อัลกอริทึมแบบผสมสำหรับการจัดกลุ่มวงจรแบบแบ่งส่วนเท่า

HYBRID ALGORITHM FOR BISECTION CIRCUIT PARTITIONING

อภิรดี ยอดเทียน

APIRADEE YODTEAN

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2544

ISBN 974-648-369-2

อัลกอริทึมแบบผสมสำหรับการจัดกลุ่มวงจรแบบแบ่งส่วนเท่า

HYBRID ALGORITHM FOR BISECTION CIRCUIT PARTITIONING

อภิรดี ยอดเทียน

APIRADEE YODTEAN

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2544

ISBN 974-648-369-2

HYBRID ALGORITHM FOR BISECTION CIRCUIT PARTITIONING

APIRADEE YODTEAN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2001

ISBN 974-648-369-2

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ อัลกอริทึมแบบผสมสำหรับการจัดกลุ่มวงจรแบบแบ่งส่วนเท่า

HYBRID ALGORITHM FOR BISECTION CIRCUIT PARTITIONING

ชื่อนักศึกษา นางสาวอภิรดี ยอดเทียน

รหัสประจำตัว 39061049

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชา วิศวกรรมไฟฟ้า

อาจารย์ผู้ควบคุมวิทยานิพนธ์ รศ.ดร.สมศักดิ์ ชุมช่วย

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
ศ.ดร.วัลลภ	สุระกำพลธร	
รศ.ดร.วันชัย	ริ้วรุจา	
รศ.ดร.มนัส	สังวรศิลป์	
รศ.ดร.สมศักดิ์	ชุมช่วย	

วัน/เดือนปี ที่สอบ 17 กรกฎาคม 2544 เวลา 12.00-13.00 น.

สถานที่สอบ ณ อาคาร 12 ชั้น ชั้น 4 (ห้อง E12-404)

บัณฑิตวิทยาลัยรับรองแล้ว

(รศ.ดร.บุญวัฒน์ อัตชู)

คณบดีบัณฑิตวิทยาลัย

วันที่ 9 **เดือน** สิงหาคม **พ.ศ.** ๒๕๔๔

หัวข้อวิทยานิพนธ์	อัลกอริทึมแบบผสมสำหรับการจัดกลุ่มวงจรแบบแบ่งส่วนเท่า
นักศึกษา	นางสาวอภิรดี ยอดเทียน
รหัสประจำตัว	39061049
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมไฟฟ้า
พ.ศ.	2544
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ. ดร. สมศักดิ์ ชุมช่วย

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้ได้ทำการทดลองปรับปรุงหาวิธีการที่ดี สำหรับแก้ปัญหาการจัดกลุ่มวงจร ซึ่งการจัดกลุ่มวงจรเป็นขั้นตอนแรกของการออกแบบวงจรรวมในระดับกายภาพแบบอัตโนมัติ การจัดกลุ่มที่ดีจะส่งผลต่อการลดความสัมพันธ์ระหว่างกลุ่มที่เกิดขึ้นซึ่งจะส่งผลต่อการลดสายสัญญาณที่ใช้เชื่อมโยงเป็นผลให้ลดเนื้อที่การวางแบบ ในการออกแบบวงจรรวมทางกายภาพแบบอัตโนมัติ (IC Physical design Automation) การใช้วิธีการจำลองอัลแนลลิ่งแก้ปัญหาการจัดกลุ่มวงจรนั้น คำตอบที่เกิดขึ้นจะมีเป็นจำนวนมาก การหาคำตอบที่ดีที่สุดจึงต้องอาศัยเวลามาก ในขณะเดียวกันถ้าใช้วิธีเจนนิติกแก้ปัญหาดังกล่าวจะส่งผลให้ต้องใช้หน่วยความจำในการประมวลผลมาก ดังนั้นขั้นตอนวิธีผสมที่นำเสนอจะช่วยลดเวลาในการทำงานและมีขั้นตอนการทำงานที่ไม่ซับซ้อน โดยการหาคำตอบเริ่มแรกจากวิธีเจนนิติก เพื่อกำหนดเป็นเกณฑ์การแบ่งกลุ่มคำตอบออกเป็นสองกลุ่มอย่างคร่าวๆ และพิจารณาเซตคำตอบที่อยู่เหนือหรือต่ำกว่าเกณฑ์ขึ้นอยู่กับฟังก์ชันเป้าหมายที่กำหนด การหาคำตอบใหม่อาศัยแนวทางวิธีการจำลองอัลแนลลิ่งและวิธีเจนนิติก ในส่วนสุดท้ายเป็นการแสดงการเปรียบเทียบผลลัพธ์ที่ได้จากวิธีการผสมกับผลลัพธ์ของวิธีอื่นๆ

Thesis Title	Hybrid Algorithm For Bisection Circuit Partitioning
Student	Miss Apiradee Yodtean
Student ID	39061049
Degree	Master of Engineering
Programme	Electrical Engineering
Year	2001
Thesis Advisor	Assoc. Prof. Dr. Somsak Choomchuay

ABSTRACT

This thesis describes the attempt to improve the optimization technique for circuit partitioning problem. Partitioning the circuit into smaller subcircuits is required as the first step in IC physical design automation in order to reduce the connection among them. The better effect will reflect the better routing area of the layout. Owing to the tremendous complexity of the recent VLSI chip, the solution set to this problem is enormous accordingly. The optimization process may take a long time to reach the desired solution when applying the simulated annealing directly. Meanwhile, the huge storage is needed when the genetic algorithm is applied. The proposed algorithm which is fallen into the class of the hybrid algorithm yields the improving speed with moderate storage. The genetic algorithm is first used to find the initial solution. The solution is then classified into upper and lower sets. The simulated annealing is next applied to the selected set, where the cost function is considered. Experiment results have shown its advantages when compared to the traditional optimization techniques.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยคำแนะนำสั่งสอน คำปรึกษา ตลอดจนความช่วยเหลือในด้านต่างๆ จาก รศ. ดร. สมศักดิ์ ชุมช่วย ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ผู้วิจัยรู้สึกซาบซึ้งในความอนุเคราะห์จากท่านและขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบคุณรุ่นพี่ๆ เพื่อนๆ และนักศึกษารุ่นน้องทุกท่านที่ช่วยเหลือให้คำแนะนำต่างๆ ขอขอบคุณมูลนิธิเพื่อการศึกษาคอมพิวเตอร์และการสื่อสาร (C&C) และบัณฑิตวิทยาลัย ที่ได้ให้ทุนสนับสนุนการศึกษาและทุนทำวิทยานิพนธ์ในครั้งนี้ ตลอดจนคุณพ่อคุณแม่ พี่และน้องที่ได้คอยให้กำลังใจและให้ทุนสนับสนุน

คุณค่าและประโยชน์อันพึงมีจากวิทยานิพนธ์ฉบับนี้ ขอมอบแด่ผู้มีพระคุณทุกท่าน

อภิรดี ยอดเทียน

สารบัญ

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

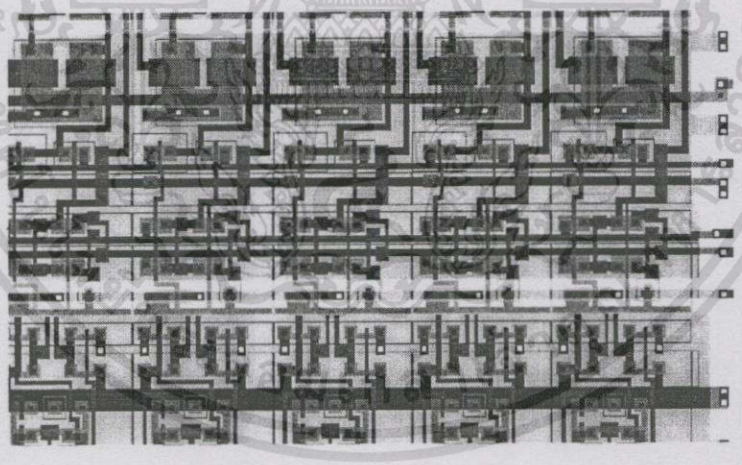
รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ที่มาของงานวิจัย

การพัฒนาวงจรรวมแต่ละประเภทขึ้นอยู่กับความเหมาะสมในการใช้งานที่แตกต่างกัน วงจรรวมมาตราฐาน (Standard Product) มีการออกแบบไม่ซับซ้อนมีขายทั่วไปตามท้องตลาด ส่วนวงจรรวมเฉพาะกิจ (ASIC : Application Specific Integrated Circuit) จะมีความซับซ้อนในการออกแบบเนื่องจากความซับซ้อนของวงจร ปัจจุบันมีการผลิตวงจรรวมเฉพาะกิจมากขึ้นและสามารถเลือกวิธีการออกแบบได้หลายวิธี เช่น Full Custom, Standard Cell, Gate Array และ FPGA [1]

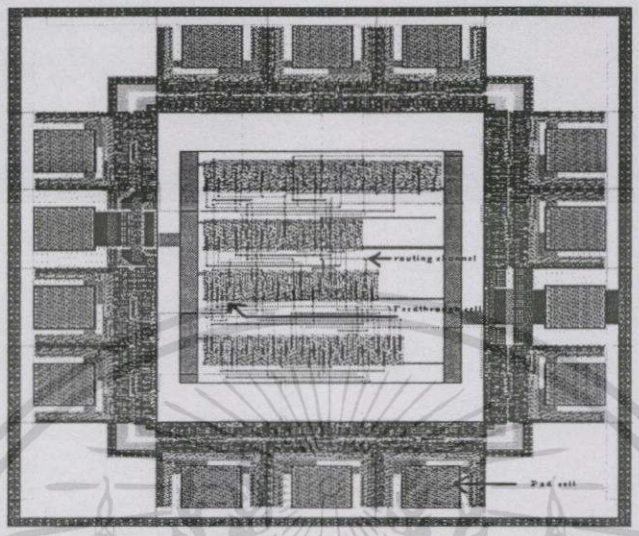
Full Custom ผู้ใช้จะออกแบบตั้งแต่ระดับทรานซิสเตอร์จนถึงวงจรรวมขนาดใหญ่ที่ทำงานได้ตามต้องการ ผู้ออกแบบจะต้องมีความสามารถ ทักษะและประสบการณ์สูง ขั้นตอนการออกแบบ Full Custom เป็นแบบลำดับชั้น (hierarchical) และแบ่งส่วนการออกแบบวงจรเป็นส่วนต่างๆอยู่ในรูปของ block หรือ unit ซึ่งแต่ละ block หรือ unit จะมีขนาดต่างๆกัน ดังรูปที่ 1 การจัดกลุ่มวงจรที่เหมาะสมของวงจรย่อยๆในแต่ละ block จึงเป็นสิ่งที่ต้องคำนึงถึง



รูปที่ 1.1 โครงสร้าง Full-Custom

เซลล์มาตราฐาน (Standard Cell) เซลล์ประเภทนี้มีคุณลักษณะที่สำคัญคือเซลล์มีความสูงคงที่แต่ความกว้างเปลี่ยนแปลงไปตามชนิดของเซลล์ มีความเหมาะสมต่อการจัดวางในลักษณะแถวต่อเนื่อง มีจุดเชื่อมต่อของเพาเวอร์และกราวด์ อยู่ภายในเซลล์ทั้งด้านซ้ายและด้านขวา มีอินพุทและเอาต์พุทอยู่ที่ด้านบนและด้านล่างของเซลล์สะดวกต่อการเลือกจุดเชื่อมต่อที่ด้านบนและด้านล่างของเซลล์ดังรูปที่ 1.2 เซลล์ประเภทนี้ไม่สามารถแก้ไขเปลี่ยนแปลงทางกายภาพของเซลล์ได้ เช่น

พวก Gates, Latches, Adderss เป็นกลุ่มวงจรในระดับ SSI, MSI ในการจัดกลุ่มวงจรที่เหมาะสมในแต่ละแถวจะช่วยลดปัญหาการดีเลย์ (การหน่วงของสัญญาณ) และเนื้อที่ของชิปลง

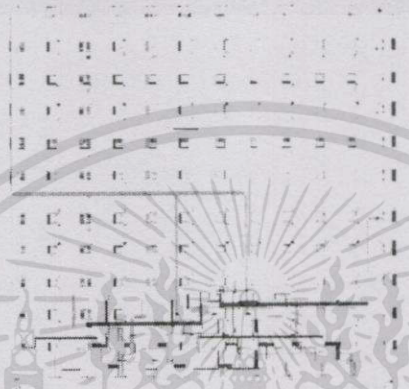


รูปที่ 1.2 โครงสร้างของ Standard Cell

Gate Array เป็นวงจรที่ผู้ผลิตจะบรรจุวงจรพื้นฐานประเภทเกต (gate) แบบต่างๆไว้ในวงจรรวมตัวเดียวกัน และบางส่วนจะมีการต่อสายภายในไว้เป็นกลุ่มวงจรเชิงเลขที่ใช้งานกันทั่วไป เช่น Decoder วงจร Register เป็นต้น ซึ่งเรียกวงจร gate array นี้ว่า Macrocell หรือ Logic-cell ดังรูปที่1.3 ผู้ใช้มีหน้าที่ต่อสายเชื่อมโยงระหว่าง Macrocell แต่ละตัวเข้าด้วยกันให้เป็นวงจรขนาดใหญ่ที่มีคุณสมบัติตามต้องการ แล้วจึงส่งไปทำการเจือสาร ดังนั้นการจัดกลุ่มวงจรในตำแหน่งต่างๆ ที่เหมาะสมจะช่วยลดปัญหาการหน่วงของสัญญาณ

รูปที่ 1.3 แสดงลักษณะทางกายภาพของแมคโครเซลล์ (Macro-Cell)

FPGA (Field Programable Gate Arrays) เป็นอุปกรณ์ที่ใช้ในการโปรแกรมวงจรที่ได้ออกแบบวงจรที่ได้ออกแบบลงไปเพื่อให้ FPGA มีฟังก์ชันการทำงานที่ต้องการ แต่มีข้อจำกัดในด้านขนาดของวงจรเพราะภายในอุปกรณ์ FPGA จะมีจำนวน gate ให้ใช้จำนวนจำกัด จึงเหมาะกับผลิตภัณฑ์ที่เป็นต้นแบบ ซึ่งเป็นการออกแบบวงจรรวมโดยการจำลองการทำงานอีกวิธีหนึ่ง โครงสร้างพื้นฐานของ FPGA แสดงในรูปที่ 1.4

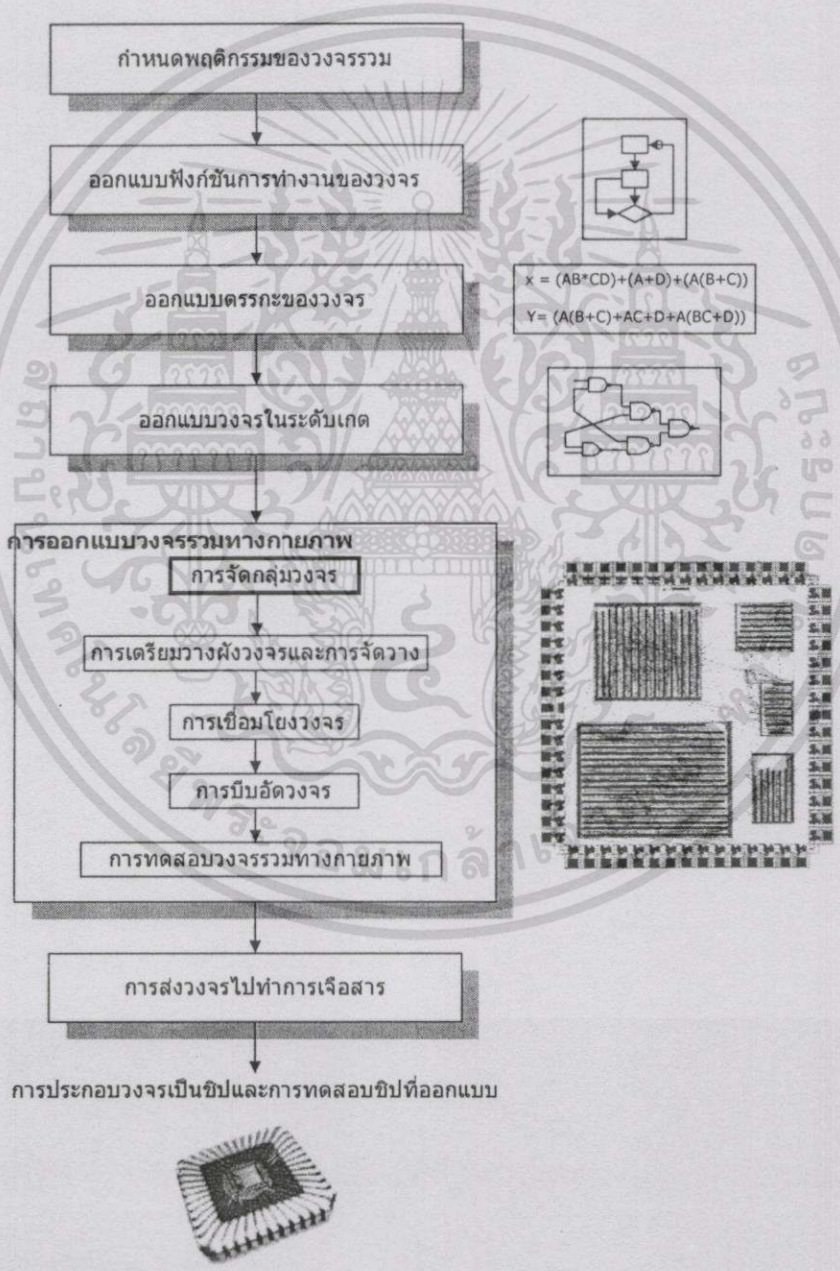


รูปที่ 1.4 แสดงโครงสร้างของ FPGA (Field Programmable Gate Arrays) ในขณะที่มีการจำลองการทำงานของวงจร

