้จะแบ่งออกเป็น 4 ส่วนคือ

ก.) การกำหนดค่าพารามิเตอร์ที่เหมาะสมก่อนการใช้งาน
ข.) การวิเคราะห์ผลลัพธ์ที่ได้จากอัลกอริทึมกำหนดเส้นทางต่าง ๆ
ค.) การวิเคราะห์สมรรถนะการทำงานเมื่อประมวลผลบนเนตเวิร์กจำลองที่มีการกำหนดค่าต้นทุนให้แก่ลิงก์ที่แตกต่างกัน
ง.) สรุปและวิเคราะห์ผล

6.2 การกำหนดค่าพารามิเตอร์ที่เหมาะสมก่อนการใช้งาน

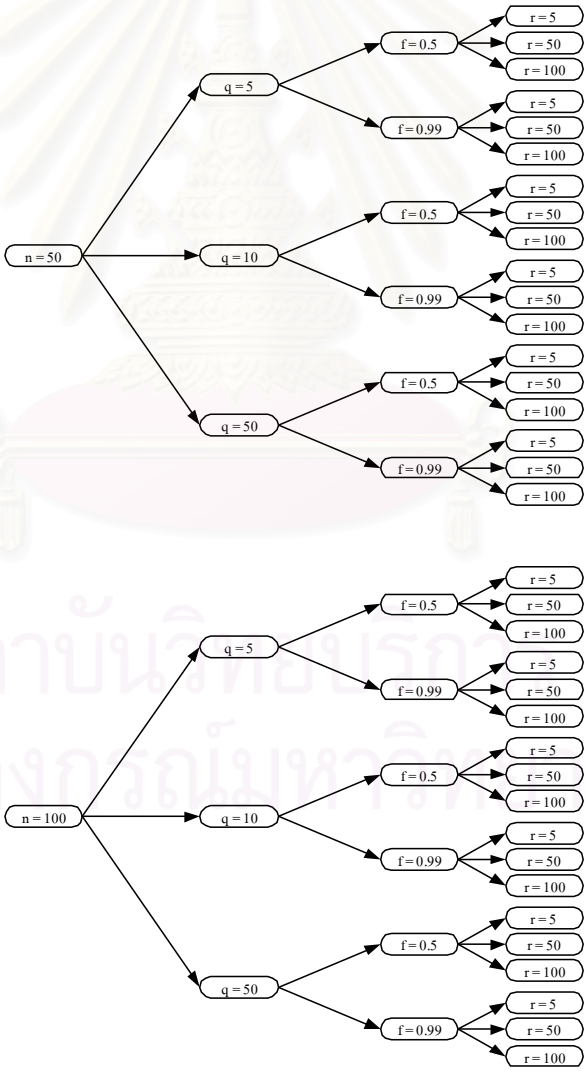
สำหรับอัลกอริทึมที่จำเป็นต้องมีการกำหนดค่าพารามิเตอร์ก่อนใช้งานจะมี 2 อัลกอริทึม คือ อัลกอริทึม SA และอัลกอริทึม Tabu ดังนี้

6.2.1 การกำหนดค่าพารามิเตอร์ของอัลกอริทึม SA

จากที่ได้กล่าวไว้แล้วในบทที่ 4 การนำอัลกอริทึม SA มาใช้งานนั้นจะมีค่าพารามิเตอร์ที่สำคัญ 3 ตัวคือ อุณหภูมิเริ่มต้น (q) ค่าการทอนอุณหภูมิ (f) และจำนวนรอบของการลดอุณหภูมิ (r) ดังนั้นก่อนการใช้งานจำเป็นจะต้องกำหนดค่าพารามิเตอร์ที่เหมาะสมกับงานที่จะใช้ก่อน

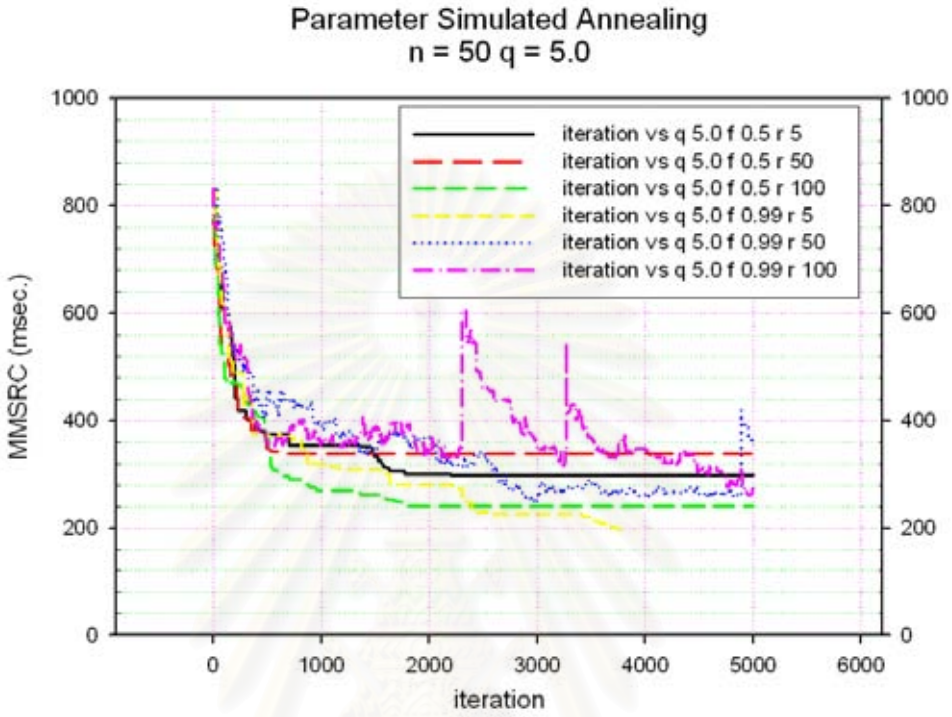
เนื่องจากอัลกอริทึม SA ถูกนำมาใช้โดยมีวัตถุประสงค์ 2 อย่างคือ อย่างแรกเป็นการนำมาใช้เพื่อเป็นเกณฑ์ในการเปรียบเทียบผลลัพธ์ที่ได้กับอัลกอริทึมฮิวริสติก โดยจะใช้เป็นอัลกอริทึม SA เพียงอย่างเดียวไม่ใช้ร่วมกับอัลกอริทึมชนิดอื่น อย่างที่สองเป็นการนำมาใช้ในส่วนของการปรับปรุงคำตอบร่วมกับอัลกอริทึมชนิดอื่นคือ อัลกอริทึม HRS-1, HRS-2, HRS-3, McGA และ RU ซึ่งส่งผลให้การทำงานในแต่ละวัตถุประสงค์มีความแตกต่างกัน ดังนั้นจึงจำเป็นจะต้องกำหนดค่าพารามิเตอร์ให้เหมาะสมตามวัตถุประสงค์ที่จะใช้งานดังต่อไปนี้

ก.) *การกำหนดค่าพารามิเตอร์ที่เหมาะสมของอัลกอริทึม SA* : ในการหาค่าพารามิเตอร์ที่เหมาะสม จะหาได้จากการทดลองประมวลผลโดยใช้ค่าพารามิเตอร์ต่าง ๆ จากนั้นตรวจสอบว่าค่าพารามิเตอร์ใดให้ค่า MMSRC และอัตราการลู่ที่ดีที่สุด โดยค่าพารามิเตอร์ที่ใช้ในอัลกอริทึม SA มีดังนี้ (รูปที่ 6.1)

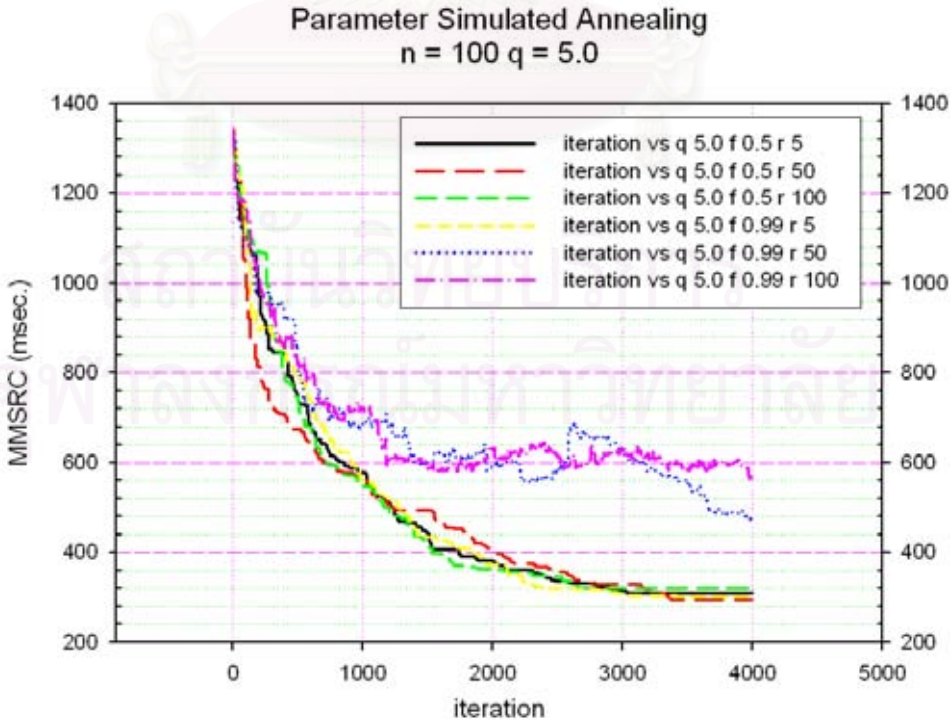


รูปที่ 6.1 ชุดค่าพารามิเตอร์ที่ใช้ประมวลผลหาค่าพารามิเตอร์ที่เหมาะสมในอัลกอริทึม SA

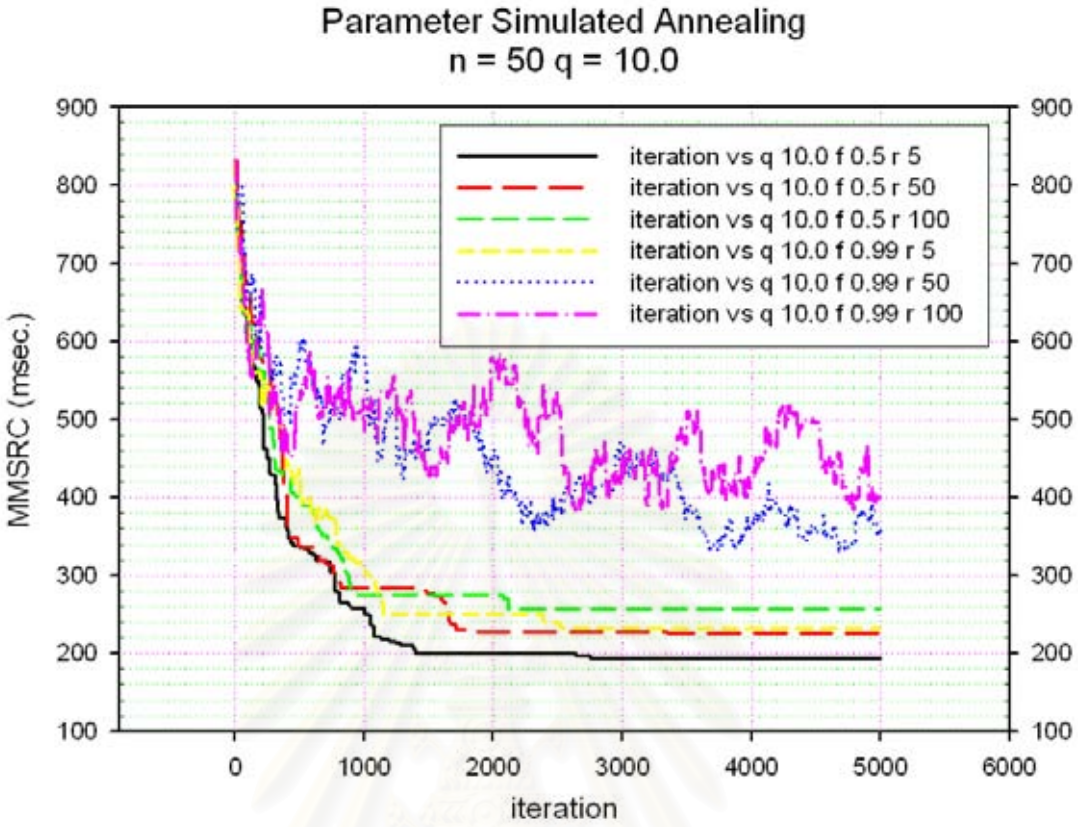
จากรูปที่ 6.1 จะใช้ค่าพารามิเตอร์ตั้งแต่ (n = 50 q = 5 f = 0.5 r = 5) จนถึง (n = 100 q = 50 f = 0.99 r = 100) เมื่อนำมาประมวลผลจะได้ผลลัพธ์ดังรูปที่ 6.2 – 6.7



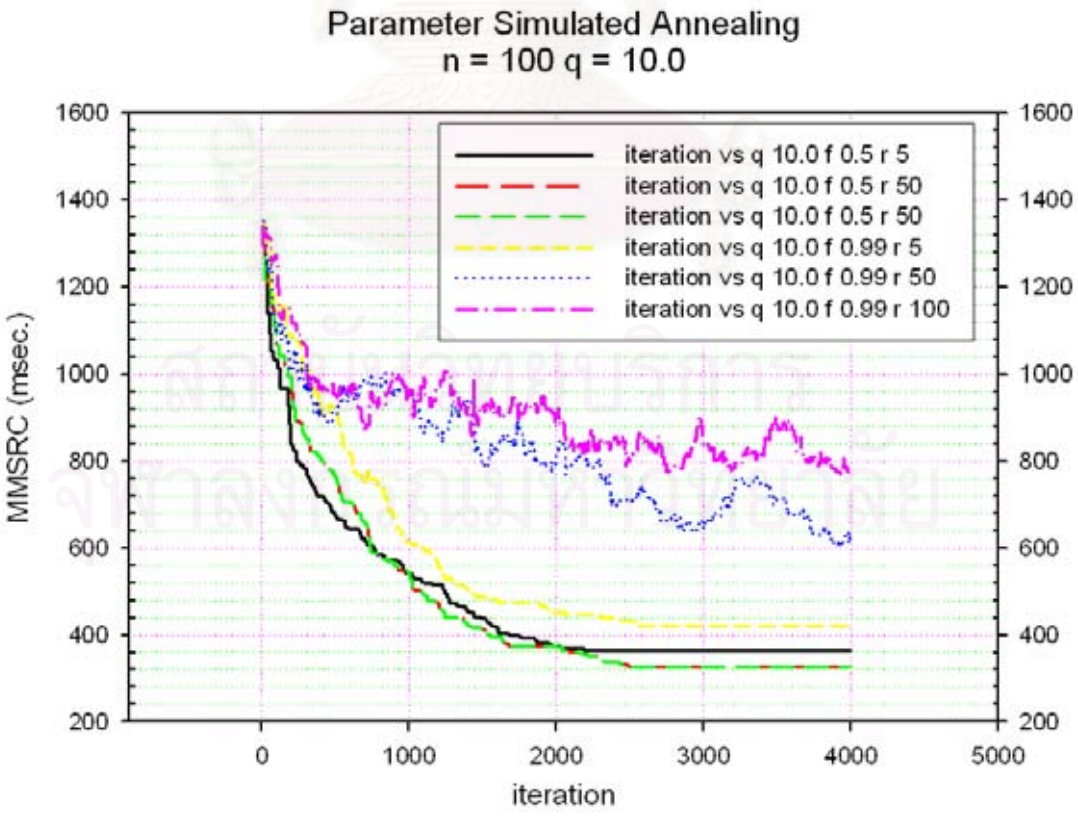
รูปที่ 6.2 การทดสอบหาค่าพารามิเตอร์ n = 50 q = 5.0



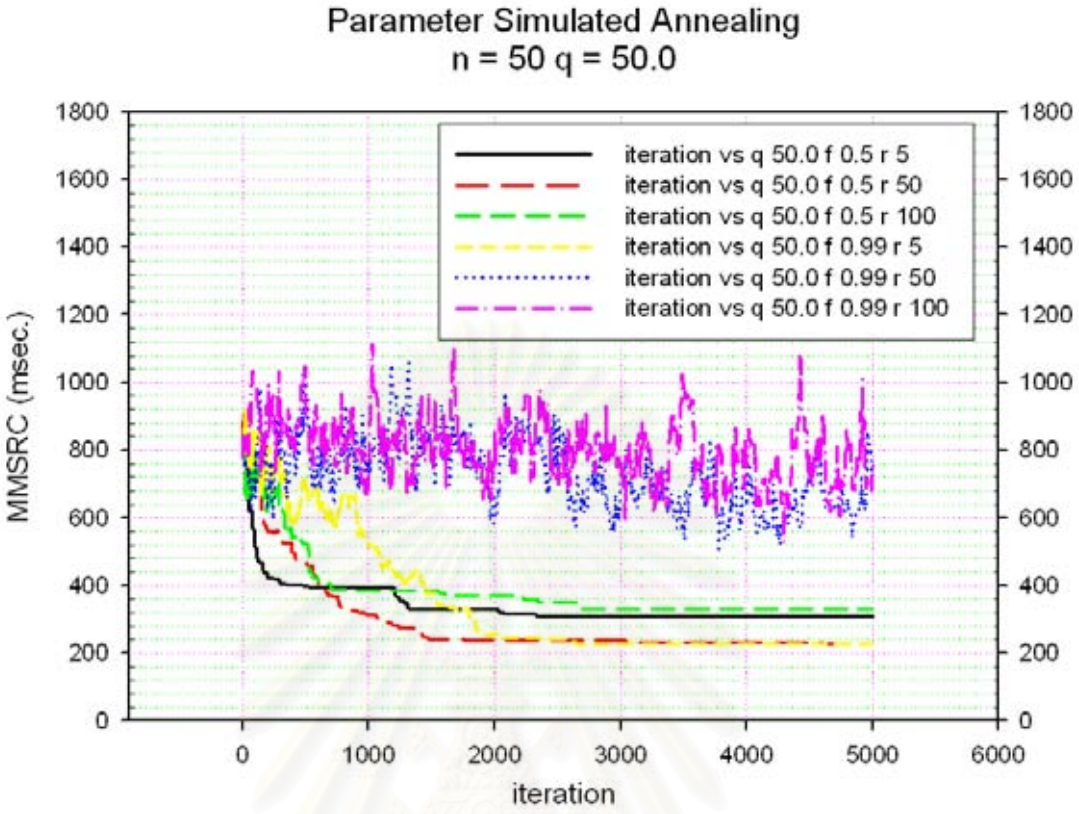
รูปที่ 6.3 การทดสอบหาค่าพารามิเตอร์ n = 100 q = 5.0



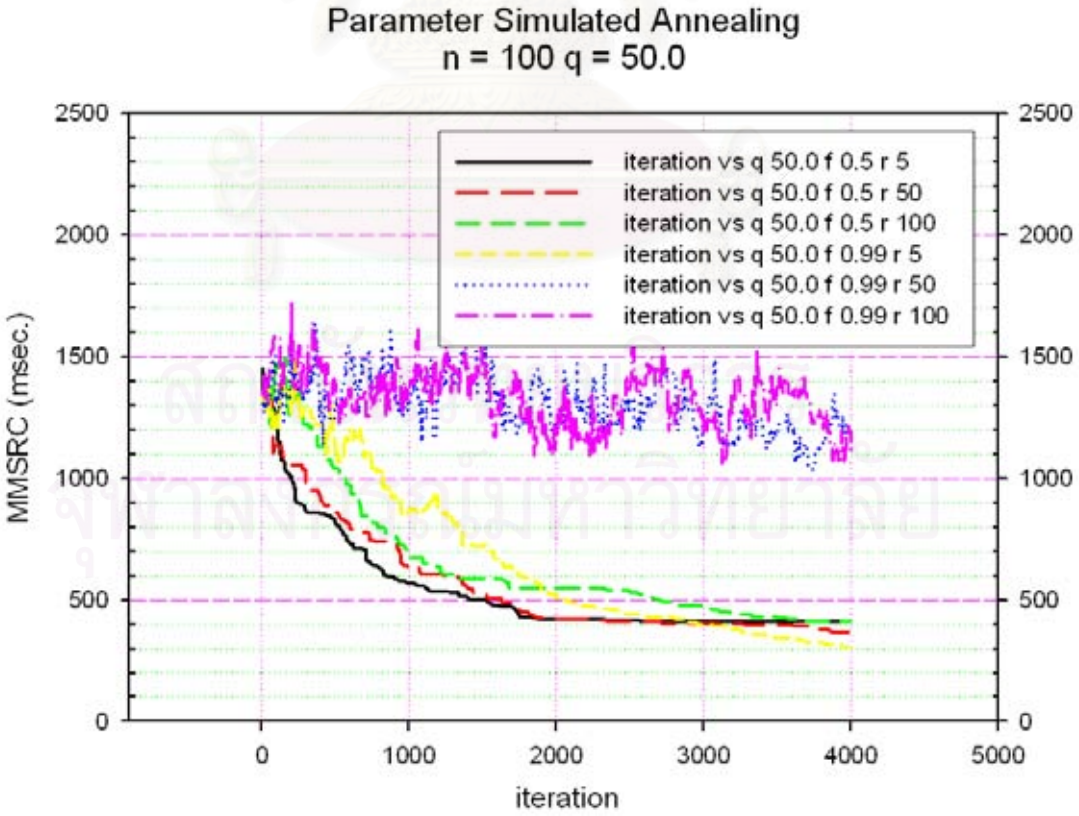
รูปที่ 6.4 การทดสอบหาค่าพารามิเตอร์ n = 50 q = 10.0



รูปที่ 6.5 การทดสอบหาค่าพารามิเตอร์ n = 100 q = 10.0



รูปที่ 6.6 การทดสอบหาค่าพารามิเตอร์ n = 50 q = 50.0

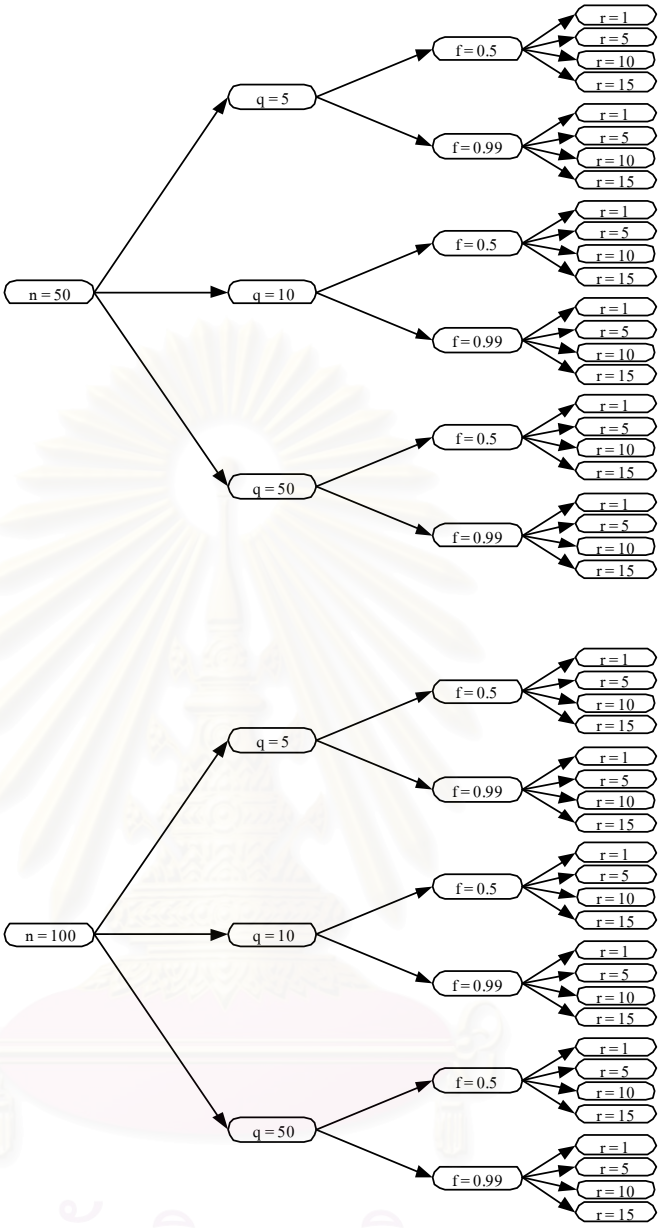


รูปที่ 6.7 การทดสอบหาค่าพารามิเตอร์ n = 100 q = 50.0

จากรูปที่ 6.2 – รูปที่ 6.7 จะเห็นได้ว่า ค่า MMSRC ที่ได้จากค่า (q = 50.0 f = 0.99 r = 50) และ (q = 50.0 f = 0.99 r = 100) จะให้แนวโน้มของค่า MMSRC ที่ไม่แน่นอนมีการแกว่งของคำตอบเกิดขึ้น (รูปที่ 6.6 และรูปที่ 6.7) ทั้งนี้เนื่องจากเมื่อใช้ค่า (q = 0.50 f = 0.99) จะส่งผลให้ความน่าจะเป็นการยอมรับคำตอบที่ด้อยลงยังคงมีค่าสูงเมื่อจำนวนรอบผ่านไป และเนื่องจากการเลือกคำตอบตัวถัดไปขึ้นกับคำตอบที่ได้รับการยอมรับก่อนหน้านั้น ดังนั้นจึงส่งผลให้ ค่าพารามิเตอร์ (q = 50 f = 0.99) ไม่สามารถให้คำตอบที่ดีได้เท่าที่ควร และยังทำให้เกิดการแกว่งของคำตอบที่ได้ในแต่ละรอบการประมวลผล

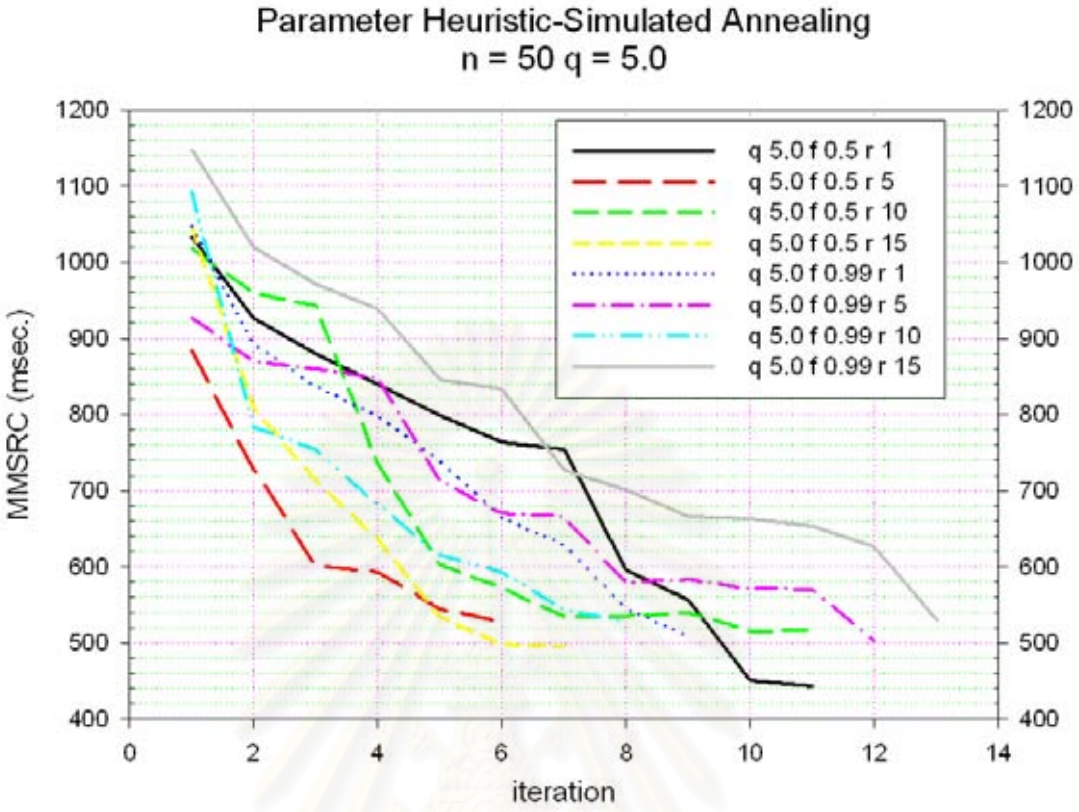
เมื่อพิจาณาที่ค่า n = 50 จากผลลัพธ์ในรูปที่ 6.2 รูปที่ 6.4 และรูปที่ 6.6 จะเห็นได้ว่าผลลัพธ์ที่ได้จากรูปที 6.2 จะให้ค่าผลลัพธ์ที่ดีที่สุด โดยจะให้ค่าผลลัพธ์ที่ดีที่สุดที่ค่าพารามิเตอร์ (q = 5.0 f = 0.99 r = 5) และเช่นเดียวกันเมื่อพิจารณาที่ค่า n = 100 (รูปที่ 6.3 รูปที่ 6.5 และรูปที่ 6.7) ผลลัพธ์ที่ได้จากรูปที่ 6.3 จะให้ค่าผลลัพธ์ที่ดีที่สุด โดยค่าพารามิเตอร์ที่ให้ค่า MMSRC ที่ต่ำที่สุดคือ (q = 5.0 f = 0.99 r = 5) ซึ่งเหมือนกับผลลัพธ์ที่ได้ที่ค่า n = 50 ทำให้วิเคราะห์ได้ว่าในการใช้งานของอัลกอริทึม SA ในลักษณะนี้จะมีความต้องการของอุณหภูมิค่าเริ่มต้นที่มีค่าต่ำ และมีการลดทอนอุณหภูมิที่ไม่มากนัก (f = 5) แต่มีการลดทอนอุณหภูมิบ่อย ๆ (r = 5) ดังนั้นค่าพารามิเตอร์ (q = 5.0 f = 0.99 r = 5) จะถูกใช้ในอัลกอริทึม SA

ข.) *การกำหนดค่าพารามิเตอร์ที่เหมาะสมของอัลกอริทึม SA กรณีใช้ร่วมกับอัลกอริทึมอื่น* :สำหรับอัลกอริทึม SA ที่ใช้ควบคู่กับอัลกอริทึม HR, McGA และ RU จะวิเคราะห์พารามิเตอร์ดังนี้

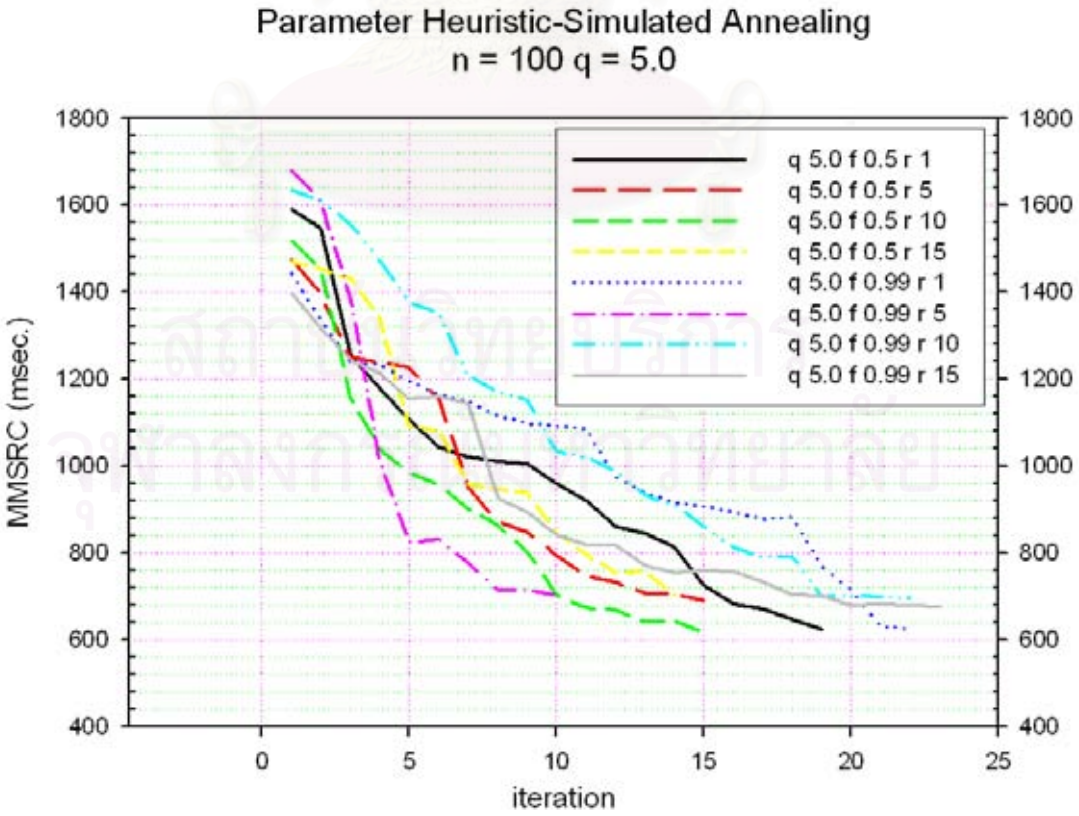


รูปที่ 6.8 ชุดค่าพารามิเตอร์ที่ใช้ประมวลผลหาค่าพารามิเตอร์ที่เหมาะสมในอัลกอริทึม SA (2)

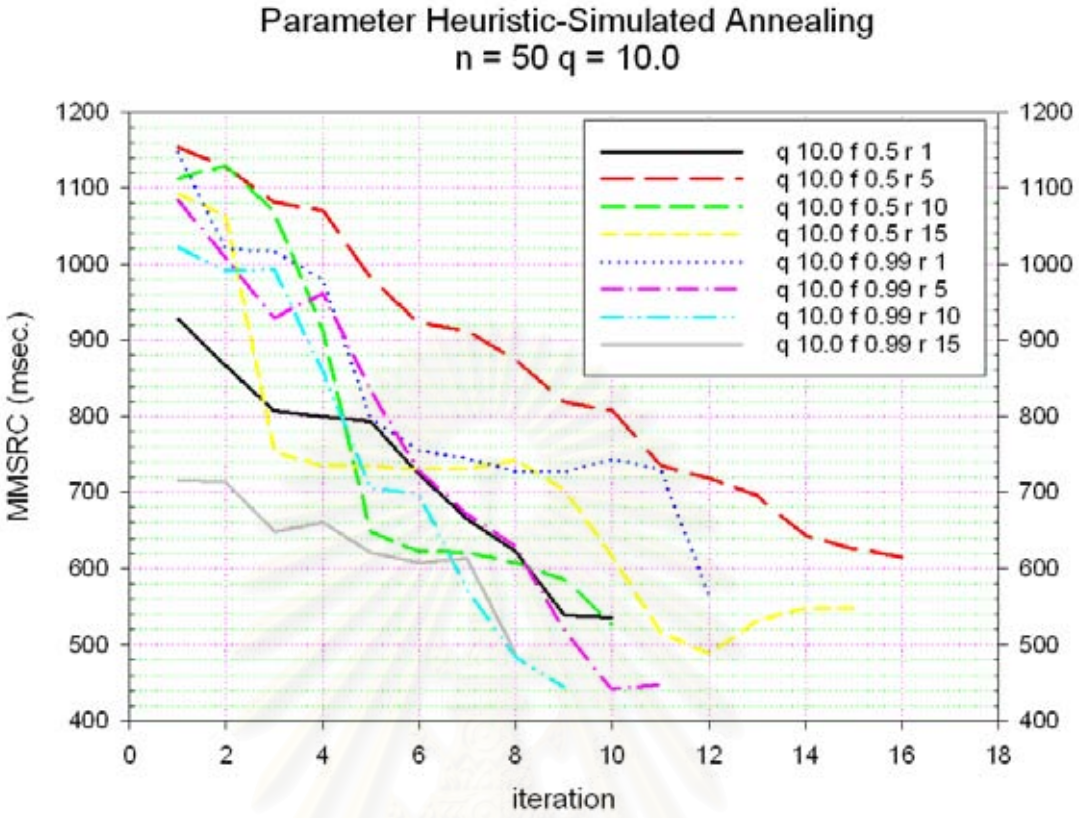
จากรูปที่ 6.8 จะใช้ค่าพารามิเตอร์ตั้งแต่ (n = 50 q = 5 f = 0.5 r = 1) จนถึง (n = 100 q = 50 f = 0.99 r = 15) เมื่อนำมาประมวลผลจะได้ผลลัพธ์ดังรูปที่ 6.9 – 6.14 ดังนี้