การผลิตวงจรรวมเฉพาะกิจ (ASIC : Application Specific Integrated Circuit) แบ่งเป็น 2 ขั้นตอนคือการออกแบบและการเจือสาร ปัจจุบันการออกแบบวงจรรวมในระดับกายภาพ (Physical Design) ได้ใช้คอมพิวเตอร์เข้ามาช่วยทำให้มีความสะดวกรวดเร็ว ตลอดจนการพัฒนาซอฟต์แวร์ช่วยออกแบบในระดับกายภาพบนเครื่อง PC ส่งผลให้อุตสาหกรรมการผลิตวงจรรวมขยายตัวได้อย่างรวดเร็ว ในการออกแบบวงจรรวมต้องผ่านขั้นตอนต่างๆดังแสดงในรูปที่ 1.5 ซึ่งเริ่มต้นจากการกำหนดการทำงานของระบบใหญ่ของวงจรรวม (System Specification) ซึ่งประกอบไปด้วยส่วนย่อยๆของระบบ ในแต่ละส่วนย่อยจะถูกกำหนดคุณสมบัติเป็นฟังก์ชัน (Functional Design) จากนั้นนำมาทำการออกแบบวงจรตรรก (Logic Design), การออกแบบวงจร (Circuit Design), การออกแบบทางกายภาพ (Physical Design) ไปสู่ขบวนการเจือสาร (Fabrication) ตามลำดับ เมื่อไปไอซีหรือชิปกลับมาแล้วก็จะทำการทดสอบการทำงานของชิปว่าตรงกับที่ได้ออกแบบหรือไม่ (Packaging and Testing) เวลาในการทำงานของแต่ละขั้นตอนจึงเป็นสิ่งที่ต้องคำนึงถึง

ในการออกแบบทางกายภาพของวงจรรวมขนาดใหญ่มาก (VLSI Physical design) ยังแบ่งขั้นตอนย่อยออกเป็น การจัดกลุ่มวงจร (Circuit Partitioning) การวางผัง (Floorplanning) การจัดวางตัวอุปกรณ์หรือกลุ่มของตัวอุปกรณ์ (Placement) การเชื่อมโยง (Routing) การบีบอัด (Compaction) และการทดสอบ (Verification) การจัดกลุ่มเซลล์ที่เหมาะสมจะส่งผลต่อการจัดวาง และผลของ

การจัดวางจะมีอิทธิพลต่อการเชื่อมโยง จากการที่ทราบจำนวนวงจรที่แน่นอน การหาคำตอบของการจัดกลุ่มวงจรจึงเป็นปัญหาการหาค่าที่เหมาะสมแบบสุ่ม (Stochastic Optimization Techniques) ของวิธีการสับเปลี่ยน (Combinatorial Optimization) ดังนั้นการจัดกลุ่มวงจรจัดว่าเป็นปัญหาเอ็นพีค็อมพลีท (NP-Complete) ซึ่งมีโอกาสเป็นไปได้หลายรูปแบบ การค้นหารูปแบบการจัดกลุ่มวงจรที่ดีที่สุด (Optimization) จึงต้องอาศัยวิธีการหาค่าที่เหมาะสมแบบสุ่ม ในปัจจุบันมีหลายวิธีเช่น Simulated Annealing Algorithm, Simulated Evolution Algorithm, Ratio Cut, Kernighan-Lin Algorithm และเทคนิคอื่นๆ



รูปที่ 1.5 ขั้นตอนการออกแบบวงจรรวม

1.2 วัตถุประสงค์ของงานวิจัย

ในการออกแบบวงจรรวมทางกายภาพแบบอัตโนมัติต้องผ่านขั้นตอนมากมาย ความรวดเร็วในการทำงานของแต่ละขั้นตอนจึงเป็นสิ่งสำคัญ ปัญหาการจัดกลุ่มวงจรเป็นปัญหาแรกของการออกแบบวงจรรวมในระดับกายภาพ การปรับปรุงวิธีการให้มีประสิทธิภาพสำหรับการจัดกลุ่มวงจร จะส่งผลต่อขบวนการถัดมาของการออกแบบวงจรรวมในระดับกายภาพให้มีประสิทธิภาพและรวดเร็วยิ่งขึ้น ในงานวิจัยนี้ได้ศึกษาวิธีการจัดกลุ่มวงจรที่มีประสิทธิภาพโดยการใช้ขั้นตอนวิธีที่สำคัญ 2 วิธี คือ วิธีการจำลองอัลแนลลิ่ง (Simulated Annealing Algorithm) และวิธีเจนนิติก (Genetic Algorithm) โดยมีจุดประสงค์เพื่อที่จะปรับปรุงขั้นตอนวิธีการดังกล่าวให้มีประสิทธิภาพมากยิ่งขึ้น โดยคาดหวังผลลัพธ์ที่จะได้เป็นความสัมพันธ์ระหว่างกลุ่มที่เกิดจากจัดวางวงจรน้อยที่สุดและเวลาที่ใช้ในการแก้ปัญหาที่เหมาะสม สิ่งต่างๆเหล่านี้จะนำไปสู่ฟังก์ชันเป้าหมายที่เหมาะสมสำหรับการแก้ปัญหาการจัดกลุ่มวงจร [2]

วิธีการจำลองอัลแนลลิ่งเป็นการคำนวณแบบสุ่มที่นำมาจากหลักการกลศาสตร์สถิติ (Statistical Mechanic) สำหรับการหาคำตอบที่เป็นค่าต่ำสุดหรือค่าสูงสุดของกลุ่มคำตอบทั้งหมด (Global Optimum) หรือใกล้เคียงของปัญหาการหาค่าความเหมาะสม เริ่มต้นจากการกำหนดคำตอบเริ่มต้น (Initial Solution) เพียงคำตอบเดียว และกำหนดอุณหภูมิเริ่มต้น (Initial Temperature) จากนั้นเป็นการกระทำซ้ำจนกระทั่งอุณหภูมิเข้าสู่ศูนย์ โดยมีอุณหภูมิเป็นตัวแปรควบคุมและการยอมรับคำตอบเกิดจากการเปรียบเทียบค่าฟังก์ชันเป้าหมายของคำตอบใหม่ (F(j)) กับคำตอบเดิม (F(i)) ถ้าใช้เวลาในการทำงานมากเท่าไร ผลลัพธ์ที่ได้จะเข้าใกล้กลุ่มคำตอบทั้งหมดมากเท่านั้น ดังนั้นวิธีการจำลองอัลแนลลิ่งจึงมีข้อด้อยในเรื่องการใช้เวลามาก ปัจจุบันได้มีการวิจัยเพื่อแก้ข้อด้อยในเรื่องของเวลากันมาก เช่น เสนอวิธีการควบคุมอุณหภูมิแบบใหม่ ผลดีของวิธีการจำลองอัลแนลลิ่งคือขั้นตอนวิธีที่จะนำไปใช้นั้นง่ายต่อการนำไปใช้งานและเหมาะสมกับการแก้ปัญหาที่มีขนาดใหญ่ [3][4][5]

วิธีเจนนิติกเป็นการจำลองกระบวนการวิวัฒนาการทางธรรมชาติ อาศัยหลักการถ่ายทอดทางพันธุกรรมเพื่อให้ได้ประชากรที่ดีขึ้นในรุ่นถัดไป โดยพิจารณาคำตอบที่มีค่าความเหมาะสมสูงสุดเริ่มต้นจากการสุ่มเลือกกลุ่มคำตอบของปัญหามาจำนวนหนึ่งเพื่อเข้ารหัสหรือแปลงค่าตัวแปรของปัญหาให้อยู่ในรูปโครงสร้างของโครโมโซมซึ่งเป็นเลขฐานสองหลายบิทโดยเลขนี้สามารถแปลงให้อยู่ในเลขฐานสิบได้โดยจะสัมพันธ์กับค่าฟังก์ชันจุดประสงค์โครโมโซมในแต่ละรุ่นจะผ่านขบวนการถ่ายทอด (Reproduction) ที่ซับซ้อนจนได้โครโมโซมรุ่นใหม่ ถ้าให้จำนวนรุ่นที่สร้างมากก็จะได้ฟังก์ชั่นจุดประสงค์หรือผลลัพธ์เข้าใกล้คำตอบที่แท้จริง (Global Optimal Solution) วิธีนี้มีขั้นตอนการทำงานที่ยุ่งยาก ใช้เนื้อที่หน่วยความจำในการประมวลผลมาก [6][7][8] เพราะการหาคำตอบหาจากกลุ่มประชากรที่สุ่มได้ในแต่ละรุ่น

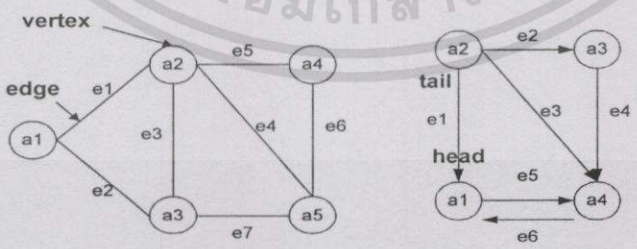
การปรับปรุงขั้นตอนวิธีการผสม (Hybrid Algorithm) เป็นหลักการใหม่ที่นำเสนอในวิทยานิพนธ์ฉบับนี้ ขั้นตอนวิธีจะอยู่บนหลักการของวิธีการจำลองอัลแนลลิ่งและวิธีเจนนิติก ซึ่งเป็นการ

ประยุกต์ใช้อัลกอริทึมเพื่อแก้ปัญหาการจัดกลุ่มวงจร ซึ่งวิธีที่นำเสนอนี้มีการปรับปรุงภายในขั้นตอนต่างๆของวิธีการจำลองอัลแนลลิ่งซึ่งเป็นผลให้ได้วิธีการดังกล่าวมีประสิทธิภาพดีขึ้น เริ่มต้นจากการคัดเลือกคำตอบอย่างคร่าวๆ โดยการตัดกลุ่มคำตอบที่ซ้ำซ้อนออกจากการกำหนดคุณสมบัติบางอย่างของโครโมโซมที่สุ่มเลือกมาจำนวนหนึ่ง แล้วหาค่าคำตอบที่ดีที่สุดจากโครโมโซมที่สุ่มนั้นมาเป็นเกณฑ์การแบ่งกลุ่มคำตอบออกเป็นสองส่วนอย่างหยาบๆ เพื่อทำการพิจารณาคำตอบในกลุ่มใดกลุ่มหนึ่งตามฟังก์ชันเป้าหมายที่กำหนด ส่งผลให้ลดสภาวะแวดล้อมของคำตอบที่จะนำมาพิจารณาลง ทำให้การยอมรับคำตอบสุดท้ายเร็วขึ้นเมื่อเทียบกับวิธีการจำลองอัลแนลลิ่ง ในขณะเดียวกันวิธีการดังกล่าวก็จะช่วยประหยัดการใช้หน่วยความจำได้มากขึ้นหากเทียบกับการทำในขั้นตอนวิธีเจนนิติกเพียงอย่างเดียว วิธีการนี้ยังคงใช้อุณหภูมิเป็นตัวควบคุมการทำงานของอัลกอริทึมเช่นเดียวกับวิธีการจำลองอัลแนลลิ่ง แต่ได้นำเทคนิคการควบคุมอุณหภูมิเข้ามาช่วยเพิ่มประสิทธิภาพให้การทำงานของวิธีการดังกล่าวให้มีประสิทธิภาพดียิ่งขึ้น [9]

1.3 ทฤษฎีที่เกี่ยวข้อง

1.3.1 ทฤษฎีกราฟ (Graph Theory)

ทฤษฎีกราฟเป็นสาขาหนึ่งในคณิตศาสตร์ที่สามารถนำมาประยุกต์ใช้ในสาขาวิชาอื่นๆได้ ทั้งนี้เพราะปัญหาต่างๆ ที่เกิดขึ้นสามารถจัดให้อยู่ในรูปของปัญหาที่เกี่ยวข้องกับการจัดการของวัตถุสิ่งของและความสัมพันธ์ระหว่างสิ่งเหล่านั้น โดยการนำเอากราฟไปใช้สร้างตัวแบบเชิงคณิตศาสตร์ (Mathematical Model) โดยที่กราฟมีโครงสร้างที่ง่าย ซึ่งประกอบไปด้วยจุดและเส้นเชื่อมโยงระหว่างจุดสองจุด จึงทำให้ง่ายต่อการนำกราฟไปใช้งาน กำหนดให้กราฟ G = (V,E) ซึ่งเซต V เป็นสมาชิกของจุดยอด (Vertex) หรือจุด (Point) หรือบัพ (Nodes) และเซต E เป็นสมาชิกของซับเซต V เป็นเส้นที่เชื่อมต่อระหว่างคู่ของจุดในกราฟซึ่งเรียกว่า ขอบ (Edge) [10] ตัวอย่างในรูปที่ 1.6 (a) แสดงกราฟ G = ($\{a_1,a_2,a_3,a_4,a_5\}$, $\{[a_1,a_2],[a_1,a_3],[a_2,a_3],[a_3,a_5],[a_2,a_5],[a_2,a_4],[a_4,a_5]\}$



(a) (b)

รูปที่ 1.6 แสดงกราฟแบบไม่มีทิศทาง

กราฟมีทิศทาง (directed graph or digraph) รูปที่ 1.6 (b) edge e1 <a1,a2> เชื่อม node a1 และ node a2 เราเรียกว่า node a1 และ a2 ว่า adjacent กัน degree ของ node n เท่ากับจำนวน edges ที่ต่อกับ node n เช่น ที่ node a1 มี degree = 3, indegree = 2, outdegree =1

1.3.2 ทฤษฎีความซับซ้อน (Complexity Theory)

เนื่องจากจำนวนผลลัพธ์ที่เป็นไปได้ (Solution Space) มีเป็นจำนวนมากไม่สามารถตรวจสอบทุกคำตอบได้ภายในเวลาพหุนาม (Polynomial Time) ดังนั้นการแก้ปัญหาการจัดวางและการเชื่อมโยง จึงเป็นการหาผลลัพธ์ที่ดีที่สุดภายใต้เงื่อนไขที่กำหนดเท่านั้น ปัญหาการจัดวางเป็นปัญหาประเภทเอ็นพีค็อมพลีท [11] (NP-Complete : Nondeterministic Polynomial Complete) ปัญหาประเภทนี้ไม่มีขั้นตอนวิธี ที่ใช้แก้ปัญหาได้โดยตรงภายในเวลาพหุนาม ต้องใช้วิธีการแปลงไปเป็นปัญหาอื่นที่สามารถแก้ได้ด้วยขั้นตอนวิธีที่ใช้เวลาพหุนามมาใช้แทน คำตอบที่ได้จากปัญหาใหม่เปรียบเสมือนเป็นคำตอบของปัญหาเดิม โดยที่ P-Problem คือ Polynomial Problem ซึ่งปัญหาที่สามารถแก้ไขได้ภายในเวลาพหุนาม ส่วน NP-Problem คือ Non-Deterministic Polynomial Problem เป็นปัญหาใดก็ตามที่ไม่สามารถแก้ได้ภายในเวลาพหุนามและ NP-Complete Problem คือ ปัญหาเอ็นพี ใดๆ ก็ตามที่สามารถแปลงจากปัญหาหนึ่งไปเป็นอีกปัญหาหนึ่งได้ภายในเวลาพหุนาม

1.3.3 การจัดลำดับและการจัดหมู่ (Combinatorial)

โดยทั่วไปปัญหา Combinatorial Optimization (CO) [12] เป็นปัญหา Optimization ที่อยู่ในรูปแบบ Minimize f(x). ตามเงื่อนไข Ax = b โดยที่ฟังก์ชัน f(x) เป็น Objective Function , A เป็น Constraint Matrix ขนาด m x n สังเกตุได้ว่าจำนวนของ Constraints สามารถมีค่าที่ใหญ่มากได้ ทุกๆ Vector x เป็นคำตอบของปัญหาผลลัพธ์ f(x) ต่ำสุดที่เกิดขึ้นเรียกว่า Optimal Solution

การจัดลำดับ (Permutation) เกิดจากการมีสิ่งของอยู่ n สิ่ง ซึ่งแต่ละสิ่งมีลักษณะแตกต่างกัน นำของ r สิ่งจาก n สิ่งมาจัดเรียงเป็นแถวตามลำดับ จำนวนวิธีที่จะกระทำได้คือ

$$P_{n,r} = \frac{n!}{(n-r)!} \text{ วิธี} \tag{1.1}$$

ถ้ามีของ n สิ่งต่างๆกัน นำของทั้ง n สิ่งมาเรียงสับเปลี่ยนแบบวงกลม จำนวนวิธีที่จะจัดได้ทั้งหมดคือ (n-1)! แต่ถ้าเลือกมาเรียงสับเปลี่ยนแบบวงกลมเพียง r สิ่ง (r<n) จำนวนวิธีที่จะจัดได้ทั้งหมดคือ

$$\frac{n!}{(n-r)!r!} \text{ วิธี} \tag{1.2}$$

การจัดหมู่ (Combination) เกิดจากการมีสิ่งของอยู่ n สิ่งซึ่งแต่ละสิ่งมีลักษณะแตกต่างกัน แล้วเลือกมา r สิ่ง (r <= n) มาจัดเป็นกลุ่มหรือหมู่ จำนวนวิธีที่เลือกได้คือ

$$Cn,r = \binom{n}{r} = \frac{n!}{(n-r)!r!} \text{ วิธี} \tag{1.3}$$

1.3.4 ปัญหาทางเลือก (Combinatorial Optimization Problem)

ปัญหาทางเลือกเป็นปัญหาที่มีคำตอบที่เป็นไปได้อยู่แล้วเป็นจำนวนมาก บางปัญหาจำนวนคำตอบที่สามารถเกิดขึ้นได้ทั้งหมดมีอัตราการเพิ่มอยู่ในรูปฟังก์ชันเอกโปเนนเชียล หรือขึ้นกับจำนวนอินพุทของปัญหา การเลือกคำตอบที่ดีที่สุดจากการหาคำตอบทั้งหมดนั้นต้องใช้เวลานานมาก ดังนั้นจึงจำเป็นต้องค้นหาคำตอบที่ดีจากบางส่วนของคำตอบทั้งหมด เราเรียกว่า Local Optimal Solution คำตอบที่ได้จะมีคุณภาพใกล้เคียงคำตอบที่ดีที่สุดเพียงใดนั้นขึ้นอยู่กับวิธีที่ใช้ว่ามีความเหมาะสมกับปัญหาที่ต้องแก้ไขนั้นเพียงใด

1.3.5 หลักการแบ่งกลุ่ม (Partitioning Groups)

การแบ่งกลุ่มแบบ Two-Way Partitioning Problem (TWPP) เป็นที่รู้จักกันดีว่าเป็นวิธีที่นิยมนำมาใช้ในการแบ่งกลุ่มในลักษณะ M-Way Partition ซึ่งมีสองวิธีคือ วิธีแรกนั้นเป็นการแบ่งกลุ่มออกเป็น 2 กลุ่มโดยแต่ละกลุ่มมีขนาดเท่ากันเรียกว่าการแบ่งกลุ่มแบบ Bisection และแต่ละกลุ่มที่ถูกแบ่งจะถูกแบ่งออกไปทีละสองกลุ่ม จนกระทั่งได้จำนวนกลุ่มที่ต้องการ ดังรูป 1.7 ก วิธีที่สองคือการแบ่งกลุ่มออกเป็น 2 กลุ่มโดยมีขนาดของกลุ่มไม่เท่ากันและเมื่อแบ่งครั้งถัดไปจะแบ่งเฉพาะกลุ่มที่มีขนาดใหญ่กว่าเท่านั้น แบ่งจนกระทั่งได้จำนวนกลุ่มตามที่ต้องการ ดังรูป 1.7 ข ข้อเสียของการแบ่งแบบนี้คือในการแบ่งครั้งถัดไปจะไม่นำส่วนที่เล็กกว่าที่ได้แบ่งไปแล้วมาพิจารณาและถ้าจำนวนข้อมูลมีมากจะต้องใช้เวลาในการแบ่งมาก

(ก) (ข)

รูปที่ 1.7 แสดงการแบ่งกลุ่มแบบสองทาง (Two Way Partitioning)

1.3.6 การค้นหาเฉพาะที่ (Local Search Algorithm)

คือการหาคำตอบที่เกิดจากการทำซ้ำโดยเริ่มต้นจากการสุ่มเลือกคำตอบมาหนึ่งคำตอบแล้วทำการหาคำตอบต่อไปโดยการสับเปลี่ยนและยอมรับคำตอบที่ดีกว่าไปเรื่อยๆวิธีการนี้จะหยุดเมื่อไม่มีการเปลี่ยนแปลงค่าคำตอบ แต่ผลลัพธ์ที่ได้อาจไม่ใช่ค่าที่เป็นคำตอบที่ดีที่สุด เพราะเป็นการค้นหาคำตอบเฉพาะที่

1.3.7 เทคนิคการสุ่ม (Sampling Techniques)

แบบแผนการสุ่มตัวอย่างแบบต่างๆ เพื่อให้ได้ตัวแทนที่ดีของประชากรที่กำลังจะศึกษา รวมทั้งวิธีการคำนวณค่าสถิติต่างๆ เพื่อนำไปใช้ในการประมาณค่าพารามิเตอร์ แผนแบบการสุ่มตัวอย่างที่สำคัญ ได้แก่ การสุ่มตัวอย่างแบบง่าย (Simple Random Sampling) การสุ่มตัวอย่างแบบแบ่งชั้นภูมิ (Stratified Sampling) การสุ่มตัวอย่างแบบระบบ (Systematic Sampling) การสุ่มตัวอย่างแบบสุ่มกลุ่ม (Cluster Sampling) และการสุ่มตัวอย่างแบบหลายชั้น (Multi-Stage Sampling) [13]

การสุ่มตัวอย่างแบบง่าย (Simple Random Sampling) คือวิธีการสุ่มเลือกหน่วยตัวอย่าง n หน่วย จากทั้งหมด N หน่วย โดยกำหนดให้ตัวอย่างแต่ละตัวอย่างที่จะเป็นไปได้ทั้งหมด มีโอกาสจะถูกเลือกได้เท่าๆกัน ซึ่งการสุ่มเลือกตัวอย่างดังกล่าวจะทำได้ทั้งชนิดไม่แทนที่กลับคืน (Without Replacement) และชนิดแทนที่กลับคืน (With Replacement) ในการสุ่มเลือกตัวอย่างชนิดไม่แทนที่กลับคืน จำนวนตัวอย่างที่จะเป็นไปได้ทั้งหมดจะเท่ากับ N_{C_n} ตัวอย่าง ดังนั้นความน่าจะเป็นที่ตัวอย่างแต่ละตัวจะถูกเลือกจะเท่ากับ $\frac{1}{N_{C_n}}$ ส่วนในการสุ่มเลือกตัวอย่างชนิดแทนที่กลับคืน จำนวนตัวอย่างที่จะเป็นไปได้ทั้งหมดจะเท่ากับ N^n ตัวอย่าง ดังนั้น ความน่าจะเป็นที่ตัวอย่างแต่ละตัวจะถูกเลือกจะเท่ากับ $\frac{1}{N^n}$

ข้อดีของการสุ่มตัวอย่างแบบง่ายพอสรุปได้คือ เป็นแผนแบบการสุ่มตัวอย่างที่ง่ายและสะดวกต่อการนำไปใช้ เหมาะสำหรับการสุ่มตัวอย่างจากประชากรที่ประกอบด้วยหน่วยตัวอย่างที่มีลักษณะคล้ายคลึงกัน มีการประมาณผลที่ง่าย สามารถคำนวณได้โดยไม่มีความยุ่งยากมากนัก สามารถปรับวิธีการประมาณได้ เป็นต้นว่า ถ้าเก็บรวบรวมข้อมูลได้เพียง n' หน่วย ($n' < n$) ก็สามารถใช้ข้อมูลที่มีอยู่ n' หน่วย ไปคำนวณหาค่าประมาณได้

ข้อเสียของการสุ่มตัวอย่างแบบง่าย คือ เป็นแผนแบบการสุ่มตัวอย่างที่ไม่เหมาะสำหรับประชากรที่ประกอบด้วยหน่วยตัวอย่างที่มีลักษณะที่แตกต่างกันมากๆ เป็นแผนแบบการสุ่มตัวอย่างที่อาจจะเสียค่าใช้จ่ายสูง เพราะว่าอาจจะต้องใช้ขนาดของตัวอย่างเป็นจำนวนมาก เพื่อควบคุมให้ความคลาดเคลื่อนอยู่ภายในขอบเขตที่ต้องการ

การสุ่มตัวอย่างแบบแบ่งชั้นภูมิ (Stratified Sampling) คือวิธีการสุ่มตัวอย่างที่มีการแบ่งประชากรออกเป็นส่วนย่อยแต่ละส่วนย่อยจะถูกเรียกว่า ชั้นภูมิ (Stratum) โดยที่หน่วยตัวอย่างแต่ละหน่วยจะถูกจัดอยู่ในชั้นภูมิใดชั้นภูมิหนึ่งเพียงชั้นภูมิเดียวเท่านั้น ต่อจากนั้นจึงทำการสุ่มเลือกหน่วยตัวอย่างมาจากแต่ละชั้นภูมิโดยวิธีใดก็ได้ การสุ่มตัวอย่างจะทำการสุ่มหน่วยตัวอย่างมาจากทุกชั้นภูมิ ถ้าหากสุ่มตัวอย่างแบบง่ายจากชั้นภูมิทุกชั้นภูมิจะเรียกว่าเป็นการสุ่มตัวอย่างแบบแบ่งชั้นภูมิชนิดสุ่มแบบง่าย (Stratified Random Sampling) ถ้าหากสุ่มตัวอย่างแบบระบบจากชั้นภูมิทุกชั้นภูมิจะเรียกว่าเป็นการสุ่มตัวอย่างแบบแบ่งชั้นภูมิชนิดสุ่มแบบระบบ (Stratified Systematic Sampling)

ข้อดีของการสุ่มตัวอย่างแบบแบ่งชั้นภูมิ พอสรุปได้ดังนี้คือ เป็นแผนแบบการสุ่มตัวอย่างที่ทำให้สามารถเลือกตัวอย่างได้จากทุกประเภทของประชากร สามารถเสนอผลการสำรวจในระดับย่อยได้เช่น ถ้าหากใช้ภาคเป็นชั้นภูมิก็จะสามารถเสนอผลการสำรวจในแต่ละภาคได้ สามารถสุ่มตัวอย่างจากแต่ละชั้นภูมิด้วยวิธีที่แตกต่างได้

ข้อเสียของการสุ่มตัวอย่างแบบแบ่งชั้นภูมิ พอสรุปได้ดังนี้ ถ้ามีการแบ่งประชากรออกเป็นชั้นภูมิมากเกินไป จะทำให้เสียเวลาและเสียค่าใช้จ่ายในการสำรวจเป็นอย่างมาก บางครั้งอาจมีปัญหาในการประมาณค่า เมื่อเก็บข้อมูลมาไม่ครบหรือไม่สมบูรณ์ จะทำให้ไม่สามารถคำนวณค่าได้ เช่น บางชั้นภูมิไม่ได้รับข้อมูลมาเลย ทั้งนี้อาจเนื่องจากมีชั้นภูมิมากเกินไปการประมาณค่าในกรณีนี้จะไม่สามารถทำได้ มีปริมาณงานเพิ่มมากขึ้น ทั้งในด้านการวางแผนการสำรวจ การเก็บรวบรวมข้อมูลและการคำนวณค่าต่างๆ

การสุ่มตัวอย่างแบบระบบ (Systematic Sampling) คือการสุ่มตัวอย่างที่ทำการเลือกหน่วยตัวอย่างแรกแบบสุ่ม จากหน่วยที่ 1 ถึง หน่วยที่ k และต่อจากนั้นจะเลือกหน่วยตัวอย่างต่อไปทุกๆ k หน่วย จนกระทั่งครบ n หน่วยตามที่ต้องการ กล่าวคือ ถ้าเลือกได้หน่วยตัวอย่างแรกเป็นหน่วยที่ i เมื่อ $1 \le i \le k$ หน่วยตัวอย่างที่จะถูกเลือกเป็นตัวอย่างคือ หน่วยตัวอย่างที่ i, i+k, i+2k, i+3k, .. , i+(n-1)k

ข้อดีของการสุ่มตัวอย่างแบบระบบคือ เป็นวิธีที่ง่าย เสียเวลาน้อย เสียค่าใช้จ่ายน้อย และมีความผิดพลาดน้อย มีประสิทธิภาพสูงเมื่อประชากรมีการเรียงลำดับของหน่วยตัวอย่างไว้เป็นอย่างดี ข้อเสียของการสุ่มตัวอย่างแบบระบบ คือ อาจจะได้ขนาดของตัวอย่างไม่ตรงตามที่ต้องการ จะได้ตัวประมาณที่เอนเอียง ถ้าหาก $N \ne kn$ และไม่สามารถหาตัวประมาณที่ไม่เอนเอียงของความแปรปรวนของตัวประมาณได้จากตัวอย่างเพียงตัวอย่างเดียวที่เลือกมาได้

การสุ่มตัวอย่างแบบสุ่มกลุ่ม (Cluster Sampling) คือวิธีการสุ่มตัวอย่างที่มีการรวมหน่วยตัวอย่างเข้าไว้เป็นกลุ่ม Clusters จำนวน N กลุ่ม แล้วทำการสุ่มเลือกตัวอย่างวิธีใดวิธีหนึ่ง การเก็บรวบรวมข้อมูลจะทำการเก็บรวบรวมจากหน่วยตัวอย่างทุกหน่วยในกลุ่มที่ถูกเลือกมาเป็นตัวอย่างเท่านั้นกลุ่มของหน่วยตัวอย่างนี้บางครั้งอาจเรียกว่าเป็นหน่วยตัวอย่างขั้นแรก (Primary Sampling Unit-psu)

ข้อดีของการสุ่มตัวอย่างแบบสุ่มกลุ่ม คือ สามารถทำกรอบตัวอย่างได้สะดวก รวดเร็ว และควบคุมการทำงานได้อย่างสะดวก ข้อเสียของการสุ่มตัวอย่างแบบสุ่มกลุ่มคือ มักจะมีประสิทธิภาพต่ำกว่าการสุ่มตัวอย่างแบบอื่นๆ เพราะว่าการสุ่มตัวอย่างแบบสุ่มกลุ่มนี้ จะเก็บรวบรวมข้อมูลจากหน่วยตัวอย่างที่อยู่ใกล้เคียงภายในกลุ่มเดียวกัน ข้อมูลที่ได้อาจไม่เป็นตัวแทนที่ดี

การสุ่มตัวอย่างแบบหลายชั้น (Multi-Stage Sampling) คือวิธีการสุ่มตัวอย่างที่ทำเป็นขั้นๆ หลายขั้นตอนด้วยกัน การสุ่มตัวอย่างแต่ละขั้นอาจจะใช้แผนแบบการสุ่มตัวอย่างแบบใดก็ได้ และการเก็บรวบรวมข้อมูลจะเก็บจากหน่วยตัวอย่างย่อยที่สุ่มเลือกมาได้ในขั้นสุดท้าย การสุ่มตัวอย่าง

แบบขั้นนั้นนิยมใช้กับประชากรที่มีขนาดใหญ่มากๆ และประชากรจะประกอบด้วยหน่วยตัวอย่าง ซึ่งแบ่งออกได้เป็นหน่วยตัวอย่างย่อยในหลายขั้น การสุ่มตัวอย่างแบบหลายขั้นมีข้อดี คือ การสร้างกรอบตัวอย่างที่สมบูรณ์และทันสมัย จะสามารถทำได้อย่างสะดวกและรวดเร็วเพราะเป็นการสร้างกรอบตัวอย่างสำหรับหน่วยตัวอย่างเป็นขั้นๆ ตามขั้นตอนของการสุ่มตัวอย่าง ทั้งยังมีแผนแบบการสุ่มตัวอย่างที่ประหยัดทั้งเวลาและค่าใช้จ่ายในการดำเนินการขั้นตอนต่างๆ เพราะว่ามีการสุ่มตัวอย่างทุกขั้นตอน และการเก็บข้อมูลจะเก็บจากหน่วยตัวอย่างที่สุ่มเลือกได้ในขั้นสุดท้ายเท่านั้น ข้อเสียที่สำคัญของวิธีนี้คือความยุ่งยากในการปะมวลผลและการวิเคราะห์ข้อมูล ซึ่งจะต้องมีการคำนวณทุกขั้นตอนที่ทำการสุ่มตัวอย่างมาไม่ว่าจะเป็นการประมาณค่าพารามิเตอร์หรือการประมาณค่าความแปรปรวนของค่าประมาณที่ได้ก็ตามแต่