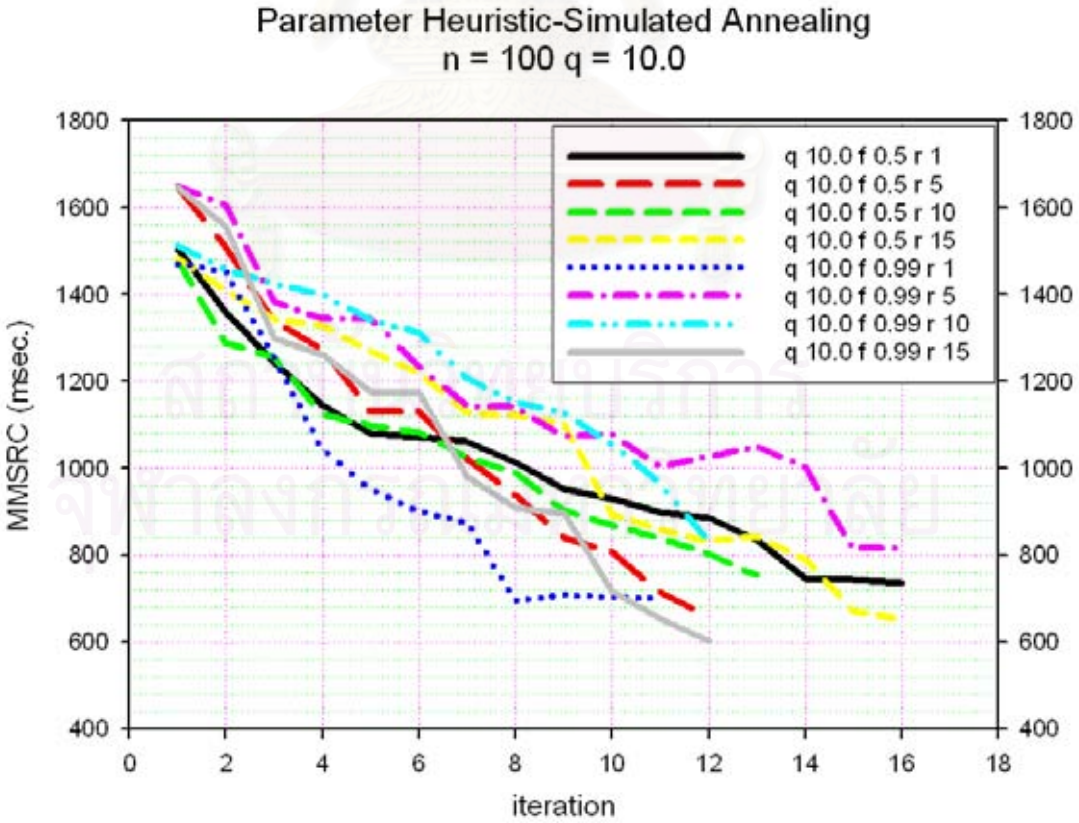
รูปที่ 6.9 การทดสอบหาค่าพารามิเตอร์ n = 50 q = 5.0



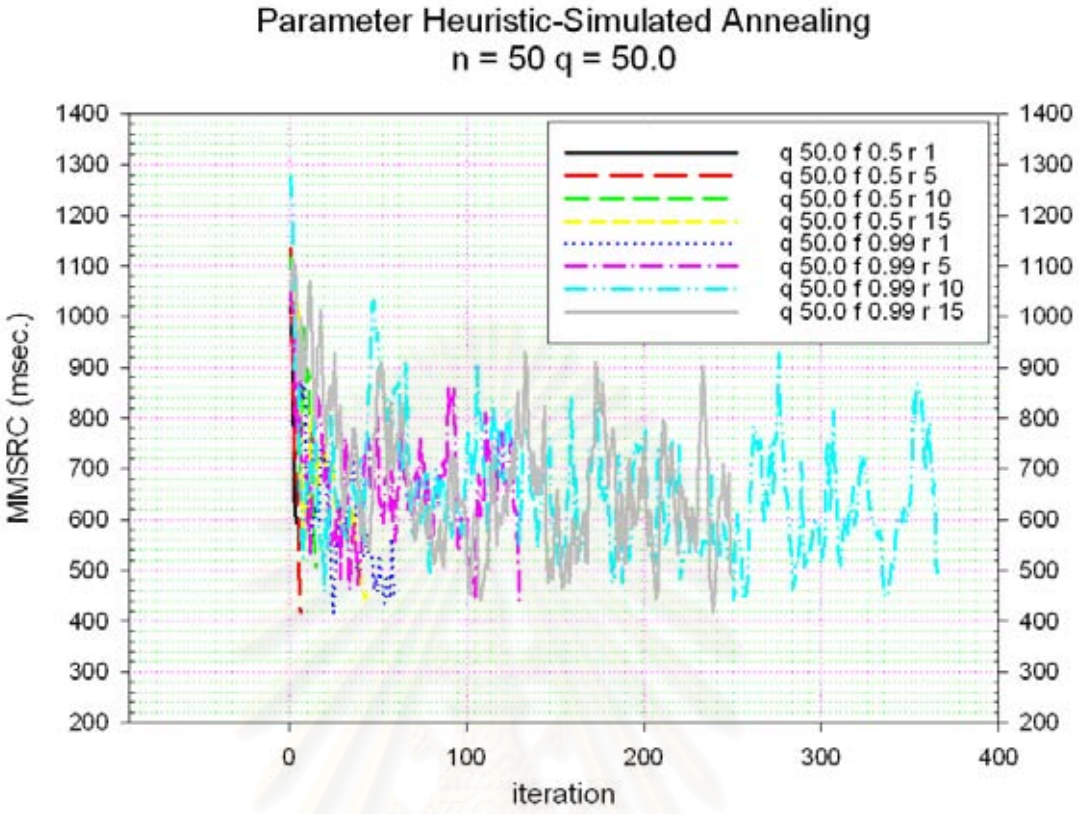
รูปที่ 6.10 การทดสอบหาค่าพารามิเตอร์ n = 100 q = 5.0



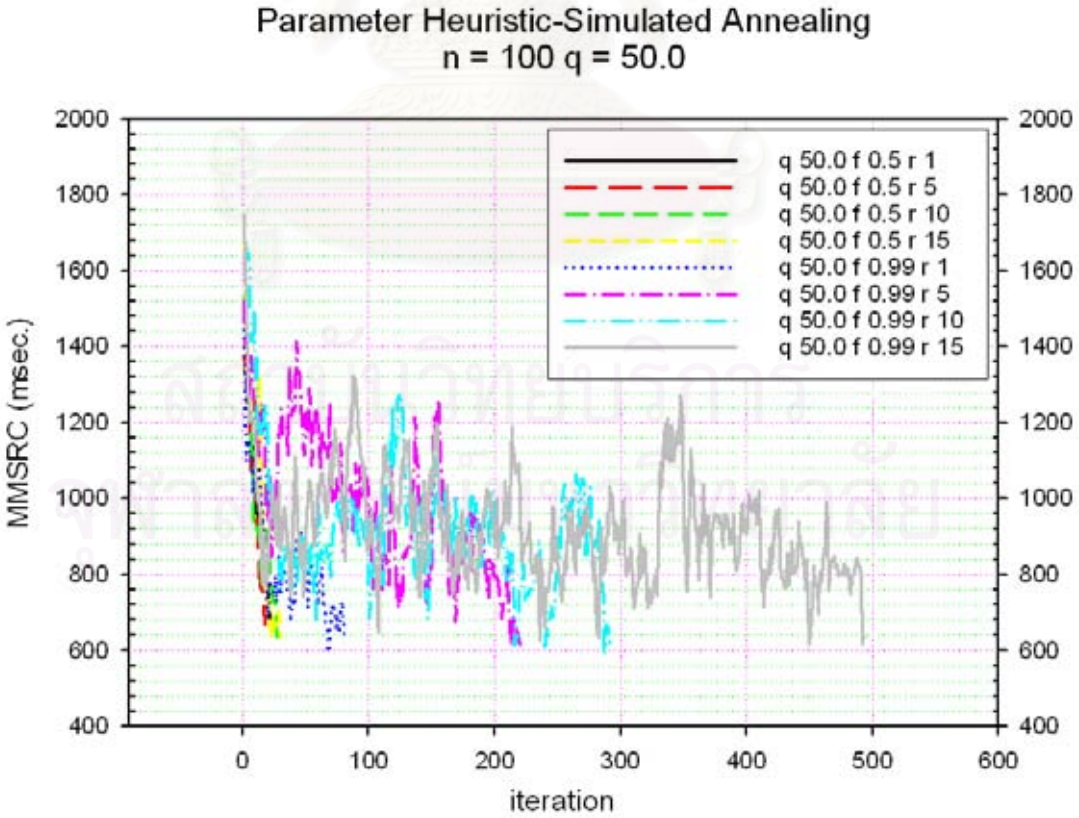
รูปที่ 6.11 การทดสอบหาค่าพารามิเตอร์ n = 50 q = 10.0



รูปที่ 6.12 การทดสอบหาค่าพารามิเตอร์ n = 100 q = 10.0



รูปที่ 6.13 การทดสอบหาค่าพารามิเตอร์ n = 50 q = 50.0

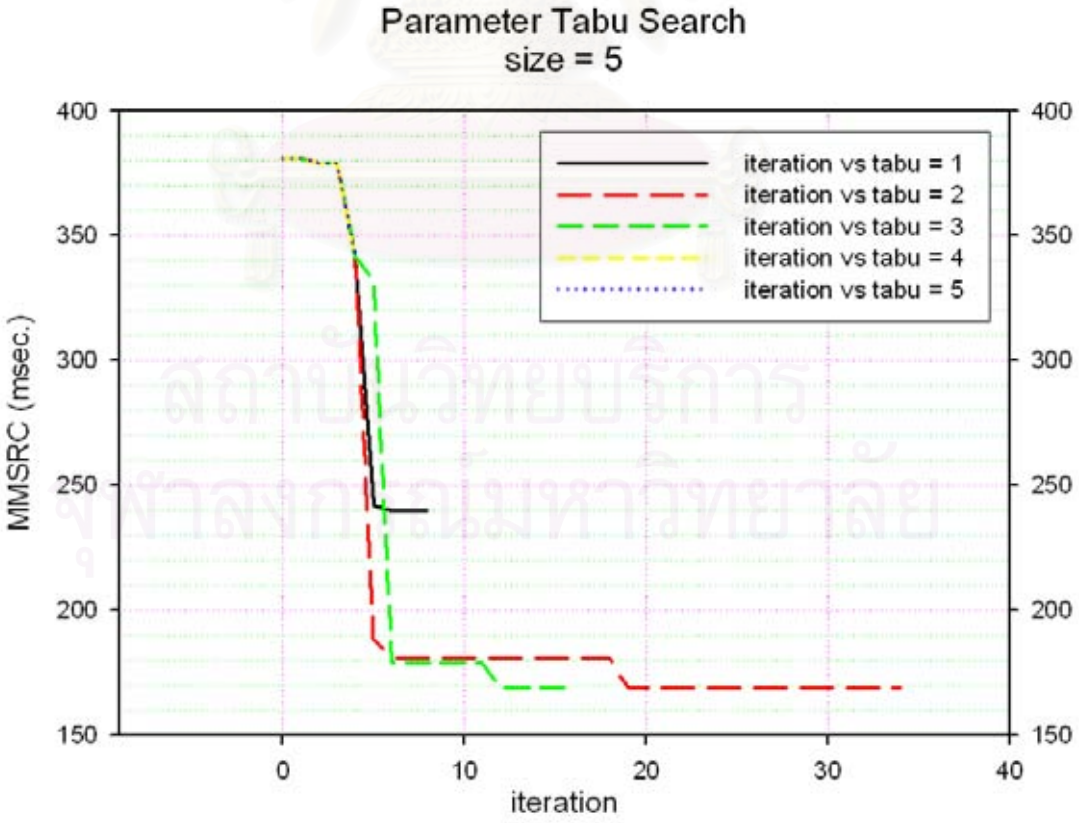


รูปที่ 6.14 การทดสอบหาค่าพารามิเตอร์ n = 100 q = 50.0

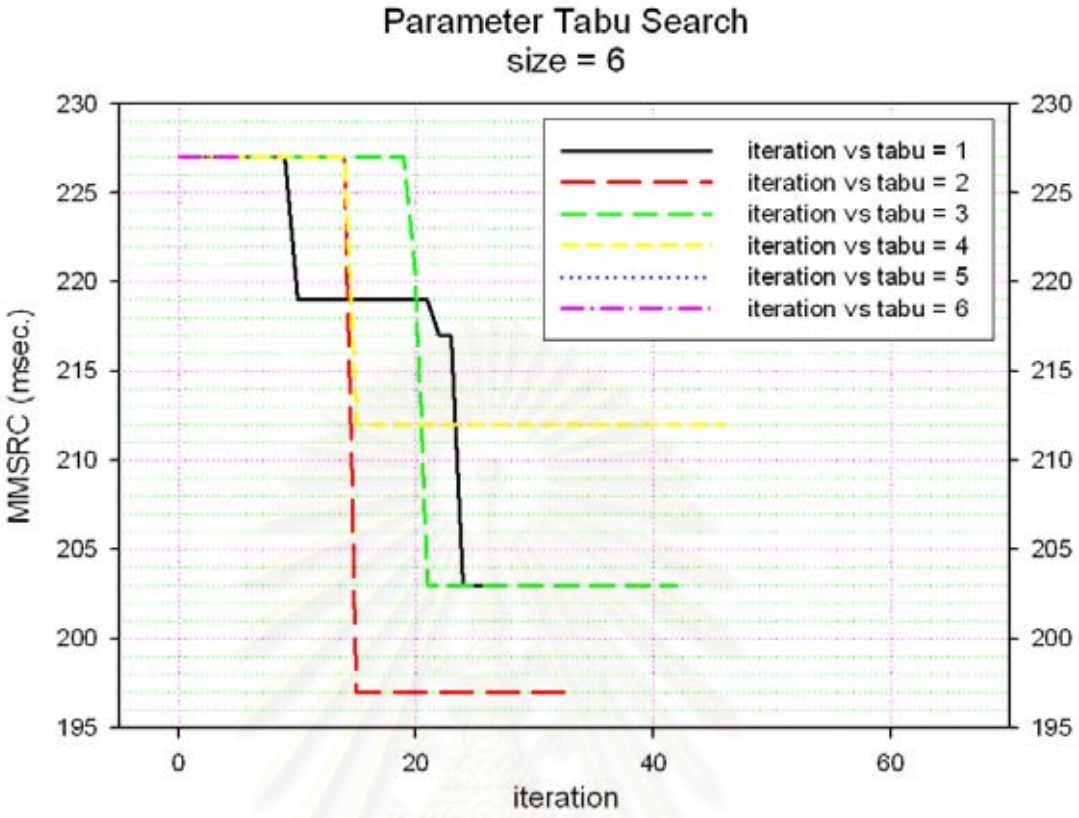
จากรูปที่ 6.13 และ 6.14 จะเห็นได้ว่าค่าพารามิเตอร์ q = 50 จะทำให้เกิดการแกว่งของค่า MMSRC เมื่อจำนวนรอบมากขึ้น ทำให้ไม่สามารถทำนายลักษณะการลู่ได้ ทั้งนี้เนื่องจากการใช้ค่าอุณหภูมิเริ่มต้นที่มีค่าสูงจะส่งผลให้ความน่าจะเป็นการยอมรับคำตอบที่ด้อยลงมีค่าสูงตามไปด้วย ดังนั้นแม้ว่าจะมีการทอนอุณหภูมิแต่ยังคงส่งผลให้ค่าความน่าจะเป็นยอมรับคำตอบที่ด้อยกว่ายังคงมีค่าสูง ซึ่งเหมือนกับผลลัพธ์ที่ได้จากการกำหนดค่าพารามิเตอร์ในรูปที่ 6.6 และรูปที่ 6.7 สำหรับรูปที่ 6.9 – 6.12 จะเห็นได้ว่า ค่าพารามิเตอร์ที่ให้ทิศทางการลู่ของ MMSRC ที่ดีที่สุดและเหมาะสมที่สุดคือค่า (q = 5.0 f = 0.50 r = 1) ทำให้วิเคราะห์ได้ว่าในการใช้งานในลักษณะนี้อัลกอริทึม SA ต้องการค่าอุณหภูมิเริ่มต้นที่ต่ำ มีการลดทอนอุณหภูมิสูงและบ่อย ดังนั้นค่าพารามิเตอร์นี้จะถูกใช้ในอัลกอริทึมฮิวริสติกที่ใช้อัลกอริทึม SA ในการปรับปรุงคำตอบต่อไป

6.2.2 การกำหนดค่าพารามิเตอร์ของอัลกอริทึม Tabu

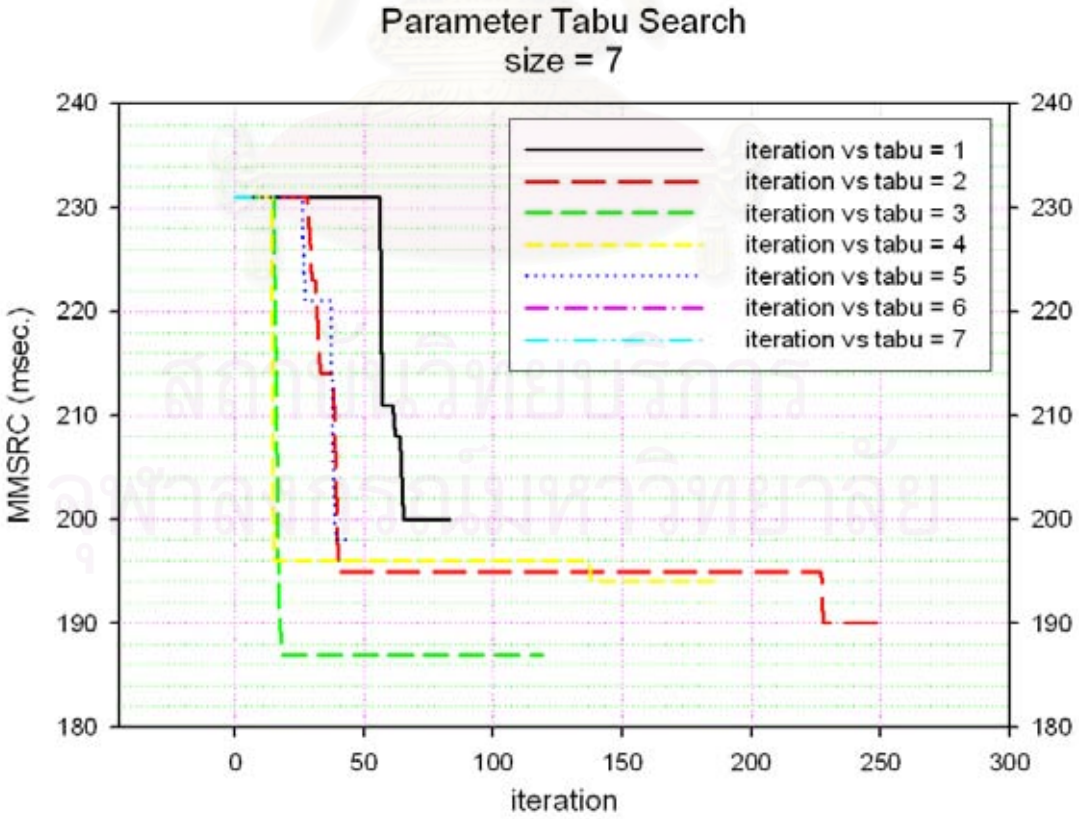
จากเนื้อหาในบทที่ 4 ค่าพารามิเตอร์ที่สำคัญในอัลกอริทึม Tabu Search คือจำนวนรอบที่ถูก Tabu ไว้ (t) สำหรับคำตอบที่ถูกปรับปรุงโดยใช้อัลกอริทึม Tabu Search นั้นจะมีขนาดตั้งแต่ 1 – 13 ซึ่งผลลัพธ์ที่ได้แสดงไว้ดังรูปที่ 6.15 – 6.23



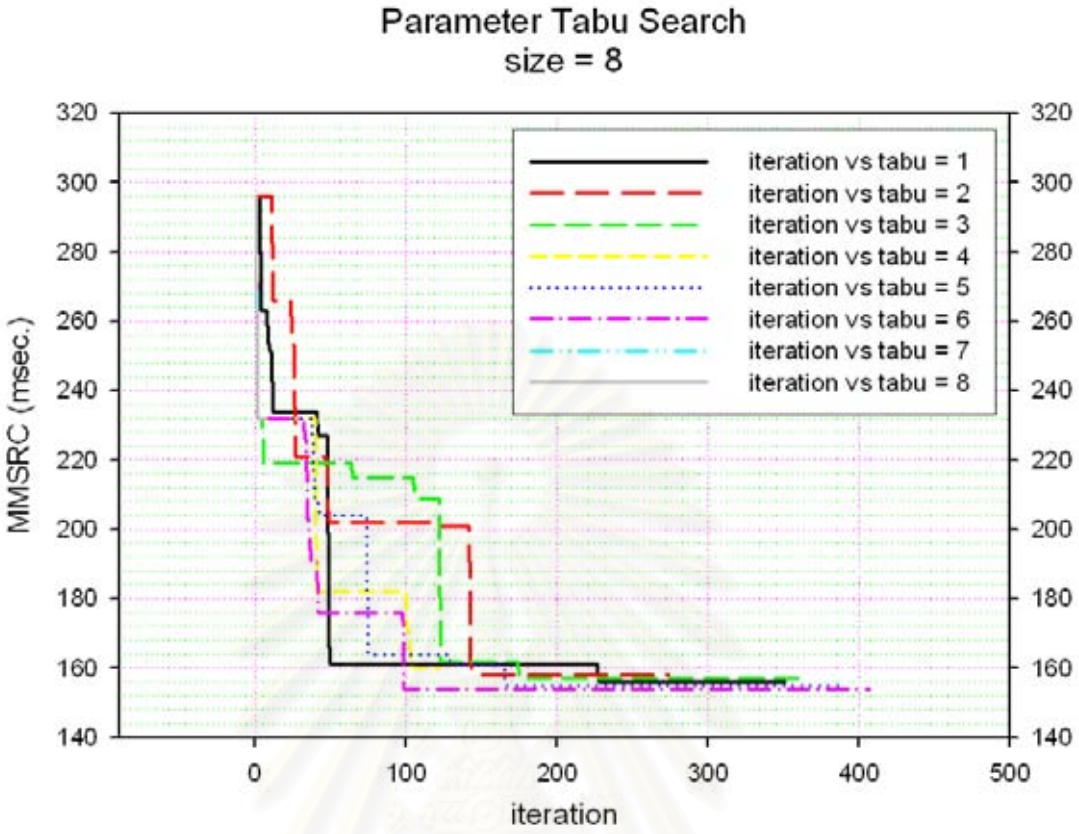
รูปที่ 6.15 การทดสอบหาค่าพารามิเตอร์ size = 5



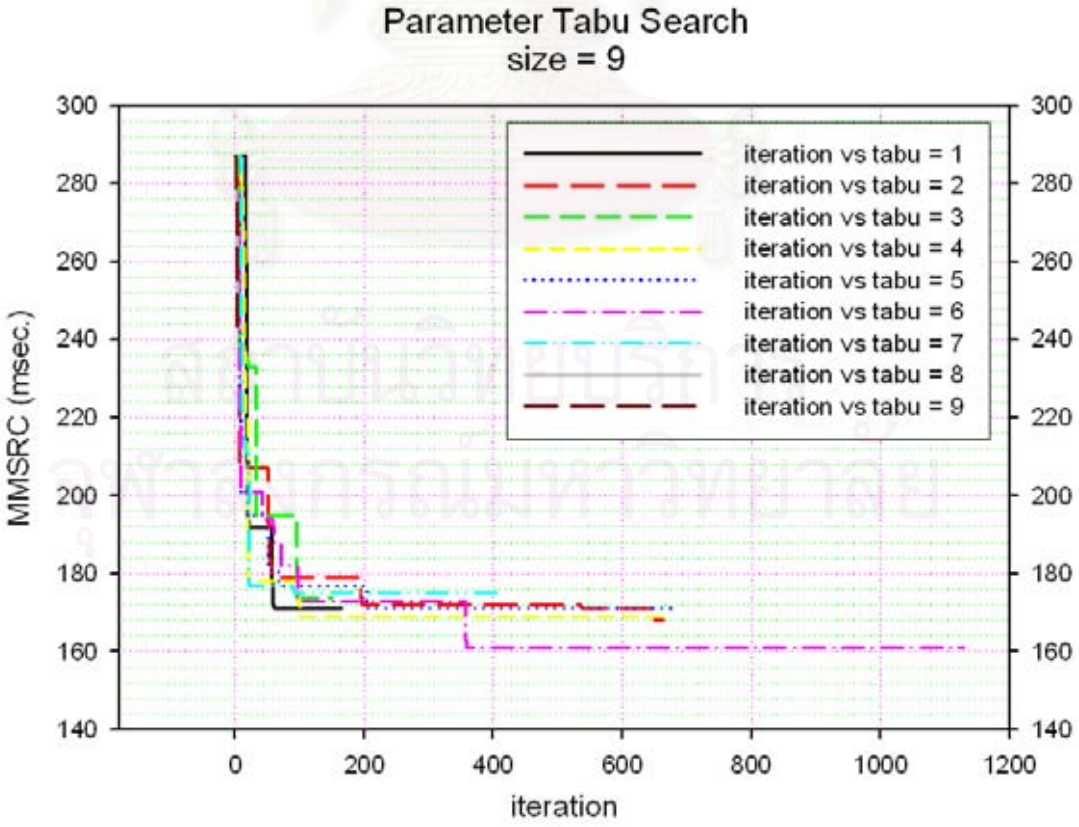
รูปที่ 6.16 การทดสอบหาค่าพารามิเตอร์ size = 6



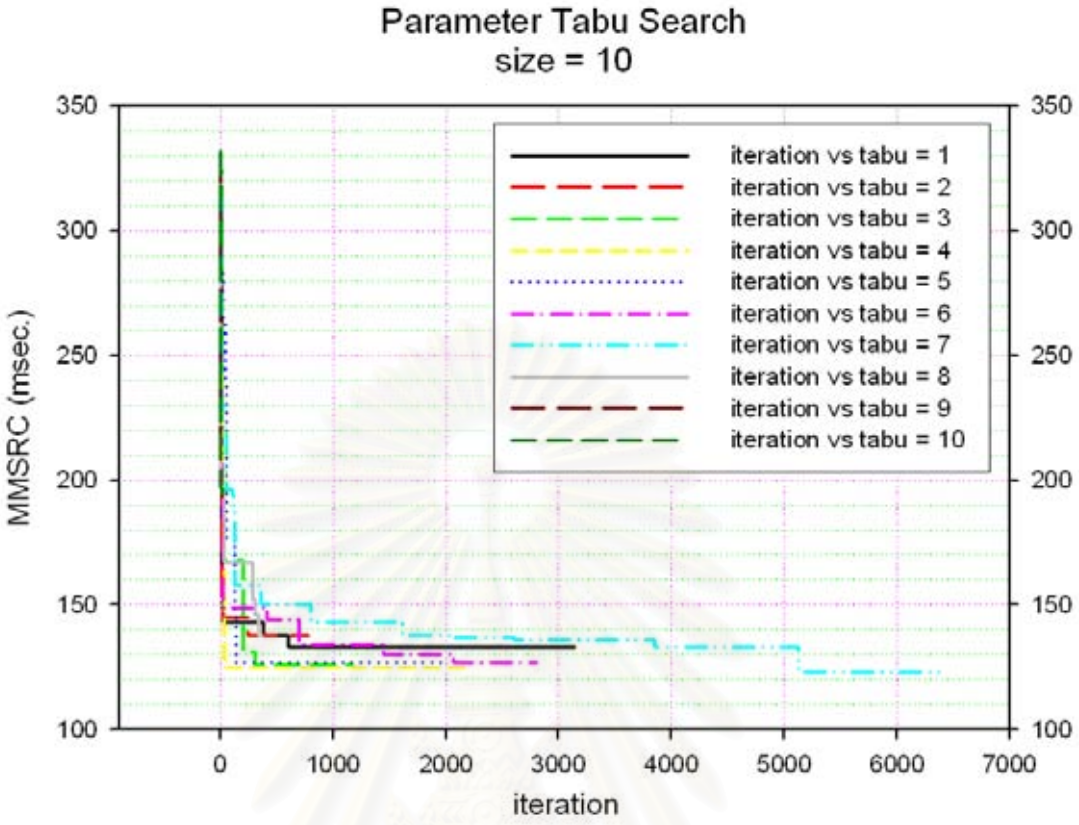
รูปที่ 6.17 การทดสอบหาค่าพารามิเตอร์ size = 7



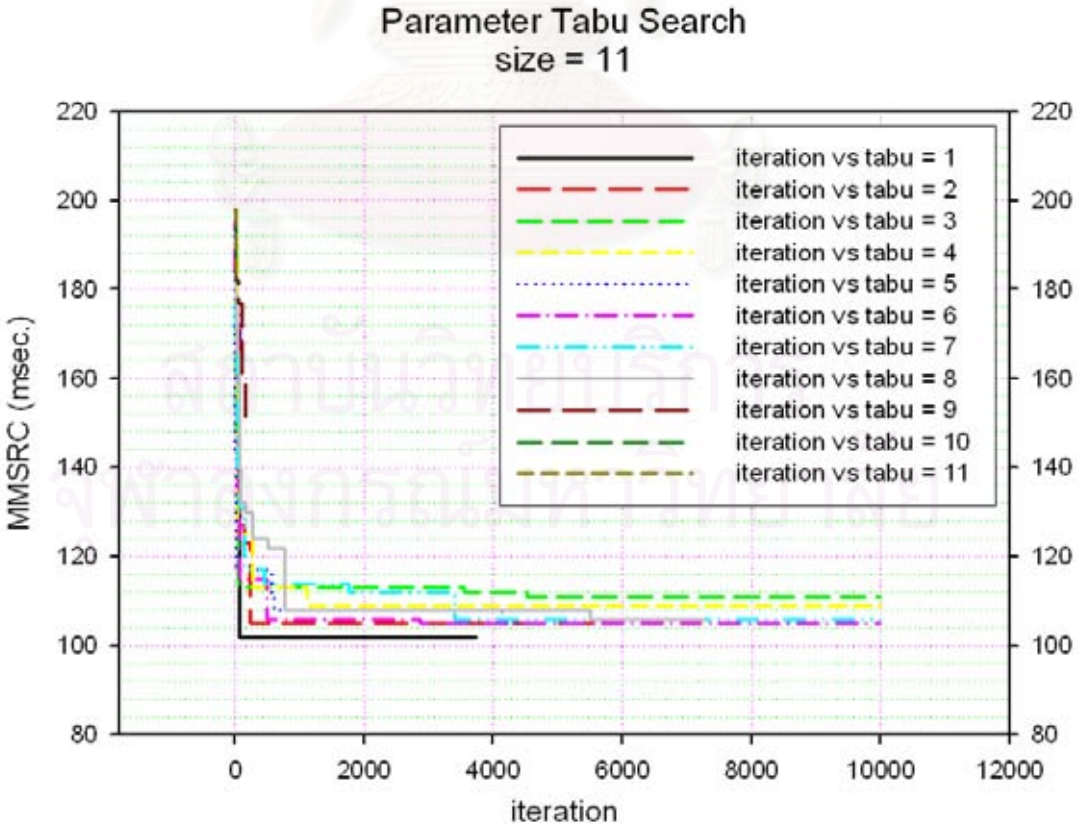
รูปที่ 6.18 การทดสอบหาค่าพารามิเตอร์ size = 8



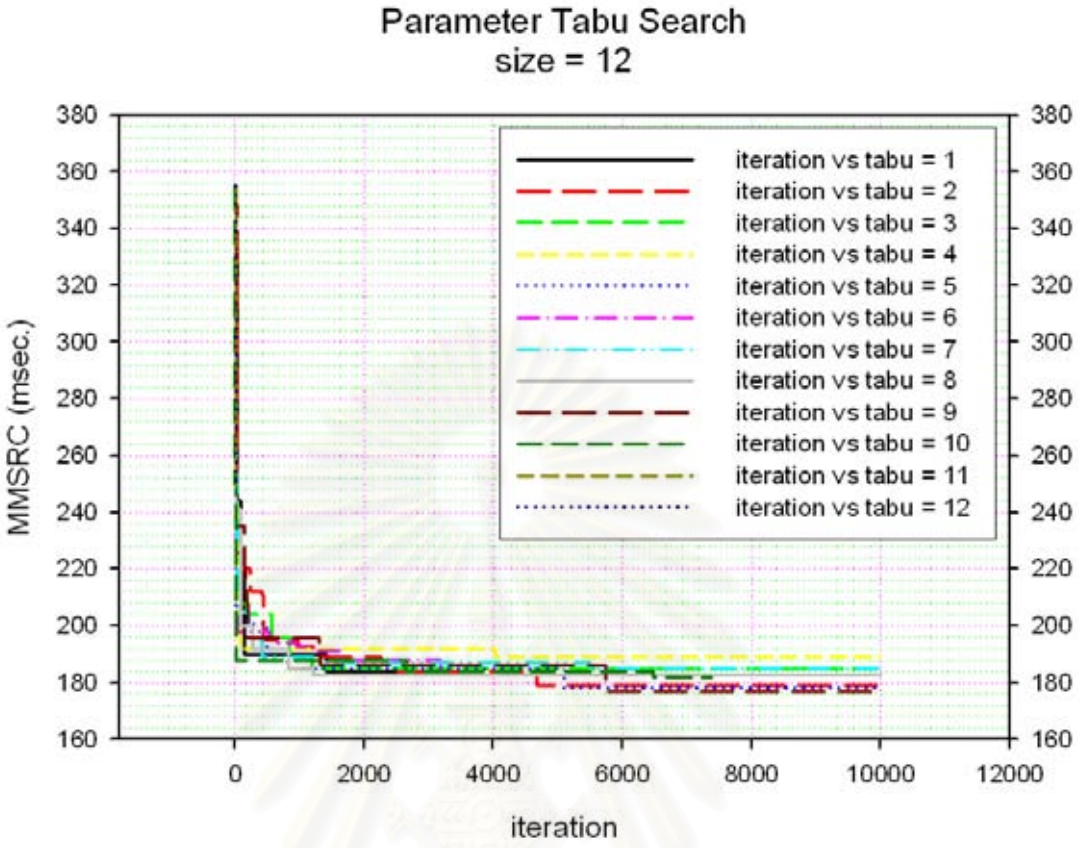
รูปที่ 6.19 การทดสอบหาค่าพารามิเตอร์ size = 9