1.4 ขอบเขตของงานวิจัย

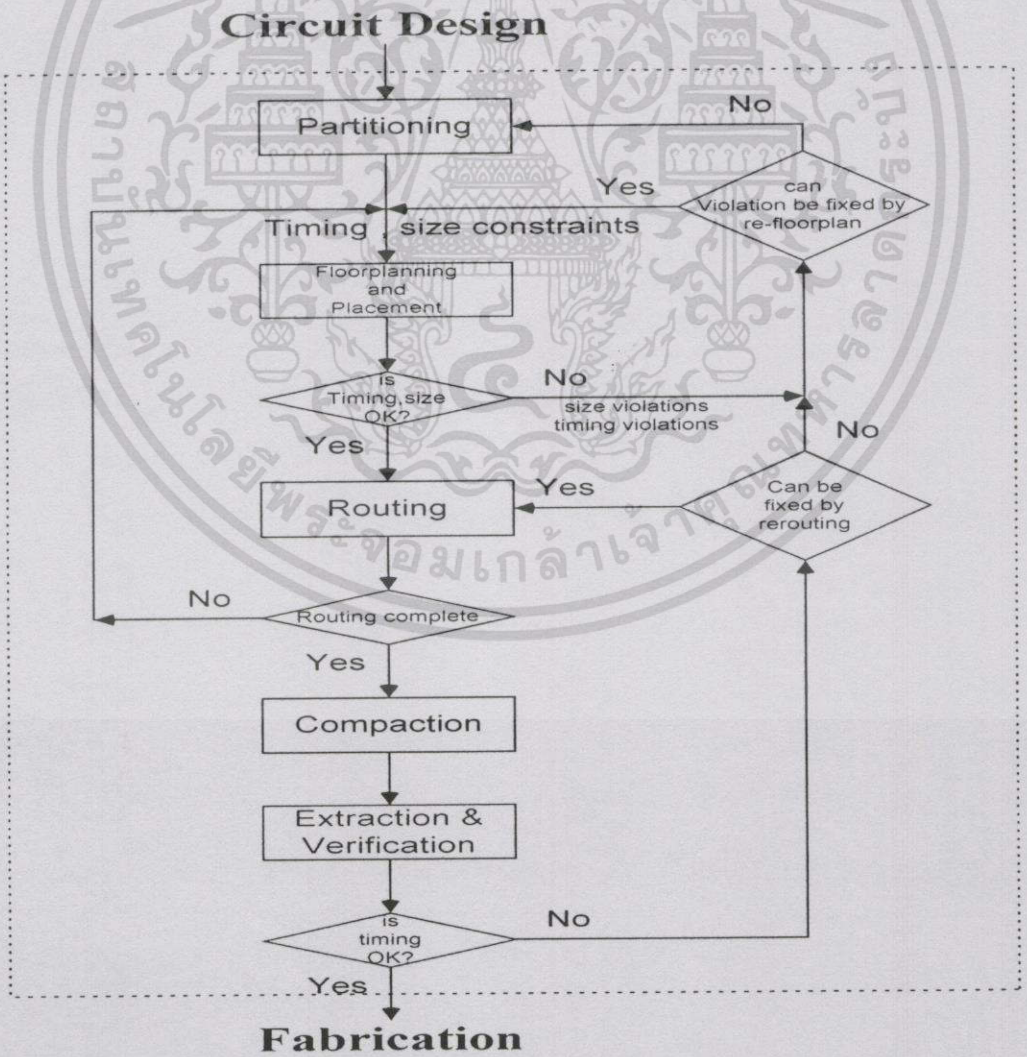
ปัจจุบันการออกแบบวงจรรวมต้องอาศัยคอมพิวเตอร์และซอฟแวร์ที่ช่วยในการออกแบบวงจรในการจำลองการทำงานของวงจรที่ออกแบบ ทำให้เกิดความรวดเร็วในการผลิตชิปเป็นอย่างมาก ดังนั้นความรวดเร็วในการออกแบบวงจรรวมจึงทำให้เกิดการพัฒนาซอฟแวร์ที่ใช้ในการออกแบบวงจรรวมไปในตัวด้วย การค้นหาขั้นตอนวิธีที่มีประสิทธิภาพซึ่งจะนำมาใช้ในการออกแบบซอฟแวร์จึงเป็นเรื่องที่ถูกนำมาพิจารณาและค้นคว้ากันอย่างต่อเนื่อง เพื่อปรับปรุงขั้นตอนการออกแบบวงจรรวมอย่างอัตโนมัติให้มีประสิทธิภาพสูงสุด ดังนั้นงานวิจัยชิ้นนี้คาดหวังว่าการปรับปรุงขั้นตอนวิธีจะส่งผลต่อการปรับปรุงซอฟแวร์ที่ช่วยในการออกแบบวงจรรวมในอนาคต ผลงานวิจัยนี้จะเริ่มต้นกล่าวถึงเนื้อหาในแต่ละบทดังนี้ บทที่ 1 เป็นบทนำกล่าวถึงที่มาของงานวิจัย วัตถุประสงค์ของงานวิจัย ทฤษฎีที่เกี่ยวข้อง และขอบเขตของงานวิจัย บทที่ 2 การจัดกลุ่มวงจร (Circuit Partitioning) กล่าวถึง การออกแบบวงจรรวมในระดับกายภาพ ปัญหาการออกแบบวงจรรวมทางกายภาพ รูปแบบและการกำหนดปัญหาการจัดกลุ่มวงจร เป้าหมายหรือข้อกำหนดของการจัดกลุ่มรูปแบบในการหาคำตอบที่ดี บทที่ 3 การหาคำตอบที่ดีที่สุด กล่าวถึง วิธีการจำลองอัลแนลลิ่ง การจัดกลุ่มวงจรโดยใช้วิธีการจำลองอัลแนลลิ่ง เทคนิคการควบคุมอุณหภูมิ วิธีเจนนิติกแบบทั่วไป การประยุกต์วิธีเจนนิติกแบบง่าย และการจัดกลุ่มวงจรโดยใช้วิธีเจนนิติก บทที่ 4 วิธีการผสม กล่าวถึงที่มาของหลักการวิธีการผสม การจัดกลุ่มวงจรโดยใช้วิธีการผสม บทที่ 5 ผลการทดลองการจัดกลุ่มวงจรแบบอัตโนมัติโดยใช้ขั้นตอนวิธีการผสม การทดสอบกับ ISPD98 Circuit Benchmark Suite ตัวอย่างการทำงานของโปรแกรม เปรียบเทียบผลการจำลองการจัดกลุ่มวงจรที่ได้จากวิธีการจำลองอัลแนลลิ่ง วิธีเจนนิติก และวิธีการผสมที่กล่าวถึง ตลอดจนการจัดกลุ่มวงจรที่ได้จากวิธีอื่นๆ เช่น FM (Fiduccia-Mattheyses), Simple Eigenvector clustering Algorithm, LA (Krishnamurthy's Look-Ahead), CLIP (Cluster Detecting Iterative-improvement Partitioner), Tabu, Eigenvector, Clustering อื่นๆ บทที่ 6 บทสรุปและวิจารณ์ วิธีการผสมกันแนวทางในการแก้ปัญหาอื่นๆ

บทที่ 2

การจัดกลุ่มวงจร (Circuit Partitioning)

โดยทั่วไป Circuit Partitioning Problem เป็น K-Way Partitioning Problem ซึ่งสามารถแสดงด้วย การแก้ปัญหาการจัดกลุ่มวงจรด้วยกราฟ (Graph Partitioning Problem) เป็นปัญหาประเภท NP Complete (Non-Deterministic Polynomial Complete) ซึ่งไม่มีวิธีการใดที่ใช้แก้ปัญหาได้โดยตรงภายในเวลาพหุนาม และปัญหาการจัดกลุ่มวงจรเป็นปัญหาแรกของการออกแบบวงจรรวมทางกายภาพแบบอัตโนมัติ ซึ่งผลของการแก้ปัญหาการจัดกลุ่มวงจรจะส่งผลต่อปัญหาการจัดวางกลุ่มวงจรและการเชื่อมโยงกลุ่มวงจร

2.1 การออกแบบวงจรรวมในระดับกายภาพ



รูปที่ 2.1 แสดงขั้นตอนการออกแบบวงจรรวมในระดับกายภาพ

จากรูปที่ 2.1 การออกแบบวงจรรวมในระดับกายภาพ แบ่งปัญหาออกเป็น 2 ปัญหาใหญ่ๆ คือ

2.1.1 ปัญหาการจัดวาง (Placement)

การจัดวางตำแหน่งวงจรเป็นขั้นตอนสุดท้ายของการจัดวางแบบลำดับชั้น ภายหลังจากการจัดกลุ่มวงจรและการวางผัง ซึ่งเป็นการหาตำแหน่งที่ดีที่สุดบนเลเอาต์ที่ให้ความยาวโดยประมาณของสายสัญญาณทั้งหมด การจัดวางเป็นการช่วยให้เราสามารถเชื่อมโยงสายสัญญาณต่างๆได้สะดวกขึ้น สำหรับจำนวน m วงจร โอกาสที่เกิดการจัดวางวงจรที่ติดกันได้ m! แบบโดยไม่รวมวงจรที่เป็น Rotation-Imaging และ Mirror-Imaging การจัดวางควรพิจารณาการ Wiring ทั้งหมดของงานโดยเป้าหมายคือให้เส้นที่เชื่อมต่อกันสั้นที่สุด หรือมีความหนาแน่นของเส้นเชื่อมต่อน้อยที่สุด การจัดวางขึ้นอยู่กับความรู้ของผู้ออกแบบในด้านโครงสร้าง และวิธีการจัดวางแบ่งออกเป็น 3 วิธีโดยคราวๆ

Constructive Initial Placement (จัดโครงสร้างเริ่มต้น) พิจารณาการเชื่อมต่อกันระหว่างวงจรเป็นสิ่งที่ต้องทำเป็นอันดับแรก

Branch-and-Bound Methods ทำการแบ่งวงจรจากกลุ่มใหญ่ๆ ให้เป็นกลุ่มย่อยโดยพิจารณาความสัมพันธ์กัน

An Interactive Re-Placement Procedure สำหรับบางวงจรที่มีการเปลี่ยนแปลงเงื่อนไขจากการเชื่อมต่อ

2.1.2 ปัญหาการเชื่อมโยง (Routing)

คือการเชื่อมต่อฟังก์ชันต่างๆ และ I/O การเชื่อมโยงจะเกิดขึ้นได้ก็ต่อเมื่อทราบตำแหน่งที่แน่นอนของจุดเชื่อมต่อนั่นคือจะต้องทราบตำแหน่งที่แน่นอนของวงจรที่ถูกจัดวางไว้เรียบร้อยแล้ว ข้อกำหนดของปัญหาการเชื่อมโยงคือจะต้องไม่เกิดการลัดวงจรขึ้นหลังจากที่มีการเชื่อมโยงและจะต้องเชื่อมโยงสายสัญญาณได้ครบถ้วน ซึ่งการเชื่อมโยงจะมีประสิทธิภาพดีเพียงใดนั้นขึ้นอยู่กับประสิทธิภาพของการจัดวาง และประสิทธิภาพการเชื่อมโยงโดยตรง

และจากรูปที่ 2.1 จะเห็นได้ว่าการจัดกลุ่มหรือการแบ่งกลุ่มวงจรจะส่งผลต่อขั้นตอนถัดมาเสมอ ดังนั้นในการวิจัยนี้เป็นการวิจัยเพื่อเน้นการจัดกลุ่มวงจรซึ่งเป็นขั้นตอนแรกของการออกแบบวงจรรวมในระดับกายภาพ ผลที่ได้จากการจัดกลุ่มวงจรที่ดีจะมีอิทธิพลต่อการจัดวางและการเชื่อมโยงที่มีประสิทธิภาพ [14]

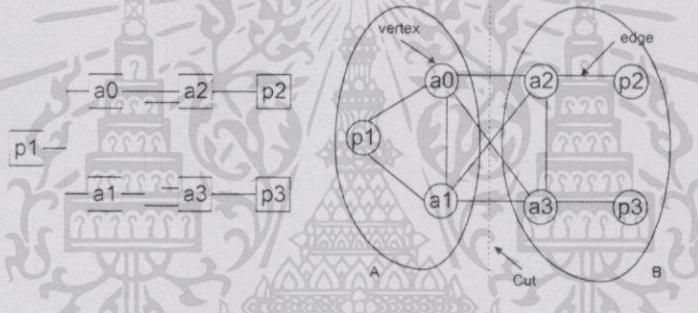
2.2 รูปแบบและการกำหนดปัญหาการจัดกลุ่มวงจร (Models and Definitions)

ปัญหาการจัดกลุ่มจัดเป็นปัญหาประเภทเอ็นพีคอมพลีท ปัญหาประเภทนี้ไม่มีวิธีการใดที่ใช้แก้ปัญหาได้โดยตรงภายในเวลาพหุนาม จึงต้องใช้วิธีการแปลงปัญหาจากปัญหาหนึ่งไปเป็นปัญหาอื่นที่สามารถแก้ไขได้ด้วยวิธีการใดๆ ที่ใช้ในเวลาพหุนาม ซึ่งคำตอบที่ได้จากการแก้ปัญหาใหม่เปรียบเสมือนเป็นคำตอบของปัญหาเดิม [15][16]

การเริ่มต้นการจัดกลุ่มวงจรจากการแทน Circuit Network ด้วยกราฟ G(V,E) ดังรูปที่ 2.2 แต่ละวงจรประกอบด้วยทรานซิสเตอร์หลายๆ ตัว กำหนด V (Vertex) แทนสมาชิกแต่ละวงจรและ E(Edge) แทนความสัมพันธ์ระหว่างวงจร ให้ C_{ij} เป็นความสัมพันธ์ของเส้นเชื่อมโยง (Edge) ที่เชื่อมต่อระหว่าง Vertex i และ j กำหนดให้เส้นตัด (Cut) แบ่งกลุ่ม A และ B ซึ่ง B = V-A จะได้ C_{AB} เป็นผลรวมของค่าความสัมพันธ์ระหว่างกลุ่มที่แบ่งทุก Vertex

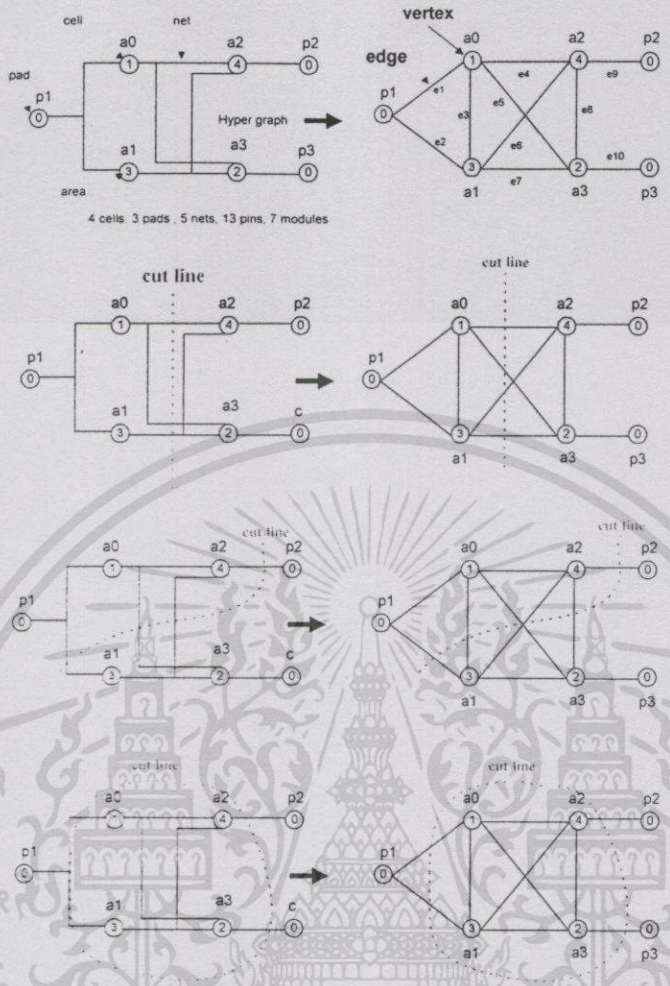
$$C_{AB} = \sum_{i \in A} \sum_{j \in B} C_{ij} \tag{2.1}$$

การแบ่งกลุ่มในที่นี้เป็นแบบ Two-Way Partitioning Problem คือแบ่งปัญหาออกเป็นสองทางหรือสองกลุ่มโดยอาศัยหลักการแบ่งแบบ Bisection คือแต่ละกลุ่มที่แบ่งจะมีขนาดเท่ากัน



รูปที่ 2.2 แสดงการแทนกลุ่มวงจร (Circuit Network) ด้วยกราฟ

อย่างเช่นวงจรที่อยู่ในรูปของ Hypergraph ถูกแทนด้วยกราฟและเมื่อทำการแบ่งกลุ่มแบบ Bisection Two-way Partitioning จะได้คำตอบที่สามารถเกิดขึ้นได้หลายวิธี ถ้าต้องหาทุกคำตอบย่อมพบคำตอบที่ดีที่สุดแน่นอน แสดงดังรูปที่ 2.3 หากในกรณีที่จำนวนวงจรมีมาก คำตอบที่สามารถเกิดขึ้นได้ย่อมมีเป็นจำนวนมาก การที่จะหาทุกคำตอบนั้นย่อมทำให้เสียเวลามาก ดังนั้นการหาคำตอบที่เหมาะสมภายในเวลาที่รวดเร็วที่สุด ดังนั้นงานวิจัยดังกล่าวจึงถูกนำมาใช้ในการออกแบบวงจรรวมแบบอัตโนมัติ



รูปที่ 2.3 การแทนปัญหาการจัดกลุ่มวงจรด้วยกราฟและการหาคำตอบที่เกิดขึ้น

2.3 เป้าหมายหรือข้อกำหนดของการจัดกลุ่ม (Cost Function or Constraints)

ปัญหาการจัดกลุ่มวงจรนี้มีเป้าหมายเพื่อจะลดความสัมพันธ์ที่เกิดจากการข้ามกลุ่มที่แบ่งให้น้อยที่สุดซึ่งจะส่งผลให้การเชื่อมโยงสายสัญญาณระหว่างกลุ่มของวงจรลดลง ดังนั้นข้อกำหนดของปัญหาที่คำนึงถึงคือ [16][17][18][19] การลดความสัมพันธ์ภายนอกระหว่างกลุ่มที่แบ่งให้น้อยที่สุด (Minimize external wiring)

$$\cos t = \sum_{e \in \Psi} w(e) \tag{2.2}$$

w(e) หรือ Weight บน Edge e ของกราฟวงจรแทนจำนวนความสัมพันธ์ระหว่างกลุ่ม Ψ เป็นจำนวนของเส้น Edge ทั้งหมดที่เกิดจากการแบ่งกลุ่ม ฟังก์ชันเป้าหมาย (Cost Function) ก็คือค่าความสัมพันธ์ระหว่างกลุ่มดังสมการที่ 2.2 ถ้าให้แบ่งกลุ่มออกเป็น k กลุ่มโดยให้ p(u) แสดงจำนวนที่แบ่งของ Node u สภาวะ $e \in \psi$ สามารถเขียนได้เป็น e = (u,v) และ p(u) ≠ p(v) สามารถเขียนเป็นสมการที่ 2.3

$$Cost = \sum_{\forall e=(u,v)\&p(u)\neq p(v)} w(e) \tag{2.3}$$

Cost เป็นผลรวมของความสัมพันธ์ระหว่างกลุ่มที่เกิดขึ้น ถ้าสามารถลดค่าความสัมพันธ์นี้ได้ จะส่งผลต่อการลดการการหน่วงเวลาของสัญญาณและลดเวลาที่ใช้ในการเชื่อมโยง

ขนาดของกลุ่มที่จัดแบ่ง (Bounded Size Partitioning) การแบ่งวงจรออกเป็น 2 ส่วนจนได้ k ส่วน เป็นปัญหา K-Way Partitioning โดยทั่วไป ข้อบังคับของขนาดถูกแสดงโดยที่ว่างของขนาดแต่ละวงจรย่อย ขนาดของวงจรย่อยที่ i กำหนดโดย $\sum_{v\in V_i} s(v)$ ถ้าขนาดของวงจรย่อยเป็น A_i จะได้สมการที่ 2.4

$$\sum_{v\in V_i} s(v) \leq A_i \tag{2.4}$$

การแบ่งกลุ่มวงจรออกเป็นสองส่วนโดยมีขนาดเท่าๆกันอย่างหยาบๆ ได้จาก

$$|V_i| = \sum_{v\in V_i} s(v) \leq \left\lceil \frac{1}{k} \sum_{v\in V} s(v) \right\rceil = \frac{1}{k}|V| \tag{2.5}$$

โดยที่ $|V_i|$ และ $|V|$ เป็นขนาดของเซต V_i และ V ตามลำดับ ถ้าสมาชิกวงจรทั้งหมดมีขนาดเดียวกัน สมการที่ 2.5 จะลดรูปเป็น

$$n_i \leq \frac{n}{k} \tag{2.6}$$

ที่ n_i และ n เป็นจำนวนสมาชิกของ V_i และ V ตามลำดับ

การจัดกลุ่มวงจรอาจจะเป็นสาเหตุที่ทำให้เกิดเส้นทางที่มีการหน่วยของสัญญาณมากที่สุด (critical path) ที่อยู่ระหว่างจำนวนครั้งการแบ่งกลุ่ม การหน่วงเวลาของสัญญาณระหว่างกลุ่มที่แบ่งเป็นองค์ประกอบที่สำคัญที่ต้องพิจารณาในการกระทำการจัดกลุ่มวงจร (partitioning high performance circuits)

2.4 เทคนิคในการหาคำตอบที่ดีแบบต่างๆ

จากการที่ปัญหาการจัดกลุ่มวงจรเป็นปัญหาประเภท NP Complete (Non-Deterministic Polynomial Complete) ซึ่งไม่มีวิธีการใดที่ใช้แก้ปัญหาได้โดยตรงภายในเวลาพหุนาม ดังนั้นปัญหาการจัดกลุ่มวงจรจึงเป็นปัญหาการหาค่าที่เหมาะสมภายในเวลาที่กำหนด ทำให้นักวิจัยหลายท่านได้ค้นหาวิธีการต่างๆ เพื่อที่จะหาคำตอบของปัญหาดังกล่าว อาทิเช่น

2.4.1 Kernighan-Lin algorithm

Kernighan-Lin algorithm [2][14][19] เป็น iterative improvement algorithm สำหรับแก้ปัญหา TWPP (Two-Way Partitioning Problem) เริ่มต้นจากการแบ่งกลุ่มในครั้งแรก (initially partitioning)

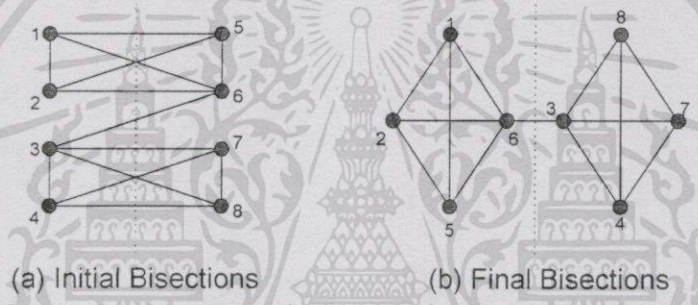
G(V,E) เป็น 2 ส่วนที่มีขนาดเท่ากัน คู่ของกลุ่มวงจร (Vertex pairs) มีการเปลี่ยนเส้นทางทำให้ค่า Cost เปลี่ยนไปจากรูปที่ 2.4 (a) แบ่งกลุ่มออกเป็น A ={1,2,3,4} กับ B = {5,6,7,8} เลือกคู่ของกลุ่มวงจรที่จะเปลี่ยนผลลัพธ์ในการลดค่าของ Cost ได้ เกิดจาก

$$D(i) = In\_edge(i) - Out\_edge(i) \tag{2.7}$$

โดยที่ In_edge(i) เป็นค่าความสัมพันธ์ระหว่าง Vertex ภายในกลุ่มเดียวกัน ส่วน Out_edge(i) เป็นค่าความสัมพันธ์ที่อยู่ต่างกลุ่ม การแก้ปัญหาอยู่ในรูปของ Square Matrix C ผลลัพธ์ของการจัดกลุ่มจะได้เซต A กับ B โดยที่ $|A| = |B| = n$ และ $A \cap B \neq \varnothing$ มีขนาดของ Cost เป็น T

$$T = \sum_{a \in A, b \in B} c_{ab} \tag{2.8}$$

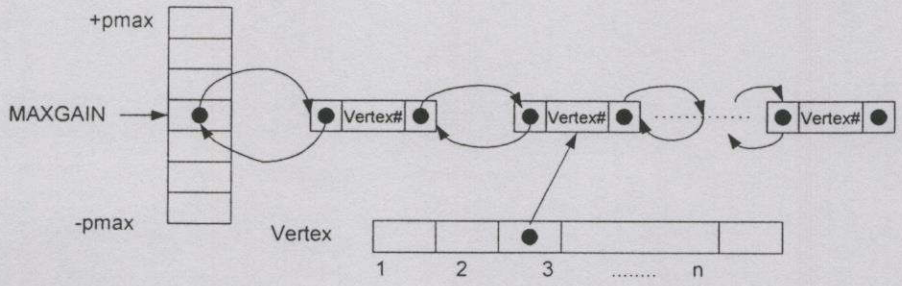
ผลลัพธ์ของการแก้ปัญหาจะแสดงอยู่ในรูปที่ 2.4 (b)



รูปที่ 2.4 การแบ่งกราฟแบบ bisection โดยใช้ KL algorithm

2.4.2 Fiduccia-Mattheyses Algorithm

Fiduccia-Mattheyses Algorithm [20] พัฒนามาจากวิธี KL โดยปรับปรุงการย้ายคู่ของ Vertex เป็นการย้ายเพียง Vertex เดียว ที่ข้ามกลุ่ม เป็นการแบ่งกลุ่มแบบมีขนาดของกลุ่มเท่าๆกัน การย้าย Vertex แบบนี้ทำให้การทำงานของวิธีการนี้เร็วกว่า KL มีโครงสร้างข้อมูลดังรูปที่ 2.5 สำหรับการเลือกย้าย Vertex ถัดไป โครงสร้างข้อมูลที่ใช้ในการเลือก Vertex อยู่ในช่วง –pmax ถึง +pmax มีลักษณะเป็น bucket



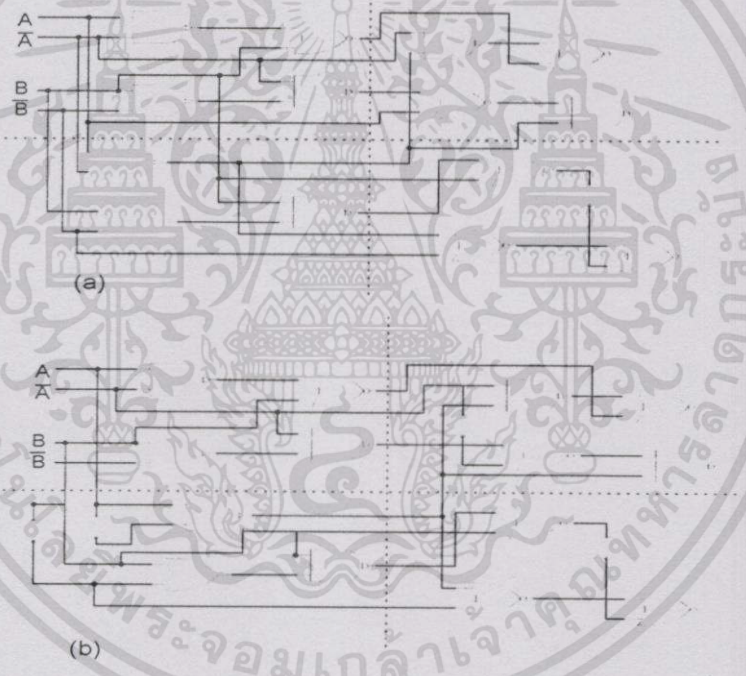
รูปที่ 2.5 โครงสร้างข้อมูลการเลือกคำตอบของวิธี Fiduccia-Mattheyses

2.4.3 Component Replication

การใช้เทคนิค Component (vertex) Replication เป็นการใช้เทคนิคของการเพิ่มวงจรลอจิกเพื่อไปลดจำนวนเส้นทางให้น้อยลง การแก้ปัญหาการจัดกลุ่มหรือแบ่งกลุ่มวงจรจาก V ออกเป็น $V_1,V_2,\ldots,V_c$ นั้นถูกกำหนดได้เป็น

$$V_i \cap V_j = \phi, i \neq j$$
$$U_{i=1}^{k} V_i = V \tag{2.9}$$

สามารถลดจำนวนของ nets ที่เกิดจากการข้ามกลุ่มได้ สังเกตจากรูปที่ 2.6 (a) ซึ่งเป็นการแบ่งกลุ่มวงจรแบบไม่อาศัย Component (vertex) Replication และเมื่อเพิ่มเกต Inverters ที่เป็นการใช้เทคนิคของ Component (vertex) Replication จะได้ดังรูปที่ 2.6(b) ซึ่งจำนวน Cost จะลดลง [21]



รูปที่ 2.6 การลดค่า Cost โดยวิธี component (vertex) replication

2.4.4 Ratio Cut [22]

Ratio Cut เป็นวิธีการแบ่งกลุ่มที่ถูกคิดขึ้นโดย Wei และ Cheng ซึ่งเป็นวิธี Metric ใหม่ในการหาตำแหน่งของกลุ่มที่แบ่งพร้อมๆกับแบ่งกลุ่มโดยคำนึงถึงขนานที่เท่าๆกัน โดยกำหนดให้ Hypergraph G(V,E) และให้ C_{ij} เป็นจำนวนเส้นของ edge ที่เชื่อมต่อ node i และ j โดย (V1,V2) เป็นกลุ่มที่ถูกแบ่ง จะได้ว่า V2 = V-V1 ดังนั้นจำนวนความสัมพันธ์ระหว่างกลุ่ม (Cost) หาได้จาก

$$C_{v_1v_2} = \sum_{i \in v_1} \sum_{j \in v_2} c_{ij} \tag{2.10}$$

Ratio Cut ที่ $R_{v_1v_2}$ ถูกกำหนดให้เป็น $R_{V_1V_2} = \frac{C_{V_1V_2}}{|V_1|x|V_2|}$, ที่ $|V_1|$ และ $|V_2|$ กำหนดเป็นเซตย่อยของ V_1 และ V_2 ตามลำดับ

ยังมีวิธีการอีกมากมายที่ใช้แก้ปัญหาการจัดกลุ่มวงจร เพื่อให้ได้ค่าที่เหมาะสมที่สุดที่นอกเหนือจากวิธีการดังกล่าวเช่น TS (TABU Search) [23] , FCB (Fuzzy-Clustering-Based) [24] และอื่นๆ

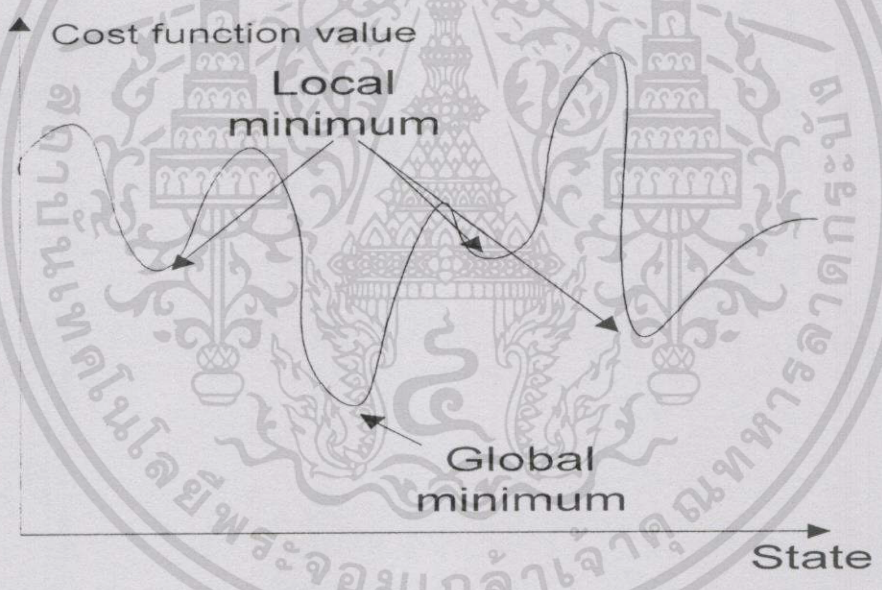
2.5 สรุป

จากที่ได้กล่าวมาข้างต้นจะเห็นได้ว่าปัญหาการจัดกลุ่มวงจรเป็นปัญหาที่มีคำตอบเกิดขึ้นได้มากมาย ถ้าหาทุกคำตอบจะพบคำตอบการจัดกลุ่มที่ดีที่สุดอย่างแน่นอน แต่นั้นหมายถึงต้องใช้เวลามากในการหาทุกคำตอบแล้วเลือกคำตอบที่ดีที่สุด ดังนั้นจึงได้มีการค้นหาเทคนิควิธีการหาค่าเหมาะสมต่างๆ มาใช้แก้ปัญหาการจัดกลุ่ม เพื่อหาคำตอบที่ดีที่สุดในเวลาที่กำหนด กล่าวได้ว่าปัญหาการจัดกลุ่มวงจรเป็นปัญหาการหาค่าที่เหมาะสม ซึ่งไม่มีขั้นตอนวิธีใดที่ใช้แก้ปัญหาได้โดยตรงภายในเวลาพหุนาม จึงต้องอาศัยวิธีการแก้ปัญหาทางเลือก (Combinatorial Optimization Problem) ซึ่งมีด้วยกันหลายวิธี ซึ่งในบทถัดไปจะเสนอวิธีจำลองอัลแนลลิ่งกับวิธีเจนนิติก ซึ่งเป็นเทคนิควิธีการหาค่าคำตอบที่นิยมใช้แก้ปัญหาการหาค่าที่ดีที่สุด

จากการที่ปัญหาการจัดกลุ่มวงจรเป็นปัญหาแรกที่ต้องคำนึงถึงของการออกแบบวงจรรวมทางกายภาพแบบอัตโนมัติ และถ้าสามารถแก้ปัญหาการจัดกลุ่มให้ได้คำตอบที่ดีที่สุด นั่นหมายถึงผลลัพธ์การจัดกลุ่มให้ค่าความสัมพันธ์ระหว่างกลุ่มออกมาน้อยที่สุด จะส่งผลต่อขั้นตอนการจัดวางที่ได้รูปแบบที่ดี ไม่ต้องย้อนกลับไปทำการจัดวางใหม่ และส่งผลต่อการเชื่อมโยงวงจรรวม โดยจะลดเวลาที่ใช้ในการเชื่อมโยงคือไม่ต้องการกลับทำการเชื่อมโยงใหม่อีก ตลอดจนวงจรที่ออกแบบได้มีค่าการหน่วงของสัญญาณน้อย

บทที่ 3
การหาคำตอบที่ดีที่สุด

การค้นหาผลลัพธ์ตามหลักการ Optimization Problem สามารถทำได้โดยการกำหนดสถานะที่เป็นไปได้ (Solution Space) และฟังก์ชันเป้าหมาย (Cost Function) เพื่อเป็นค่าวัดเชิงปริมาณในการค้นหาผลลัพธ์ที่ให้ค่าฟังก์ชันเป้าหมายออกมาต่ำสุดภายในเงื่อนไขที่กำหนด เริ่มต้นด้วยการกำหนดสถานะหลังจากนั้นเป็นการทำซ้ำโดยเลือกสถานะใหม่กับสถานะเดิม ถ้าค่าฟังก์ชันเป้าหมายใหม่น้อยกว่า (หรือมากกว่า) ค่าฟังก์ชันเป้าหมายเดิม เราจะยอมรับสถานะใหม่แล้วกลับไปทำซ้ำเช่นนี้จนครบทุกสถานะรอบข้าง จากหลักการข้างต้นเราเรียกว่า Local Search หรือ Iteractive Improvement Algorithm นั่นเอง ผลลัพธ์ที่ได้จะมีคุณภาพเพียงใดขึ้นอยู่กับวิธีการว่าสามารถให้ค่าผลลัพธ์ได้ใกล้เคียงกับ Global Optimal Solution เพียงใด



รูปที่ 3.1 Local and Global Optimum

วิธีการที่ใช้หาค่าเหมาะสมมีหลายวิธี แต่สำหรับผลงานวิจัยนี้ได้ศึกษาวิธีจำลองอัลแนลลิ่ง Simulated Annealing Algorithm (SA) และวิธีเจนนิติก Genetic Algorithm (GA) ซึ่งทั้งสองขั้นตอนวิธีอาศัยหลักการสุ่มและการทำซ้ำเพื่อให้ได้คำตอบที่ใกล้เคียงคำตอบที่ดีที่สุด เทคนิคในการหาคำตอบของแต่ละเทคนิคก็มีความจำเป็นเพราะว่าเกี่ยวข้องกับเวลาในการแก้ปัญหาให้ถูกต้องด้วยเวลาที่เหมาะสม ปัญหาการ Optimization มีคำตอบมากมายขึ้นอยู่กับขนาดของปัญหา ซึ่งเป็นแบบ Exponential ดังนั้น การแก้ปัญหาจึงต้องใช้ เทคนิคเฉพาะของแต่ละปัญหา (Heuristic Technique) เพื่อให้การแก้ปัญหานั้นสะดวกและรวดเร็วขึ้น

3.1 วิธีการจำลองอัลแนลลิ่ง (Simulated Annealing Algorithm)