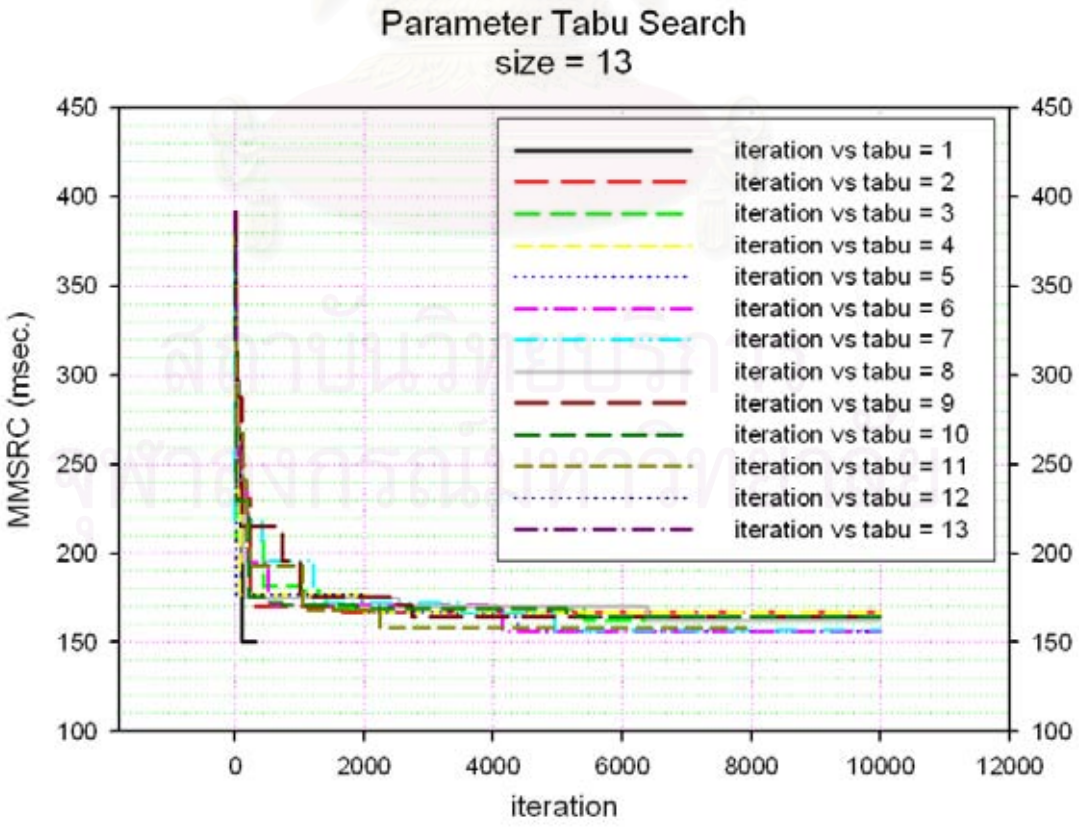
รูปที่ 6.20 การทดสอบหาค่าพารามิเตอร์ size = 10



รูปที่ 6.21 การทดสอบหาค่าพารามิเตอร์ size = 11



รูปที่ 6.22 การทดสอบหาค่าพารามิเตอร์ size = 12



รูปที่ 6.23 การทดสอบหาค่าพารามิเตอร์ size = 13

จากรูปที่ 6.15 – 6.23 จะเห็นได้ว่า ค่าพารามิเตอร์ *t* ที่เหมาะสมในแต่ละค่าขนาดของคำตอบมีไม่เท่ากัน หมายความว่าค่าพารามิเตอร์ *t* ขึ้นอยู่กับขนาดของคำตอบ โดยผลลัพธ์ที่ได้สรุปได้ดังนี้

ขนาดของคำตอบ 5	ค่าพารามิเตอร์ t ที่เหมาะสม 2
ขนาดของคำตอบ 6	ค่าพารามิเตอร์ t ที่เหมาะสม 2
ขนาดของคำตอบ 7	ค่าพารามิเตอร์ t ที่เหมาะสม 3
ขนาดของคำตอบ 8	ค่าพารามิเตอร์ t ที่เหมาะสม 5
ขนาดของคำตอบ 9	ค่าพารามิเตอร์ t ที่เหมาะสม 6
ขนาดของคำตอบ 10	ค่าพารามิเตอร์ t ที่เหมาะสม 6
ขนาดของคำตอบ 11	ค่าพารามิเตอร์ t ที่เหมาะสม 6
ขนาดของคำตอบ 12	ค่าพารามิเตอร์ t ที่เหมาะสม 6
ขนาดของคำตอบ 13	ค่าพารามิเตอร์ t ที่เหมาะสม 6

ค่าพารามิเตอร์นี้จะถูกใช้ในอัลกอริทึมฮิวริสติกที่ใช้อัลกอริทึม Tabu ในการปรับปรุงคำตอบต่อไป

6.3 การวิเคราะห์ผลลัพธ์ที่ได้จากอัลกอริทึมกำหนดเส้นทางต่าง ๆ

เมื่อมีการวิเคราะห์หาค่าพารามิเตอร์ที่เหมาะสมกับการใช้งานแล้ว ขั้นต่อไปคือการนำค่าพารามิเตอร์ที่เหมาะสมเหล่านั้นไปใช้ในการประมวลผลจริง โดยจะใช้อัลกอริทึมต่าง ๆ ดังนี้

ก.) อัลกอริทึม Brute Force Search (BF)

ข.) อัลกอริทึม Simulated Annealing (SA)

ค.) อัลกอริทึม ฮิวริสติกแบบที่ 1 โดยใช้อัลกอริทึม Simulated Annealing ในการปรับปรุงคำตอบ (HRS-1)

ง.) อัลกอริทึม ฮิวริสติกแบบที่ 2 โดยใช้อัลกอริทึม Simulated Annealing ในการปรับปรุงคำตอบ (HRS-2)

จ.) อัลกอริทึม ฮิวริสติกแบบที่ 3 โดยใช้อัลกอริทึม Simulated Annealing ในการปรับปรุงคำตอบ (HRS-3)

ฉ.) อัลกอริทึม ฮิวริสติกแบบที่ 1 โดยใช้อัลกอริทึม Tabu Search ในการปรับปรุงคำตอบ (HRT-1)

ช.) อัลกอริทึม ฮิวริสติกแบบที่ 2 โดยใช้อัลกอริทึม Tabu Search ในการปรับปรุงคำตอบ (HRT-2)

ซ.) อัลกอริทึม ฮิวริสติกแบบที่ 3 โดยใช้อัลกอริทึม Tabu Search ในการปรับปรุงคำตอบ (HRT-3)

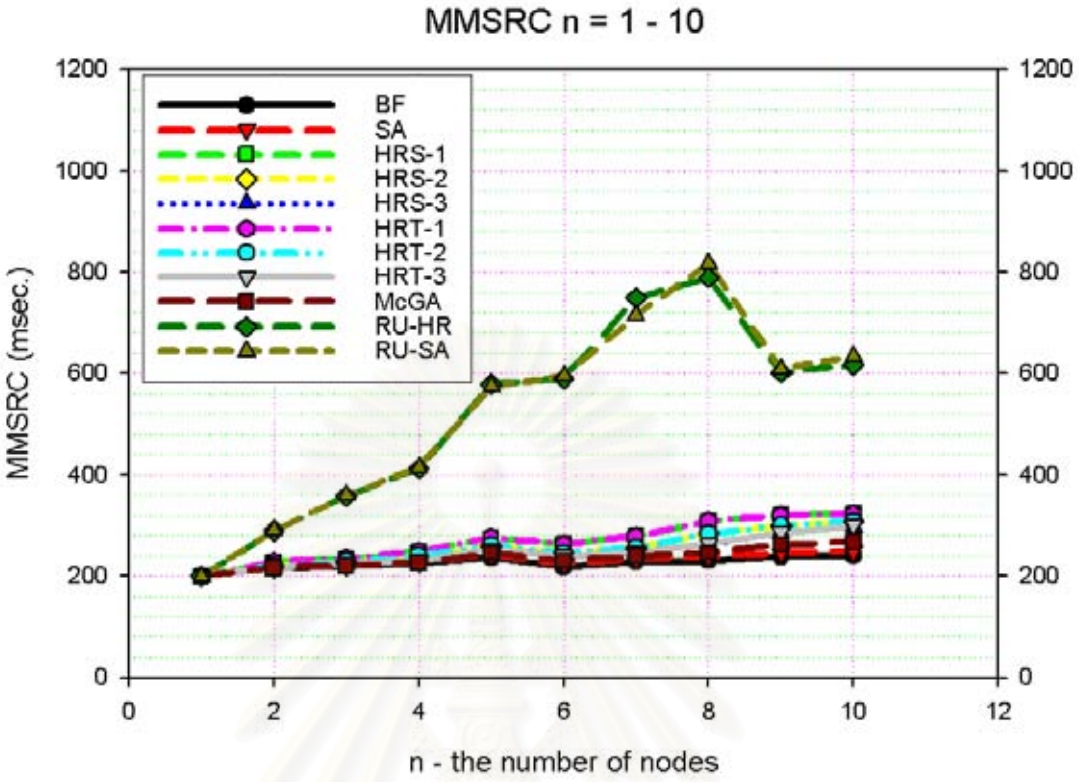
ฌ.) อัลกอริทึมประยุกต์คอมแพคเจเนติก (McGA)

ญ.) อัลกอริทึมกำหนดเส้นทางแบบอิสระยูนิฟอร์มจำนวนรอบการประมวลผลเท่ากับอัลกอริทึม HRS-1 (RU-HR)

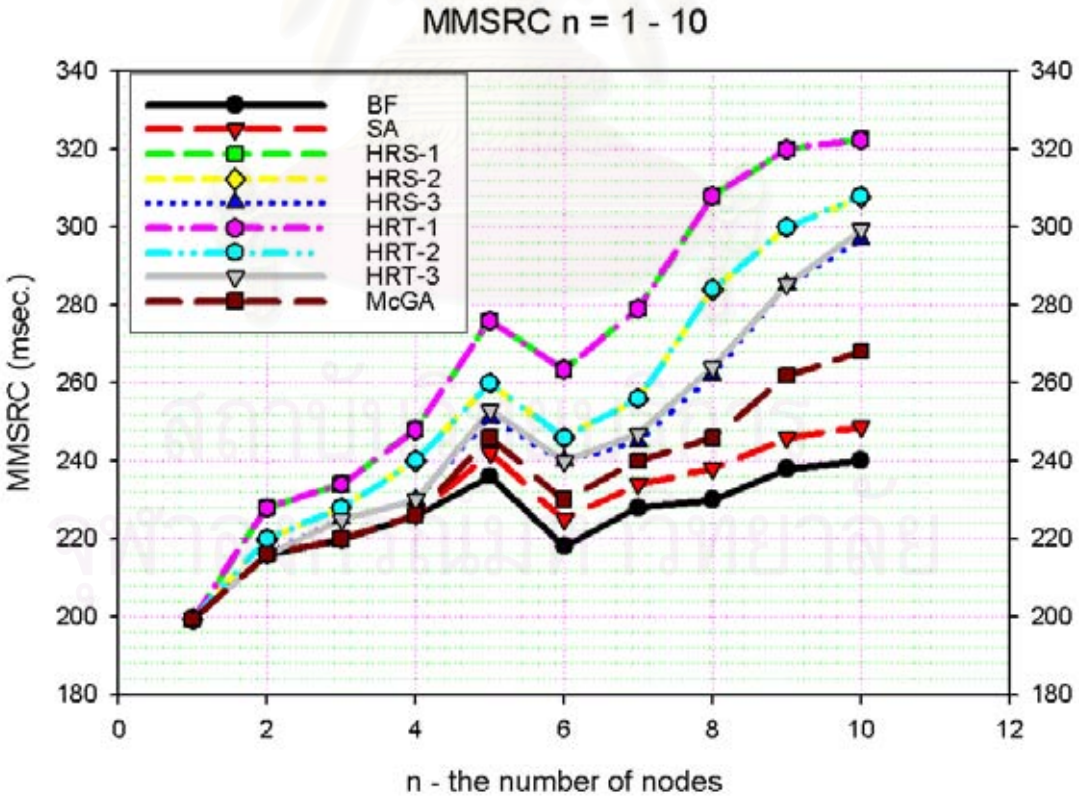
ฎ.) อัลกอริทึมกำหนดเส้นทางแบบอิสระยูฟอร์มจำนวนรอบการประมวลผลเท่ากับอัลกอริทึม SA (RU-SA)

โดยในแต่ละอัลกอริทึมจะทำการวิเคราะห์ค่า MMSRC และเวลาที่ใช้ในการประมวลผลรวมไปถึงจำนวนโมบายล์เอเจนต์ที่ใช้ เทียบกับจำนวนโนดที่ใช้ในเนตเวิร์ก โดยวิธีทำการจำลองจะทำการกำหนดค่าเมตริกซ์ต้นทุนให้กับแต่ละค่า n (จำนวนโนดในเนตเวิร์ก) เป็นจำนวน 10 ชุดสำหรับค่า n = 1, 2, 3, 4, 5, 6, 7, 8, 9 และจำนวน 100 ชุดสำหรับค่า n = 10, 20, 30, 40, 50, 60, 70, 80, 90 และ 100 เหตุผลที่ทำการแบ่งช่วงค่า n เนื่องจากที่ ค่า n มีค่าสูง (ตั้งแต่ 10 ขึ้นไป) BF จะใช้เวลาในการประมวลผลที่ค่อนข้างสูงต่อ 1 รอบของการประมวลผล และเป็นสาเหตุที่ทำการจำลองเมตริกซ์ต้นทุนเพียง 10 ชุด ดังนั้นค่า n ช่วง 1-10 จะทำการวิเคราะห์เปรียบเทียบทุกอัลกอริทึม และค่า n ช่วง 10-100 จะทำการวิเคราะห์เปรียบเทียบอัลกอริทึม SA, HRS-1, HRS-2, HRS-3, HRT-1, HRT-2 และ HRT-3

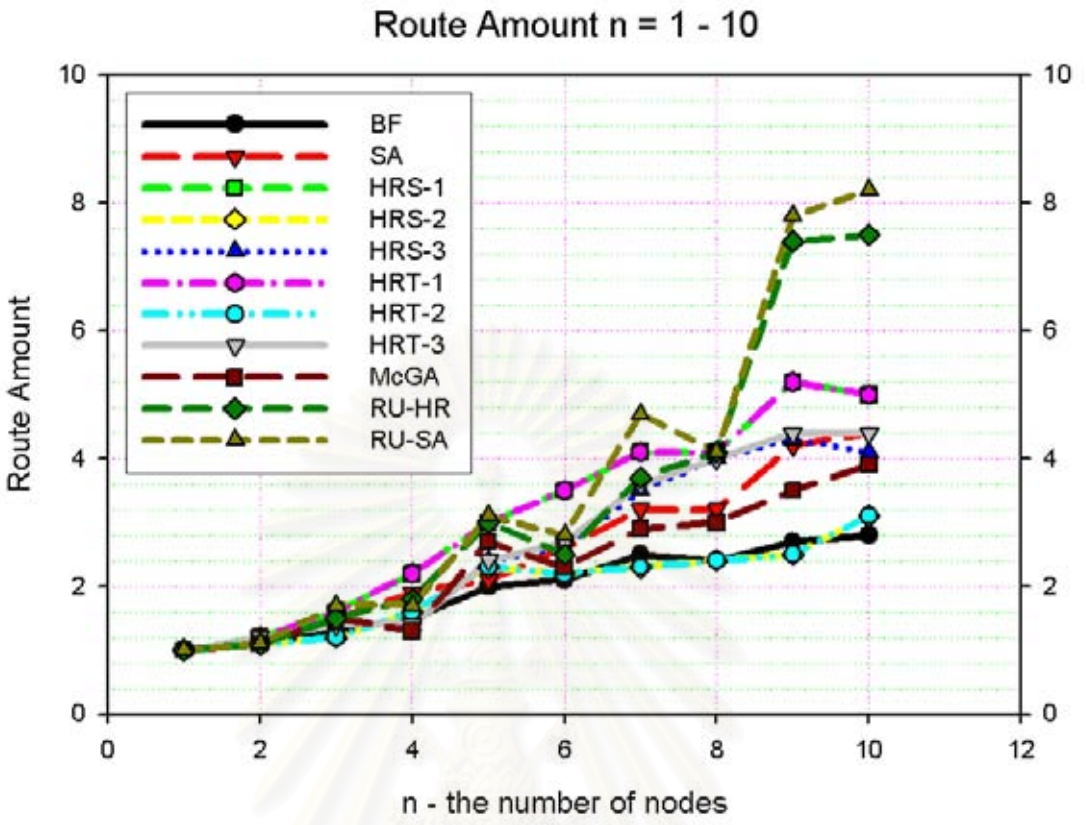
หลังจากที่แต่ละค่า n ได้ทำการจำลองการประมวลผลครบ 100 ชุด (10 ชุดสำหรับ n = 1 – 10) จะนำผลที่ได้มาหาค่าเฉลี่ยซึ่งได้ผลลัพธ์ดังรูปที่ 6.24 – 6.33 ดังนี้



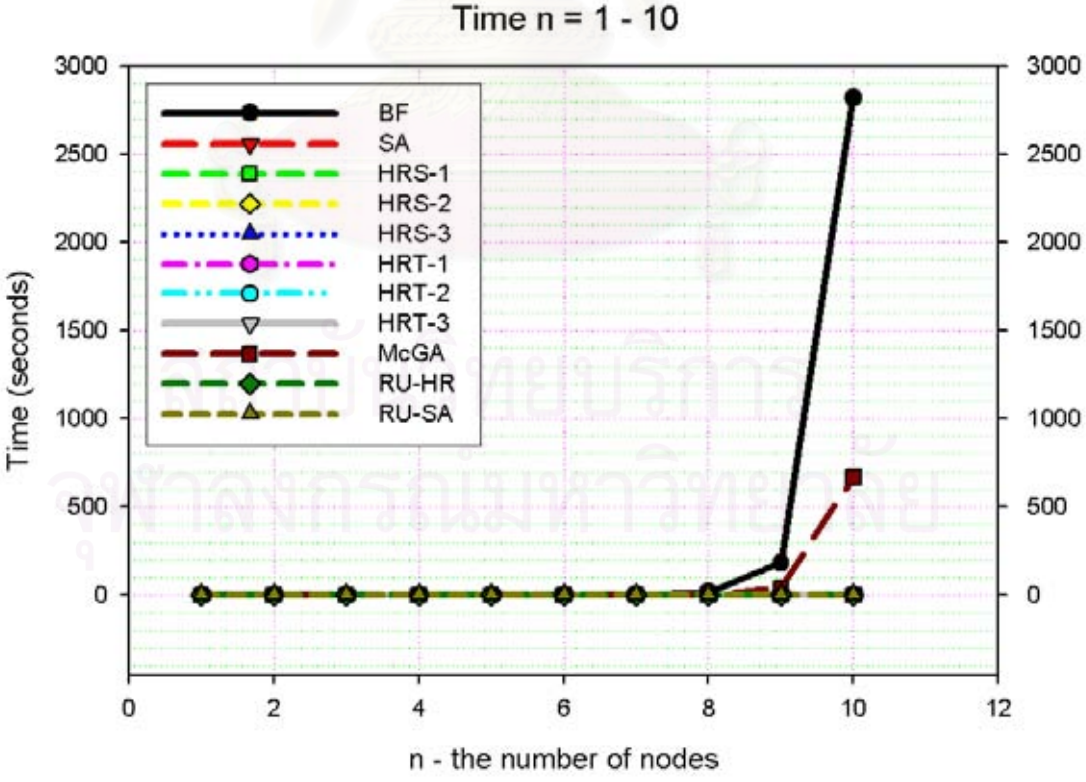
รูปที่ 6.24 MMSRC n = 1 – 10



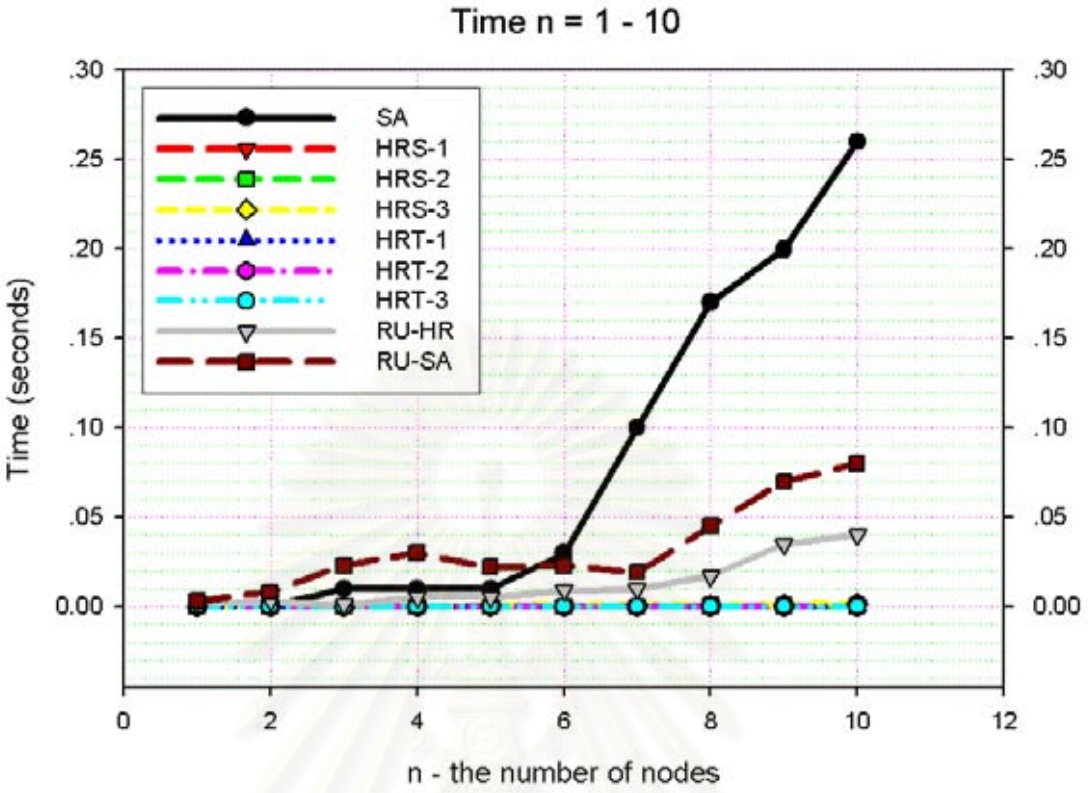
รูปที่ 6.25 MMSRC n = 1 – 10 (เมื่อไม่นำ RU-HR และ RU-SA มาพิจารณา)



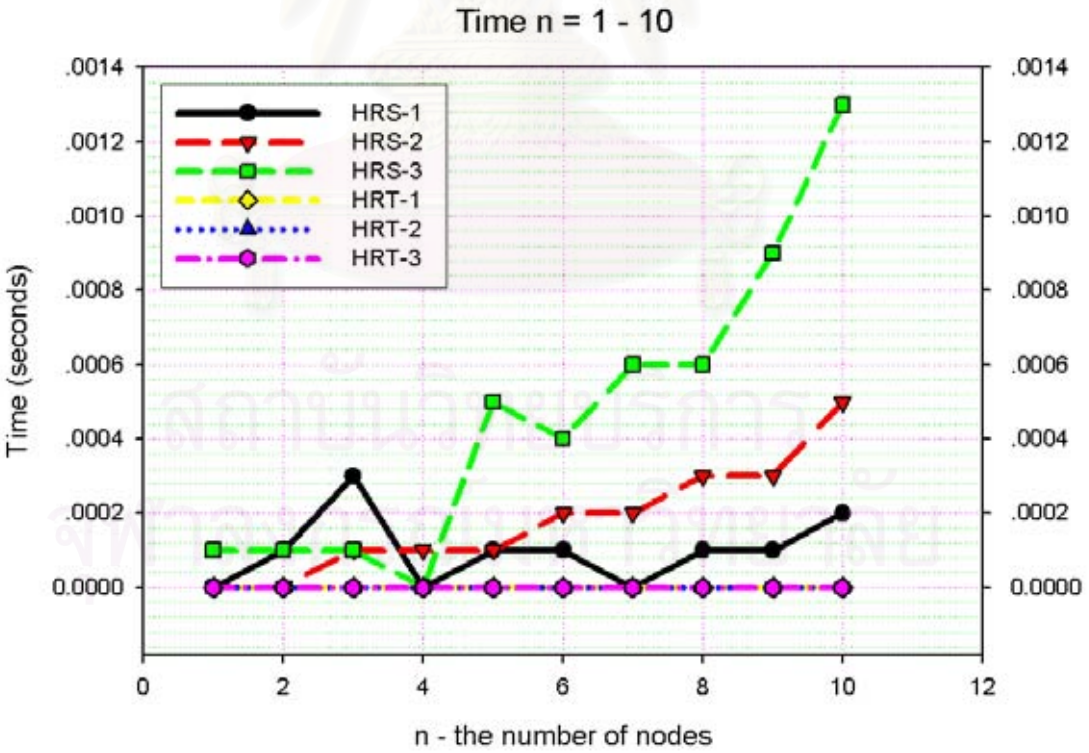
รูปที่ 6.26 Route Amount n = 1 – 10



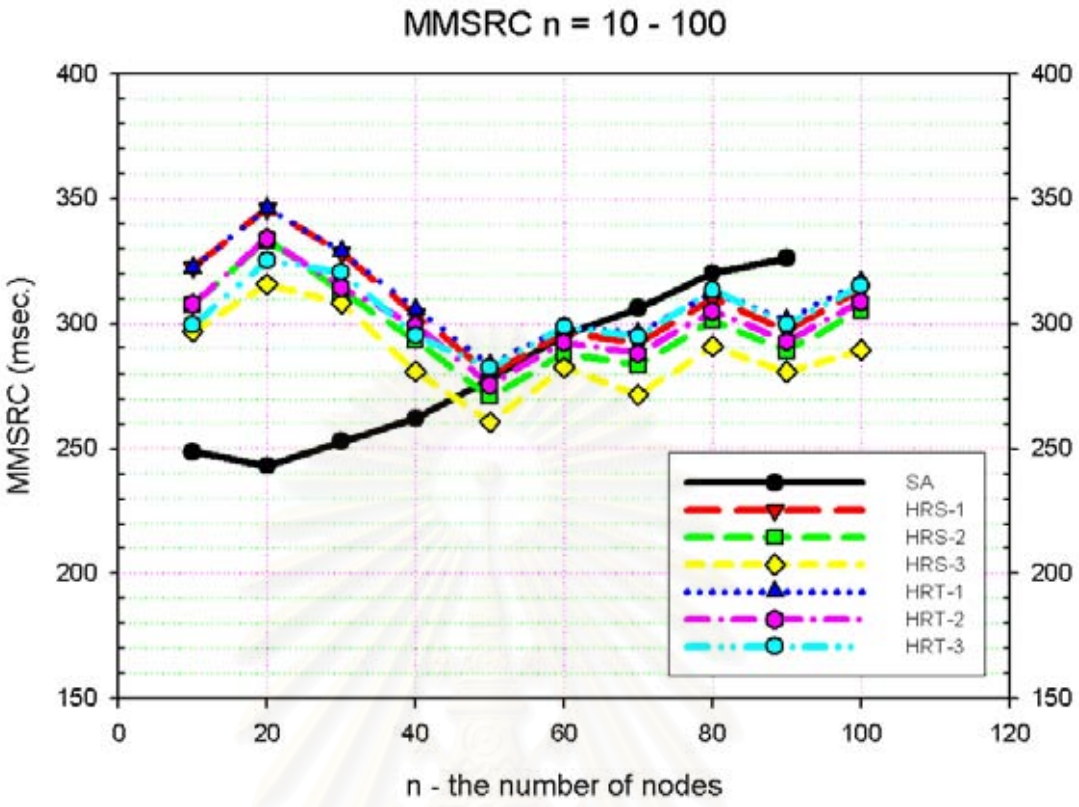
รูปที่ 6.27 Time n = 1 – 10



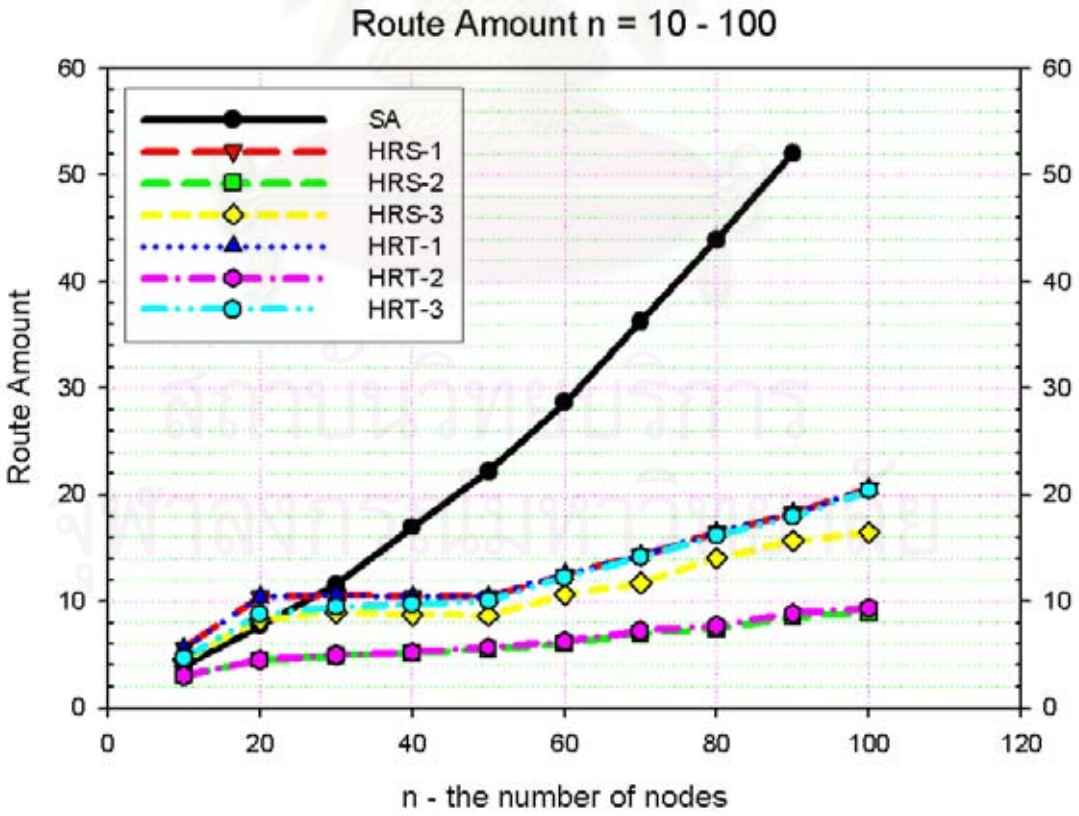
รูปที่ 6.28 Time n = 1 – 10 (เมื่อไม่นำ BF และ McGA มาพิจารณา)



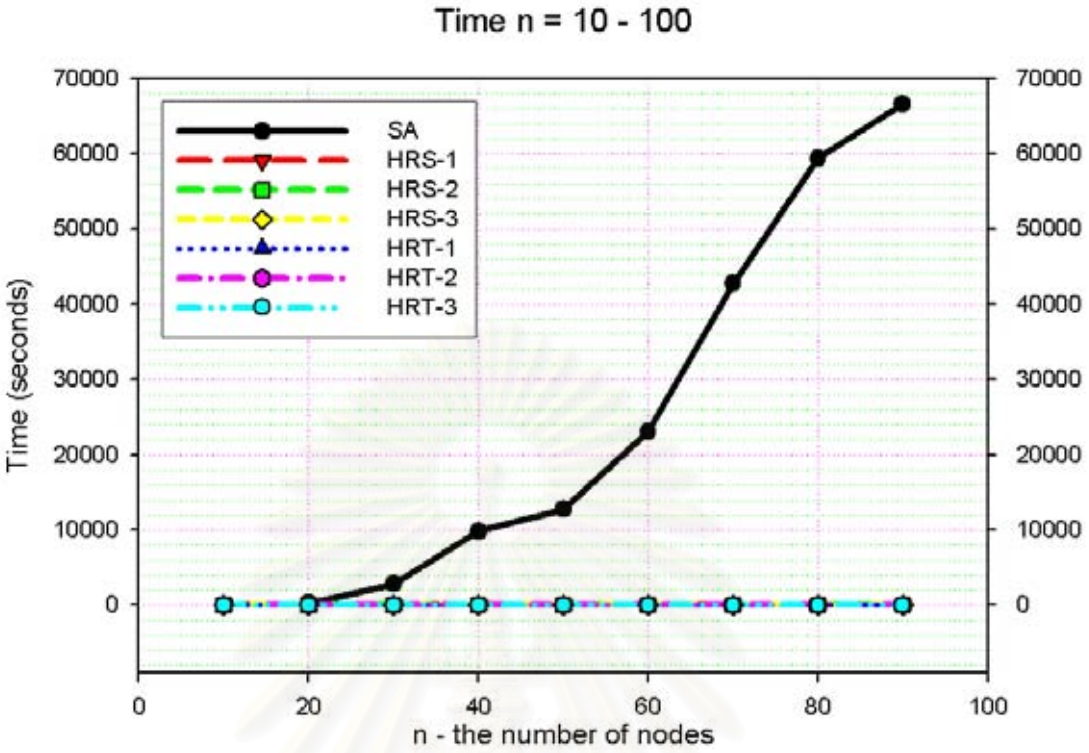
รูปที่ 6.29 Time n = 1 – 10 (เมื่อพิจารณาเฉพาะอัลกอริทึมฮิวริสติก)



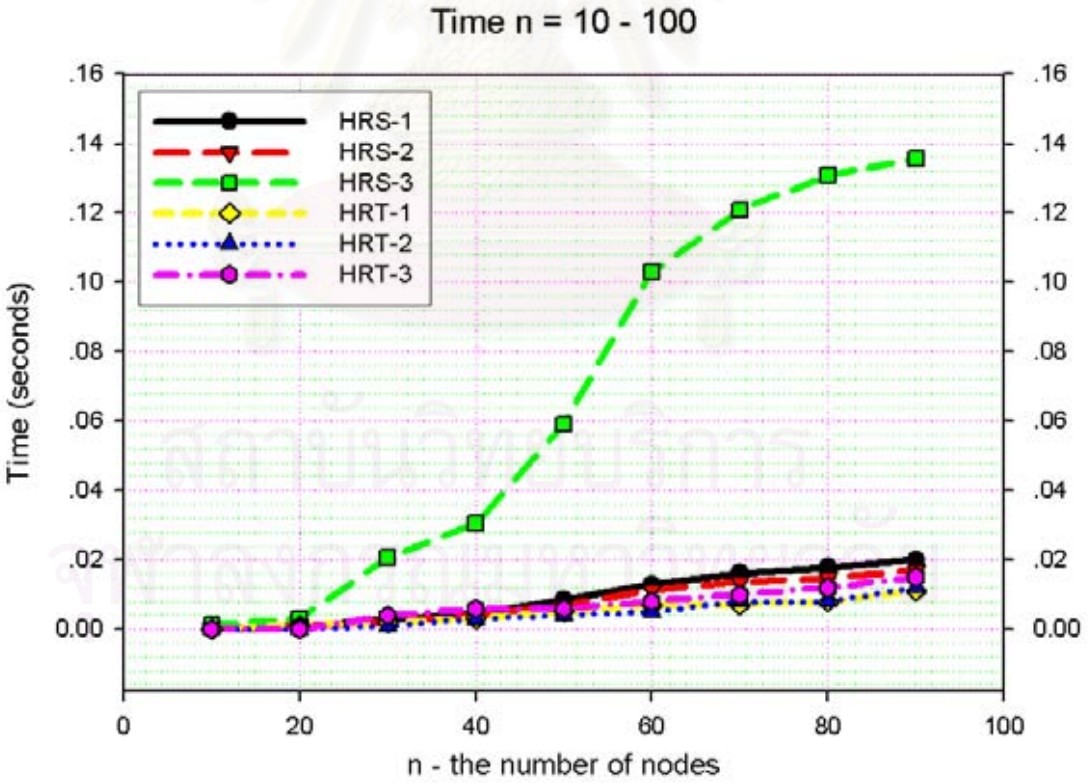
รูปที่ 6.30 MMSRC n = 10 – 100



รูปที่ 6.31 Route Amount n = 10 – 100



รูปที่ 6.32 Time n = 10 – 100



รูปที่ 6.33 Time n = 10 – 100 (เมื่อพิจารณาเฉพาะอัลกอริทึมฮิวริสติก)

สำหรับตัวอย่างของเส้นทางที่เกิดจากอัลกอริทึม BF, SA, McGA, HRS-1, HRS-2, HRS-3, RU-SA และ RU-HR มีแสดงไว้ในภาคผนวก ข.

การวิเคราะห์ผลจากรูปที่ 6.24 – รูปที 6.33 จะแยกตามหัวข้อย่อยดังนี้

6.3.1 MMSRC เมื่อ n = 1 – 10

จากรูปที่ 6.24 อัลกอริทึม RU-HR และ RU-SA จะให้ค่า MMSRC ที่มีค่ามากที่สุด เมื่อเปรียบเทียบกับค่า MMSRC ที่ได้จากอัลกอริทึมอื่น เนื่องจากอัลกอริทึม RU-HR และ RU-SA มีหลักการสุ่มเลือกเส้นทางแบบอิสระ ไม่มีแนวคิดของการพัฒนาคำตอบ ดังนั้นค่า MMSRC ที่ได้จึงมีค่ามากกว่าเมื่อเปรียบเทียบกับ MMSRC ที่ได้จากอัลกอริทึมชนิดอื่น ๆ สำหรับค่า MMSRC ที่ได้จากอัลกอริทึม RU-HR จะมีค่าใกล้เคียงกับค่า MMSRC ที่ได้จากอัลกอริทึม RU-SA ทั้งนี้จากที่กล่าวไว้แล้วอัลกอริทึมทั้ง 2 ไม่มีแนวคิดของการพัฒนาคำตอบ อาศัยวิธีสุ่มเลือกเส้นทางแบบอิสระ ดังนั้นแม้ว่าจำนวนรอบของการสุ่มจะมากขึ้น ค่าคำตอบที่ได้จะไม่แตกต่างจากเดิมมากนัก

จากรูปที่ 6.25 จะเห็นได้ว่า ค่า MMSRC ที่ได้จากแต่ละอัลกอริทึม (ยกเว้นอัลกอริทึม RU-HR และ RU-SA) มีค่าใกล้เคียงกันมาก โดยค่า MMSRC ที่ได้จากอัลกอริทึม BF จะให้ค่าที่ต่ำที่สุด รองมาคือค่า MMSRC ที่ได้จากอัลกอริทึม SA อัลกอริทึม McGA และอัลกอริทึมฮิวริสติก โดยค่า MMSRC ที่ได้จากอัลกอริทึม BF จะให้ค่าที่ต่ำที่สุด เนื่องมาจากอัลกอริทึม BF มีหลักการที่จะคำนวณค่าต้นทุนของคำตอบที่เป็นไปได้ทั้งหมด แล้วนำค่าต้นทุนของแต่ละคำตอบที่ได้มาเปรียบเทียบกันเพื่อให้ได้ค่าดีที่สุด ดังนั้นจึงเป็นที่แน่นอนว่าผลลัพธ์ที่ได้เมื่อเทียบกับอัลกอริทึมอื่นย่อมจะให้ค่าที่ดีที่สุด แต่อย่างไรก็ตามเนื่องจากเป็นการคำนวณต้นทุนของคำตอบที่เป็นไปได้ทั้งหมด ดังนั้นจึงต้องใช้เวลาและหน่วยความจำในการประมวลผลที่มากซึ่งจะกล่าวไว้ในหัวข้อ 6.3.3 สำหรับอัลกอริทึม SA จะให้ค่า MMSRC ที่ดีรองลงมาจากอัลกอริทึม BF โดยจะให้ผลลัพธ์ที่เท่ากับอัลกอริทึม BF ในช่วง n = 1 – 4 และจะเริ่มให้ค่า MMSRC ที่แตกต่างกันออกไปตั้งแต่ค่า n = 5 เป็นต้นไป เนื่องจากอัลกอริทึม SA จะไม่ใช้วิธีการคำนวณต้นทุนของค่าคำตอบที่เป็นไปได้ทั้งหมดเหมือนกับอัลกอริทึม BF โดยคำตอบที่นำมาพิจารณาจะเป็นเพียงส่วนหนึ่งของคำตอบที่เป็นไปได้ทั้งหมด ซึ่งชุดของคำตอบที่จะนำมาพิจารณาจะเริ่มต้นโดยการสุ่มเลือกคำตอบขึ้นมาก่อนหนึ่งชุดเพื่อเป็นค่าคำตอบเริ่มต้น จากนั้นจึงใช้คำตอบเริ่มต้นนี้เป็นแนวทางในการหาคำตอบตัวถัดไปเพื่อที่จะนำมาคำนวณค่าต้นทุนเพื่อเปรียบเทียบหาค่าคำตอบที่ให้ค่าต้นทุนที่ดีที่สุด ดังนั้นจะเห็นได้ว่าช่วงค่า n = 1 – 4 นั้นชุดคำตอบที่เป็นไปได้ทั้งหมดยังคงมีค่าไม่มาก จึงเป็นไปได้ว่า คำตอบที่ให้ค่าที่ดีที่สุดอยู่ในชุดคำตอบที่อัลกอริทึม SA เลือกออกมา เมื่อ n มีค่ามากขึ้นโอกาสที่คำตอบที่ดีที่สุดจะอยู่ในชุดคำตอบที่เลือกออกมาย่อมมีค่าน้อยลง จึงทำให้ค่า MMSRC ที่ได้จึงไม่ดีเท่ากับผลลัพธ์ที่ได้จากอัลกอริทึม BF แต่ยังคงมีความใกล้เคียงกัน สำหรับการทำงานของอัลกอริทึม McGA เกิดจากการสุ่มเลือกเส้นทางให้กับโนดโดยใช้เวกเตอร์ความน่าจะเป็นของแต่ละโนด แล้วทำการ

ปรับปรุงคำตอบโดยใช้ SA จากนั้นจะทำการปรับค่าเวกเตอร์ความน่าจะเป็นเมื่อ MMSRC ของคำตอบที่ได้มีค่าที่ดีขึ้น ดังนั้นจะสังเกตได้ว่า การเลือกคำตอบตัวถัดไป (เส้นทางที่จะกำหนดให้กับโมบายล์เอเจนต์) ของ McGA จะไม่ได้มาจากคำตอบปัจจุบันโดยตรง แต่จะมาจากเวกเตอร์ความน่าจะเป็น ซึ่งเป็นลักษณะการค้นหาคำตอบโดยใช้ตัวแปรสุ่มมาช่วยในการปรับปรุงคำตอบจึงส่งผลให้อัลกอริทึม SA ให้ค่า MMSRC ที่ต่ำกว่าอัลกอริทึม McGA สำหรับอัลกอริทึมฮิวริสติก จากรูปจะเห็นได้ว่า ผลลัพธ์ที่ได้จะให้ค่าที่ใกล้เคียงกับอัลกอริทึม BF อัลกอริทึม SA และอัลกอริทึม McGA ในช่วง n ที่มีค่าต่ำ ๆ แต่จะเริ่มมีความแตกต่างของค่า MMSRC อย่างเห็นได้ชัดเมื่อค่า n มีค่ามากขึ้น เนื่องจากการรวมกลุ่มของอัลกอริทึมฮิวริสติก เป็นการรวมกลุ่มโดยอาศัยค่าต้นทุนแลน และต้นทุนแวนในการจัดเส้นทางให้กับโนด โดยโนดที่เชื่อมกันด้วยลิงก์ต้นทุนแลนจะอยู่ในกลุ่มเดียวกัน แต่อย่างไรก็ตามกลุ่มของโนด บางกลุ่มอาจจะอยู่ใกล้กันและเมื่อรวมกันแล้วอาจให้ค่า MMSRC ที่ลดลงจึงส่งผลให้ค่า MMSRC ที่ได้จาก McGA มีค่าที่ต่ำกว่าค่าที่ได้จากอัลกอริทึมฮิวริสติก

นอกจากนี้จะเห็นได้ว่า ค่า MMSRC ที่ได้จากอัลกอริทึม HRS และ HRT จะให้ค่า MMSRC ที่ใกล้เคียงกันมาก เนื่องมาจากที่ค่า n มีค่าน้อย ๆ ความหลากหลายของคำตอบที่ยังมีไม่มากเมื่อใช้วิธีการเลือกคำตอบตัวถัดไปโดยอาศัยการสุ่มเข้ามาช่วย จึงเป็นไปได้ว่า คำตอบที่ดีที่สุดในแต่ละอัลกอริทึมจะอยู่ในเซตคำตอบที่พิจารณาเหมือนกัน เมื่อพิจารณาค่า MMSRC ที่ได้จากอัลกอริทึมฮิวริสติกด้วยกันจะพบว่า ค่า MMSRC ที่ได้จากอัลกริ HR-3 จะให้ค่าที่ต่ำที่สุด รองลงมาคืออัลกอริทึม HR-2 แล้วตามด้วยอัลกอริทึม HR-1 โดยสาเหตุที่อัลกอริทึม HR-2 ให้ค่า MMSRC ที่ดีกว่าอัลกอริทึม HR-1 อันเนื่องมาจากอัลกอริทึม HR-2 เป็นการนำอัลกอริทึม HR-1 มาพัฒนาอีกครั้งโดยอาศัยแนวคิดการรวมกลุ่มของเซิร์ฟเวอร์ที่อยู่ใกล้กันให้อยู่ในเส้นทางเดียวกันจนกว่า จะให้ค่า MMSRC ที่ไม่ดีกว่าเดิมจึงยุติการรวมกลุ่มของเซิร์ฟเวอร์ ดังนั้นจึงเป็นที่แน่นอนว่า อัลกอริทึม HR-2 ควรจะให้ค่าที่ดีกว่าผลลัพธ์ที่ได้จากอัลกอริทึม HR-1 แต่สิ่งที่จะต้องพิจารณาเพิ่มคือเวลาในการประมวลผล โดยคิดว่าเวลาประมวลผลที่ใช้มากขึ้นในอัลกอริทึม HR-2 จะคุ้มค่ากับค่า MMSRC ที่ลดลงมาจากอัลกอริทึม HR-1 หรือไม่ ซึ่งจะวิเคราะห์ไว้ในหัวข้อ 6.4.3 สำหรับค่า MMSRC ที่มีค่าต่ำที่สุด (เมื่อเปรียบเทียบกับอัลกอริทึมฮิวริสติกด้วยกัน) จะเป็นค่า MMSRC ที่ได้จากอัลกอริทึม HR-3 โดยอัลกอริทึม HR-3 ให้ค่า MMSRC ที่ดีที่สุด ดีกว่าค่า MMSRC ที่ได้จากอัลกอริทึม HR-2 มีเหตุผลมาจากในการรวมกลุ่มของเซิร์ฟเวอร์ให้อยู่ในเส้นทางเดียวกันของอัลกอริทึม HR-3 นั้นเมื่อพบว่ามีกลุ่มของเซิร์ฟเวอร์ที่สามารถรวมกันได้ จะนำมาจัดเรียงลำดับใหม่อีกครั้งหนึ่ง (โดยการจัดเรียงลำดับคือการนำอัลกอริทึม Simulated Annealing หรืออัลกอริทึม Tabu Search มาปรับปรุงนั่นเอง) ดังนั้นผลลัพธ์ที่ได้จึงมีแนวโน้มที่จะดีกว่าผลลัพธ์ที่ได้จากอัลกอริทึม HR-2 แต่เป็นที่แน่นอนว่าเวลาในการประมวลผลย่อมเพิ่มขึ้น (โดยจะอธิบายไว้ในหัวข้อ 6.3.3)

6.3.2 Route-Amount เมื่อ n = 1 – 10

สำหรับจำนวนเส้นทางที่ใช้ (Route-Amount) จะบ่งบอกถึงจำนวนโมบายล์เอเจนต์ที่ปล่อยเข้าสู่เนตเวิร์ก และยิ่งจำนวนโมบายล์เอเจนต์ที่ใช้มากเท่าใด แนวโน้มที่จะใช้ปริมาณทราฟฟิกจะมีปริมาณมากขึ้นเท่านั้น จากรูปที่ 6.26 จะเห็นได้ว่าจำนวนโมบายล์เอเจนต์ที่ได้จากอัลกอริทึม RU-HR และ RU-SA จะให้จำนวนโมบายล์เอเจนต์มากที่สุด โดยอัลกอริทึมที่เหลือจะให้จำนวนโมบายล์เอเจนต์ที่ใกล้เคียงกัน นอกจากนี้จะเห็นได้ว่าจำนวนโมบายล์เอเจนต์ที่ใช้จะมีแนวโน้มที่มากขึ้นเมื่อจำนวนเซิร์ฟเวอร์ในเนตเวิร์กมีปริมาณมากขึ้น

สำหรับจำนวนโมบายล์เอเจนต์ที่ใช้งานในช่วง n = 1 – 10 จะยังไม่เห็นถึงความแตกต่างได้ชัดเจน แต่จะเห็นได้ชัดเจนเมื่อพิจารณาในช่วง n = 10 – 100 ดังนั้นจะวิเคราะห์จำนวนโมบายล์เอเจนต์อีกครั้งหนึ่งในหัวข้อ 6.3.5 ต่อไป

6.3.3 Time เมื่อ n = 1 – 10

จากหัวข้อที่ 6.3.1 ในการเปรียบเทียบผลลัพธ์ที่ได้จากแต่ละอัลกอริทึมนั้นนอกจากจะวิเคราะห์ค่า MMSRC, Route-Amount แล้วยังต้องพิจารณาถึงเวลาที่ใช้ในการประมวลผลด้วยเช่นกัน จากรูปที่ 6.27 เป็นการเปรียบเทียบเวลาที่ใช้ในการประมวลผลของทุกอัลกอริทึม จะเห็นได้ว่า อัลกอริทึม BF จะใช้เวลาในการประมวลผลมากที่สุดเมื่อเทียบกับเวลาที่ใช้ในการประมวลผลจากอัลกอริทึมอื่น ๆ เนื่องจากตามที่ได้กล่าวไว้แล้วว่า อัลกอริทึม BF เป็นการคำนวณค่าต้นทุนของคำตอบที่เป็นไปได้ทั้งหมด ดังนั้นเวลาที่ใช้ในการประมวลผลจะประกอบด้วย 2 ส่วนที่สำคัญคือ การค้นหาคำตอบที่เป็นไปได้ทั้งหมด และการคำนวณต้นทุนของแต่ละคำตอบ ในส่วนของการค้นหาคำตอบที่เป็นไปได้ทั้งหมดนั้นจะมีคำนวณที่สำคัญอยู่ 2 ส่วนคือการจัดกลุ่ม และการสลับลำดับ ดังนั้นสมมติว่าจำนวนโนดที่ใช้พิจารณามีค่าเท่ากับ n เมื่อทำการสลับลำดับจะมีรูปแบบได้ทั้งหมด n! รูปแบบ และแต่ละรูปแบบจะต้องมาผ่านการจัดกลุ่ม โดยจำนวนกลุ่มเปรียบเสมือนจำนวนโมบายล์เอเจนต์ที่ใช้ ดังนั้นในแต่ละแบบจะจัดกลุ่มได้เท่ากับ $\sum_{i=0}^{n-1} C_i^{n-1}$ ดังนั้นกรณี n = 10 รูปแบบของคำตอบที่เป็นไปได้ทั้งหมดจะเท่ากับ $10!(\sum_{i=0}^{9} C_i^{9})$ (ประมาณ 1,857,945,600 รูปแบบ) และแต่ละคำตอบที่เป็นไปได้จะต้องมีการนำไปคำนวณค่าต้นทุนเพื่อนำมาเปรียบเทียบหาค่าที่ดีที่สุด ซึ่งตัวอย่างการคำนวณจะเห็นได้ว่าเป็นเรื่องปกติที่อัลกอริทึม BF จะใช้เวลาในการประมวลผลค่อนข้างสูง อัลกอริทึมที่ใช้เวลาประมวลผลรองลงมาคืออัลกอริทึม McGA โดยสาเหตุที่อัลกอริทึม McGA ใช้เวลาประมวลผลที่มากกว่าอัลกอริทึม SA เนื่องจาก ถ้ามองอัลกอริทึม McGA ในส่วนของการทำงานในแต่ละรอบของการประมวลผลจะจะประกอบไปด้วย 2 ส่วนที่สำคัญคือ ส่วนแรก

ทำหน้าที่กำหนดเส้นทางให้กับโนดโดยใช้อัลกอริทึม McGA และส่วนที่สองทำหน้าที่นำชุดของเส้นทางที่ได้ไปทำการปรับปรุงคำตอบโดยใช้อัลกอริทึม SA ดังนั้นจึงส่งผลให้อัลกอริทึม McGA จึงใช้เวลาในการประมวลผลมากกว่าอัลกอริทึม SA

สำหรับอัลกอริทึม SA จะใช้เวลาการประมวลผลที่ลดลงมา แต่ยังคงมีค่ามากเมื่อนำไปพิจารณาเปรียบเทียบกับอัลกอริทึมฮิวริสติก ดังรูปที่ 6.28 โดยจากรูปจะเห็นได้ว่าเวลาที่ใช้ในการประมวลผลของอัลกอริทึม SA จะให้ค่าที่มากขึ้นอย่างเห็นได้ชัดตั้งแต่ n = 6 เป็นต้นไป โดยในช่วงค่า n = 1 – 5 ค่าคำตอบที่เป็นไปได้ยังมีค่าไม่มาก ดังนั้นเวลาที่ใช้ในการประมวลผลของ SA จึงใกล้เคียงกับอัลกอริทึมฮิวริสติก แต่เมื่อ n มีค่าตั้งแต่ 6 เป็นต้นไปเวลาที่ใช้จะเริ่มให้ความแตกต่างกัน โดยมีสาเหตุเนื่องมาจากในการคำนวณหาคำตอบที่ให้ค่าต้นทุนที่ดีที่สุดนั้น อัลกอริทึม SA จะทำการแยกเป็นกรณีย่อย ๆ สมมติว่าใช้จำนวนโนดเท่ากับ n อัลกอริทึมจะแยกคำนวณตั้งแต่กรณีใช้เส้นทางเดียว ใช้จำนวนเส้นทางเท่ากับ 2 ใช้จำนวนเส้นทางเท่ากับ 3 จนถึงกรณีใช้จำนวนเส้นทางเท่ากับ n (อัลกอริทึม BF มีลักษณะการคำนวณแยกกรณีจำนวนเส้นทางเช่นเดียวกัน) ดังนั้นจะเห็นได้ว่าเมื่อเทียบกับอัลกอริทึมฮิวริสติกที่มีการกำหนดจำนวนเส้นทางตายตัวตั้งแต่เริ่มต้น (โดยใช้อัลกอริทึม HR-1 ในหัวข้อ 6.2.1) ดังนั้นเวลาที่ใช้ในกรณีของอัลกอริทึม SA ย่อมจะให้ค่าเวลาการประมวลผลที่มากขึ้นเมื่อจำนวนโนดในเนตเวิร์กเพิ่มขึ้น สำหรับเวลาที่ใช้ในการประมวลผลเมื่อพิจารณาเฉพาะอัลกอริทึมฮิวริสติก ผลลัพธ์ที่ได้จะแสดงไว้ดังรูปที่ 6.29 จากรูปจะเห็นได้ว่า เวลาประมวลผลที่ได้จาก HRT จะให้ค่าเวลาที่ต่ำกว่าเมื่อเทียบกับเวลาประมวลผลที่ได้จากอัลกอริทึม HRS และจากรูปที่ 6.29 จะเห็นได้ว่าเมื่อเปรียบเทียบเฉพาะอัลกอริทึม HRS ด้วยกัน เวลาประมวลผลที่ได้จาก HRS-1 จะใช้เวลาโดยรวมที่ต่ำที่สุดรองมาคืออัลกอริทึม HRS-2 และ HRS-3 โดยอัลกอริทึม HRS-2 จะใช้เวลาการประมวลผลที่มากกว่าอัลกอริทึม HRS-1 เนื่องจากอัลกอริทึม HRS-2 เป็นการนำผลลัพธ์ที่ได้จากอัลกอริทึม HRS-1 มาใช้อีกทีหนึ่งโดยการรวมกลุ่มของโนดที่อยู่ใกล้กัน ดังนั้นเวลาประมวลผลที่ใช้เพิ่มขึ้นจากอัลกอริทึม HRS-1 คือเวลาที่ใช้ในการหาและรวมกลุ่มของโนดที่อยู่ใกล้กันนั่นเอง สำหรับอัลกอริทึม HRS-3 จะใช้แนวคิดเช่นเดียวกับแบบที่ 2 แต่มีวิธีในการรวมกลุ่มของโนดที่ใกล้กันที่แตกต่างจากอัลกอริทึม HRS-2 โดยอัลกอริทึม HRS-3 เมื่อได้กลุ่มของโนดที่จะมารวมกันเป็นเส้นทางเดียวกันแทนที่จะคำนวณรูปแบบของการรวม (ซึ่งมี 4 แบบจากรูปที่ 6.4) แล้วหารูปแบบที่ให้ค่า MMSRC ที่ต่ำที่สุดดังเช่นอัลกอริทึม HRS-2 แต่อัลกอริทึม HRS-3 จะนำโนดที่มีใน 2 กลุ่มนั้นมาจัดเรียงลำดับใหม่โดยใช้อัลกอริทึม SA ดังนั้นจะเห็นได้ว่าเวลาประมวลผลที่อัลกอริทึม HRS- 3 เพิ่มขึ้นมาคือเวลาที่ใช้ในส่วนของการรวมจัดเรียงลำดับโนดใหม่โดยใช้อัลกอริทึม SA นั่นเอง อย่างไรก็ตามจากรูปที่ 6.29 แม้จะมีความแตกต่างด้านเวลาของแต่ละอัลกอรทึม แต่ในการใช้งานจริงความแตกต่างระดับ 1 มิลลิวินาทีนั้นถือว่าไม่มีผลมากนัก ดัง

นั้นในการเปรียบเทียบสมรรถนะของอัลกอริทึมฮิวริสติกด้วยกันเวลาประมวลผลแทบจะไม่มีผลมากนัก สำหรับการวิเคราะห์ช่วง n = 10 – 100 จะกล่าวในหัวข้อต่อไป

6.3.4 MMSRC เมื่อ n = 10 – 100

จากรูปที่ 6.30 จะเห็นได้ว่าผลลัพธ์สามารถแบ่งออกเป็น 2 ช่วงคือ ช่วง n = 10 – 50 และช่วง n = 50 – 100 โดยค่า MMSRC ที่ได้จากอัลกอริทึม SA ในช่วง n = 10 – 50 จะให้ค่าดีที่สุดเมื่อเปรียบเทียบกับอัลกอริทึมฮิวริสติก แต่จะเริ่มใกล้เคียงและมีค่ามากขึ้นเมื่อเปรียบเทียบกับอัลกอริทึมฮิวริสติก ในช่วงค่า n = 50 – 100 โดยสาเหตุเนื่องมาจากหลักการหาคำตอบถัดไปของอัลกอริทึม SA จะใช้วิธีการสุ่มสลับตำแหน่ง ซึ่งการสลับตำแหน่งนี้เองเป็นข้อด้อยของอัลกอริทึม SA กล่าวคือวิธีการเลือกคำตอบถัดไปโดยวิธีสลับตำแหน่งจะไม่สามารถทำให้เกิดคำตอบที่เป็นไปได้ทั้งหมด จะมีคำตอบบางตัวที่ไม่สามารถเกิดขึ้นได้เลย โดยจะยกตัวอย่างดังต่อไปนี้

ตัวอย่างที่ 6.1 *ตัวอย่างประกอบการอธิบายการทำงานของอัลกอริทึม Simulated Annealing*

กำหนดให้คำตอบเริ่มต้นคือ [1 – 2 – 3 – 4]

ดังนั้นจากแนวคิดของอัลกอริทึม Simulated Annealing จะทำการสลับตำแหน่งทีละคู่เพื่อเป็นการกำหนดค่าคำตอบใหม่ดังนี้

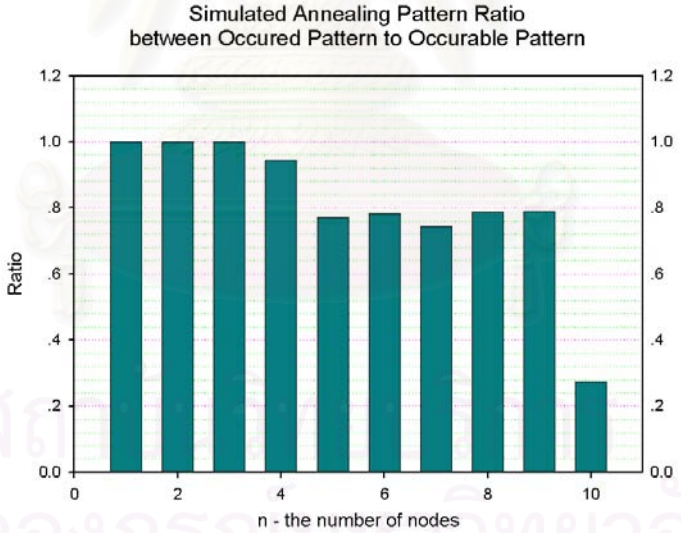
ครั้งที่ 0 (คำตอบเริ่มต้น)	[1 – 2 – 3 – 4]
ครั้งที่ 1	[2 – 1 – 3 – 4]
ครั้งที่ 2	[2 – 3 – 1 – 4]
ครั้งที่ 3	[2 – 4 – 1 – 3]
ครั้งที่ 4	[4 – 2 – 1 – 3]
ครั้งที่ 5	[1 – 2 – 4 – 3]
ครั้งที่ 6	[1 – 3 – 4 – 2]
ครั้งที่ 7	[4 – 3 – 1 – 2]
ครั้งที่ 8	[4 – 3 – 2 – 1]
ครั้งที่ 9	[4 – 1 – 2 – 3]
ครั้งที่ 10	[1 – 4 – 2 – 3]
ครั้งที่ 11	[3 – 4 – 2 – 1]
ครั้งที่ 12	[3 – 4 – 1 – 2]
ครั้งที่ 13	[3 – 1 – 4 – 2]
ครั้งที่ 14	[3 – 1 – 2 – 4]

ครั้งที่ 15 [1 – 3 – 2 – 4]

จากตัวอย่างที่ 6.1 จะเห็นได้ว่า ถ้ากำหนดให้ขนาดของเส้นทางเท่ากับ 4 โนดรูปแบบที่เป็นไปได้ทั้งหมดควรจะมี 24 รูปแบบ (4!) แต่ในตัวอย่างสามารถเกิดขึ้นได้เพียง 16 รูปแบบ โดยจะมีรูปแบบที่ไม่สามารถเกิดขึ้นได้คือ

[2 – 1 – 4 – 3] [2 – 3 – 4 – 1] [1 – 4 – 3 – 2] [2 – 4 – 3 – 1]
[3 – 2 – 1 – 4] [3 – 2 – 4 – 1] [4 – 1 – 3 – 2] [4 – 2 – 3 – 1]

ดังนั้นจะเห็นได้ว่า หลักการสลับตำแหน่งเพื่อให้เกิดคำตอบถัดไปของอัลกอริทึม Simulated Annealing จะไม่สามารถก่อให้เกิดรูปแบบของคำตอบที่เป็นไปได้ทั้งหมด บางรูปแบบอาจจะไม่สามารถเกิดขึ้นได้ ดังนั้นจึงมีความเป็นไปได้ที่คำตอบที่ดีที่สุดอาจจะอยู่ในชุดคำตอบที่ไม่สามารถเกิดขึ้นได้เช่นเดียวกัน จึงส่งผลให้คำตอบที่ได้จากอัลกอริทึม SA ด้อยกว่าอัลกอริทึม BF สำหรับอัตราส่วนจำนวนรูปแบบที่เกิดขึ้นเทียบกับจำนวนรูปแบบที่เป็นไปได้ทั้งหมดเมื่อทำการเก็บค่า 100 รอบ (สำหรับค่า n = 1–7 และจำนวน 10 รอบสำหรับค่า n = 8 – 10) แล้วหาค่าเฉลี่ยจะได้ผลลัพธ์ดังรูปที่ 6.34



รูปที่ 6.34 อัตราส่วนรูปแบบที่เกิดขึ้นต่อรูปแบบที่เป็นไปได้ทั้งหมด

จากรูปที่ 6.34 จะเห็นได้ว่าช่วงค่า n = 1 – 4 จะให้อัตราส่วนรูปแบบที่เกิดขึ้นต่อรูปแบบที่เป็นไปได้ทั้งหมด เฉลี่ยได้ใกล้เคียง 1 และเมื่อ n มีค่ามากขึ้นจะได้อัตราส่วนอยู่ที่ประมาณ 0.7 – 0.8 ดังนั้นจำนวนรูปแบบที่ไม่สามารถเกิดขึ้นได้จะมีประมาณร้อยละ 20 ถึง 30 จึงส่งผลให้ค่า MMSRC ที่ได้จากอัลกอริทึม SA มีค่าไม่ดีเท่ากับค่า MMSRC ที่ได้จากอัลกอริทึม BF ซึ่งสอดคล้องกับผลลัพธ์ในหัวข้อ 6.3.1 (รูปที่ 6.24 – รูปที่ 6.25)

สำหรับผลลัพธ์ที่ได้จากอัลกอริทึมฮิวริสติกทั้ง 6 อัลกอริทึมจะเห็นได้ว่า ผลลัพธ์ที่ได้จากอัลกอริทึม HRS-3 จะให้ค่าที่ดีที่สุด แต่ HRT-3 จะให้ค่าที่ดีรองจาก HRS-3 ในช่วงค่า n = 10 – 20 และค่า MMSRC จะเริ่มมากขึ้นจนกระทั่งมากที่สุดที่ค่า n = 100 เมื่อไม่นำ HRT-3 มาพิจารณาจะพบว่าอัลกอริทึมฮิวริสติกที่ให้ผลลัพธ์ที่ดีที่สุดจะเรียงตามลำดับได้ดังนี้คือ HRS-3, HRS-2, HRT-2, HRS-1 และ HRT-1 โดยผลลัพธ์ที่ได้จะเหมือนกับผลลัพธ์ที่ได้จากการวิเคราะห์ในหัวข้อ 6.3.1 (MMSRC n = 1 – 10) สำหรับ HRT-3 จะให้ผลลัพธ์ที่ดีในช่วงค่า n ต่ำ ๆ แต่จะให้ค่าผลลัพธ์ที่ด้อยที่ค่า n สูง ๆ จากผลลัพธ์สามารถวิเคราะห์ได้ว่าวิธีการกำหนดคำตอบถัดไปโดยการคงตำแหน่งไว้ระยะหนึ่งของอัลกอริทึม Tabu Search จะให้ประสิทธิภาพที่ไม่ดีนักที่จำนวนโนดมีค่ามาก ๆ โดยจะสังเกตเห็นได้ว่า HRT-2 เป็นอัลกอริทึมที่ใช้อัลกอริทึม Tabu Search เช่นเดียวกัน แต่เป็นการใช้อัลกอริทึม Tabu Search เพียงครั้งเดียวในตอนกำหนดเส้นทางรอบแรก หลังจากนั้นการรวมกลุ่มของโนดจะใช้วิธีเลือกแบบที่ดีที่สุดจาก 4 แบบที่เป็นไปได้ (รูปที่ 6.4) แต่จะไม่มีการใช้อัลกอริทึม Tabu Search อีก ส่วน HRT-3 จะใช้อัลกอริทึม Tabu Search ทุกครั้งที่มีการรวมกลุ่ม ซึ่งการรวมกลุ่มนี้เองที่ทำให้จำนวนโนดเพิ่มมากขึ้นดังนั้นทำให้สรุปได้ว่า อัลกอริทึม Tabu Search เหมาะสำหรับการกำหนดให้กับเส้นทางที่มีจำนวนโนดไม่มากนัก

จากรูปที่ 6.30 จะเห็นได้ว่าผลลัพธ์ที่ได้จะไม่มีลักษณะที่เป็นแนวโน้มหรือมีการลู่ในทิศทางใดทิศทางหนึ่งเมื่อจำนวนโนดมากขึ้น ทั้งนี้เนื่องจากในการกำหนดเส้นทางให้กับโมบายล์เอเจนต์นั้นสามารถที่จะกำหนดให้กับโมบายล์เอเจนต์ได้มากกว่าหนึ่งตัว ดังนั้นในแต่ละค่า n จะมีพารามิเตอร์อีกตัวหนึ่งที่ไม่เท่ากันคือ จำนวนเส้นทางที่ใช้ หรือจำนวนโมบายล์เอเจนต์ที่ปล่อยเข้าสู่เนตเวิร์กนั่นเอง ดังนั้นค่า MMSRC คือผลลัพธ์ที่ได้จากการกำหนดเส้นทางให้กับตัวโมบายล์เอเจนต์ไม่ใช่ผลลัพธ์โดยตรงเนื่องจากจำนวนโนดในเนตเวิร์กที่มากขึ้น เพราะฉะนั้นค่าที่ควรจะมีลักษณะที่เป็นแนวโน้ม หรือมีการลู่ เมื่อจำนวนโนดในเนตเวิร์กมากขึ้น ควรจะเป็นค่าจำนวนเส้นทางที่ใช้ หรือจำนวนโมบายล์เอเจนต์ที่ปล่อยเข้าสู่เนตเวิร์กนั่นเอง ซึ่งจะวิเคราะห์ไว้ในหัวข้อที่ 6.3.5

6.3.5 Route-Amount เมื่อ n = 10 – 100

สำหรับจำนวนเส้นทางที่ใช้เมื่อเปรียบเทียบกับจำนวนโนดที่เพิ่มขึ้นสามารถอธิบายได้ดังรูปที่ 6.31 จากรูปจะเห็นได้ว่า ผลลัพธ์ที่ได้มีแนวโน้มมากขึ้นเมื่อจำนวนโนดในเนตเวิร์กมากขึ้น โดยอัลกอริทึม SA จะใช้จำนวนเส้นทางมากที่สุดเมื่อเปรียบเทียบกับอัลกอริทึมอื่น ซึ่งมีสาเหตุมาจาก จากที่ได้ยกตัวอย่างไว้ในตัวอย่างที่ 6.1 จะเห็นได้ว่าในอัลกอริทึม SA จะมีรูปแบบจำนวนหนึ่งที่ไม่สามารถเกิดขึ้นได้ และจากหลักการกำหนดเส้นทางของอัลกอริทึม SA จะทำการ

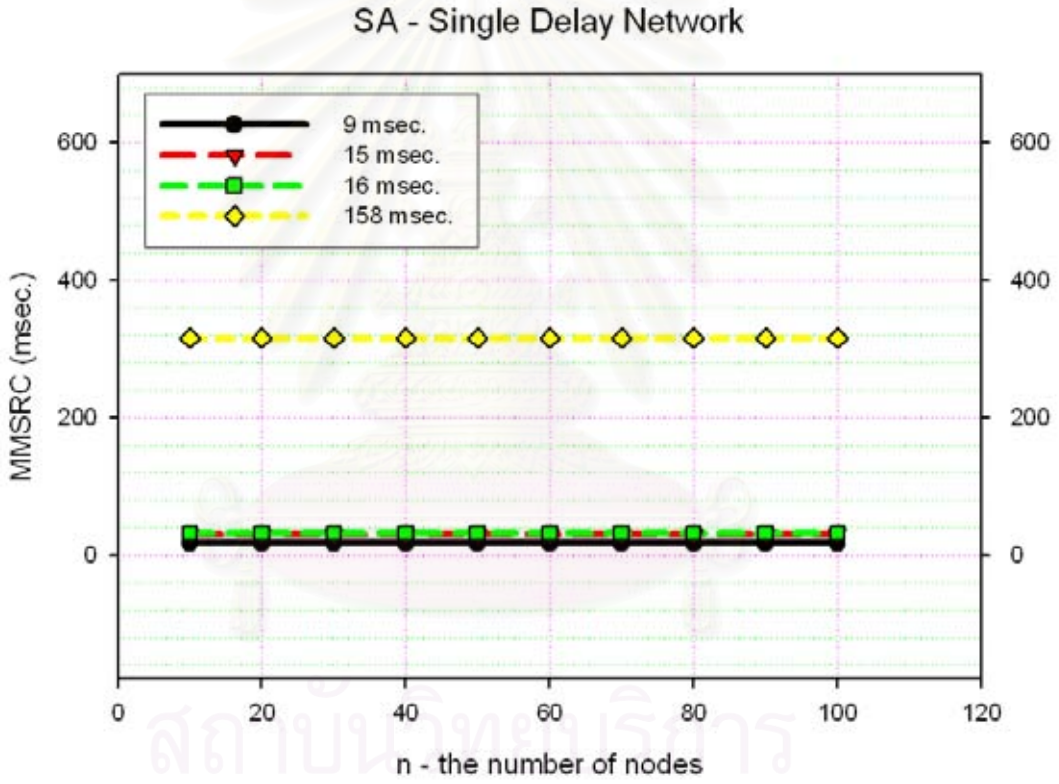
ตรวจสอบหาค่าที่ดีที่สุดในแต่ละค่าจำนวนเส้นทางที่ใช้ แล้วจึงนำผลลัพธ์ที่ดีที่สุดในแต่ละค่าจำนวนเส้นทางมาเปรียบเทียบกันอีกครั้งหนึ่งเพื่อให้ได้ค่าจำนวนเส้นทางที่ให้ผลลัพธ์ที่ดีที่สุด ดังนั้นจะเห็นได้ว่า ในขั้นตอนของการหาค่าคำตอบที่ดีที่สุดในแต่ละค่าจำนวนเส้นทางที่ใช้ คำตอบที่ได้อาจจะไม่ดีที่สุด (เพราะอาจเป็นไปได้ที่คำตอบที่ดีที่สุดอยู่ในจำนวนรูปแบบที่ไม่สามารถเกิดขึ้นได้) จึงทำให้มีความคลาดเคลื่อนของผลลัพธ์ที่ควรจะเป็น สำหรับผลลัพธ์ที่ได้จากอัลกอริทึมฮิวริสติกด้วยกันจะเห็นได้ว่า HRS-2 และ HRT-2 จะให้จำนวนเส้นทางที่ใช้ที่ต่ำทีสุด ตามมาด้วย HRS-3, HRT-3 และ HRS-1 HRT-1 (ซึ่งให้ผลลัพธ์ที่ใกล้เคียงกัน) โดยเหตุผลที่ HR-2 (HRS-2 และ HRT-2) ใช้จำนวนเส้นทางที่น้อยกว่า HR-3 (HRS-3 และ HRT-3) เนื่องจาก ในการรวมกลุ่มของโนดที่อยู่ใกล้กันของ HR-2 นั้นจะใช้วิธีการไล่จับคู่ของกลุ่มไปเรื่อย ถ้าคู่ใดให้ผลลัพธ์ที่ดี จะนำคู่นั้นรวมเป็นกลุ่มเดียวกัน แล้วนำกลุ่มที่ได้นั้นไปไล่จับกลุ่มอีกที ทำเช่นนี้เรื่อย ๆ จนไม่สามารถเกิดการพัฒนาคำตอบ (อธิบายไว้ในหัวข้อ 6.2.2) ซึ่งต่างจาก HR-3 ที่ใช้วิธีสร้างลิสต์ขึ้นมาใหม่ แล้วนำแต่ละคู่ที่อยู่ในลิสต์มองเป็นกลุ่มเดียวกัน ดังนั้นจะเห็นได้ว่า ในการรวมกลุ่มแบบ HR-3 นั้นกลุ่มที่อยู่ใกล้กันไม่จำเป็นต้องให้ผลลัพธ์โดยรวมที่ดีก็ได้ ผลลัพธ์ที่ได้อาจจะดีกว่าเดิม หรือด้อยกว่าเดิมได้ แต่ใน HR-2 จะรวมกันได้ก็ต่อเมื่อให้ค่า MMSRC ที่น้อยกว่าค่าพิกัด (ซึ่งค่าพิกัดคือค่า MMSRC ก่อนที่จะมีการรวมกลุ่ม) จึงส่งผลให้ HR-2 สามารถที่จะรวมกลุ่มของโนดได้มากกว่าดังผลลัพธ์ที่ได้ในรูปที่ 6.31