3.1.1 วิธีการจำลองอัลแนลลิ่ง

อัลแนลลิ่งเป็นรูปแบบหนึ่งสำหรับแก้ปัญหาแบบ Combinatorial Optimization โดย Simulated Annealing เป็น Heuristic Solution Strategy Optimization ได้หลายรูปแบบ ซึ่งเรียกว่า "Annealing Algorithm " เกิดขึ้นจากความคิดของ Kirkpatrick et al [2][3][5] วิธีการจำลองอัลแนลลิ่งมีลักษณะคล้ายกับ Iterative Improvement ซึ่งเป็นการนำ Local Search Algorithm มาใช้กับ Metropolis Algorithm วิธีการจำลองอัลแนลลิ่งมีลักษณะการพิจารณาที่เป็นแบบกระตุ้น (Uphill) หรือการยอมรับคำตอบใหม่ที่มีค่าไม่ดีกว่าคำตอบก่อน และเมื่อพิจารณาคำตอบที่เกิดขึ้นทั้งหมดแล้วจะพบว่ามีคำตอบที่เป็น Local มากมายและจะพบคำตอบที่ดีที่ยอมรับได้

เปรียบเทียบหลักการทำงานของวิธีการนี้เสมือนการค่อยๆ เย็นตัวลง (Physical Annealing) ของของแข็ง โดยให้ความร้อนกับมันแล้วค่อยๆลด อุณหภูมิลงอย่างช้าๆ จนเข้าสู่จุดสมดุลซึ่งการจำลองกระบวนการนี้ คล้ายกับรูปแบบของการ Combinatorial Optimization [3][4][5][9]

เป้าหมายทางด้านกายภาพคือการจัดเรียงกลุ่มของอะตอม (Configuration) ให้มีพลังงานต่อระบบน้อยที่สุด ดังนั้นกระบวนการจำลองนี้คือทำอย่างไรให้ได้จุดสมดุลทาง Thermodynamic สำหรับแต่ละอุณหภูมิในกระบวนการลดอุณหภูมิลง คล้ายกับการควบคุมการเย็นตัวลงนั้นเอง วิธีการนี้ถูกใช้ใน Metropolis Algorithms ดังรูปที่ 3.2

```
M = number of moves to attempt
T = current temperature;
For m = 1 to M{
   Generate a random move, e.g., move a particle;
   Evaluate the change in energy, ΔE;
   if ΔE < 0    /* downhill move : accept it */
         then  accept this move , and update configuration;
   else      /* uphill move : accept maybe */
         accept with probability P = exp (-ΔE/T) ;
         update configuration if accepted
END
```

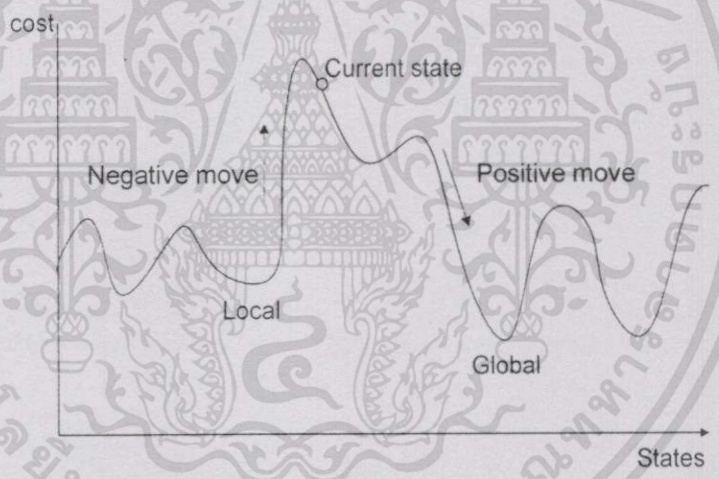
รูปที่ 3.2 Metropolis Algorithm

โดยอยู่บนหลักการ Iterative Improvement การกระตุ้นและจะเคลื่อนไปยังตำแหน่งใหม่ ในการเคลื่อนไปยังตำแหน่งใหม่ จะพิจารณาว่า พลังงาน ΔE ที่เกิดจากความแตกต่างระหว่างพลังงานใหม่

กับพลังงานเก่ามีการเปลี่ยนแปลงไปในทางที่ต่ำกว่าหรือสูงกว่า ถ้า $\Delta E < 0$ หรือพลังงานลดลงคือค่า Configuration ที่ให้ค่า Cost (พลังงานยืดเหนี่ยว) มีค่าพลังงานน้อยกว่าแล้ว จะใช้ค่านั้นสำหรับการเคลื่อนไปยังตำแหน่งใหม่ ถ้า $\Delta E > 0$ หรือพลังงานเพิ่มขึ้นก็จะยอมรับค่า Configuration ใหม่ การเพิ่มของพลังงานที่สูงกว่าในการกระตุ้น จะถูกประมาณโดย อุณหภูมิ T ขณะนั้น ณ ที่อุณหภูมิสูงๆ โอกาสที่จะเกิดการกระตุ้น (Uphill) ของพลังงานจะมากและในอุณหภูมิที่ต่ำกว่าก็จะเกิดโอกาสที่น้อยกว่า Metropolis Algorithms นี้อยู่ในรูปแบบของ Boltzmann Distribution ซึ่งโอกาสหรือความน่าจะเป็นที่จะเกิดการ Uphill ของขนาดพลังงาน ΔE ณ อุณหภูมิ T คือ

$$P[accept] = e^{-\frac{\Delta E}{T}} \tag{3.1}$$

โดยจะกำหนดรูปแบบมาตราฐานของค่าเลขสุ่ม (Random Number (R)) ซึ่งอยู่ในช่วง 0 ถึง 1 และทำการเปรียบเทียบกับ P[Accept] ถ้า R < P[Accept] จะยอมรับผลนี้



รูปที่ 3.3 กราฟฟังก์ชันเป้าหมายที่ได้จากวิธีการจำลองอัลแนลลิ่ง

ช่วงของการยอมรับที่จะเกิดการกระตุ้นนั้นเราเรียกว่า จุดเลื่อนล้ำ (Landscape) ซึ่งการยอมรับจะอยู่ในช่วงของค่าเลขสุ่มดังกล่าว และอยู่ในช่วงของข้อมูลที่กำหนดให้

การประยุกต์ใช้วิธีการจำลองอัลแนลลิ่งเพื่อแก้ปัญหาใดๆ ต้องพิจารณาถึง

ก. การกำหนดสถานะที่เป็นไปได้ เป็นการกำหนดเซ็ตของผลลัพธ์ที่จะถูกเลือกด้วยค่าฟังก์ชันเป้าหมาย ดังนั้น ถ้าสามารถกำหนดเซ็ตของสถานะที่เป็นไปได้ทั้งหมดให้แคบได้เท่าใด จำนวนของสถานะรอบข้างก็จะลดลงตาม ทำให้ระยะเวลาในการค้นหาสั้นลง

ข. ฟังก์ชันเป้าหมาย (Objective Function) เป็นเครื่องมือวัดคุณภาพของสถานะในการเปลี่ยนสถานะแต่ละครั้งซึ่งจำเป็นต้องคำนวณหาค่าฟังก์ชันเป้าหมายใหม่ เพื่อเปรียบเทียบกับค่า

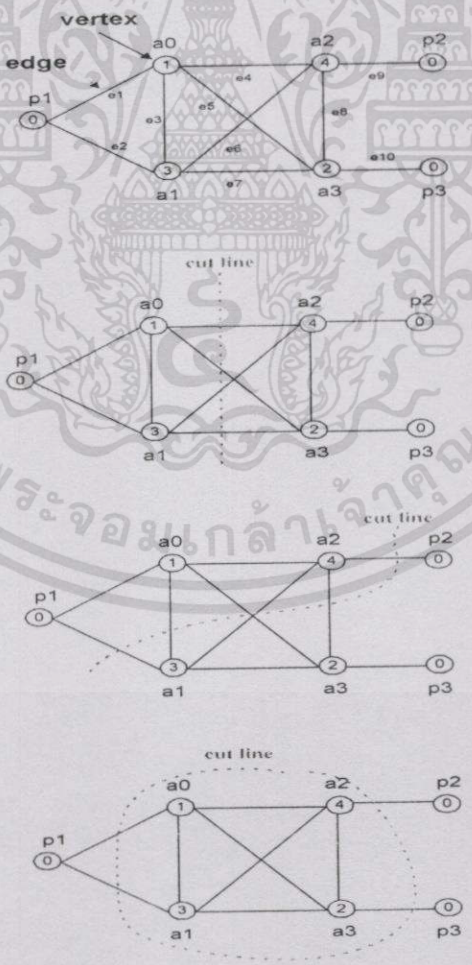
ฟังก์ชันเป้าหมายเดิม ดังนั้นฟังก์ชันเป้าหมายที่กำหนดขึ้นควรเป็นฟังก์ชันที่ง่ายต่อการคำนวณและนำไปสู่วัตถุประสงค์ที่ต้องการ

ค. วิธีการเปลี่ยนสถานะที่นิยมนำมาประยุกต์ใช้มี 2 รูปแบบด้วยกันคือ การย้าย (Move) และการสลับที่ (Exchange) จะใช้รูปแบบใดขึ้นอยู่กับคุณลักษณะของปัญหาเป็นสำคัญ

ง. แผนการควบคุมอุณหภูมิ (Annealing Schedule) หรือ Cooling Schedule เป็นการควบคุมค่าอุณหภูมิซึ่งประกอบด้วย อุณหภูมิเริ่มต้น อุณหภูมิที่ลดลง อุณหภูมิสุดท้าย และจำนวนการเปลี่ยนแปลง (Transition) ตามทฤษฎีทางคณิตศาสตร์ของ Markov Chain พิสูจน์ได้ว่าวิธีจำลองอัลแนลลิ่ง สามารถที่จะให้คำตอบออกมาเป็น Global Optimal Solution เมื่อมีการทดลองเปรียบเทียบคำตอบใหม่กับคำตอบเดิม ด้วยจำนวนครั้งเปรียบเทียบที่มากพอ

3.1.2 การจัดกลุ่มวงจรโดยใช้วิธีการจำลองอัลแนลลิ่ง

การจำลองอัลแนลลิ่งสำหรับการแก้ปัญหาการจัดกลุ่มวงจร [25][26] โดยพิจารณาการแก้ปัญหาสองทางอาศัยหลักการ Bisection คือแต่ละกลุ่มที่แบ่งจะมีขนาดเท่ากันหรือใกล้เคียงกันมาก



รูปที่ 3.4 ตัวอย่างคำตอบที่เป็นไปได้จากการแทนการจัดกลุ่มวงจรด้วยกราฟ

ก. การเริ่มต้นการจัดกลุ่มวงจรจากการแทนผังวงจร Circuit Network ด้วยกราฟ G(V,E) ดังรูปที่ 3.4 ซึ่งแต่ละวงจรประกอบด้วยทรานซิสเตอร์หลายๆตัว กำหนด V (Vertex) แทนสมาชิกแต่ละวงจรและ E (Edge) แทนความสัมพันธ์ระหว่างวงจร ให้ C_{ij} เป็นความสัมพันธ์ของเส้นเชื่อมโยง (Edge) ที่เชื่อมต่อระหว่าง Vertex i และ j

ข. กำหนดให้เส้น Cut แบ่งกลุ่ม A และ B ซึ่ง B = V-A คำนวณหาค่าฟังก์ชันเป้าหมาย C_{AB} ฟังก์ชันเป้าหมายของการจัดกลุ่มวงจรหาได้จากการใช้ Ratio Cut [27] C_{AB} เป็นผลรวมของค่าความสัมพันธ์ระหว่างกลุ่มที่แบ่งทุก Vertex ดังสมการที่ 3.2 และขนาดเท่ากันระหว่างกลุ่มที่แบ่ง Imbalance (A,B) ดังสมการที่ 3.3

$$C_{AB} = \sum_{i \in A} \sum_{j \in B} C_{ij} \tag{3.2}$$

$$\text{Imbalance (A,B)} = \text{Size of A} - \text{Size of B}$$

$$= \sum_{v \in A} s(v) - \sum_{v \in B} s(v) \tag{3.3}$$

โดยที่ s(v) เป็นขนาดของ Vertex v

ค. Accepting Vertex Moves : M เป็นจำนวนของการเปลี่ยนแปลง (Move States) ต่อการทำซ้ำ สำหรับการย้ายค่า Vertex V ถูกเลือกอย่างสุ่มซึ่งจะย้ายจากกลุ่มหนึ่งไปยังกลุ่มอื่น ก็ต่อเมื่อมีการยอมรับการย้าย ซึ่งการย้ายจะถูกยอมรับถ้า ΔGain น้อยกว่าหรือเท่ากับศูนย์หรือค่าความน่าจะเป็นของ Boltzmann Distribution น้อยกว่าค่า R ที่มีค่าอยู่ระหว่าง 0 ถึง 1 เราจะยอมรับคำตอบใหม่ของการเปลี่ยนแปลง ΔGain หมายถึงการเปรียบเทียบฟังก์ชันเป้าหมายคำตอบใหม่กับคำตอบเดิม

ง. Stopping Criteria : ภายหลังจากการกระทำซ้ำ อุณหภูมิ T เป็นตัวกำหนดการลดลงของแฟกเตอร์ α ที่ $0 < \alpha < 1$ ขั้นตอนวิธีจะหยุดเมื่ออุณหภูมิเริ่มต้น (T_0) ลดลงจนเข้าสู่ศูนย์

3.1.3 เทคนิคการควบคุมอุณหภูมิ

ปัจจุบันได้มีการศึกษาวิธีการจำลองอัลแนลลิ่งกันอย่างกว้างขวางเพื่อพยายามแก้ไขข้อเสียในเรื่องของเวลาโดยการเสนอวิธีการควบคุมอุณหภูมิแบบใหม่ เช่น Quick Annealing, Fast Simulated Anneling และ Very Fast Simulated Annealing วิธีจำลองอัลแนลลิ่งนั้นเสนอโดย Kirlpatrick et al [28] ในปี 1983 วิธีการควบคุมอุณหภูมิดังสมการที่ 3.4

$$T_k = \alpha T_{k-1} = \alpha^k T_0 \tag{3.4}$$

ซึ่ง $\alpha \in [0.1]$ เป็นอัตราคงที่ ค่า α ที่ใช้จะอยู่ระหว่าง $0.8 < \alpha < 0.99$

Geman [28] กล่าวว่าถึงเงื่อนไขสำหรับการหาผลลัพธ์ของวิธีอัลแนลลิ่งที่จะได้ค่าที่ดีที่สุด ที่นิยมใช้กันมากคือ Boltzman Annealing หรือ Classical Simulated Annealing ซึ่งขึ้นอยู่กับลักษณะทางกายภาพ และใช้การกระจายของ Gaussian สำหรับการเปลี่ยนค่า

$$g(x,s_k)=\frac{1}{\sqrt{2\pi s_k}}e^{-\frac{(x-x_0)^2}{2s_k}} \tag{3.5}$$

ซึ่ง x_0 เป็นค่าปัจจุบันของค่า x ที่ดีที่สุดการควบคุมอุณหภูมิหาได้จาก

$$T_k=\frac{T_0}{\ln(k+1)}, k=1,...,\infty \tag{3.6}$$

การพิสูจน์หาคำตอบของวิธีนี้เกิดจากการลดลงของอุณหภูมิจนถึงจุดพอใจ

$$\begin{aligned}\sum_{k=k_0}^{\infty}g(x,s_k)&=\sum_{k=k_0}^{\infty}\left[\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi s_k^i}}e^{-\left(x^i-x_0^i\right)^2/2s_k^i}\right]\\&=c_0\sum_{k=k_0}^{\infty}\left[\prod_{i=1}^{n}\sqrt{\ln(k+1)}e^{\ln(k+1)}\right]\\&\geq c_0\sum_{k=k_0}^{\infty}e^{-\ln(k+1)}\\&=c_0\sum_{k=k_0}^{\infty}\frac{1}{k+1}\qquad=\infty,\end{aligned} \tag{3.7}$$

ที่ c_0 เป็นค่าคงที่ที่กำหนดขึ้นเอง และ k_0 เป็นค่าควบคุมอุณหภูมิที่กำหนดขึ้นเอง และค่า i จะเป็นตัวบอกลำดับในกลุ่มของค่าที่ดีที่สุด

Szu [28] เสนอ Fast Annealingในปี 1987 ซึ่งเป็นการค้นหาเฉพาะที่และประกอบไปด้วยการกระโดดข้ามค่าที่เป็นคำตอบไปได้ไกลเป็นครั้งคราว (Long Jumps) วิธีนี้เกิดจากการปรับปรุง Boltzmann ฟังก์ชันการกระจาย ซึ่งเรียกว่า Cauchy Distribution จะใช้แทน Gaussian Distribution โดยที่ Cauchy Distribution มีสมการที่ 3.8 ดังนี้

$$g(x, s_k) = \frac{s_k}{\pi\left[(x - x_0)^2 + s_k^2\right]} \tag{3.8}$$

มีความเป็นไปได้สำหรับค่า x ที่อยู่ห่างจากค่า x_0 มากกว่าค่า Gaussian Distribution นั้นคือความเป็นไปได้ของ การกระโดดไปได้ไกลเป็นครั้งคราว ซึ่งจะส่งผลให้การหาคำตอบออกจาก Local Minima ในลักษณะเดียวกันเราแทนค่าที่ใช้ควบคุมอุณหภูมิด้วยการลดอุณหภูมิที่เร็วกว่าโดย

$$T_k = \frac{T_0}{1 + k} \tag{3.9}$$

การพิสูจน์เพื่อหาผลลัพธ์โดยวิธีนี้จะลดเวลาได้เป็นที่น่าพอใจ

$$\begin{aligned}
\sum_{k=k_0}^{\infty} g(x, s_k) &= \sum_{k=k_0}^{\infty}\left[\prod_{i=1}^{n} \frac{s_k^i}{\pi\left[\left(x^i - x_0^i\right)^2 + \left(s_k^i\right)^2\right]}\right] \\
&\geq \sum_{k=k_0+1}^{\infty}\left[\prod_{i=1}^{n} \frac{s_0^i}{k\pi\left[\left(x^i - x_0^i\right)^2 + \left(s_0^i\right)^2\right]}\right] \\
&\geq c_0 \sum_{k=k_0+1}^{\infty} \frac{1}{k} = \infty,
\end{aligned} \tag{3.10}$$

ซึ่ง C_0 เป็นค่าคงที่ที่กำหนดและ k_0 เป็นค่าควบคุมอุณหภูมิที่กำหนด โดยที่ i เป็นค่าลำดับชั้นในกลุ่มของค่า x ที่ Optimization

3.2 วิธีเจนนิติก (Genetic Algorithm)

เป็นวิธีการจำลองรูปแบบของวิธีการทางชีววิทยาในการให้กำเนิดประชากรรุ่นใหม่หรือขยายพันธุ์ในรุ่นลูก รุ่นหลานต่อไปซึ่งวิธีเจนนิติกอาศัยพื้นฐานความคิดของการวิวัฒนาการทางธรรมชาติของการถ่ายทอดทางพันธุกรรม โดยนักชีววิทยาเมนเดล (Mendel) บิดาแห่งวิชาพันธุศาสตร์เขาค้นพบยีนส์ (Genes) ที่เป็นตัวกำหนดลักษณะภายนอกต่างๆทางพันธุกรรมของสิ่งมีชีวิต ซึ่งยีนส์จะมีการเรียงตัวต่อกันเป็นแถวยาวเรียกว่า โครโมโซม (Chromosome) โดยอยู่กันเป็นคู่ๆในเซลล์สิ่งมีชีวิตในแต่ละยีนส์จะมีลักษณะต่างๆกัน ซึ่งเรียกว่า แอลลี (Allele) ซึ่งลักษณะที่แตกต่างกันของยีนส์

ในแต่ละตำแหน่งยีนส์เดียวกันของคู่โครโมโซม เรียกว่า ยีโนไทป์ (Genotype) และลักษณะภายนอกที่ปรากฏออกมาให้เห็นเรียกว่า ฟีโนไทป์ (Phenotype) ยกตัวอย่างคู่โครโมโซมของคนซึ่งจะมี 23 คู่ 46 โครโมโซม แต่ละโครโมโซมจะประกอบด้วยยีนส์ราวๆ 1250 ยีนส์ อาทิเช่น สีตา สีผม และอื่นๆ ในโครโมโซมA มีแอลลีของยีนส์ลักษณะสีตาสีดำ ในขณะที่โครโมโซม B มีแอลลีของยีนส์ลักษณะสีตาสีน้ำตาล

การถ่ายทอดลักษณะทางพันธุกรรมของสิ่งมีชีวิตเกิดจากการแบ่งตัวของเซลล์เมื่อมีการเจริญเติบโตหรือทดแทนเซลล์เก่าที่ตายไปและเมื่อมีการสร้างเซลล์สืบพันธุ์ การแบ่งเซลล์มีด้วยกัน 2 แบบ คือ แบบไมโทซิส (Mitosis) เป็นการแบ่งเซลล์ร่างกาย (Somatic Cell) ซึ่งเป็นกระบวนการเพิ่มจำนวนเซลล์โดยที่โครโมโซมยังคงมีจำนวนเท่ากับเซลล์เริ่มต้น และอีกแบบหนึ่งคือ แบบไมโอซิส (Meiosis) เป็นการแบ่งเซลล์สืบพันธุ์ (Gamete) แต่ละเซลล์จะมีจำนวนโครโมโซมเพียงครึ่งหนึ่งของจำนวนโครโมโซมของเซลล์ร่างกาย [29]

ก. การแบ่งเซลล์แบบไมโทซิส เริ่มต้นจากเซลล์เดิม 1 เซลล์ ที่มีจำนวนโครโมโซม 2 ชุด หรือดิพลอยด์เซลล์ เมื่อผ่านการแบ่งเซลล์จนสมบูรณ์จะได้เซลล์ใหม่ 2 เซลล์ที่มีจำนวนโครโมโซมจำนวน 2 ชุดเท่าเดิมและมีโครงสร้างของดีเอ็นเอเหมือนเดิมทุกประการ การแบ่งเซลล์แบบนี้มี 2 ช่วงระยะด้วยกันคือ ช่วงอินเตอร์เฟส และไมโทซิส

ข. การแบ่งเซลล์แบบไมโอซิส แบ่งออกเป็นช่วงอินเตอร์เฟสและช่วงไมโอซิส 2 ขั้นตอนต่อเนื่องกันคือ ขั้นตอนไมโอซิส 1 และไมโอซิส 2 และแต่ละขั้นตอนแบ่งออกเป็น 4 ระยะย่อยเช่นเดียวกับช่วงไมโทซิส ในกระบวนการนี้จะเริ่มจากดิพลอยด์เซลล์เพียง 1 เซลล์ผลลัพธ์ที่ได้คือ แฮพลอยด์เซลล์ 4 เซลล์ ดีเอ็นเอจะจำลองตัวเองในระยะอินเตอร์เฟสทำให้เกิดโครมาทิด 2 ข้าง ในระยะโพรเฟส 1 ซึ่งจะมีการเข้าคู่แนบชิดกันของโครโมโซมคู่เหมือนและเกิดการแลกเปลี่ยนดีเอ็นเอบางส่วนระหว่างซิสเตอร์โครมาทิด (Sister Chromatid) โดยการไขว้เปลี่ยนกันของโครมาทิด เรียกว่าการเกิดครอสซิ่งโอเวอร์ (Crossing Over) ซึ่งจะทำให้เกิดการสลับที่ของยีนส์ในโครโมโซมคู่เหมือน ประโยชน์ของการ ครอส์โอเวอร์คือทำให้เกิดลักษณะที่แตกต่างกันได้มากขึ้น ทำให้สิ่งมีชีวิตรุ่นต่อๆไปมีความหลากหลายมากยิ่งขึ้น ถ้าการเกิดเซลล์ใหม่ไม่มีการครอสโอเวอร์แล้วโครโมโซมที่มียีนส์ลักษณะใดก็จะมีลักษณะนั้นอยู่ตลอดไปทำให้รุ่นต่อๆ มามีลักษณะเดิมๆ โอกาสที่สิ่งมีชีวิตจะมีการปรับตัวให้ดีขึ้นมีได้ยากกว่าการเปลี่ยนลักษณะยีนส์ใหม่หลายๆ แบบมากขึ้น จากนั้นจะมีการเรียงตัวในสภาพพร้อมที่จะแยกกันของโครโมโซมคู่เหมือนในระยะเมทาเฟส 1 และแยกออกไปที่แต่ละขั้วในระยะอะนาเฟส 1 เมื่อจบระยะเทโลเฟส 1 จะมีการแบ่งนิวเคลียสและเซลล์ออกเป็น 2 เซลล์และเข้าสู่ขั้นตอนไมโอซิส 2 ซึ่งจะคล้ายกับขั้นตอนไมโอซิส 1 โดยผ่านระยะโพรเฟส 2 เมทาเฟส 2 อะนาเฟส 2 และเทโลเฟส 2 ตามลำดับ และเกิดการแบ่งไซโทพลาสซึมอย่างรวดเร็วทำให้ได้แฮพลอยด์ 4 เซลล์

การแบ่งเซลล์แบบไมโอซิสทำให้เกิดการสร้างเซลล์สืบพันธุ์ซึ่งมีจำนวนโครโมโซมลดลงเป็นครึ่งหนึ่งของเซลล์ร่างกาย กระบวนการสร้างเซลล์สืบพันธุ์ เรียกว่าแกมีโทเจเนซิส (Gametogenesis) การแบ่งเซลล์แบบไมโอซิสเป็นกระบวนการสำคัญที่จะรักษาจำนวนโครโมโซมของสิ่งมีชีวิตให้คงที่ การลดจำนวนโครโมโซมลงครึ่งหนึ่งเป็นการแยกคู่โครโมโซมคู่ที่เหมือนกันออกไปอยู่เซลล์ใหม่แต่ละเซลล์ซึ่งจะเจริญต่อไปเป็นเซลล์สืบพันธุ์ที่มีจำนวนโครโมโซมเท่ากันทุกเซลล์และแต่ละเซลล์จะมีปริมาณยีนส์เท่ากัน การแยกคู่ออกจากกันของโครโมโซมคู่เหมือนเป็นกระบวนการที่เป็นอิสระต่อกัน (Independent Assortment) ทำให้เซลล์สืบพันธุ์ที่สร้างขึ้นอาจมีโครโมโซมที่มาจากพ่อบางส่วนจากแม่บางส่วนปะปนกัน การแลกเปลี่ยนชิ้นส่วนของโครโมโซมคู่เหมือนระหว่างการเกิดครอสซิ่งโอเวอร์เป็นผลให้มีการรวมตัวกันใหม่ของยีนส์ (Gene Recombination) ดังแสดงในรูปที่ 3.5 ซึ่งจะถ่ายทอดผ่านเซลล์สืบพันธุ์ไปสู่รุ่นลูก ทำให้เกิดความแตกต่างของยีนส์ในรุ่นต่อๆไป ตัวอย่างเช่นดีเอ็นเอของคนทุกๆคนไม่เหมือนกัน แต่ละคนจะมีดีเอ็นเอที่เป็นเอกลักษณ์ของตนเองแม้พี่น้องที่เกิดจากพ่อแม่เดียวกัน ยกเว้นในกรณีของฝาแฝดแท้ซึ่งเกิดจากไข่ใบเดียวกันที่จะมีดีเอ็นเอเหมือนกันทุกประการ

ส่วนการมิวเตชัน (Mutation) หรือการผ่าเหล่า คือการเปลี่ยนแปลงลักษณะของยีนส์ไปจากเดิมที่ควรจะเป็นตามการถ่ายทอดแต่กลับเปลี่ยนไปเป็นแบบอื่นๆ ซึ่งเป็นสาเหตุของการเกิดลักษณะที่แปลกๆออกไป เพราะธรรมชาติจะค่อยๆปรับตัวเองให้เหมาะสมกับสภาพแวดล้อมซึ่งจะเป็นไปอย่างช้า แต่การผ่าเหล่านั้นทุกลักษณะในแต่ละยีนส์จะมีโอกาสเปลี่ยนแปลงไปจากเดิมได้เลยถ้าเกิดความเหมาะสมกับสภาพแวดล้อมขณะนั้นๆ ก็จะถูกคัดเลือกให้คงอยู่ต่อไป [7][30]

รูปที่ 3.5 การเกิดครอสซิ่งโอเวอร์ในช่วงไมโอซิส 1 โดยการแลกเปลี่ยนส่วนของดีเอ็นเอ ระหว่างซิสเตอร์โครมาทิด (Sister Chromatid) ในโครโมโซมคู่เหมือน

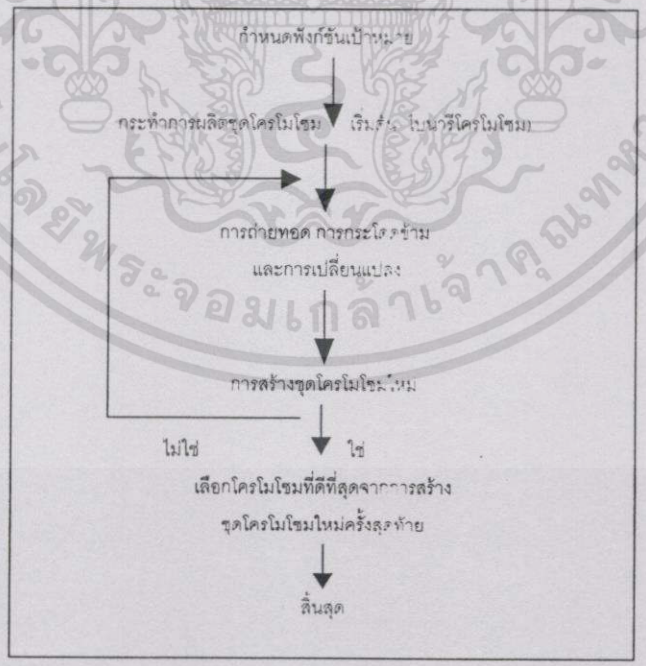
3.2.1 หลักการประยุกต์ใช้วิธีเจนนิติกแบบทั่วไป

ลักษณะของวิธีเจนนิติกเป็นวิธีการหาค่าที่เหมาะสมแบบสุ่ม โดยหลักการพื้นฐานของเจนนิติกแล้วจะเป็นแบบการสุ่ม ที่มีหลักการการคาดหวังคำตอบใหม่จากสถิติคำตอบเดิมที่ดี โดยการเลียนแบบวิธีการคัดเลือกทางธรรมชาติพันธุกรรม เกิดจากการรวมกันหรือสลับเปลี่ยนตัวแปรต่างๆโดยอาศัยหลักการสุ่ม เพื่อให้ได้ประชากรที่ดีขึ้น ซึ่งวิธีเจนนิติกจะเป็นการหาค่าที่เหมาะสมที่มีค่าสูงสุด (Maximization) โดยจะกำหนดคำตอบขึ้นมาหลายๆคำตอบ แต่ละคำตอบเรียกว่าโครโมโซม ในแต่ละโครโมโซมจะเป็นลักษณะของสตริง (String) ที่มีอยู่หลายบิต การหาคำตอบเกิดจากผลลัพธ์ของกลุ่มค่าตัวแปรที่เป็นฟังก์ชันเป้าหมายของปัญหาและอาศัยการถ่วงน้ำหนักความเหมาะสมของแต่ละคำตอบ ในรูปที่ 3.6 แสดงไดอะแกรมการทำงานของเจนนิติกแบบทั่วไป [30][31]

3.2.2 การประยุกต์วิธีเจนนิติกแบบง่าย

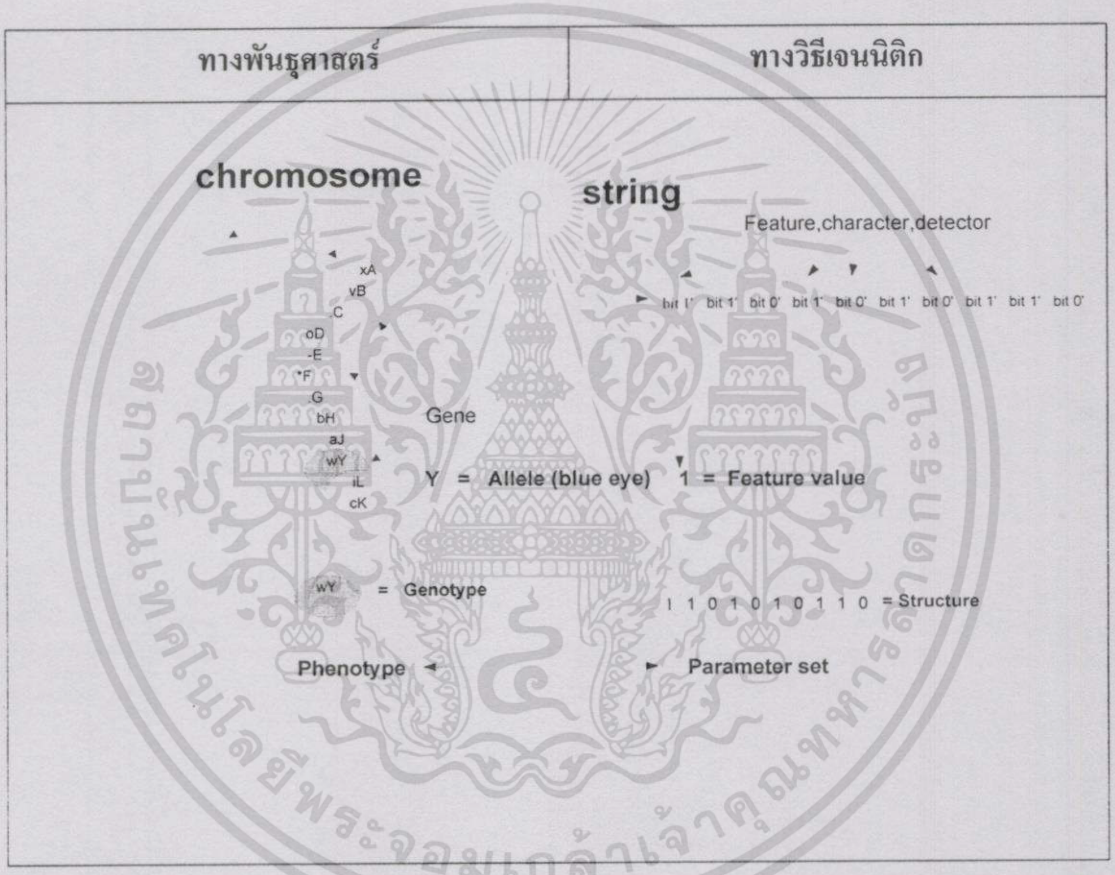
การหาคำตอบที่ดีที่สุดของปัญหาโดยวิธีนี้มีพื้นฐานอยู่บนผลลัพธ์จากการหาคำตอบที่ผ่านมาแล้วและจะไม่พิจารณาขั้นตอนของการแก้ปัญหา แต่จะพิจารณาโดยตัดสินว่าคำตอบใหม่ที่ได้รับดีขึ้นหรือไม่ หรือเป็นคำตอบที่ใกล้เคียงคำตอบที่ต้องการหรือไม่จากฟังก์ชันเป้าหมาย โดยมีขั้นตอนการพิจารณาหาคำตอบดังนี้

ก. วิเคราะห์ปัญหาเพื่อกำหนดฟังก์ชันเป้าหมาย ซึ่งเป็นฟังก์ชันที่แสดงความสัมพันธ์ของแต่ละตัวแปรหรือเงื่อนไขของปัญหาที่ระบุคำตอบใดคำตอบหนึ่งที่ได้จากการสุ่มหาจากคำตอบทั้งหมดที่สามารถเป็นไปได้ ณ ค่าตัวแปรหรือข้อกำหนดชุดนั้น