6.3.6 Time เมื่อ n = 10 – 100

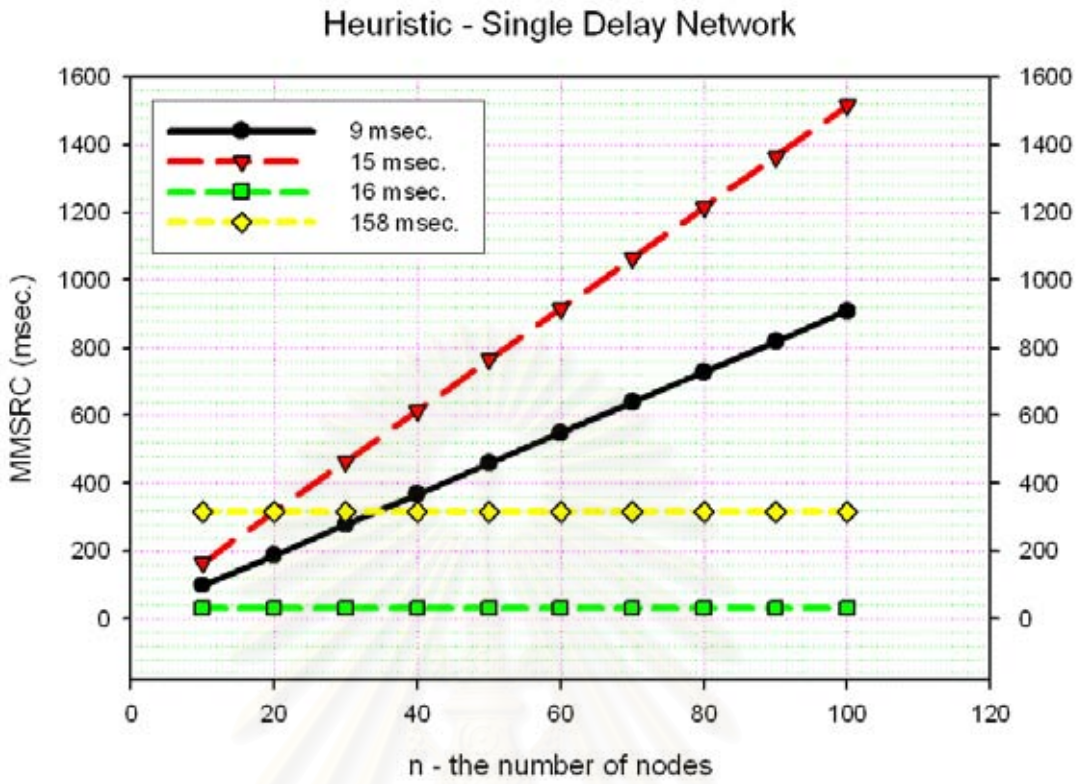
จากรูปที่ 6.32 จะเห็นได้ว่า อัลกอริทึม SA จะใช้เวลาในการประมวลผลที่มากที่สุด เนื่องจากอัลกอริทึม SA จะทำการหาค่าที่ดีที่สุดในแต่ละค่าจำนวนเส้นทางแล้วจึงนำค่าที่ดีที่สุดในแต่ละเส้นทางนั้นมาเปรียบเทียบกันอีกครั้งหนึ่ง ซึ่งต่างจากอัลกอริทึมฮิวริสติกที่มีการกำหนดจำนวนเส้นทางที่ตายตัวตั้งแต่เริ่มต้นโดยใช้อัลกอริทึมฮิวริสติก-1 ดังนั้นจึงเป็นผลทำให้อัลกอริทึม SA ใช้เวลาในการประมวลผลที่มากที่สุด สำหรับผลลัพธ์ที่ได้จากรูปที่ 6.33 จะเห็นได้ว่า เมื่อเปรียบเทียบเวลาประมวลผลชองอัลกอริทึมฮิวริสติกด้วยกัน อัลกอริทึมฮิวริสติกที่ใช้อัลกอริทึม Tabu Search จะใช้เวลาในการประมวลผลที่น้อยกว่า อัลกรอทึมฮิวริสติกที่ใช้อัลกอริทึม SA และนอกจากนี้ HRS-3 จะให้เวลาในการประมวลผลที่มากที่สุด รองลงมาคือ HRS-2, HRS-1, HRT-3, HRT-2 และ HRT-1 ซึ่งผลลัพธ์ที่ได้จะสอดคล้องกับผลลัพธ์ที่ได้จากกรณี n = 1 – 10 กล่าวคือ อัลกอริทึมฮิวริสติกแบบที่ 3 จะใช้เวลามากที่สุดเนื่องจากต้องมีการใช้อัลกอริทึม SA หรือ Tabu Search ประมวลผลทุกครั้งที่มีการรวมกลุ่มของโนด ในขณะที่อัลกอริทึม ฮิวริสติกแบบที่ 2 จะใช้การไล่จับกลุ่มของโนดที่เพิ่มเติมมากจากอัลกอริทึมแบบที่ 1 ซึ่งส่งผลให้อัลกอริทึมฮิวริสติกแบบที่ 1 ใช้เวลาในการประมวลผลที่ต่ำที่สุด

6.4 การวิเคราะห์สมรรถนะการทำงานเมื่อประมวลผลบนเนตเวิร์กจำลองที่มีการกำหนดค่าต้นทุนให้แก่ลิงก์ที่แตกต่างกัน

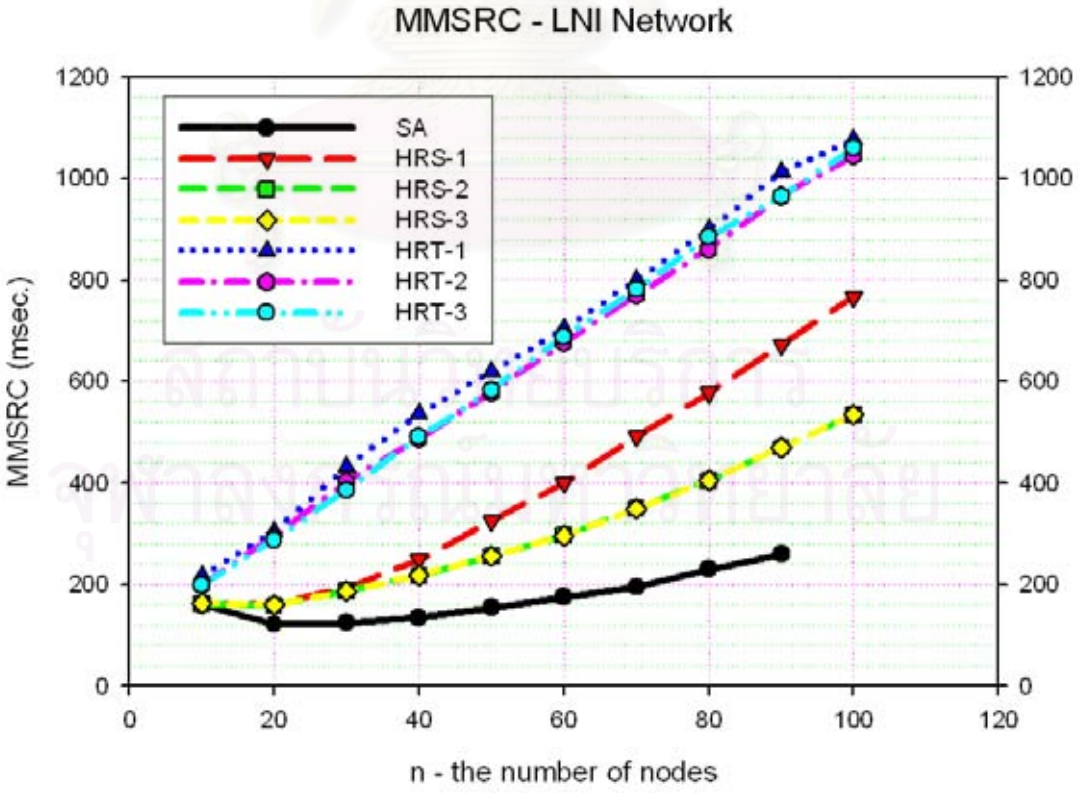
นอกจากการเปรียบเทียบประสิทธิภาพการทำงานของแต่ละอัลกอริทึมในเรื่องเวลาประมวลผล และค่า MMSRC แล้วยังมีการวิเคราะห์เชิงเปรียบเทียบสมรรถนะการทำงานของอัลกอริทึมต่างๆ เมื่อประมวลผลบนเนตเวิร์กที่มีลักษณะการกำหนดค่าต้นทุนที่แตกต่างกัน (โดยวิธีกำหนดค่าต้นทุนของเนตเวิร์กได้กล่าวอธิบายไว้ในหัวข้อ 6.5) โดยจะทำการวิเคราะห์ที่ค่า n = 10 – 100 และค่า MMSRC ที่ได้จะได้มาจากค่าเฉลี่ยของเมตริกซ์ต้นทุนของเนตเวิร์ก 100 ชุดในแต่ละค่า n ซึ่งผลลัพธ์ที่ได้แสดงไว้ดังรูปที่ 6.35 – รูปที่ 6.42



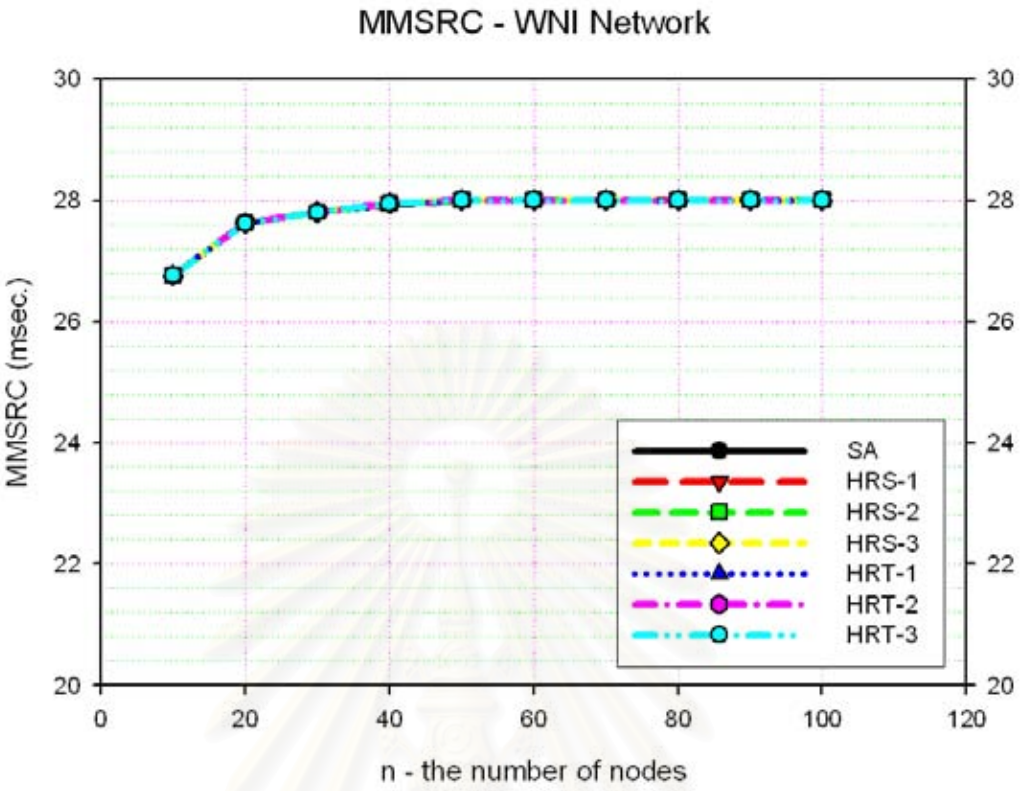
รูปที่ 6.35 ค่า MMSRC ที่ได้จากการประมวลผลบนเนตเวิร์ก
ที่ใช้ค่าต้นทุนลิงก์ค่าเดียวโดยใช้อัลกอริทึม SA



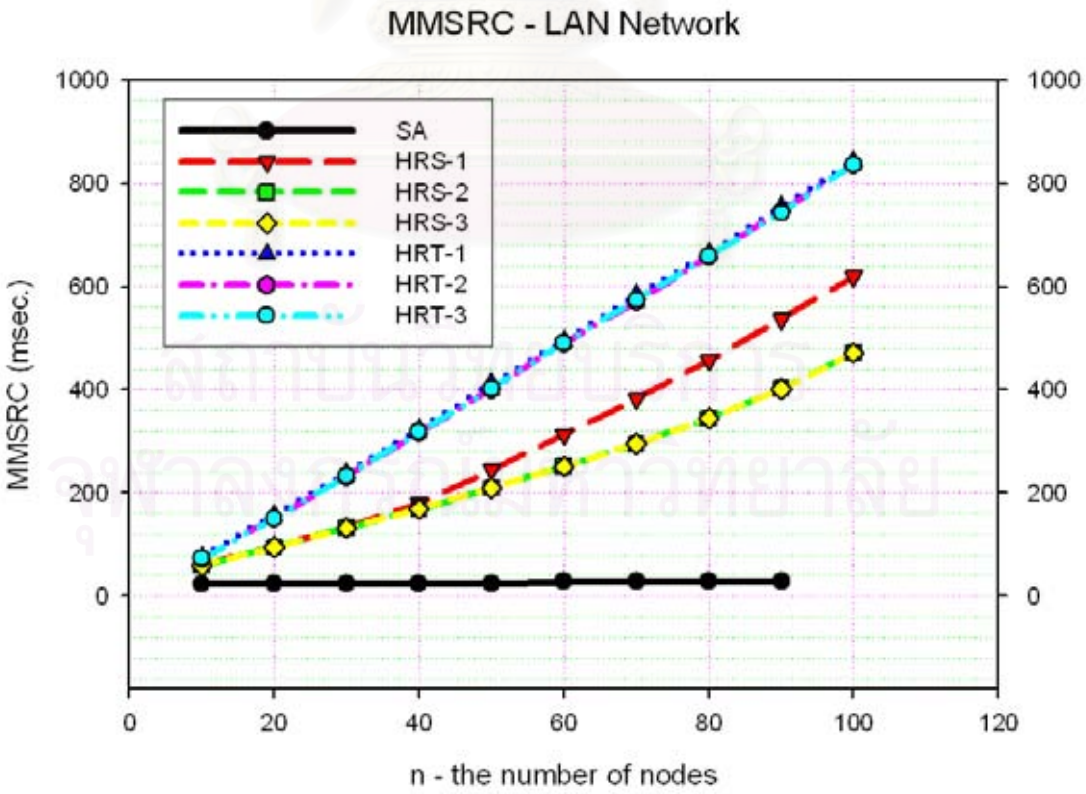
รูปที่ 6.36 ค่า MMSRC ที่ได้จากการประมวลผลบนเนตเวิร์ก
ที่ใช้ค่าต้นทุนลิงก์ค่าเดียวโดยใช้อัลกอริทึม ฮิวริสติก



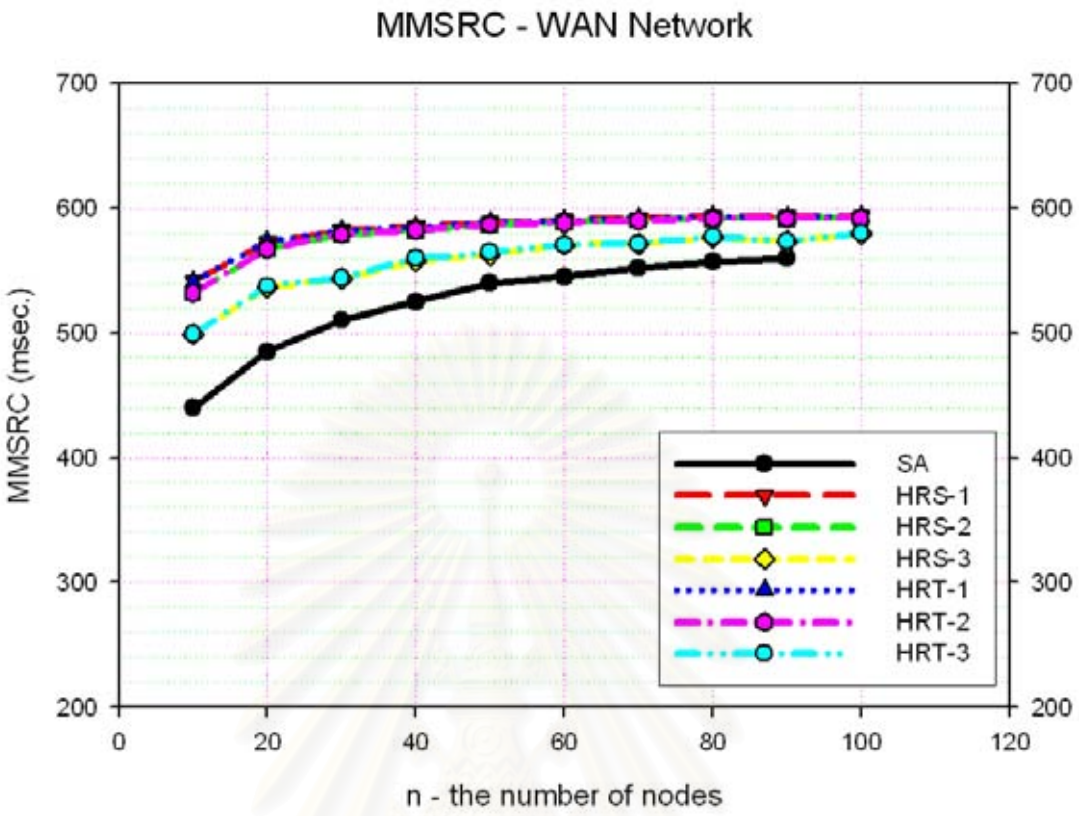
รูปที่ 6.37 ค่า MMSRC จากการประมวลผลบน LNI เนตเวิร์ก



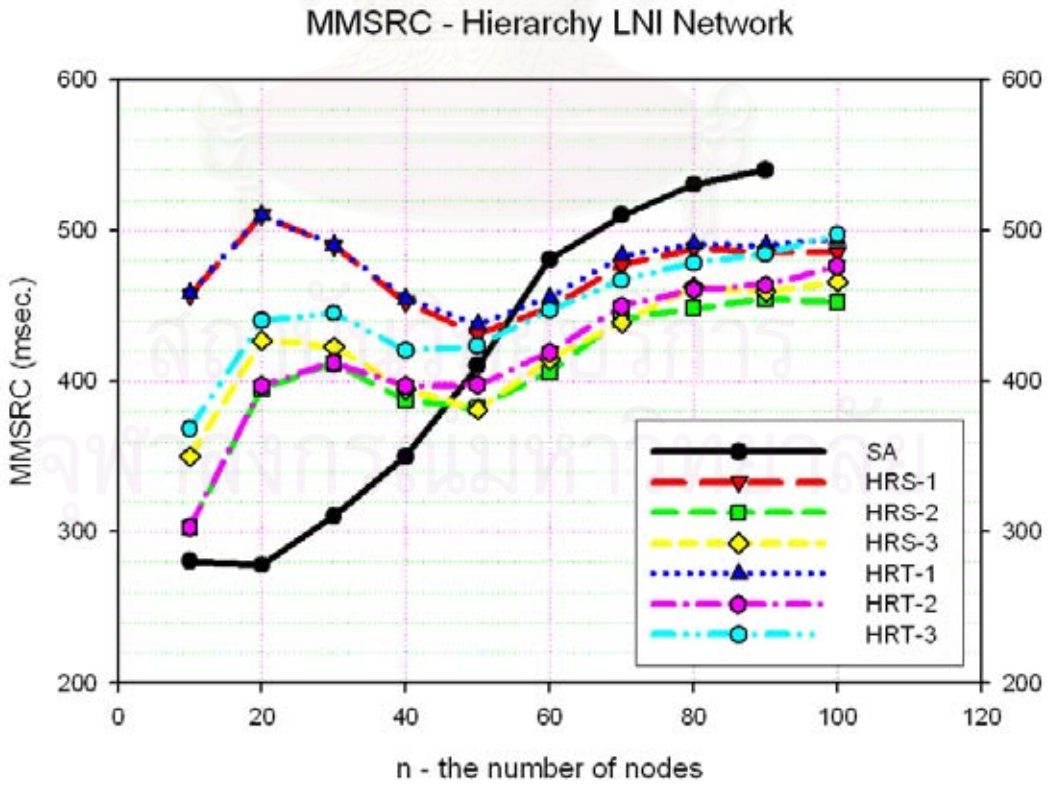
รูปที่ 6.38 ค่า MMSRC จากการประมวลผลบน WNI เนตเวิร์ก



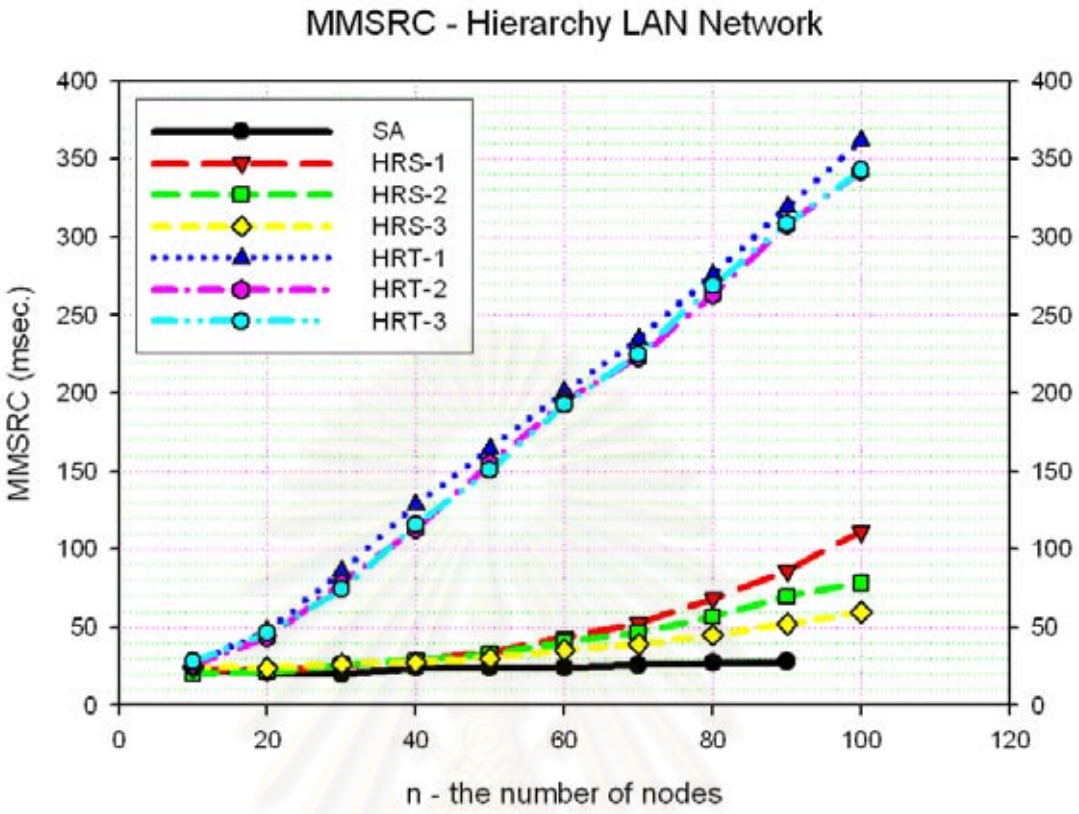
รูปที่ 6.39 ค่า MMSRC จากการประมวลผลบน L-Net เนตเวิร์ก



รูปที่ 6.40 ค่า MMSRC จากการประมวลผลบน W-Net เนตเวิร์ก



รูปที่ 6.41 ค่า MMSRC จากการประมวลผลบน Hierarchy LNI เนตเวิร์ก



รูปที่ 6.42 ค่า MMSRC จากการประมวลผลบน Hierarchy L-Net เนตเวิร์ก

6.4.1 ค่า MMSRC ที่ได้จากการประมวลผลบนเนตเวิร์กที่ใช้ค่าต้นทุนลิงก์ค่าเดียว

จากค่า MMSRC ที่ได้จากเนตเวิร์กที่ใช้ค่าต้นทุนค่าเดียวในรูปที่ 6.35 และรูปที่ 6.36 จะเห็นได้ว่า ในกรณีการประมวลผลโดยใช้อัลกอริทึม SA ค่า MMSRC ที่ได้จะมีค่าคงที่ตลอดทุกค่า n โดยค่าที่ได้จะเป็น 2 เท่าของค่าต้นทุนที่ใช้ในแต่ละเนตเวิร์ก ทั้งนี้เนื่องจาก ลักษณะของต้นทุนของลิงก์ในเนตเวิร์กทำให้ค่าต้นทุนที่ได้จากแต่ละเส้นทางเท่ากับ ค่าต้นทุนของลิงก์คูณกับจำนวนลิงก์ที่เชื่อมต่อแต่ละโนดไว้ ดังตัวอย่างที่ 6.2

ตัวอย่างที่ 6.2 *ตัวอย่างค่าต้นทุนของเส้นทางกรณี n = 10 ค่าต้นทุนลิงก์ 9เท่ากับมิลลิวินาที*

เส้นทางประกอบไปด้วย 1 โนด ค่าต้นทุนเท่ากับ 9 x 2 = 18
เส้นทางประกอบไปด้วย 2 โนด ค่าต้นทุนเท่ากับ 9 x 3 = 27
เส้นทางประกอบไปด้วย 3 โนด ค่าต้นทุนเท่ากับ 9 x 4 = 36
เส้นทางประกอบไปด้วย 4 โนด ค่าต้นทุนเท่ากับ 9 x 5 = 45
เส้นทางประกอบไปด้วย 5 โนด ค่าต้นทุนเท่ากับ 9 x 6 = 54
เส้นทางประกอบไปด้วย 6 โนด ค่าต้นทุนเท่ากับ 9 x 7 = 63

เส้นทางประกอบไปด้วย 7 โนด ค่าต้นทุนเท่ากับ 9 x 8 = 72
เส้นทางประกอบไปด้วย 8 โนด ค่าต้นทุนเท่ากับ 9 x 9 = 81
เส้นทางประกอบไปด้วย 9 โนด ค่าต้นทุนเท่ากับ 9 x 10 = 90
เส้นทางประกอบไปด้วย 10 โนด ค่าต้นทุนเท่ากับ 9 x 11 = 99

จากตัวอย่างที่ 6.2 จะเห็นได้ว่ากรณีเส้นทางที่ประกอบไปด้วย 1 โนดจะให้ค่าต้นทุนที่ต่ำที่สุด ซึ่งเป็นการกำหนดจำนวนเส้นทางเท่ากับจำนวนโนดนั่นเอง และเช่นเดียวกันกับกรณีเนตเวิร์กที่ใช้ค่าต้นทุน 15, 16 และ 158 มิลลิวินาที ที่ทุกช่วงค่า n ค่าต้นทุนที่ต่ำที่สุดคือต้นทุนที่ได้จากกรณีเส้นทางที่ประกอบไปด้วย 1 โนด ดังนั้นค่า MMSRC ที่ได้ในรูปที่ 6.35 จึงเป็น 2 เท่าของค่าต้นทุนที่ใช้ในเนตเวิร์ก

สำหรับกรณีประมวลผลโดยใช้อัลกอริทึมฮิวริสติก (ในที่นี้ผลลัพธ์ที่ได้จากอัลกอริทึม HRS-1, HRS-2, HRS-3, HRT-1, HRT-2 และ HRT-3 ให้ค่าเท่ากัน) จะให้ค่า MMSRC ที่แตกต่างจากค่า MMSRC ที่ได้จากรูปที่ 6.36 โดยที่ค่าต้นทุน 9 และ 15 มิลลิวินาที ค่า MMSRC ที่ได้จะมีลักษณะเป็นสัดส่วนกับค่า n ทั้งนี้เนื่องจากกรณี 9 และ 15 มิลลิวินาที อัลกอริทึมฮิวริสติกจะมองเนตเวิร์กที่ประกอบไปด้วยโนดอยู่ใกล้กันหมด ดังนั้นการกำหนดเส้นทางที่ได้จากอัลกอริทึมฮิวริสติกจึงเป็นลักษณะใช้เส้นทางเดียว จึงส่งผลให้ที่แต่ละค่า n ค่า MMSRC ที่ได้จะเท่ากับ ค่าต้นทุนคูณกับจำนวนลิงก์ (n+1) ทำให้ค่า MMSRC ที่ได้เป็นสัดส่วนแบบเชิงเส้นกับค่า n สำหรับกรณีค่าต้นทุนเท่ากับ 16 และ 158 มิลลิวินาที ค่าต้นทุนทั้งคู่เป็นค่าต้นทุนแวน ดังนั้นอัลกอริทึมฮิวริสติกจะมองเนตเวิร์กที่ใช้ค่าต้นทุน 2 ค่านี้เป็นเนตเวิร์กที่ประกอบไปด้วยโนดที่อยู่ห่างกัน ดังนั้นอัลกอริทึมฮิวริสติกจึงกำหนดเส้นทางในลักษณะใช้จำนวนเส้นทางเท่ากับจำนวนโนดเหมือนกับผลที่ได้จากอัลกอริทึม SA ซึ่งจะให้ค่าต้นทุนที่ต่ำที่สุดเท่ากับ 2 เท่าค่าต้นทุนที่ใช้ ค่า MMSRC ที่ได้จึงมีค่าคงที่ตลอดช่วงค่า n

จากค่า MMSRC ที่ได้ในรูปที่ 6.36 ชี้ให้เห็นถึงข้อด้อยของอัลกอริทึมฮิวริสติกที่ใช้กับเนตเวิร์กที่มีค่าต้นทุนค่าเดียว ซึ่งเป็นข้อด้อยในส่วนของขั้นตอนการทำงานทางตรรกะ แต่อย่างไรก็ตามลักษณะเนตเวิร์กเช่นนี้ไม่เป็นจริงในทางปฏิบัติ ดังนั้นลักษณะตัวอย่างของเนตเวิร์กแบบนี้กล่าวไว้เพื่อแสดงให้เห็นถึงขั้นตอนการทำงานของอัลกอริทึมฮิวริสติกที่ชัดเจนยิ่งขึ้น

6.4.2 ค่า MMSRC ที่ได้จากการประมวลผลบนเนตเวิร์กแบบ LNI

จากรูปที่ 6.37 จะเห็นได้ว่า อัลกอริทึม SA จะให้ค่า MSMRC ที่ต่ำกว่าค่า MMSRC ที่ได้จากอัลกอริทึมฮิวริสติกทั้ง 6 แบบ ทั้งนี้เนื่องจากการกำหนดค่าต้นทุนของ LNI Network ทำให้

อัลกอริทึมฮิวริสติกมองเนตเวิร์กประกอบไปด้วยกลุ่มของเนตเวิร์กเพียงกลุ่มเดียว เพราะทุกโนดยกเว้นโนดเริ่มต้นเชื่อมกันด้วยลิงก์ที่เป็นค่าต้นทุนแลน ดังนั้นอัลกอริทึมฮิวริสติกจะใช้เส้นทางเดียวเหมือนกับกรณีเนตเวิร์กที่ใช้ค่าต้นทุนเดียว 9 และ 15 มิลลิวินาที ซึ่งต่างจากอัลกอริทึม SA ที่ใช้การคำนวณทดลองใช้เส้นทางตั้งแต่ 1 เส้นทางไปจนถึงจำนวนเส้นทางเท่ากับจำนวนโนด ดังนั้นอัลกอริทึม SA จึงมีโอกาสที่จะใช้เส้นทางมากกว่า 1 เส้นทางได้ จึงส่งผลให้ค่า MMSRC ที่ได้ต่ำกว่าค่า MMSRC ที่ได้จากอัลกอริทึมฮิวริสติก แต่อย่างไรก็ตามในการใช้งานจริง โอกาสที่จะมีเนตเวิร์กที่ประกอบไปด้วยโนดจำนวนมาก ๆ และเชื่อมต่อกันด้วยค่าต้นทุนแลน แทบจะเป็นไปไม่ได้ ดังนั้นเพื่อความเหมาะสมจึงคิดกรณีเนตเวิร์กแบบ LNI ที่มีลักษณะการกำหนดแบบระดับชั้น (Hierarchy) ซึ่งจะกล่าวไว้ในหัวข้อ 6.4.6 ต่อไป

6.4.3 ค่า MMSRC ที่ได้จากการประมวลผลบนเนตเวิร์กแบบ WNI

จากรูปที่ 6.38 จะเห็นได้ว่า ค่า MMSRC ที่ได้จากอัลกอริทึม SA และอักลอริทึมฮิวริสติกจะให้ค่าเท่ากันหมด ทั้งนี้เนื่องจาก การกำหนดค่าต้นทุนในเนตเวิร์กแบบ WNI จะส่งผลให้อัลกอริทึมฮิวริสติกประกอบไปด้วยโนดที่อยู่ห่างกัน ดังนั้นจึงมีการกำหนดเส้นทางที่ใช้จำนวนเส้นทางเท่ากับจำนวนโนด ผลลัพธ์ที่ได้จึงเท่ากับค่าที่มากที่สุดของเส้นทางที่เชื่อมต่อระหว่างโนดใด ๆ กับโนดเริ่มต้น

6.4.4 ค่า MMSRC ที่ได้จากการประมวลผลบนเนตเวิร์กแบบ L-Net

สำหรับค่า MMSRC ที่ได้จากเนตเวิร์กแบบ L-Net จะสังเกตได้ว่า จะมีลักษณะคล้ายกับผลลัพธ์ที่ได้ในรูปที่ 6.37 กล่าวคือ อัลกอริทึมฮิวริสติกจะให้ค่า MMSRC ที่เป็นสัดส่วนแบบเชิงเส้นกับค่า n แต่อัลกอริทึม SA จะให้ค่าที่ค่อนข้างคงที่ ทั้งนี้เนื่องจากที่กล่าวอธิบายไว้ในหัวข้อ 6.4.1 เนื่องจากการกำหนดค่าต้นทุนแบบ L-Net โนดทุกโนดจะเชื่อมต่อกันด้วยค่าต้นทุนแลนทั้งหมด ทำให้อัลกอริทึมฮิวริสติกมีการกำหนดจำนวนเส้นทางเพียงเส้นทางเดียว ซึ่งแตกต่างจากอัลกอริทึม SA ที่มีการตรวจสอบทดลองใช้เส้นทางตั้งแต่ 1 เส้นทางไปจนถึงจำนวนเส้นทางเท่ากับจำนวนโนด และกรณีนี้ทำให้การใช้จำนวนเส้นทางเท่ากับจำนวนโนด ให้ค่า MMSRC ที่ต่ำกว่าการใช้เส้นทางเพียงเส้นทางเดียว

6.4.5 ค่า MMSRC ที่ได้จากการประมวลผลบนเนตเวิร์กแบบ W-Net

จากกำหนดค่าต้นทุนของเนตเวิร์ก W-Net จะเห็นได้ว่า ค่า MMSRC ที่ได้จากแต่ละอัลกอริทึมจะมีค่าใกล้เคียงกันมาก และจะใกล้เคียงกันมากยิ่งขึ้นที่ค่า n ค่าสูง ๆ ทั้งนี้เนื่องจาก

การกำหนดค่าต้นทุนแบบ W-Net ทำให้อัลกอริทึมฮิวริสติกมองเนตเวิร์กประกอบไปด้วยโนดที่ไม่อยู่ใกล้กันเลย ดังนั้นจึงใช้จำนวนเส้นทางเท่ากับจำนวนโนด แต่สำหรับอัลกอริทึม SA จะให้ค่า MMSRC ที่ต่ำกว่าทั้งนี้อาจเนื่องมาจาก อาจเกิดกรณีบางเส้นทางผ่านโนดมากกว่า 1 โนดได้ ดังนั้นจึงส่งผลให้ค่า MMSRC ที่ได้มีค่าลดต่ำลง

6.4.6 ค่า MMSRC ที่ได้จากการประมวลผลบนเนตเวิร์กแบบ HLNI

จากที่ได้กล่าวไว้ในหัวข้อ 6.4.2 เพื่อความสอดคล้องกับการใช้งานจริง จึงมีการกำหนดเนตเวิร์กแบบ HLNI (วิธีการกำหนดค่าต้นทุนอธิบายไว้ในบทที่ 5) จากรูปที่ 6.41 จะเห็นได้ว่า ค่า MMSRC ที่ได้จะแตกต่างจากผลลัพธ์ในรูปที่ 6.347 โดยสิ้นเชิง แต่จะใกล้เคียงกับผลลัพธ์ที่ได้ในรูปที่ 6.30 โดยอัลกอริทึม MMSRC จะให้ค่า ที่ดีในช่วงค่า n ต่ำ ๆ แต่จะให้ค่าที่แย่กว่าอัลกอริทึมฮิวริสติกที่ค่า n มีค่าสูง ๆ เมื่อพิจารณาค่า MMSRC ที่ได้จากอัลกอริทึมฮิวริสติกด้วยกัน จะเห็นได้ว่า HRS-1 และ HRT-1 จะให้ค่า MMSRC ที่มากที่สุด สอดคล้องกับผลลัพธ์ที่ได้ในรูปที่ 6.30 แต่ผลลัพธ์ที่ได้จาก HR-2 (HRS-2 และ HRT-2) จะให้ค่า MMSRC ที่ต่ำกว่า HR-3 ทั้งนี้คาดว่าเป็นผลเนื่องมาจากส่วนของการรวมเส้นทางเข้าด้วยกัน ซึ่งในอัลกอริทึม HR-2 จะใช้วิธีลองจับคู่เส้นทางแบบ หัวท้ายชนกัน ถ้าให้ค่า MMSRC ที่ดีขึ้น ก็รวมเข้าด้วยกัน ทำทุกเส้นทางที่มี ซึ่งต่างจากอัลกอริทึม HR-3 ที่ใช้วิธีตรวจสอบค่าเวลาหน่วงว่า เส้นทางคู่ใด มีค่าต้นทุนลิงก์ที่เชื่อมต่อระหว่างโนดในแต่ละเส้นทางที่สั้นที่สุด ก็จะรวมกัน แล้วนำไปปรับปรุงคำตอบใหม่โดยใช้อัลกอริทึม SA หรือ Tabu ทำจนครบทุกคู่ของเส้นทางที่มี ดังนั้นจากแนวความคิดการรวมกลุ่มของทั้ง 2 แบบ การรวมกลุ่มแบบ HR-3 ควรจะให้ค่า MMSRC ที่ดีกว่า แต่อย่างไรก็ตามการปรับปรุงคำตอบโดยใช้ SA และ Tabu ต่างก็ขึ้นกับตัวแปรสุ่ม จึงอาจเป็นได้ว่า ในเนตเวิร์กบางรูปแบบ การปรับปรุงคำตอบที่ได้ ให้ค่าที่ไม่ดีเท่าที่ควร จึงส่งผลให้การรวมกลุ่มที่ได้ให้ค่า MMSRC ที่ด้อยกว่า HR-2 ดังนั้นจึงไม่สามารถบอกได้ว่าระหว่างอัลกอริทึม HR-2 และ HR-3 อัลกอริทึมใด ดีที่สุด แต่บอกได้เพียงว่า อัลกอริทึมใด มีแนวโน้มที่ให้ค่า MMSRC ที่ดีกว่า แต่อย่างไรก็ตามผลลัพธ์ที่ได้จากอัลกอริทึมทั้ง 2 อัลกอริทึม ให้ค่า MMSRC ที่ใกล้เคียงกัน

6.4.7 ค่า MMSRC ที่ได้จากการประมวลผลบนเนตเวิร์กแบบ HL-Net

จากรูปที่ 6.42 ผลลัพธ์ที่ได้จะสังเกตได้ว่า ยังคงมีลักษณะคล้ายกับผลลัพธ์ที่ได้จากรูปที่ 6.39 โดยอัลกอริทึมฮิวริสติกยังคงให้ค่า MMSRC ที่เป็นสัดส่วนกับค่า n แต่ให้ค่าที่ลดต่ำลงมากมากเมื่อเทียบกับรูปที่ 6.39 ทั้งนี้เนื่องจากการกำหนดแบบระดับชั้น ส่งผลให้อัลกอริทึมฮิวริสติกมองเนตเวิร์กประกอบไปด้วยกลุ่มของโนด ดังนั้นการกำหนดเส้นทางจะไม่ใช่การกำหนดโดยใช้เส้นทางเดียวอีกต่อไป แต่เป็นการกำหนดเส้นทางที่มีจำนวนเส้นทางที่สอดคล้องกับจำนวนกลุ่ม

ของโนดที่มีในเนตเวิร์ก ดังนั้นค่า MMSRC ที่ได้จึงมีค่าที่ต่ำลงมา และใกล้เคียงกับค่า MMSRC ที่ได้จากอัลกอริทึม SA

6.5 สรุป

จากการวิเคราะห์ที่กล่าวไว้ทั้งหมดนี้ทำให้สรุปได้ว่า อัลกอริทึม BF จะให้คำตอบที่ดีที่สุด แต่ใช้เวลาในการประมวลผลที่มากที่สุด ซึ่งประโยชน์ของอัลกอริทึม BF คือใช้เป็นเกณฑ์ในการเปรียบเทียบผลลัพธ์ที่ได้จากคำตอบของอัลกอริทึมอื่นว่าให้ผลลัพธ์ที่ใกล้เคียงกับคำตอบที่ดีที่สุดเพียงใด สำหรับอัลกอริทึม SA จะให้คำตอบที่ใกล้เคียงกับอัลกอริทึม BF และยังใช้เวลาที่น้อยกว่าอัลกอริทึม BF พอสมควร แต่ยังไม่น้อยเพียงพอที่จะสามารถนำมาใช้งานได้จริง ดังนั้นจึงต้องนำอัลกอริทึม SA ไปประยุกต์ใช้กับอัลกอริทึมอื่น ซึ่งได้คิดอัลกอริทึมฮิวริสติกขึ้นมา 3 แบบ โดยแต่ละแบบจะเลือกใช้อัลกอริทึม SA หรืออัลกอริทึม Tabu Search มาใช้ในการปรับปรุงคำตอบ รวมเป็น 6 วิธี คือ HRS-1, HRS-2, HRS-3, HRT-1, HRT-2 และ HRT-3 ซึ่งผลลัพธ์แสดงให้เห็นว่า HRS-3 จะให้คำตอบที่เหมาะสมที่สุด แต่ในส่วนของเวลาประมวลผลอาจจะไม่ดีไปกว่าอัลกอริทึมฮิวริสติกตัวอื่น แต่เมื่อพิจารณาในความเป็นจริง ถือว่าใกล้เคียงกันมาก นอกจากนี้จำนวนเส้นทางที่ใช้ยังได้ผลลัพธ์ที่ใกล้เคียงกับอัลกอริทึมอื่นด้วยเช่นเดียวกัน แต่จะมากกว่า HRS-2 และ HRT-2 อยู่เพียงเล็กน้อย

สำหรับในส่วนของการจัดกลุ่มของอัลกอริทึมฮิวริสติก เพื่อแสดงให้เห็นถึงประสิทธิภาพการทำงานเมื่อเปรียบเทียบกับอัลกอริทึมอื่น ๆ ที่มีในปัจจุบัน จึงนำอัลกอริทึม cGA มาทำการประยุกต์ใช้งานการกำหนดเส้นทาง คืออัลกอริทึม McGA ซึ่งผลลัพธ์แสดงให้เห็นว่า ค่า MMSRC ที่ได้จากอัลกอริทึม McGA ให้ค่าที่ดีกว่าอัลกอริทึมฮิวริสติก แต่ใช้เวลาประมวลผลที่ค่อนข้างมากโดยเป็นรองจากอัลกอริทึม BF ซึ่งใช้เวลาประมวลผลมากที่สุด ดังนั้นจึงอาจจะมองได้ว่าอัลกอริทึมฮิวริสติกอาจจะมีข้อดีเพียงเวลาประมวลผลที่ต่ำเพียงประการเดียว ดังนั้นเพื่อแสดงให้เห็นข้อเด่นในส่วนของค่า MMSRC จึงมีการนำอัลกอริทึมกำหนดเส้นทางอิสระนำมาเปรียบเทียบ โดยผลลัพธ์ที่ได้จากอัลกอริทึมกำหนดเส้นทางแบบอิสระจะให้ค่า MMSRC ที่สูงมาก ดังนั้นจึงเป็นเกณฑ์ในการเปรียบเทียบที่เห็นได้ชัดเจน ที่แสดงให้เห็นว่า แม้อัลกอริทึมฮิวริสติกจะให้ค่า MMSRC ที่ด้อยกว่าอัลกอริทึมชนิดอื่น ๆ แต่ความแตกต่างที่เกิดขึ้นเมื่อเปรียบเทียบกับการกำหนดแบบอิสระแล้วถือว่าใกล้เคียงกัน

จากการทดลองประมวลผลบนเนตเวิร์กชนิดอื่นที่แตกต่างกัน จะเห็นได้ว่า ในบางกรณี อัลกอริทึมฮิวริสติกจะให้ค่า MMSRC ที่ด้อยกว่า SA ที่ค่อนข้างมาก รวมไปถึงข้อด้อยทางตรรกะเมื่อประมวลผลบนเนตเวิร์กที่มีการกำหนดค่าต้นทุนแบบค่าเดียว อย่างไรก็ตามเมื่อพิจารณา

เนตเวิร์กที่สอดคล้องกับการใช้งานจริง อย่างเช่น เนตเวิร์ก WAN, HLNI, HL-Net จะเห็นได้ว่า ค่า MMSRC ที่ได้จากอัลกอริทึมฮิวริสติกจะให้ผลลัพธ์ที่ใกล้เคียงกับอัลกอริทึม SA และเมื่อพิจารณาอัลกอริทึมฮิวริสติกด้วยกัน อัลกอิรทึม HRS-3 ยังคงมีแนวโน้มที่ให้ผลลัพธ์ที่ดีที่สุด ดังนั้นจากการวิเคราะห์ที่ผ่านมาทั้งหมดในส่วนของเวลาประมวลผล ค่า MMSRC จำนวนเส้นทางที่ใช้ ทำให้สรุปได้ว่า อัลกอริทึม HRS-3 เหมาะสมที่สุดสำหรับการนำไปใช้ในการกำหนดเส้นทางให้กับโมบายล์เอเจนต์

บทที่ 7

การวิเคราะห์การกำหนดเส้นทางกรณีย้อนเส้นทางเดิมได้

7.1 กล่าวนำ

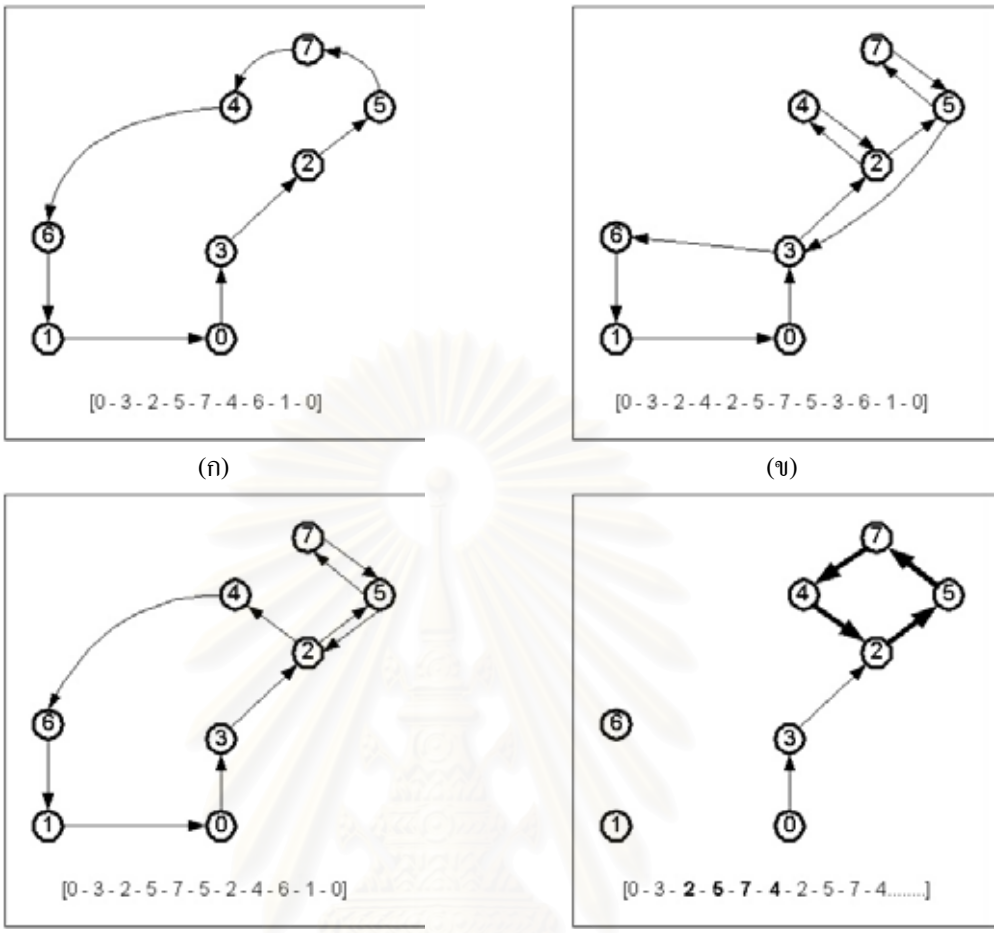
จากการศึกษาอัลกอริทึมต่าง ๆ รวมไปถึงการวิเคราะห์ผลที่ได้จากอัลกอริทึมกำหนดเส้นทางที่กล่าวไว้ในบทที่ 4 ถึงบทที่ 6 อัลกอริทึมทั้งหมดนั้นมีข้อกำหนดของการกำหนดเส้นทางที่เป็นลักษณะไม่มีการย้อนกลับทางเส้นเดิม ตามคุณลักษณะของ *ปัญหาการเดินทางของเซลส์แมน* (Traveling Salesman Problem: TSP) จึงอาจเกิดความคิดว่า การกำหนดเส้นทางให้กับโมบายล์เอเจนต์จำเป็นต้องมีคุณลักษณะ *ไม่มีการย้อนกลับเส้นทางเดิม* เหมือนกับปัญหาการเดินทางของเซลส์แมนหรือไม่ ดังนั้นเนื้อหาในบทนี้จะเป็นการวิเคราะห์เพิ่มเติมกรณีการกำหนดเส้นทางให้กับโมบายล์เอเจนต์แบบอนุญาตให้ผ่านเส้นทางเดิมได้ โดยจะแบ่งเนื้อหาออกเป็น 2 ส่วนดังนี้

ก.) วิธีกำหนดเส้นทางให้กับโมบายล์เอเจนต์แบบอนุญาตให้ผ่านเส้นทางเดิมได้

ข.) การวิเคราะห์ผลลัพธ์ที่ได้จากการกำหนดเส้นทางแบบอนุญาตให้ผ่านเส้นทางเดิมได้

7.2 วิธีกำหนดเส้นทางให้กับโมบายล์เอเจนต์แบบอนุญาตให้ผ่านเส้นทางเดิมได้

จากอัลกอริทึมกำหนดเส้นทางในบทที่ 4 และบทที่ 5 จะเห็นได้ว่า กรณีที่ไม่มีการย้อนกลับเส้นทางเดิม ลักษณะของเส้นทางจะมีความยาวของเส้นทางคงที่เท่ากับจำนวนโนด ตัวอย่างเช่น สมมติให้จำนวนโนดเท่ากับ 7 ลักษณะเส้นทางอาจจะเป็น [3 – 2 – 5 – 7 – 4 – 6 – 1] โดยจะมีค่าความยาวของเส้นทางเท่ากับ 7 แต่เมื่อพิจารณากรณีอนุญาตให้ผ่านเส้นทางเดิมได้ ลักษณะเส้นทางอาจจะเป็น [3 – 2 – 4 – 2 – 5 – 7 – 5 – 3 – 6 – 1] ซึ่งมีความยาวของเส้นทางเท่ากับ 10 หรืออาจจะเป็น [3 – 2 – 5 – 7 – 5 – 2 – 4 – 6 – 1] ซึ่งมีความยาวของเส้นทางเท่ากับ 9 จะเห็นได้ว่ากรณีจำนวนโนดเท่ากับ 7 ความยาวของเส้นทางที่เป็นไปได้จะมีค่าไม่คงที่ และไม่สามารถคาดเดาได้เนื่องจากอาจเกิดกรณีวนซ้ำ (Loop) ซึ่งจะทำให้ความยาวของเส้นทางยิ่งยาวขึ้นไปอีก ดังรูปที่ 7.1



รูปที่ 7.1 ตัวอย่างรูปแบบเส้นทางที่สามารถย้อนเส้นทางเดิมได้

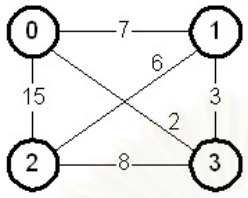
จากรูปที่ 7.1 จะเห็นได้ว่า อัลกอริทึมกำหนดเส้นทางแบบอนุญาตให้ผ่านเส้นทางเดิมได้จะมีลักษณะของคำตอบที่มีความยาวของเส้นทางที่ไม่แน่นอน จะทำให้เกิดปัญหาทีมีความซับซ้อนมากขึ้น ดังนั้นจึงจำเป็นต้องมีการพัฒนาเทคนิคที่ใช้ในการแก้ปัญหาที่มีลักษณะเฉพาะนี้ โดยนำเมตริกซ์ต้นทุนที่มีอยู่แล้วมาทำการหาค่าต้นทุนของลิงก์ที่เชื่อมแต่ละคู่โนดที่มีค่าต่ำที่สุด แล้วเก็บค่าต้นทุนที่ได้เป็นเมตริกซ์ชุดใหม่เรียกว่า รีดิวซ์เมตริกซ์ (Reduced Matrix) แล้วนำรีดิวซ์เมตริกซ์ที่ได้ไปผ่านอัลกอริทึมกำหนดเส้นทางที่กล่าวไว้ในบทที่ 4 และบทที่ 5

การทำรีดิวซ์เมตริกซ์มีแนวคิดว่า ค่าต้นทุนของลิงก์ที่เชื่อมต่อระหว่างคู่โนดใด ๆ อาจได้มาจากค่าต้นทุนในเมตริกซ์ต้นทุน หรือเป็นผลรวมของค่าต้นทุนของลิงก์ที่เชื่อมต่อผ่านโนดอื่น แล้วเก็บค่าที่ต่ำที่สุดไว้ในรีดิวซ์เมตริกซ์ เพื่อเป็นการทำความเข้าใจการทำงานจะยกตัวอย่างการทำรีดิวซ์เมตริกซ์ ดังตัวอย่างที่ 7.1

ตัวอย่างที่ 7.1 *ตัวอย่างการทำรีดิวซ์เมตริกซ์*

กำหนดให้จำนวนโนดเท่ากับ 3

สมมติให้ค่าต้นทุนที่เชื่อมต่อแต่ละคู่โนดใด ๆ และค่าเมตริกซ์ต้นทุน แสดงไว้ดังรูปที่ 7.2



$$\begin{bmatrix} 0 & 7 & 15 & 2 \\ 7 & 0 & 6 & 3 \\ 15 & 6 & 0 & 8 \\ 2 & 3 & 8 & 0 \end{bmatrix}$$

รูปที่ 7.2 ตัวอย่างเนตเวิร์กกรณี n = 3 และค่าเมตริกซ์ต้นทุน

จากรูปที่ 7.2 เมื่อพิจารณาการหาค่าต้นทุนระหว่างคู่โนดใด ๆ โดยผ่าน 1 โนดจะได้ค่าต้นทุนดังนี้

คู่โนด 0 – 1

[0 – 2 – 1] ค่าต้นทุนเท่ากับ 15 + 6 = 21 มิลลิวินาที

[0 – 3 – 1] ค่าต้นทุนเท่ากับ 2 + 3 = 5 มิลลิวินาที

คู่โนด 0 – 2

[0 – 1 – 2] ค่าต้นทุนเท่ากับ 7 + 6 = 13 มิลลิวินาที

[0 – 3 – 2] ค่าต้นทุนเท่ากับ 2 + 8 = 10 มิลลิวินาที

คู่โนด 0 – 3

[0 – 1 – 3] ค่าต้นทุนเท่ากับ 7 + 3 = 10 มิลลิวินาที

[0 – 2 – 3] ค่าต้นทุนเท่ากับ 15 + 8 = 23 มิลลิวินาที

คู่โนด 1 – 2

[1 – 0 – 2] ค่าต้นทุนเท่ากับ 7 + 15 = 22 มิลลิวินาที

[1 – 3 – 2] ค่าต้นทุนเท่ากับ 3 + 8 = 11 มิลลิวินาที

คู่โนด 1 – 3

[1 – 0 – 3] ค่าต้นทุนเท่ากับ 7 + 2 = 9 มิลลิวินาที

[1 – 2 – 3] ค่าต้นทุนเท่ากับ 6 + 8 = 14 มิลลิวินาที

คู่โนด 2 – 3

[2 – 0 – 3] ค่าต้นทุนเท่ากับ 15 + 2 = 17 มิลลิวินาที

[2 – 1 – 3] ค่าต้นทุนเท่ากับ 6 + 3 = 9 มิลลิวินาที

สำหรับคู่โนด 1 – 0, 2 – 0, 2 – 1, 3 – 0, 3 – 1, 3 – 2 จะมีค่าเท่ากับค่าต้นทุนที่ได้จากคู่โนด 0 – 1, 0 – 2, 1 – 2, 0 – 3, 1 – 3, 2 – 3 ตามลำดับ

เมื่อพิจารณาค่าต้นทุนที่ต่ำที่สุดที่ได้จากเส้นทางที่ผ่าน 1 โนดกับค่าต้นทุนในเมตริกซ์ต้นทุนจะได้รีดิวซ์เมตริกซ์ดังรูปที่ 7.3

$$\begin{bmatrix} 0 & 5 & 10 & 2 \\ 5 & 0 & 6 & 3 \\ 10 & 6 & 0 & 8 \\ 2 & 3 & 8 & 0 \end{bmatrix}$$

รูปที่ 7.3 รีดิวซ์เมตริกซ์กรณีผ่าน 1 โนด

และเช่นเดียวกันกรณีพิจารณาค่าต้นทุนที่ผ่าน 2 โนดจะได้ค่าเมตริกซ์ต้นทุนดังรูปที่ 7.4

$$\begin{bmatrix} 0 & 7 & 11 & 2 \\ 7 & 0 & 6 & 3 \\ 11 & 6 & 0 & 8 \\ 2 & 3 & 8 & 0 \end{bmatrix}$$

รูปที่ 7.4 รีดิวซ์เมตริกซ์กรณีผ่าน 2 โนด

สำหรับการนำรีดิวซ์เมตริกซ์ไปใช้งาน สามารถนำไปใช้ได้ 2 วิธีคือ การนำค่าต้นทุนในรีดิวซ์เมตริกซ์ไปใช้ในการคำนวณค่า MMSRC ของคำตอบ (เส้นทาง) ที่มีอยู่แล้วจากอัลกอริทึมต่าง ๆ โดยจะเรียกวิธีนี้ว่า *วิธีแทนค่า* และอีกวิธีหนึ่งคือการนำรีดิวซ์เมตริกซ์ไปประมวลผลใหม่โดยใช้อัลกอริทึมกำหนดเส้นทางที่มีอยู่ ซึ่งจะเรียกวิธีนี้ว่า *วิธีประยุกต์* ซึ่งการใช้งานทั้ง 2 วิธีอธิบายไว้ในตัวอย่างที่ 7.2

ตัวอย่างที่ 7.2 *ตัวอย่างการนำรีดิวซ์เมตริกซ์ไปใช้งานโดยใช้อัลกอริทึม BF*

กำหนดให้ผลลัพธ์ที่ได้จากอัลกอริทึม BF โดยใช้เมตริกซ์ต้นทุนในตัวอย่างที่ 7.1 คือ

[0 – 1 – 2 – 3 – 0] ค่า MMSRC เท่ากับ 23 มิลลิวินาที

วิธีแทนค่า :

จากตัวอย่างที่ 7.1 ค่าต้นทุนที่เชื่อมโยงระหว่างโนด 0 และ 1 มีค่าเท่ากับ 5 และเกิดจากการผ่านโนดที่ 3 ก่อน ดังนั้น เส้นทางที่เชื่อมโยงระหว่างโนด 0 และ 1 ในคำตอบที่ได้ ([0 – 1 – 2 – 3 – 0]) จะต้องถูกแทนด้วย [0 – 3 – 1] และเมื่อคำนวณค่าต้นทุนจะได้ค่า MMSRC ดังนี้

[0 – 3 – 1 – 2 – 3 – 0] ค่า MMSRC เท่ากับ 21 มิลลิวินาที

วิธีประยุกต์ :

สำหรับวิธีประยุกต์จะเป็นการนำรีดิวซ์เมตริกซ์ไปประมวลผลใหม่โดยใช้อัลกอริทึมต่าง ๆ (ในที่นี้ใช้อัลกอริทึม BF) ซึ่งผลลัพธ์ที่ได้มีค่าดังนี้

[0 – 3 – 1 – 0] [0 – 2 – 0] ค่า MMSRC เท่ากับ 20 มิลลิวินาที

แต่เนื่องจากการทำรีดิวซ์เมตริกซ์ในตัวอย่างที่ 7.1 เส้นทางที่เชื่อมต่อระหว่างโนด 0 และ 1 จะถูกแทนด้วย [0 – 3 – 1] และเส้นทางที่เชื่อมต่อระหว่างโนด 0 และ 2 จะถูกแทนด้วย [0 – 3 – 2] ดังนั้นเมื่อแทนค่าเส้นทาง จะได้เส้นทางที่แท้จริงดังนี้คือ

[0 – 3 – 1 – 3 – 0] [0 – 3 – 2 – 3 – 0]

จากตัวอย่างที่ 7.2 จะเห็นได้ว่าวิธีแทนค่า และวิธีประยุกต์จะให้ค่า MMSRC ที่ลดลง (ดีขึ้น) โดยวิธีประยุกต์มีแนวโน้มที่จะให้ค่า MMSRC ที่ดีกว่า ทั้งนี้เนื่องจากการวิธีประยุกต์เป็นการนำรีดิวซ์เมตริกซ์ไปประมวลผลใหม่ ซึ่งทำให้เกิดการพัฒนาของคำตอบที่ดีขึ้น และดีกว่าวิธีแทนค่า โดยการวิเคราะห์จะอธิบายไว้ในหัวข้อต่อไป

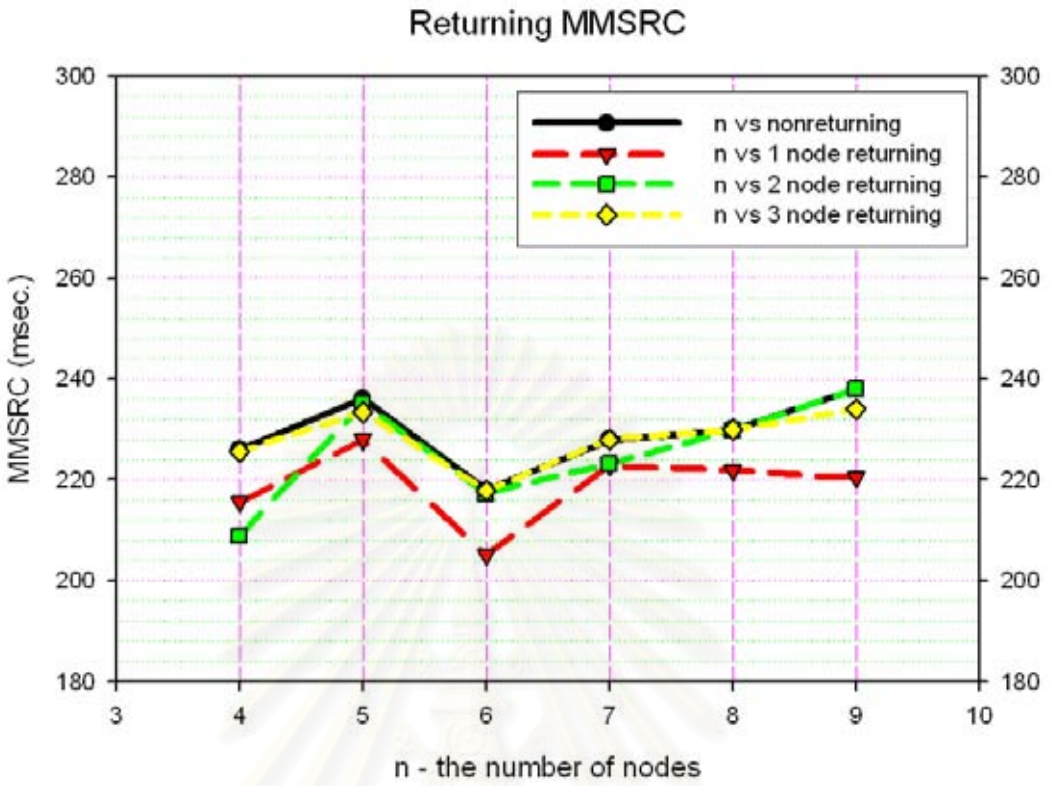
7.3 การวิเคราะห์ผลลัพธ์ที่ได้จากการกำหนดเส้นทางแบบอนุญาตให้ผ่านเส้นทางเดิมได้

จากตัวอย่างที่ 7.2 การใช้งานรีดิวซ์เมตริกซ์โดยนำไปผ่านอัลกอริทึมกำหนดเส้นทางจะส่งผลให้อัลกอริทึมกำหนดเส้นทางที่มีหลักการแบบไม่อนุญาตย้อนเส้นทางเดิมได้ เสมือนทำงานแบบอนุญาตให้ผ่านเส้นทางเดิมได้ และเนื่องจากการทำรีดิวซ์เมตริกซ์กรณี ผ่าน 2 โนด หรือ 3 โนดจะต้องใช้กับกรณีจำนวนโนดตั้งแต่ 3 หรือ 4 โนดขึ้นไปตามลำดับ ดังนั้นในการเปรียบเทียบวิเคราะห์ การทำรีดิวซ์เมตริกซ์จะวิเคราะห์จำนวนโนดตั้งแต่ 4 โนดขึ้นไป โดยเนื้อหาในหัวข้อนี้จะแบ่งเป็นหัวข้อย่อยต่อไปนี้

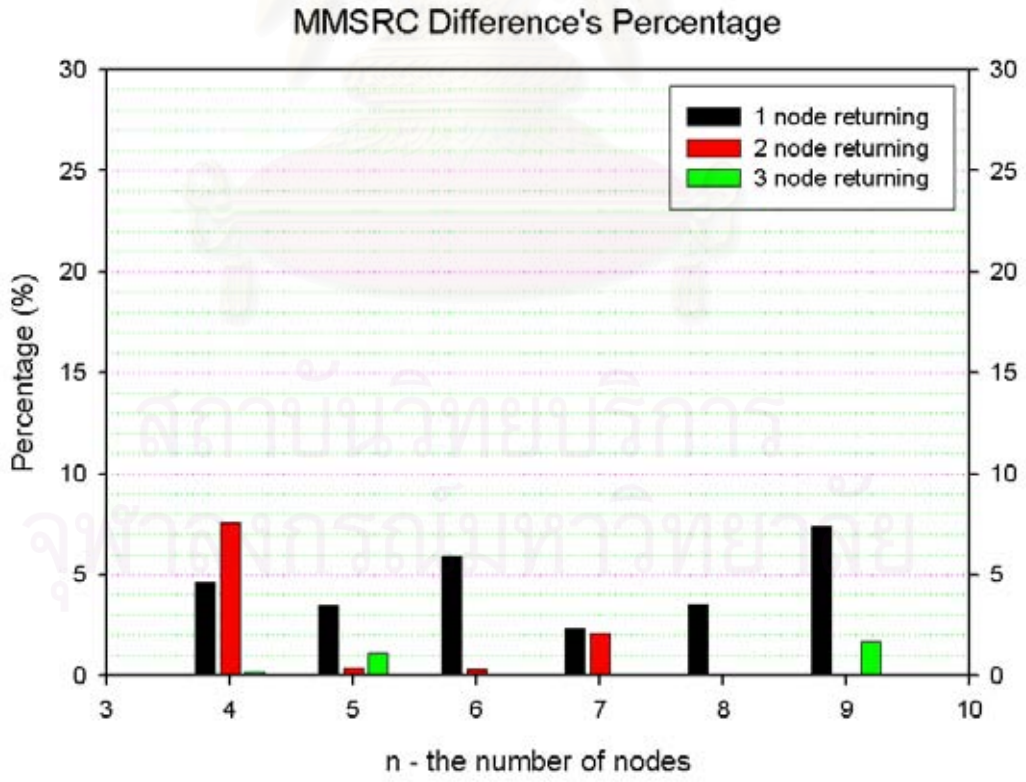
ก.) การวิเคราะห์ผลการทำรีดิวซ์เมตริกซ์เมื่อผ่านจำนวนโนดที่แตกต่างกัน
ข.) การวิเคราะห์การกำหนดเส้นทางโดยใช้วิธีประยุกต์
ค.) การวิเคราะห์การกำหนดเส้นทางโดยใช้วิธีแทนค่าเทียบกับวิธีประยุกต์

7.3.1 การวิเคราะห์ผลการทำรีดิวซ์เมตริกซ์เมื่อผ่านจำนวนโนดที่แตกต่างกัน

การวิเคราะห์การทำรีดิวซ์เมตริกซ์จะนำอัลกอริทึม BF มาประมวลผลเพื่อพิจารณาค่า MMSRC ที่ได้จากรีดิวซ์เมตริกซ์ โดยแต่ละเมตริกซ์ต้นทุนจะทำรีดิวซ์เมตริกซ์ขึ้นมา 3 ชุดคือ กรณีผ่าน 1 โนด 2 โนดและ 3 โนด ซึ่งผลที่ได้แสดงไว้ดังรูปที่ 7.5 และรูปที่ 7.6



รูปที่ 7.5 ค่า MMSRC ที่ได้จากอัลกอริทึม BF โดยใช้รีดิวซ์เมตริกซ์ และเมตริกซ์ต้นทุน

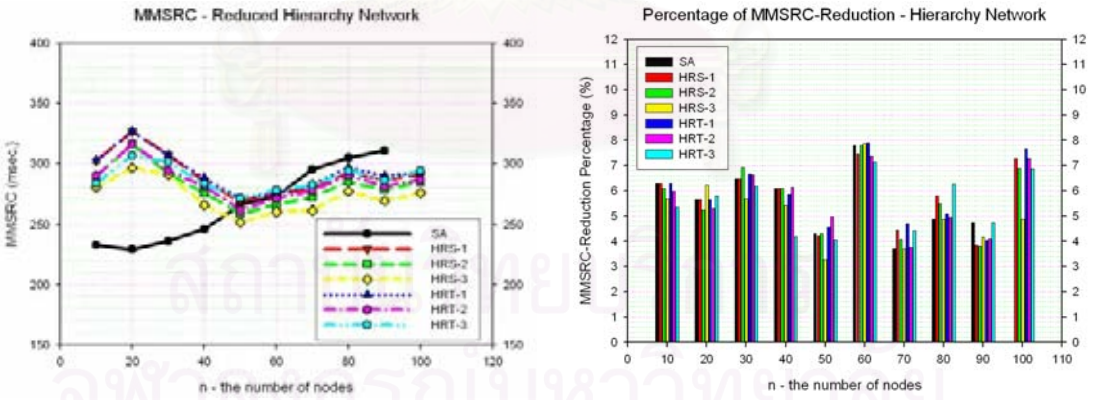


รูปที่ 7.6 ความแตกต่างเป็นร้อยละของค่า MMSRC
ของรีดิวซ์เมตริกซ์เมื่อเทียบกับเมตริกซ์ต้นทุน

จากรูปที่ 7.5 จะเห็นได้ว่า ค่า MMSRC ที่ได้จากรีดิวซ์เมตริกซ์จะมีค่าที่ต่ำกว่าค่า MMSRC ที่ได้จากเมตริกซ์ต้นทุน เมื่อเปรียบเทียบค่า MMSRC จากรีดิวซ์เมตริกซ์ด้วยกันแล้วจะเห็นได้ว่า ค่า MMSRC ที่ได้จากรีดิวซ์เมตริกซ์กรณีผ่าน 1 โนดจะให้ค่า MMSRC ที่ต่ำที่สุด ส่วนรีดิวซ์เมตริกซ์กรณีผ่าน 2 โนดและ 3 โนดจะให้ค่า MMSRC ที่ใกล้เคียงกับค่า MMSRC ที่ได้จากเมตริกซ์ต้นทุน ดังนั้นจะเห็นได้ว่าการทำรีดิวซ์เมตริกซ์ จะส่งผลให้ค่า MMSRC ที่ได้จะมีค่าต่ำกว่าการประมวลผลบนเมตริกซ์ต้นทุนแบบปกติ แต่จะให้ความแตกต่างที่น้อยมากไม่เกิน 10% นอกจากนี้การทำรีดิวซ์เมตริกซ์ที่ผ่านจำนวนโนดที่มากขึ้นไม่ได้ส่งผลให้ค่า MMSRC ที่ได้ลดลงตาม กล่าวคือกรณีผ่าน 2 โนดและ 3 โนดไม่ได้ให้ค่า MMSRC ที่ดีกว่ากรณีผ่าน 1 โนด

7.3.2 การวิเคราะห์การกำหนดเส้นทางโดยใช้วิธีประยุกต์

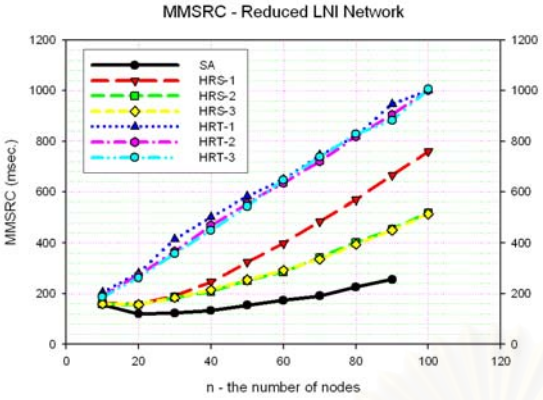
จากผลการวิเคราะห์ในหัวข้อ 7.3.1 แสดงให้เห็นว่า การทำรีดิวซ์เมตริกซ์ที่ผ่านจำนวนโนดมากขึ้นไม่ได้ส่งผลให้ค่า MMSRC ที่ได้ดีขึ้นตาม โดยกรณีผ่าน 1 โนดจะให้ค่า MMSRC ที่ต่ำที่สุด ดังนั้นเนื้อหาในหัวข้อย่อยนี้จะแสดงการเปรียบเทียบค่า MMSRC ที่ได้จากรีดิวซ์เมตริกซ์กรณีผ่าน 1 โนดโดยใช้อัลกอริทึม SA, HRS-1, HRS-2, HRS-3, HRT-1, HRT-2 และ HRT-3 โดยจะทำการวิเคราะห์บนเนตเวิร์กระดับชั้น (ที่กล่าวไว้ในบทที่ 4) LNI, WNI, L-Net, W-Net, HLNI และ HL-Net ซึ่งผลลัพธ์ที่ได้แสดงไว้ดังนี้



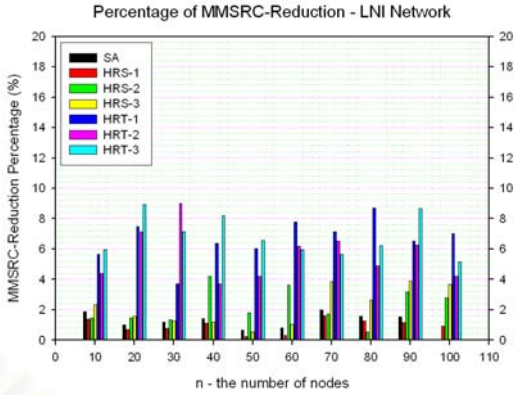
(ก) (ข)

รูปที่ 7.7 (ก) ค่า MMSRC ที่ได้จากเนตเวิร์กแบบระดับชั้น

รูปที่ 7.7 (ข) ค่าความแตกต่างเป็นร้อยละเมื่อเทียบกับค่า MMSRC ที่ไม่ได้ทำรีดิวซ์เมตริกซ์



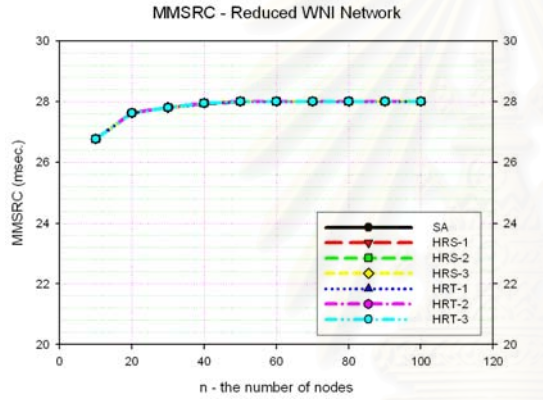
(ก)



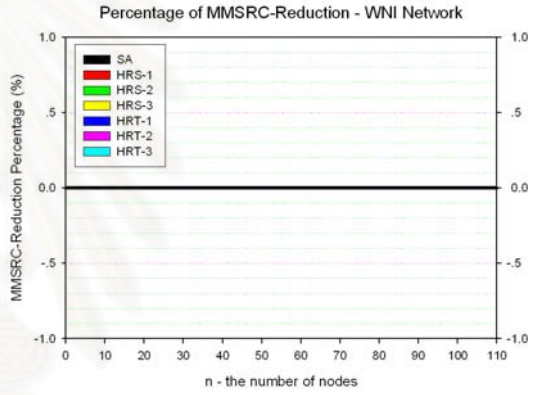
(ข)

รูปที่ 7.8 (ก) ค่า MMSRC ที่ได้จากเนตเวิร์กแบบ LNI

รูปที่ 7.8 (ข) ค่าความแตกต่างเป็นร้อยละเมื่อเทียบกับค่า MMSRC ที่ไม่ได้ทำรีดิวซ์เมตริกซ์



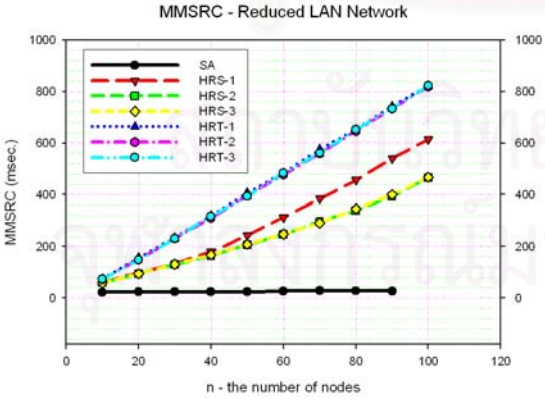
(ก)



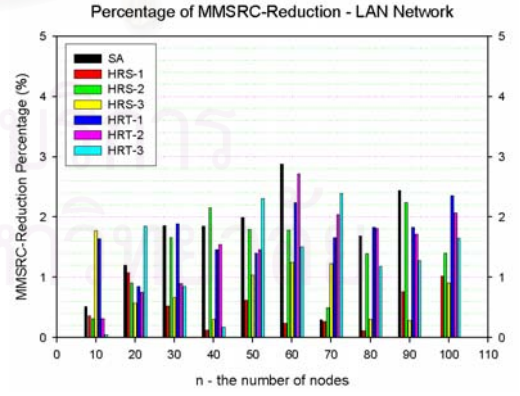
(ข)

รูปที่ 7.9 (ก) ค่า MMSRC ที่ได้จากเนตเวิร์กแบบ WNI

รูปที่ 7.9 (ข) ค่าความแตกต่างเป็นร้อยละเมื่อเทียบกับค่า MMSRC ที่ไม่ได้ทำรีดิวซ์เมตริกซ์



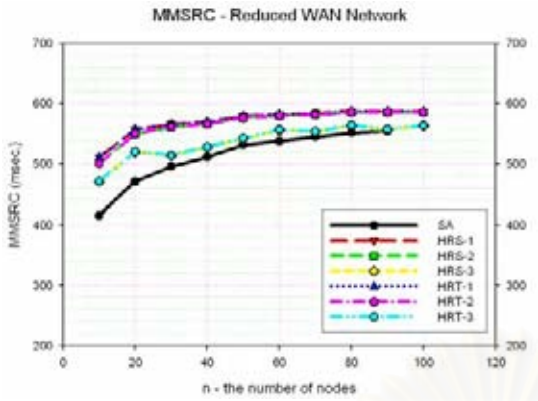
(ก)



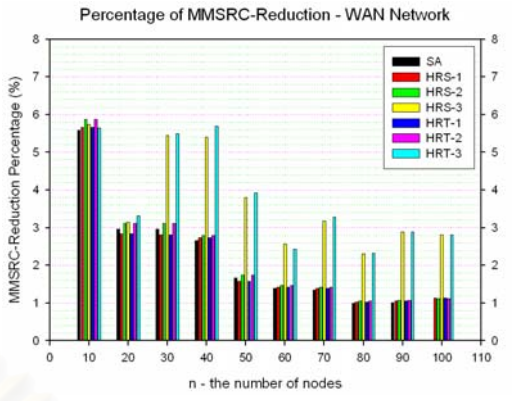
(ข)

รูปที่ 7.10 (ก) ค่า MMSRC ที่ได้จากเนตเวิร์กแบบ L-Net

รูปที่ 7.10 (ข) ค่าความแตกต่างเป็นร้อยละเมื่อเทียบกับค่า MMSRC ที่ไม่ได้ทำรีดิวซ์เมตริกซ์



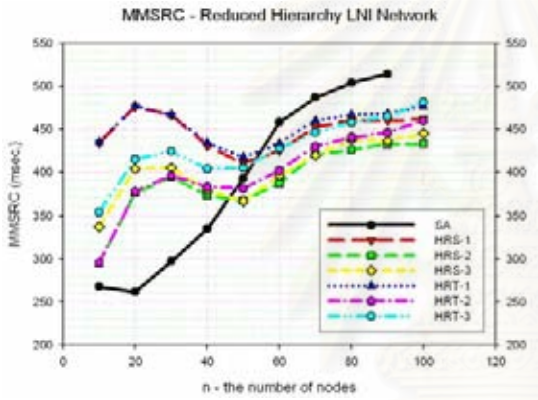
(ก)



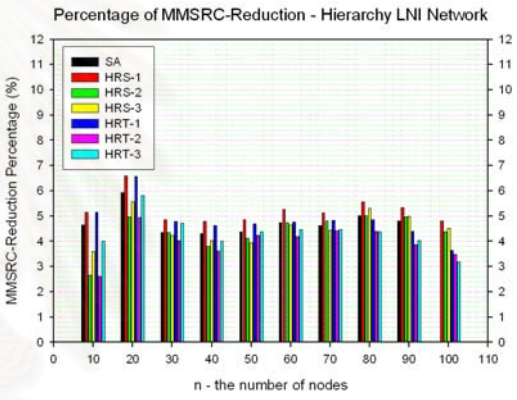
(ข)

รูปที่ 7.11 (ก) ค่า MMSRC ที่ได้จากเนตเวิร์กแบบ W-Net

รูปที่ 7.11 (ข) ค่าความแตกต่างเป็นร้อยละเมื่อเทียบกับค่า MMSRC ที่ไม่ได้ทำรีดิวซ์เมตริกซ์



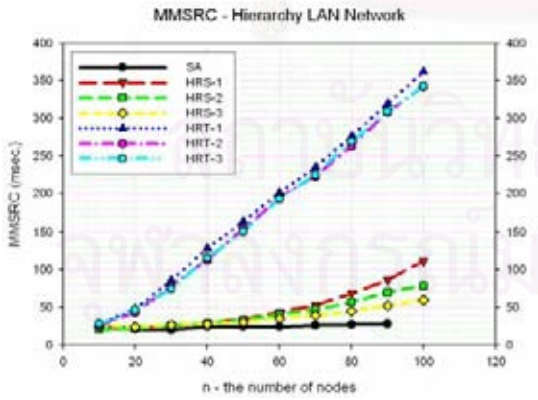
(ก)



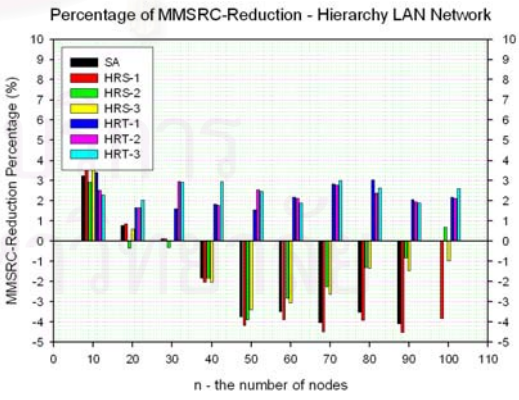
(ข)

รูปที่ 7.12 (ก) ค่า MMSRC ที่ได้จากเนตเวิร์กแบบ HLNI

รูปที่ 7.12 (ข) ค่าความแตกต่างเป็นร้อยละเมื่อเทียบกับค่า MMSRC ที่ไม่ได้ทำรีดิวซ์เมตริกซ์



(ก)



(ข)

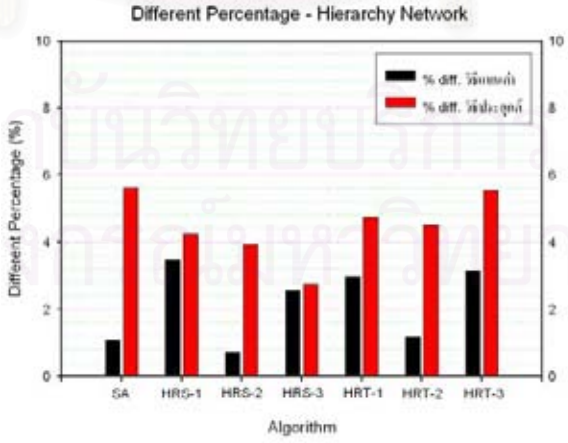
รูปที่ 7.13 (ก) ค่า MMSRC ที่ได้จากเนตเวิร์กแบบ HL-Net

รูปที่ 7.13 (ข) ค่าความแตกต่างเป็นร้อยละเมื่อเทียบกับค่า MMSRC ที่ไม่ได้ทำรีดิวซ์เมตริกซ์

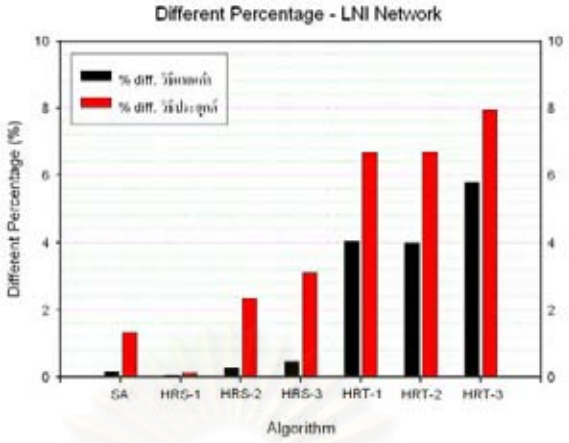
จากรูปที่ 7.7 (ก) – รูปที่ 7.13 (ก) จะเห็นได้ว่า ค่า MMSRC ที่ได้มีความคล้ายคลึงกับผลลัพธ์ที่ได้จากรูปที่ 6.30 และรูปที่ 6.37 – รูปที่ 6.42 โดยเมื่อวิเคราะห์ความแตกต่างเป็นร้อยละของค่า MMSRC ที่ได้จากวิธีประยุกต์เทียบกับค่า MMSRC กรณีไม่ใช้รีดิวซ์เมตริกซ์ จะเห็นได้ว่าความแตกต่างที่เกิดขึ้นจะมีค่าไม่เกินร้อยละ 10 และนอกจากนี้กรณีรูปที่ 7.13 (ข) จะเห็นได้ว่าที่ค่า n บางค่าจะให้ความแตกต่างเป็นร้อยละมีค่าติดลบ หมายความว่า ในบางกรณีการนำรีดิวซ์เมตริกซ์มาใช้ในวิธีประยุกต์ไม่ได้ทำให้ค่า MMSRC มีค่าลดลง (ดีขึ้น) เสมอไป ทั้งนี้เนื่องจาก กรณีรูปที่ 7.13 (ข) (Hierarchy LAN Network) นั้นเมื่อทำรีดิวซ์เมตริกซ์ จะทำให้ลักษณะการเชื่อมโยงระหว่างโนดในเนตเวิร์กที่มองโดยรีดิวซ์เมตริกซ์ มีรูปร่างที่แตกต่างไปจากเดิมโดยจะมีลักษณะการเกาะกลุ่มกันมากขึ้น และอาจทำให้ขนาดของกลุ่มบางกลุ่มมีจำนวนโนดมากขึ้น ดังนั้นเมื่อจำนวนโนดในกลุ่มมากขึ้นค่าต้นทุนรวมย่อมมีแนวโน้มที่มากขึ้นตามมา จึงส่งผลให้ค่า MMSRC ที่ได้มีค่ามากขึ้น หรือด้อยลง และเนื่องจากผลจากวิธีประยุกต์ทำให้บางกรณี ไม่ทำให้ค่า MMSRC ที่ได้มีค่าดีขึ้น ดังนั้นหัวข้อต่อไปจึงเป็นการวิเคราะห์ถึงการนำวิธีแทนค่ามาใช้ เทียบกับวิธีประยุกต์

7.3.3 การวิเคราะห์การกำหนดเส้นทางโดยใช้วิธีแทนค่า

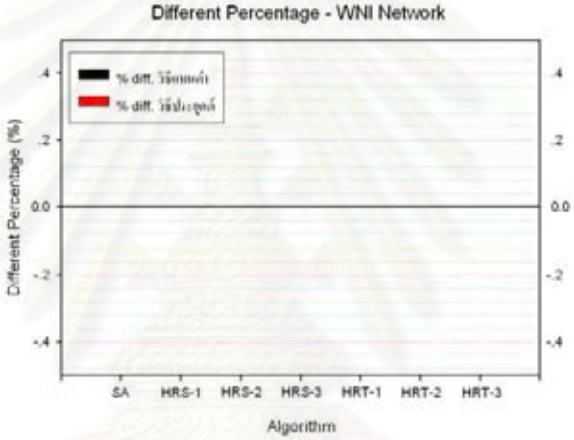
จากที่ได้อธิบายไว้ในตัวอย่างที่ 7.1 และตัวอย่างที่ 7.2 การนำรีดิวซ์เมตริกซ์มาใช้งานสามารถแบ่งได้เป็น 2 วิธีคือ วิธีแทนค่า และวิธีประยุกต์ และผลจากหัวข้อที่ 7.3.2 แสดงให้เห็นว่าวิธีประยุกต์นั้นไม่ได้ให้ค่า MMSRC ที่ดีขึ้นเสมอไป ดังนั้นในหัวข้อนี้จะเป็นการวิเคราะห์การนำรีดิวซ์เมตริกซ์ไปใช้งานโดยวิธีแทนค่า โดยจะวิเคราะห์ที่จำนวนโนดเท่ากับ 50 ซึ่งผลลัพธ์ที่ได้แสดงไว้ในรูปที่ 7.14 – รูปที่ 7.20 ดังนี้



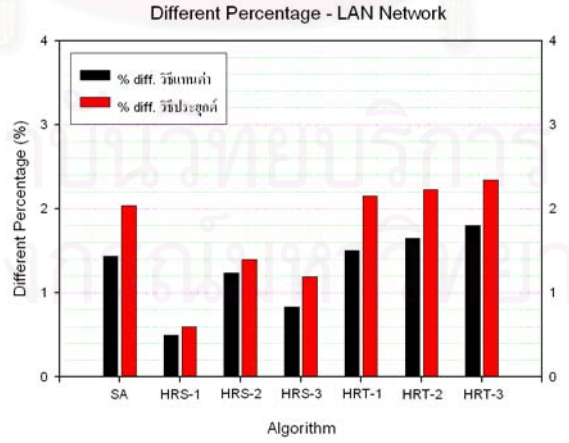
รูปที่ 7.14 ความแตกต่างเทียบเป็นร้อยละเมื่อใช้รีดิวซ์เมตริกซ์โดยวิธีแทนค่าและวิธีประยุกต์ กรณี Hierarchy Network



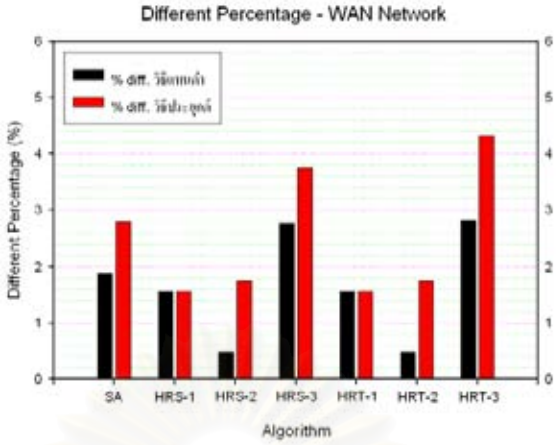
รูปที่ 7.15 ความแตกต่างเทียบเป็นร้อยละเมื่อใช้รีดิวซ์เมตริกซ์โดยวิธีแทนค่าและวิธีประยุกต์ กรณี LNI Network



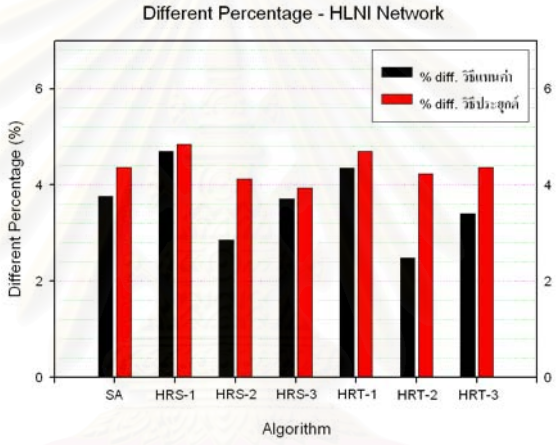
รูปที่ 7.16 ความแตกต่างเทียบเป็นร้อยละเมื่อใช้รีดิวซ์เมตริกซ์โดยวิธีแทนค่าและวิธีประยุกต์ กรณี WNI Network



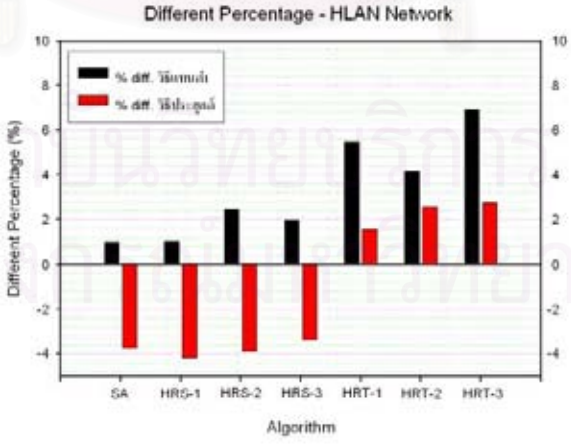
รูปที่ 7.17 ความแตกต่างเทียบเป็นร้อยละเมื่อใช้รีดิวซ์เมตริกซ์โดยวิธีแทนค่าและวิธีประยุกต์ กรณี LAN Network



รูปที่ 7.18 ความแตกต่างเทียบเป็นร้อยละเมื่อใช้รีดิวซ์เมตริกซ์โดยวิธีแทนค่าและวิธีประยุกต์ กรณี WAN Network



รูปที่ 7.19 ความแตกต่างเทียบเป็นร้อยละเมื่อใช้รีดิวซ์เมตริกซ์โดยวิธีแทนค่าและวิธีประยุกต์ กรณี Hierarchy LNI Network



รูปที่ 7.20 ความแตกต่างเทียบเป็นร้อยละเมื่อใช้รีดิวซ์เมตริกซ์โดยวิธีแทนค่าและวิธีประยุกต์ กรณี Hierarchy LAN Network

จากรูปที่ 7.14 – รูปที่ 7.20 จะเห็นได้ว่า ผลลัพธ์ที่ได้จากวิธีแทนค่า โดยส่วนใหญ่จะให้ผลลัพธ์ที่ด้อยกว่า ผลลัพธ์ที่ได้จากวิธีประยุกต์ แต่อย่างไรก็ตามผลลัพธ์ที่ได้จากวิธีแทนค่า จะไม่มีกรณีใดเลยที่ให้ค่า MMSRC ที่ด้อยกว่าเดิม เหมือนวิธีประยุกต์ โดยสรุปกล่าวคือ วิธีแทนค่าจะให้ผลลัพธ์ที่ด้อยกว่าวิธีประยุกต์เล็กน้อย แต่จะไม่เกิดกรณีให้ค่า MMSRC ที่ด้อยกว่าเดิม เมื่อนำรีดิวซ์เมตริกซ์มาใช้

จากการวิเคราะห์ข้างต้นจะเห็นได้ว่า แม้ว่าวิทยานิพนธ์ฉบับนี้มีการกำหนดเส้นทางของโมบายล์เอเจนต์ โดยนำคุณลักษณะ *การไม่ย้อนเส้นทางเดิม* ของ *ปัญหาเซลส์แมน* มาใช้งาน ซึ่งอาจทำให้เกิดความคิดในเรื่องของความจำเป็นที่จะต้องนำคุณสมบัตินี้มาใช้หรือไม่ แต่จากความพยายามที่จะแสดงให้เห็นถึงความแตกต่างในส่วนของการย้อนกลับเส้นทางเดิมได้ โดยใช้อัลกอริทึมที่มีอยู่นี้ แสดงให้เห็นว่า แม้จะมีการอนุญาตให้ย้อนเส้นทางเดิมได้ ผลลัพธ์ที่ได้จะแตกต่างจากกรณีไม่มีการย้อนกลับเส้นทางเดิมได้อยู่ในช่วงร้อยละ 10 ซึ่งสามารถยอมรับได้ นอกจากนี้ยังนับเป็นข้อดีเพิ่มเติมอีกอย่างหนึ่งสำหรับอัลกอริทึมกำหนดเส้นทางแบบไม่มีการย้อนกลับของเส้นทางเดิม กล่าวคือ การสร้างอัลกอริทึม โดยมีหลักการอนุญาตให้ย้อนเส้นทางเดิมได้นั้นจะมีความซับซ้อน และมีการใช้ทรัพยากรหน่วยความจำที่ค่อนข้างมาก และเวลาประมวลผลที่ค่อนข้างสูง ซึ่งแตกต่างจากกรณีไม่มีการย้อนเส้นทางเดิมที่มีการใช้หน่วยความจำ และเวลาประมวลผลที่ค่อนข้างต่ำกว่า โดยให้ผลลัพธ์ที่ใกล้เคียงกัน

บทที่ 8

บทสรุปและข้อเสนอแนะ

8.1 บทสรุป

จากปริมาณข้อมูลที่สามารถสืบค้นได้บนเนตเวิร์กที่มีปริมาณมาก ได้มีแนวคิดต่าง ๆ ที่จะอำนวยความสะดวกในการสืบค้นข้อมูลให้ตรงกับความต้องการของผู้ใช้บริการ ได้แก่ วิธีการสืบค้นข้อมูลแบบ VSM (Vector Space Model) และ IL (Inverted List) โดยวิธีการเหล่านี้จำเป็นจะต้องมีเซิร์ฟเวอร์ที่ทำหน้าที่เก็บฐานข้อมูลซึ่งมีจำนวนมากอยู่ในเนตเวิร์ก ดังนั้นการสืบค้นข้อมูลโดยใช้โมบายล์เอเจนต์นั้นจำเป็นจะต้องมีการกำหนดเส้นทาง หรือลำดับก่อนหลังของเซิร์ฟเวอร์ที่โมบายล์เอเจนต์ใช้ในการประมวลผล โดยวิทยานิพนธ์ฉบับนี้ได้เสนออัลกอริทึมที่ใช้ในการกำหนดเส้นทางให้กับโมบายล์เอเจนต์ พร้อมทั้งปรับปรุง และประยุกต์อัลกอริทึมที่นิยมใช้ในปัจจุบันให้เข้ากับงานสืบค้นข้อมูล เพื่อใช้ในการวิเคราะห์เชิงเปรียบเทียบกับอัลกอริทึมฮิวริสติกที่คิดขึ้น

จากผลงานวิจัยและการวิเคราะห์บทต่าง ๆ จะเห็นได้ว่า การทำงานการสืบค้นข้อมูลในปัจจุบันมีการพัฒนาแนวคิดต่าง ๆ ที่ต่างจากทฤษฎีในยุคแรก ๆ เช่น Inverted List ซึ่งได้พัฒนาแนวคิดมาจากหลักการเดิมอยู่พอสมควร (Vector Space Model : VSM) ซึ่งผลลัพธ์ที่ได้ทำให้สามารถทำการสืบค้นข้อมูลที่มีขนาดฐานข้อมูลที่มีขนาดใหญ่ได้ สำหรับวิธีการกำหนดเส้นทางให้กับโมบายล์เอเจนต์จะเห็นได้ว่า วิธีการปรับปรุงคำตอบต่าง ๆ ของอัลกอริทึมที่นิยมใช้ในปัจจุบัน เช่น Brute Force Search, Simulated Annealing, Tabu Search และ Modified Compact Genetic จะให้คำตอบที่ดีกว่าการใช้วิธีแบบฮิวริสติกที่คิดขึ้น แต่อย่างไรก็ตามเวลาที่ใช้ในแต่ละอัลกอริทึมมีค่าค่อนข้างสูงเมื่อพิจารณาถึงการใช้งานจริงในการกำหนดเส้นทางเริ่มต้นให้กับโมบายล์เอเจนต์ แม้ว่าผลลัพธ์ที่ได้จากอัลกอริทึมฮิวริสติกจะมีผลลัพธ์ที่ไม่ดีกว่าอัลกอริทึมมาตรฐานแต่ยังมีความใกล้เคียงของค่าผลลัพธ์ที่ได้ เมื่อนำวิธีกำหนดเส้นทางแบบอิสระนำมาร่วมพิจารณา นอกจากนี้เวลาที่ใช้ในการกำหนดเส้นทางเริ่มต้น อัลกอริทึมฮิวริสติกจะใช้เวลาที่น้อยมาก โดยเวลาที่ใช้อยู่ในระดับมิลลิวินาที

จากการทำงานของอัลกอริทึมต่าง ๆ ที่กล่าวมาแล้วนั้น ทุกอัลกอริทึมมีพื้นฐานการทำงานในลักษณะไม่มีการย้อนกลับของเส้นทาง ซึ่งเป็นคุณลักษณะหนึ่งที่มีใน *ปัญหาของเซลส์แมน* เพื่อแสดงให้เห็นถึงความแตกต่างที่ได้เมื่อพิจารณาการกำหนดเส้นทางแบบย้อนเส้นทางได้ จึงนำวิธีรีดิวซ์เมตริกซ์มาประยุกต์ใช้ เพื่อให้อัลกอริทึมที่มีอยู่เสมือนกำหนดเส้นทางเริ่มต้นให้กับ

โมบายล์เอเจนต์แบบย้อนเส้นทางเดิมได้ โดยการใช้งานรีดิวซ์เมตริกว์ในวิทยานิพนธ์นี้แบ่งออกเป็น 2 แบบคือ วิธีแทนค่า และวิธีประยุกต์ ซึ่งจากผลลัพธ์แสดงให้เห็นว่า โดยส่วนใหญ่วิธีแทนค่าจะให้ผลลัพธ์ที่ด้อยกว่าผลลัพธ์จากวิธีประยุกต์ แต่อย่างไรก็ตามวิธีประยุกต์จะให้ผลลัพธ์ที่แย่กว่าเดิม (เมื่อไม่ใช้รีดิวซ์เมตริกซ์) ในบางกรณี จึงนับเป็นข้อด้อยของวิธีประยุกต์ อย่างไรก็ตามจากผลลัพธ์โดยรวม การนำรีดิวซ์เมตริกซ์มาใช้งานจะเห็นได้ว่า การกำหนดเส้นทางแบบย้อนกลับได้นั้นจะให้ค่าผลลัพธ์ที่ใกล้เคียงกับผลลัพธ์ที่ได้จากการกำหนดเส้นทางแบบไม่ย้อนเส้นทางเดิม ซึ่งทำให้เห็นข้อดีเพิ่มเติมของการกำหนดเส้นทางแบบไม่ย้อนกลับ กล่าวคือ เนื่องจากการทำงานของอัลกอริทึมแบบย้อนเส้นทางเดิมได้นั้นจะมีความซับซ้อน และใช้เนื้อที่หน่วยความจำที่มากกว่า และยังใช้เวลาประมวลผลที่ยาวนานกว่า ความซับซ้อน และเวลาประมวลผลที่น้อยกว่าของอัลกอริทึมกำหนดเส้นทางแบบไม่ย้อนกลับเส้นทางเดิมที่ให้ผลลัพธ์ที่ใกล้เคียงกัน น่าจะเป็นทางเลือกที่เหมาะสมในการกำหนดเส้นทางเริ่มต้นให้กับโมบายล์เอเจนต์

8.2 ข้อเสนอแนะ

จากที่กล่าวไว้ในบทสรุปเรื่องแนวทางการทำงานของอัลกอริทึมกำหนดเส้นทางแบบไม่มีการย้อนกลับนั้น ในส่วนที่แสดงให้เห็นถึงการวิเคราะห์การทำงานแบบย้อนกลับเส้นทางเดิมได้นั้น ทำโดยการแปลงรูปแบบของเมตริกซ์ต้นทุนให้เป็นรีดิวซ์เมตริกซ์แล้วนำมาประมวลผลโดยอัลกอริทึมที่ออกแบบโดยไม่มีการย้อนกลับ แม้ว่าผลลัพธ์ที่ได้จะแสดงให้เห็นว่าการกำหนดแบบย้อนกลับเส้นทางเดิมได้จะให้ผลลัพธ์ที่ใกล้เคียงแบบไม่มีการผ่านเส้นทางเดิม แต่การทำงานที่กล่าวนั้นเป็นเพียงแนวทางให้อัลกอริทึมกำหนดเส้นทางแบบไม่ย้อนกลับเส้นทางเดิม เสมือนทำงานแบบย้อนกลับเส้นทางเดิมได้ ดังนั้นการเขียนโปรแกรมโดยใช้แนวทางการย้อนกลับเส้นทางเดิมได้ อาจจะให้ผลแตกต่างจากนี้ได้ เพราะฉะนั้นหากมีกรณีศึกษาเพิ่มเติมในเรื่องของการกำหนดเส้นทางแบบย้อนกลับ น่าจะมีการศึกษาถึงวิธีที่นำอัลกอริทึมมาตรฐานที่มีอยู่ในปัจจุบันเช่น Brute Force Search, Simulated Annealing, Tabu Search หรือ Genetic Algorithm มาประยุกต์ใช้งานการกำหนดเส้นทางเริ่มต้นโดยอนุญาตให้ย้อนกลับเส้นทางเดิมได้ เพราะปัจจุบันเอกสารทางวิชาการส่วนใหญ่จะเน้นการนำอัลกอริทึมมาตรฐานที่มีไปใช้กับปัญหาของเซลส์แมน แต่ไม่ได้พิจารณาถึงการกำหนดเส้นทางแบบย้อนกลับเส้นทางเดิมได้

รายการอ้างอิง

Alfonso Fuggetta, Gian Pietro Picco, and Giovanni Vigna, "Understanding Code Mobility", *IEEE Transactions on Software Engineering.,* May 1998.

Akhil Sahai, and Christine Morin, "Towards Distributed and Dynamic Network Management", *IEEE/IFIP Network Operation and Management Symposium*, 1998.

Anselm Lingnau, Oswald Drobnik, and Johann Wolfgang Goethe, "Making Mobile Agents Communicate : A Flexible Approach", *IEEE Proceedings*, 1996.

Antonella Di Stefano, and Corrado Santoro, "NetChaser : Agent Support for Personal Mobility", *IEEE Internet Computing*, Mar-Apr 2000.

Brian Brewington, Robert Gray, Katsuhiro Moizumi, David Kotz, George Cybenko and Daniel Rus, "Mobile Agents in Distributed Information Retrieval", *Thayer School of Engineering, Dartmouth College,* 1999.

Damianos Gavalas, Dominic Greenwood, Mohammed Ghanbari, and Mike O'Mahony, "An Infrastructure for Distributed and Dynamic Network Management based on Mobile Agent Technology", *IEEE International Conference* , 1999.

DataBeam Corporation, "A Primer on the H.323 Series Standard", *White Paper*, May 1998.

Dik L. Lee, "Document Ranking and the Vector-Space Model", March/April 1999.

Elin Wedlund, and Henning Schulzrinne, "Mobility Support using SIP", *Columbia University*.

Euihyun Jung, Yong-Jin Park, and Chulhye Park, "Mobile Agent Network for Supporting Personal Mobility", *Twelfth International Conference*, 1998.

Gatot Susilo, Andrzej Bieszczad, and Bernard Pagurek, "Infrastructure for Advanced Network Management based on Mobile Code", *Network Operations and Management Symposium*, 1998.

George Cybenko, Robert Gray, Yunxin Wu, Alexy Khrabrov, "Information Architecture and Agents", *Dartmouth College,* 1994.

George R. Harik, Fernando G. Lobo and David E. Goldberg, "Compact Genetic Algorith", *IEEE Transaction on Evolutionary Computation, Vol.3,No.4.* Nov,1999.

Giacomo Cabri, Letizia Leonardi, and Franco Zambonelli, "Mobile-Agent Coordination Models for Internet Applications", *IEEE Computer*, Feb 2000.

Guedes L.A., Oliveira P. C., Faina L. F. and Cardozo E., "QoS Agency : An Agent-based Architecture for Supporting Quality of Service in Distributed Multimedia Systems", *IEEE Conference on Protocols for Multimedia Systems - Multimedia Networking (PROMSMmNet'97)* , 1997.

Handley M., Schulzrinne H., Schooler E., and Rosenberg J., "SIP : Session Initiation Protocol", *RFC 2543*, *Standards Track*, Mar 1999.

Hans-Gunter Stein, and Michael Breu, "An Architecture for Automated Information Retrieval", {hgs|breu}@fast.de.

Henning G. Schulzrinne, and Jonathan D. Rosenberg, "The Session Initiation Protocol : Providing Advanced Telephony Services Across the Internet", *Bell Labs Technical journal*, Oct-Dec 1998.

Hongsong Ma, and Sun Teck Tan, "Dynamic Mobile Agent Based Distributed Network Management Using Internet Technology and CORBA", *IEEE SMC '99 Conference Proceedings*, 1999.

Ichiro Satoh, "A Mobile Agent-Based Framework for Active Networks", *IEEE SMC '99 Conference Proceedings*, 1999.

James Gosling, and Jenry McGilton, "Tha Java™ Language Environment : A White Paper", *Sun Microsystems*, May 1996.

Jihoon Yang, Prashant Pai, Vasant Honavar, and Les Miller, "Mobile Intelligent Agents for Document Classification and Retrieval : A Machine Learning Approach", *Dept. of Computer Science, Iowa State University*.

Jihoon Yang, Vasant Honavar, Les Miller, and Johnny Wong, "Intelligent Mobile Agents for Information Retrieval and Knowledge Discovery from Distributed Data and Knowledge Sources", *Information Technology Conference*, 1998.

Jin-Wook Back, Jae-Heung Yeo, Gyu-Tae Kim, Heon-Young Yeom, "Cost Effective Mobile Agent Planning for Distributed Information Retrieval", *Seoul National University*.

Joachim Baumann, Fritz Hohl, Nikolaos Radouniklis, Markus Straber, and Kert Rothermel, "Communication Concepts for Mobile Agent Systems", *University of Stuttgart, Germany*.

Jonathan Dale, "A Mobile Agent Architecture for Distributed Information Management", *Department of Electronics and Computer Science,* Sep,1997.

Karmouch A., and Horlait E., "Agent-based Multimedia Communications on Internet", *IEEE Canadian Conference*,1998.

Kazuhiro Minami and Toshihito Suzuki, "JMT (Java-Based Moderator Templates for Multi-Agent Planning", *OOPSLA'97 Workshop*.

Kevin D. Smith, Raman B. Paranjape, "Mobile Agents for Web-Based Medical Image Retrieval", *IEEE Proceeding,* May,1999.

Luis Moura Silva, Guilherme Soares, Paulo Martins, Victor Batista, and Luis Santos, "The Performance of Mobile Agent Platforms", *IEEE First International Symposium* , 1999.

Magedanz T., Rothermel K., and Krause S., "Intelligent Agents : An Emerging Technology for Next Generation Telecommunications?", *IEEE Computer Societies*, 1996.

Magedanz T., and Eckardt T., "Mobile Software Agents : A New Paradigm for Telecommunications Management", *IEEE Proceedings*, 1996.

Mahbub Hassan, Alfandika Nayandoro, and Mohammed Atiquzzaman, "Internet Telephony : Services, Technical challenges, and Products", *IEEE Communication Magazine*, Apr 2000.

Mamadou Tadiou Kone, and Tatsuo Nakajima, "An Architecture for a QoS-based Mobile Agent System", *IEEE Proceedings*, 1998.

Marcelo Goncalves Rubinstein, and Otto Carlos Muniz Bandeira Duarte, "Service Location for Mobile Agent Systems", *SBT/IEEE International*, 1998.

Marcelo Goncalves, and Otto Carlos Muniz Bandeira Duarte, "EVALUATING THE PERFORMANCE OF MOBILE AGENTS IN NETWORK MANAGEMENT", *Global Telecommunications Conference-Globalcom'99*, 1999.

Menelaos K. Perdikeas, Fotis G. Chatzipapadopoulos, and Iakovos S. Venieris, "An Evaluation Study of Mobile Agent Technology : Standardization, Implementation and Evolution", *IEEE International Conference*, 1999.

Michael W. Berry, Merray Browne, "Understanding Search Engine : Mathematical Modeling and Text Retrieval" *, SIAM : Society for Industrial and Applied Mathematics Philadelphia*.

Michael W. Berry, Zlatko Drmac, Elizabeth R. Jessup, "Matrices, Vector Spaces, and Information Retrieval", *SIAM REVIEW, Vol.41,No.2,pp.335-362*,1999.

Dr.Mohan Chellappa, "Text Retrieval – A Trendy Cocktail to Address the Data World", *IEEE*, 1994.

Onn Shehory, Kartia Sycara, Prasad Chalasani and Somesh Jha, "Agent Cloning : An approach to agent mobility and resource allocation", *The Robotics Institute, Carnegie Mellon University*.

Puliafito A., Riccobene S., and Scarpa M., "An analytical comparison of the client-server, remote evaluation and mobile agents paradigms", *IEEE Proceedings*, 1999.

Raymie Stata, Krishna Bharat, Farzin Maghoul, "The Term Vector Database : fast access to indexing trees for web pages", http://www9.org/w9cdrom/159/159.html

Robert L. Pop, Bohdan P. Maksymiuk, and Michael R. Poreda, "Efficient Information Retrieval On the World Wide Web Using Adaptable and Mobile Java Agents", *IEEE International Conference ,* 1998.

Ronald L. Rardin, "Optimization in Operations Research",*Prentice-Hall International,Inc.*, 1998.

Rothermel K., Hohl F., Radouniklis N., "Mobile Agent Systems : What is Missing ?", *Institute for Parallel and Distributed High-Performance Systems (IPVR).*

Sun Microsystems, "JAIN™ : Integrated Network APIs for the Java™ Platform", http://java.sun.com/products/jain, Nov 2000.

Sun Microsystems, "Jini™ Architecture Specification", Oct 2000.

Tak W. Yan, Hector Garcia Molina, "Index Structures for Information Filtering Under the Vector Space Model", *IEEE*, 1994.

University of Glasgow, "IDOMENEUS Technology Transfer Server Database", *Department of Computing Science,* 1999.

Uyless Black, "Network Management Standards : SNMP, CMIP, TMN, MIBs, and Object Libraries", *New York, McGraw-Hill*, 1995.

Vu Anh Pham and Ahmed Karmouth, "Mobile Software Agents : An Overview", *IEEE Communication Magazine.,* July 1998, pp.26-37.

Van C.J. Rijsbergen, "Information Retrieval", *Department of Computer Science, University of GLASGOW*.

Wen-Shyen E. Chen, C. Y. Lin, and Yao-Nan Lien, "A Mobile Agent Infrastructure with Mobility and Management Support", *IEEE International Workshops ,* 1999.

Wenping Chang, and Radu Popescu-Zeletin, "Mobile Agent based Service Provisioning in Integrated Networks", *IEEE Proceedings*, 2000.

Wolfgang Theilmann, Kurt Rothermel, "Disseminating Mobile Agents for Distributed Information Filtering",*IEEE*,1999.

Yu MIYOSHI, Kantaro KAMAHORA, Youg-Jin PARK, Yoshiyori URANO, and Hideyoshi TOMINAGA, "An Advanced Network Management System with Mobile Agents", *Dept. of Electronics Information and Communication Engineering, Waseda University*.

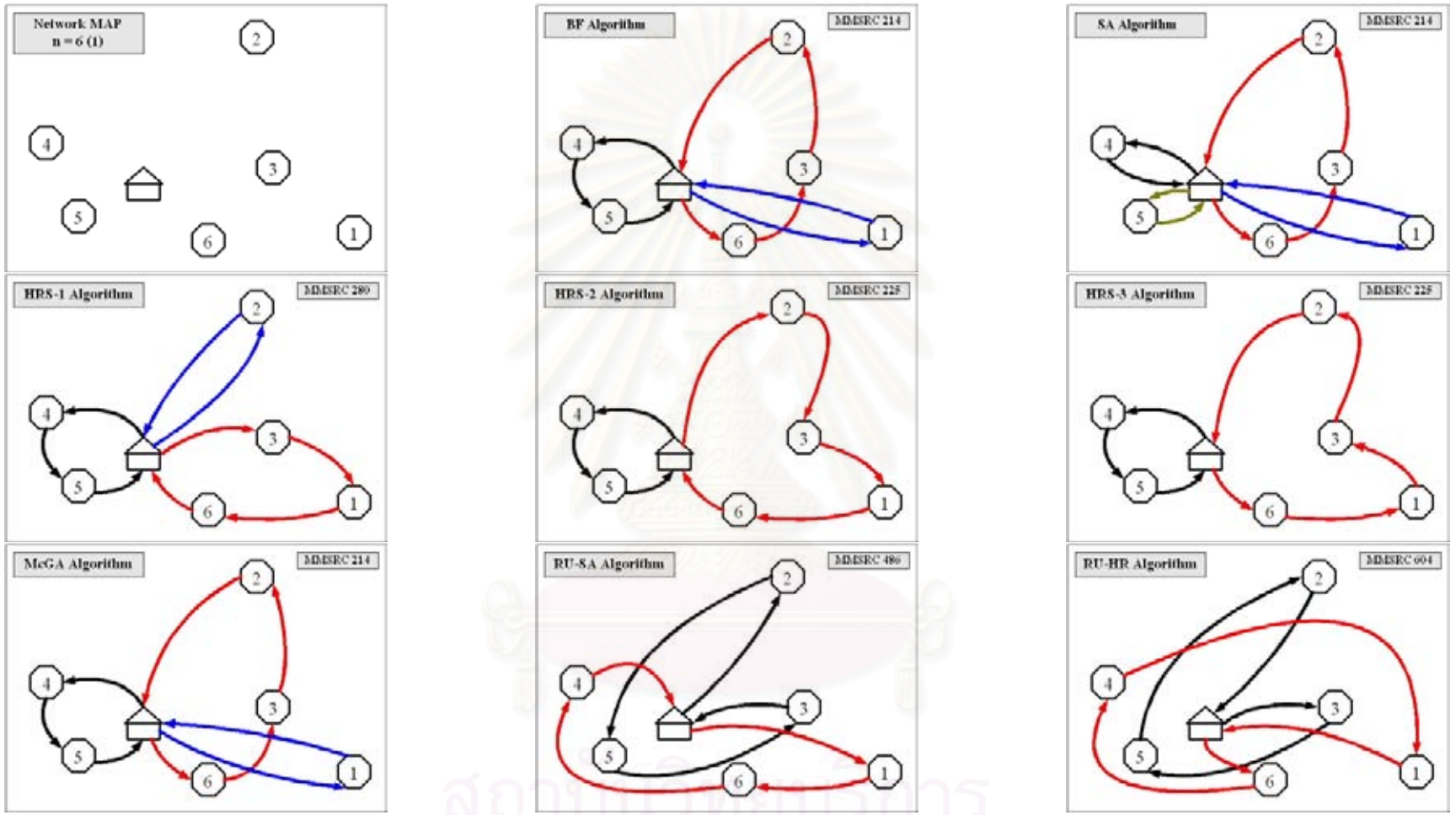
ภาคผนวก

ภาคผนวก ก.

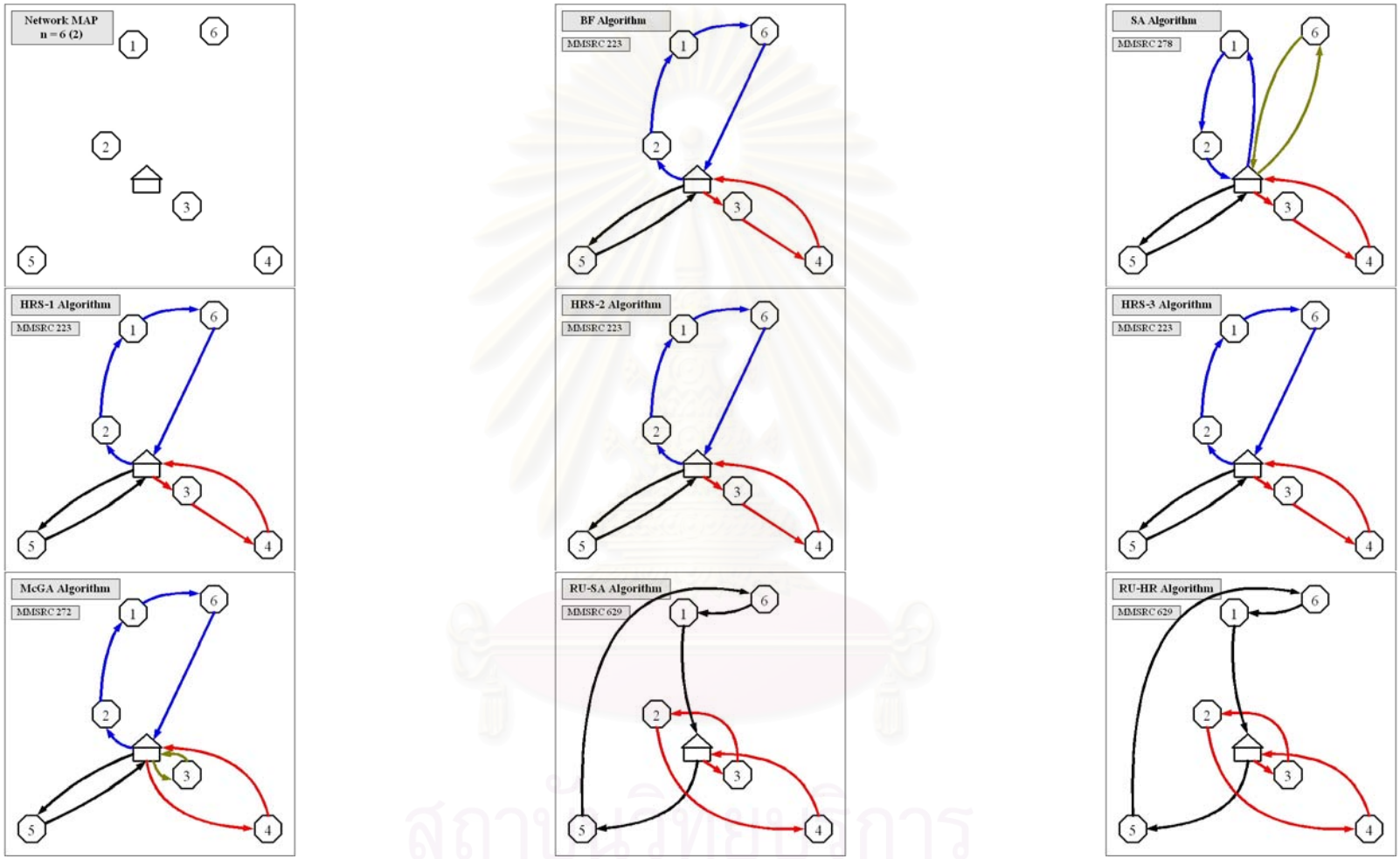
ภาคผนวก ก. จะเป็นส่วนที่แสดงถึงตัวอย่างของเส้นทางที่กำหนดโดยอัลกอริทึมกำหนดเส้นทางต่าง ๆ โดยในแต่ละรูปจะประกอบไปด้วย 9 รูปย่อย ดังนี้คือ

1.) ผังของเนตเวิร์กจำลอง
2.) เส้นทางที่กำหนดโดยอัลกอริทึม BF
3.) เส้นทางที่กำหนดโดยอัลกอริทึม SA
4.) เส้นทางที่กำหนดโดยอัลกอริทึม HRS-1
5.) เส้นทางที่กำหนดโดยอัลกอริทึม HRS-2
6.) เส้นทางที่กำหนดโดยอัลกอริทึม HRS-3
7.) เส้นทางที่กำหนดโดยอัลกอริทึม McGA
8.) เส้นทางที่กำหนดโดยอัลกอริทึม RU-SA
9.) เส้นทางที่กำหนดโดยอัลกอริทึม RU-HR

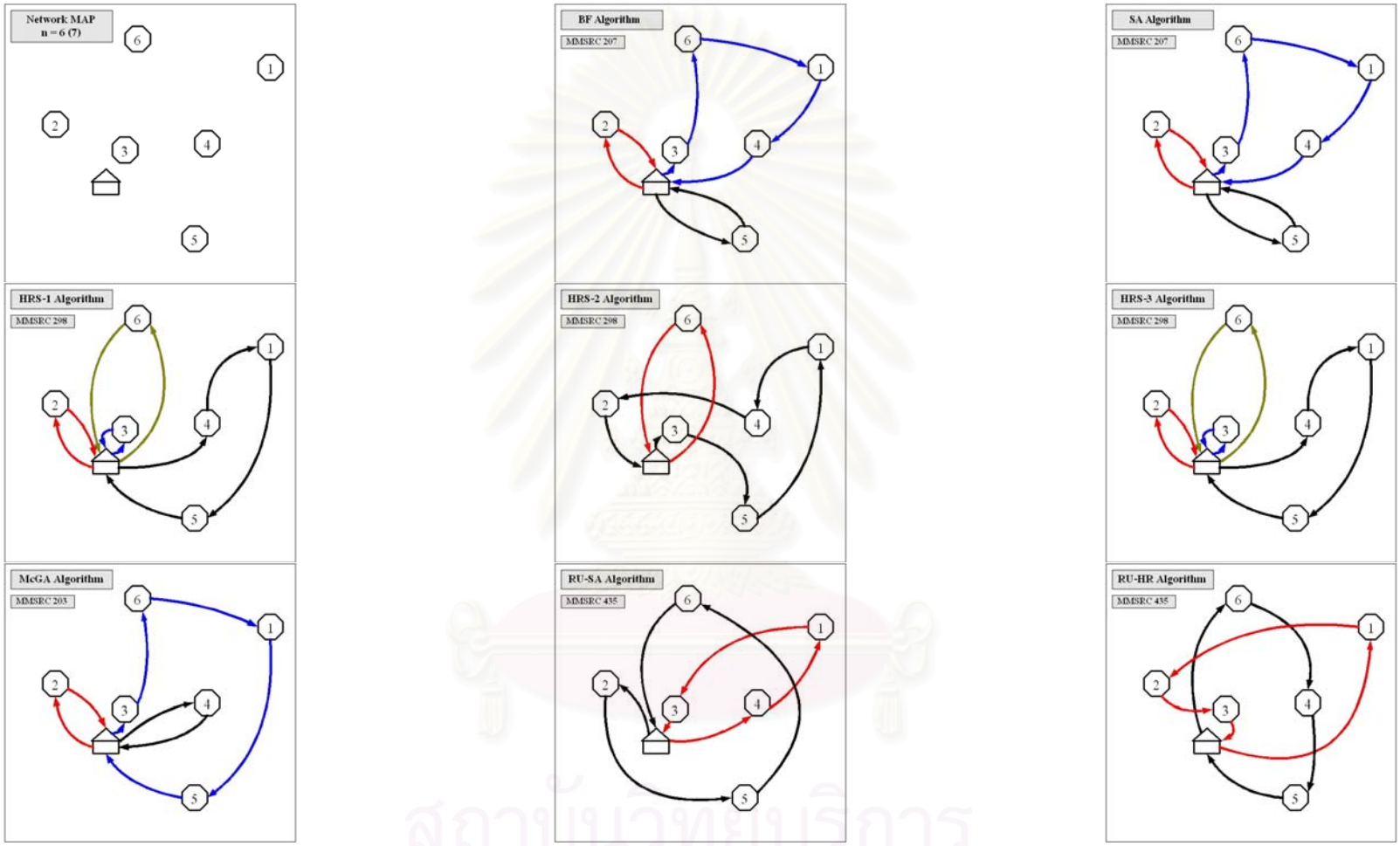
โดยจะยกตัวอย่างทั้งหมด 6 ชุดคือ ตัวอย่างกรณี n = 6 จำนวน 3 ชุด และตัวอย่างกรณี n = 8 จำนวน 3 ชุด ดังรูปที่ ก1 – รูปที่ ก6 ดังนี้



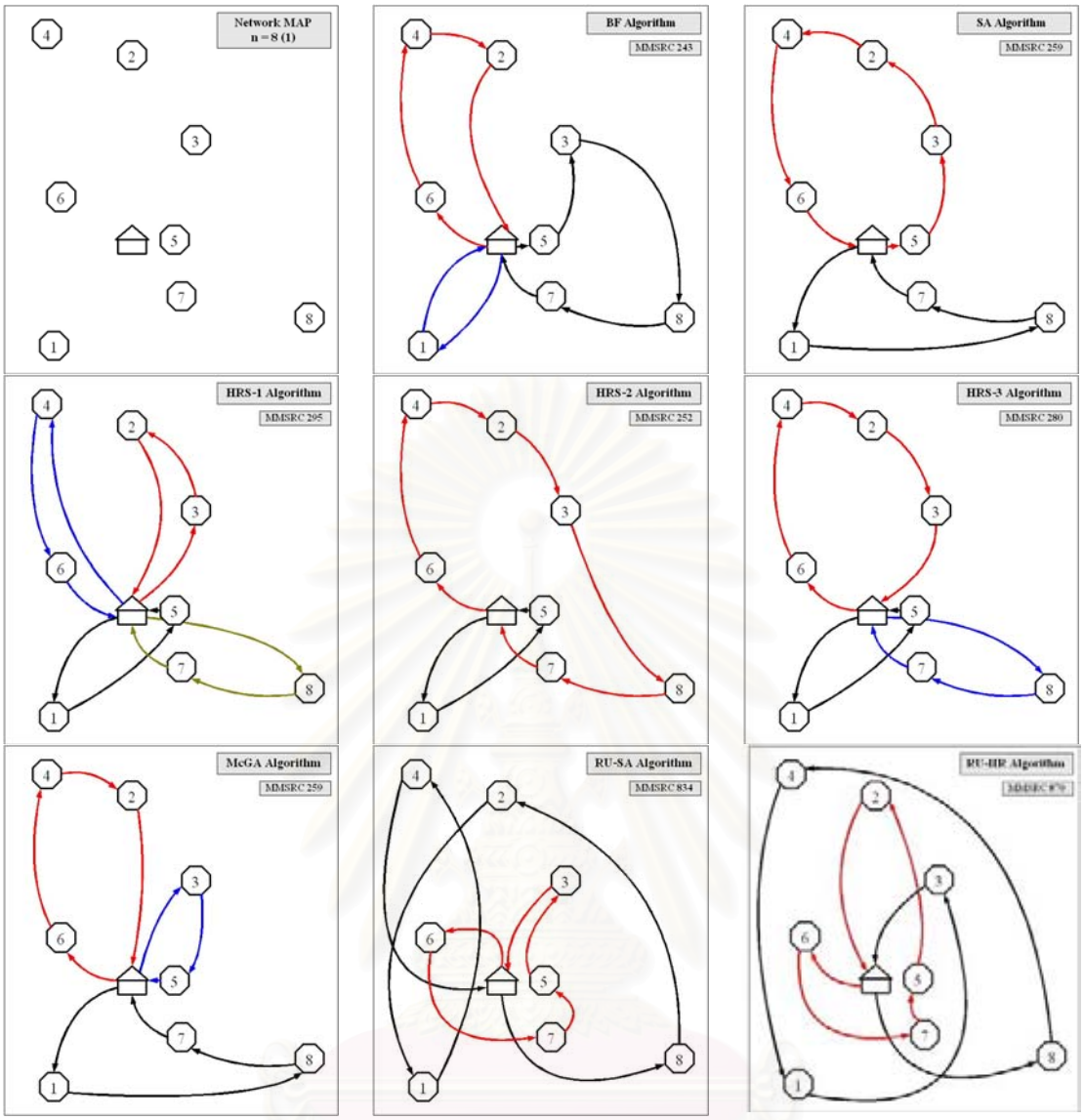
รูปที่ ก1 เส้นทางที่กำหนดโดยอัลกอริทึมกำหนดเส้นทางต่าง ๆ เมื่อ n = 6 (1)



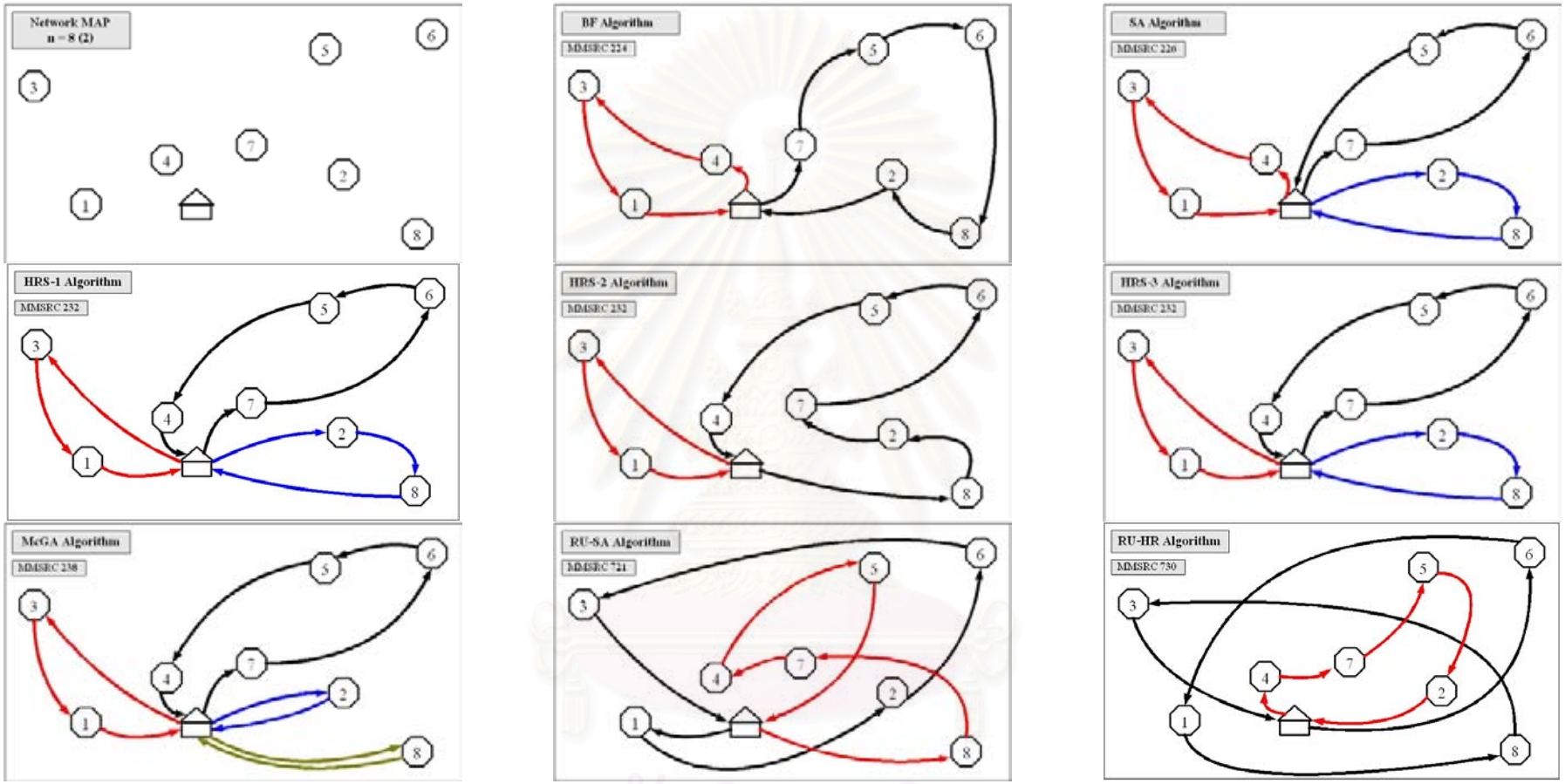
รูปที่ ก2 เส้นทางที่กำหนดโดยอัลกอริทึมกำหนดเส้นทางต่าง ๆ เมื่อ n = 6 (2)



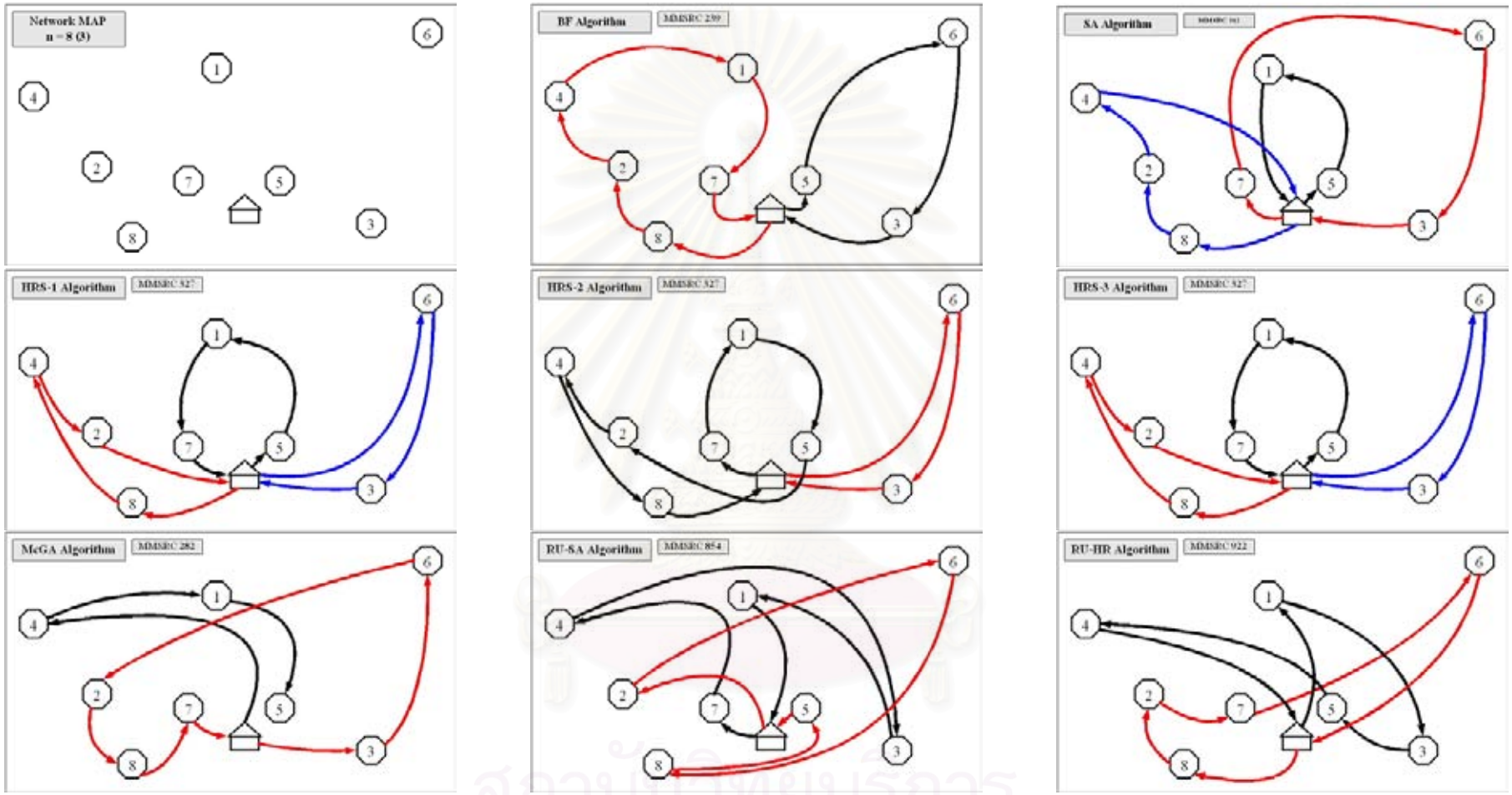
รูปที่ ก3 เส้นทางที่กำหนดโดยอัลกอริทึมกำหนดเส้นทางต่าง ๆ เมื่อ n = 6 (3)



รูปที่ ก4 เส้นทางที่กำหนดโดยอัลกอริทึมกำหนดเส้นทางต่าง ๆ เมื่อ n = 8 (1)



รูปที่ ก5 เส้นทางที่กำหนดโดยอัลกอริทึมกำหนดเส้นทางต่าง ๆ เมื่อ n = 8 (2)



รูปที่ ก6 เส้นทางที่กำหนดโดยอัลกอริทึมกำหนดเส้นทางต่าง ๆ เมื่อ n = 8 (3)

ประวัติผู้เขียนวิทยานิพนธ์

นายสมชัย แสงทองสกุลเลิศ เกิดวันที่ 11 เมษายน พ.ศ. 2521 จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2541 และเข้ารับการศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมไฟฟ้า จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2542