รูปที่ 3.6 ไดอะแกรมการทำงานของวิธีการเจนนิติกแบบทั่วไป

ข. กำหนดรูปแบบโครโมโซม เป็นการมองปัญหาเทียบเท่ากับโครโมโซมซึ่งประกอบด้วยยีนส์ลักษณะต่างๆ ซึ่งเป็นลำดับข้อมูลต่างๆที่เมื่อแปลความหมายแล้วให้ค่าคำตอบของปัญหา ยีนส์จะเป็นตัวแปรของเงื่อนไข หรือข้อกำหนดต่างๆ ของปัญหา ดังนั้นการกำหนดรูปแบบโครโมโซมของแต่ละปัญหาโดยการแปลงองค์ประกอบของปัญหาให้อยู่ในรูปลำดับของยีนส์บนโครโมโซมเรียกว่าสตริง (String) จะประกอบไปด้วยบิท (Bit) ที่เป็นอักขระ (Character) ของลักษณะที่เป็นไปได้ของแต่ละยีนส์ที่ประกอบรวมกันเป็นค่าความยาวของโครโมโซม รูปที่ 3.7 แสดงความหมายเปรียบเทียบคำศัพท์ ที่ใช้ในทางพันธุศาสตร์กับทางวิธีเจนนิติก



รูปที่ 3.7 การแทนคำศัพท์ ที่ใช้ในทางพันธุศาสตร์กับทางวิธีเจนนิติก

ค. กำหนดจำนวนรุ่นสำหรับการวิวัฒนาการของกลุ่มคำตอบ

ง. สร้างประชากรโครโมโซมต้นกำเนิดโดยการสุ่มในรุ่นแรก รุ่นถัดมาสร้างจากประชากรต้นกำเนิดที่เกิดจากการสุ่มค่าบิทในแต่ละโครโมโซม

จ. วิเคราะห์ค่าความเหมาะสมแต่ละโครโมโซม และตรวจสอบเงื่อนไขความพอใจ

ฉ. การคัดเลือกเพื่อสร้างกลุ่มกำหนดสายพันธ์ (Mating Pool) คือชุดโครโมโซมพ่อแม่ที่ถูกคัดเป็นต้นแบบ โดยอาศัยการจำลองการคัดเลือกทางพันธุกรรม โดยพิจารณาค่าถ่วงน้ำหนักจากค่า

ความเหมาะสมของแต่ละโครโมโซม หากโครโมโซมใดมีค่าความเหมาะสมมากก็จะมีโอกาสถูกคัดเลือกเป็นต้นแบบมาก

ช. การครอสโอเวอร์ เป็นการกระโดดข้ามหรือสลับกันระหว่างโครโมโซมในการแลกเปลี่ยนส่วนของโครโมโซมพ่อแม่ ตามการกำหนดอัตราความน่าจะเป็นของการครอสโอเวอร์

ซ. การมิวเตชัน เป็นตัวดำเนินการผ่าเหล่าตัวหนึ่งที่อาจช่วยให้โครโมโซมมีค่าความเหมาะสมดีขึ้น

3.2.3 การจัดกลุ่มวงจรโดยใช้วิธีเจนนิติก

วิธีเจนนิติกสำหรับแก้ปัญหาการจัดกลุ่มวงจรอยู่บนหลักการของ Bui's [32][33] แบ่งการแก้ปัญหาออกเป็น 2 ส่วน คือ ขั้นตอนในการเตรียมการและขั้นตอนในการทำงาน

ขั้นตอนในการเตรียมการนั้นเป็นส่วนของการปรับปรุงรูปแบบของปัญหาให้เหมาะสมกับการนำเสนอ ประกอบด้วย

ก. การกำหนดฟังก์ชันตามความเหมาะสม คือ ผลงานของค่าความสัมพันธ์ระหว่างกลุ่มที่แบ่งดังสมการที่ 3.14 ซึ่งคำตอบที่ดีที่สุด คือค่าความสัมพันธ์ระหว่างกลุ่มที่แบ่งมีค่าน้อยที่สุด และขนาดของกลุ่มที่แบ่งมีค่าประมาณเกือบเท่ากัน

ข. กำหนดรูปแบบโครโมโซม จากการสุ่มโครโมโซมมา P โครโมโซมที่มีค่าสุ่มอยู่ระหว่าง 1 ถึง 2^{c}-2 (c เป็นจำนวนทั้งหมดของวงจร) แต่ละโครโมโซมแทนด้วยการแบ่งกลุ่มเป็นสองส่วนที่มีขนาดเท่ากัน ซึ่งสัมพันธ์กับการแบ่งกลุ่มออกเป็นสองส่วนที่มีขนาดเท่ากัน (Bisection) ของกราฟ จำนวนของยีนส์ในแต่ละโครโมโซมมีจำนวนเท่ากับ C ตัว แทนค่าในแต่ละยีนส์ด้วย 0 หรือ 1 เท่านั้น ซึ่งยีนส์จะมีค่าเป็น 0 ถ้ายีนส์ อยู่ทางซ้ายของ Bisection และมีค่าเป็น 1 ถ้ายีนส์อยู่ทางขวาของ Bisection ดังนั้นในหนึ่งโครโมโซมจะมีจำนวน 0 และ 1 เกือบจะเท่ากันหรือเท่ากันเช่นโครโมโซม [00111] แทนกราฟที่มี 5 วงจรในกลุ่มแรก มีวงจรลำดับที่ 1 และ 2 อยู่ในกลุ่มเดียวกันส่วนอีกกลุ่มหนึ่งมีวงจรในลำดับที่ 3, 4, 5 อยู่ ในส่วนของขั้นตอนการทำงาน แสดงขั้นตอนวิธีในรูปที่ 3.8

```
Genetic Algorithm preprocess
begin
      create initial population of  fixed size P
      do
        choose parent1 and parent2 from the population
        offspring = crossover(parent1,parent2)
        mutation(offspring)
        local_improvement(offspring)
        replace(population,offspring)
      until (stopping criteria met )
      report the best answer
end
```

รูปที่ 3.8 แสดงขั้นตอนวิธีการเจนนิติกสำหรับการจัดกลุ่มวงจร

Create Initial Population : จำนวนประชากร Pโครโมโซมได้จากการสุ่มซึ่งเป็นประชากรรุ่นแรก แต่ละโครโมโซมเป็นคำตอบที่อยู่ในเซตของคำตอบ

Choose Parents : แต่ละลำดับหรือแต่ละโครโมโซมมีค่าความเหมาะสม (Fitness) ซึ่งเป็นตัวบอกคุณภาพของโครโมโซมในแต่ละลำดับดูได้จาก Bui [26] จะคำนวณหาค่าความเหมาะสม F_i สำหรับแต่ละโครโมโซมได้จาก

$$F_i = (C_w - C_i) + \frac{C_w - C_b}{3} \tag{3.14}$$

ซึ่ง C_w เป็น Cutsize ที่ใหญ่ที่สุดในประชากรแต่ละรุ่น C_b เป็น Cutsize ที่น้อยที่สุดในประชากรแต่ละรุ่นและ C_i เป็น Cutsize ของแต่ละโครโมโซมที่ i และมีค่าขนาดของกลุ่มที่แบ่งเกือบเท่ากัน

$$\text{Im}\, balance(A,B) = \sum_{v \in A} s(v) - \sum_{v \in B} s(v) \tag{3.15}$$

s(v) เป็นขนาดของ Vertex v แต่ละโครโมโซมจะถูกพิจารณาสำหรับการเลือกคู่พ่อแม่ (Parents) ความน่าจะเป็นของการเลือกคู่พ่อแม่ของแต่ละคู่นั้นจะเป็นสัดส่วนพอเหมาะกับค่าความเหมาะสมและกล่าวว่าความน่าจะเป็นที่โครโมโซมที่ดีจะถูกเลือกเป็นสี่เท่าของความน่าจะเป็นที่โครโมโซมไม่ดีจะถูกเลือกมาเป็นคำตอบ ดังนั้นจำนวนโครโมโซม P ตัวถูกจัดลำดับตามค่าความเหมาะสมและฟังก์ชันการกระจายความน่าจะเป็น (Probability Distribution Function) จะถูกสร้างขึ้นโดยที่ Probability Factor (r) ซึ่งหาได้จากสมการ

$$r = 4^{\frac{1}{P-1}} \tag{3.16}$$

สมมุติว่าความน่าจะเป็นที่จะกำหนดแต่ละโครโมโซมเป็นลำดับเรขาคณิตซึ่งผลรวมของความน่าจะเป็นเท่ากับ S หาจาก

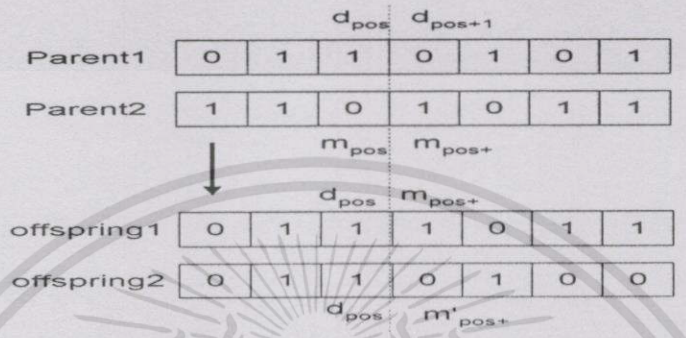
$$S = 1 + r + r^2 + \ldots + r^{P-1} = \frac{1 - r^P}{1 - r} \tag{3.17}$$

เพราะฉะนั้นความน่าจะเป็นที่แต่ละโครโมโซมจะถูกเลือกคือ Pr{i} จึงหาได้จาก

$$\Pr\{i\} = \frac{r^{i-1}}{S} \tag{3.18}$$

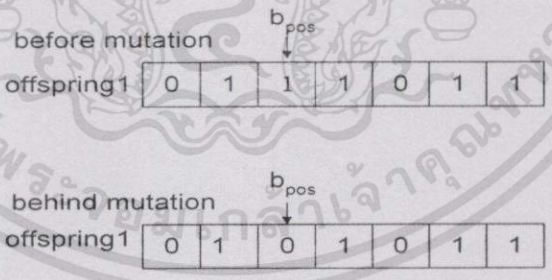
การครอสโอเวอร์ (Crossover): เป็นตัวการในการข้ามพันธุ์แลกเปลี่ยนส่วนของโครโมโซมเพื่อสร้างชุดโครโมโซมใหม่หรือโครโมโซมลูกเรียกว่า Offspring ในที่นี้การครอสโอเวอร์เป็นแบบ 1

จุด (One point Crossover) ซึ่งตำแหน่งที่จะครอสโอเวอร์ให้เป็น pos ได้จากการสุ่ม การแลกเปลี่ยนค่าของแต่ละบิตเกิดตั้งแต่ตำแหน่งที่ pos+1 เป็นต้นไป แสดงได้ดังรูปที่ 3.9 ค่า offspring1 เกิดจากการเชื่อมค่าครึ่งทางซ้ายของ parent1 กับครึ่งทางขวาของ parent2 ในขณะที่ offspring2 เกิดจากการเชื่อมต่อของครึ่งทางซ้ายของ parent1 กับค่าตรงข้ามของครึ่งทางขวาของ parent2



รูปที่ 3.9 การครอสโอเวอร์แบบ 1 จุด

การมิวเตชัน(Mutation): การกลายพันธุ์เป็นตัวช่วยให้โครโมโซมมีค่าความเหมาะสมดีขึ้น เพราะหลังจากการครอสโอเวอร์แล้วค่า Bit Pattern ที่ได้อาจจะมีค่าไม่เหมาะสมจึงต้องมีการปรับให้ดีขึ้น ค่า offspring จะถูกกลับค่าของบิตในตำแหน่งบิตที่สุ่มได้ ดังนั้นขบวนการกลายพันธ์ (Mutation Procedure) กำหนดค่า b ซึ่งเป็นตำแหน่งที่จะกลับค่าบิต (จาก 1 เป็น 0 หรือจาก 0 เป็น 1)



รูปที่ 3.10 การมิวเตชันของบิท

Local_improvement and Replace (Update Population) : การสร้าง 2 offspring ไปเพิ่มขนาดของประชากรเป็น P+2 แต่เราต้องการรักษาขนาดประชากร P และเป้าหมายของขั้นตอนวิธีก็คือต้องการผลลัพธ์ที่ดีในรุ่นถัดมา ดังนั้นโครโมโซมที่มีค่าความเหมาะสมต่ำสุด 2 โครโมโซมจะถูกตัดออกไปจากกลุ่มประชากร

Stopping Criteria : ใช้สำหรับการเปลี่ยนรุ่นถัดไปและเป็นตัวกำหนดการหยุดทำงานของขั้นตอนวิธี ถ้าไม่มีการเปลี่ยนแปลงภายหลังรุ่นใหม่ที่เกิดแล้วขั้นตอนวิธีจะหยุดทำงาน

3.2.4 การปรับปรุงประสิทธิภาพของวิธีเจนนิติก

ประสิทธิภาพในการค้นหาคำตอบของปัญหาต่างๆ จะแตกต่างกันออกไป ยิ่งปัญหาที่มีเงื่อนไขซับซ้อนมากๆแล้ว การดำเนินการจะลำบากและใช้เวลานานในการค้นหาคำตอบ GA จะไม่สามารถหาคำตอบที่ดีที่สุดได้ในขอบเขตที่กำหนด จึงได้มีการปรับปรุงประสิทธิภาพของ GA ในส่วนต่างๆ เช่น ส่วนของการคัดเลือกต้นแบบในการสร้างกลุ่มกำหนดสายพันธุ์ (mating pool) วิธีการสุ่มโครโมโซม การกำเนินประชากรรุ่นใหม่ด้วยการรักษาโครโมโซมที่ดีในแบบต่างๆ [30]

วัตถุประสงค์ของการหาคำตอบของปัญหามีอยู่ 2 อย่างคือ หาค่าสูงสุด (Maximize utility Function) กับค่าต่ำสุด (Minimize cost function) GA เป็นการหาค่าคำตอบสูงสุด โดยทั่วไปฟังก์ชันเป้าหมายจะเป็นฟังก์ชันความเหมาะสมที่ให้ค่าข้อมูลดิบ (raw fitness) เป็นค่าความเหมาะสม วิธีการคัดเลือกต้นแบบโดยทั่วไปของ GA มี 2 รูปแบบคือ

3.2.4.1 แบบอ้างอิงค่าความเหมาะสม (fitness-based) เป็นรูปแบบการคัดเลือกต้นแบบโดยวัดประสิทธิภาพของแต่ละโครโมโซมจากความเหมาะสมโดยตรง คือ โครโมโซมที่มีค่าความเหมาะสมสูง จะมีโอกาสถูกเลือกเป็นต้นแบบมากกว่าโครโมโซมที่มีค่าความเหมาะสมต่ำ โดยส่วนมากแล้วโครโมโซมในรุ่นแรกๆ มักจะเป็นโครโมโซมที่ไม่ดี แต่บางกรณีก็พบว่าได้โครโมโซมที่ดีเกิดขึ้น ส่งผลให้ค่าคาดหวังที่จะสุ่มได้ของโครโมโซมนั้นสูงมากกว่าโครโมโซมอื่นๆมาก และทำให้โครโมโซมนั้นถูกส่งเป็นต้นแบบในจำนวนที่มาก เป็นเหตุให้เกิดจุดจบก่อน (Premature Convergence) คือ หาคำตอบได้ก่อนซึ่งไม่ใช่คำตอบที่ดีที่สุด หรืออีกกรณีคือการดำเนินการของ GA ในรุ่นหลังๆ โครโมโซมต่างๆ เริ่มดีขึ้น ความแตกต่างของแต่ละโครโมโซมน้อยลงมาก ทำให้ค่าความคาดหวังที่จะสุ่มได้ของแต่ละโครโมโซมใกล้เคียงกันมาก แสดงว่าจำนวนโครโมโซมต้นแบบของแต่ละโครโมโซมก็จะใกล้เคียงกัน มีผลทำให้ความสามารถในการค้นหาคำตอบของ GA นั้นลดลงจนเหมือนกับการค้นหาคำตอบแบบสุ่ม จึงได้ทำการปรับปรุงฟังก์ชันความเหมาะสมโดยปรับสัดส่วนค่าความเหมาะสม (Scaling Fitness) เพื่อปรับค่าคาดหวังที่จะสุ่มได้ของแต่ละโครโมโซมให้พอเหมาะมากขึ้น มี 3 วิธีหลักๆ

ก. แบบหน้าต่าง (Windowing) เป็นวิธีการปรับค่าความเหมาะสมของแต่ละโครโมโซมโดยพิจารณาจากค่าเหมาะสมที่ไม่ดี ดังสมการที่ 3.19

$$F = \begin{cases} f - f_{worst}^{'} & for \quad MAX(f) \\ f_{worst}^{'} - f & for \quad MIN(f) \end{cases} \tag{3.19}$$

ใช้ค่าต่ำสุดของการหาค่าสูงสุดหรือค่าสูงสุดของการหาค่าต่ำสุดของปัญหาเป็นค่า $f_{worst}^{'}$ แต่ในการค้นหาคำตอบของ GA จะไม่สามารถระบุค่าต่ำสุดของปัญหาได้ ดังนั้นวิธีการกำหนด $f_{worst}^{'}$ ของ GA โดยกำหนดให้

$f_{worst}^{'}$ = ค่าที่ไม่ดีที่สุดของโครโมโซมทั้งหมดในรุ่นนั้น

W = ขนาดหน้าต่างหรือจำนวนรุ่นที่ผ่านมา

f'_{worst} = f'_{worst} ในรุ่นเริ่มต้น และเปลี่ยนแปลงเมื่อเกิดค่า f'_{worst} ที่ดีขึ้น ถ้า W = 0

f'_{worst} ของ W รุ่นที่ผ่านมา ถ้า W > 0

การกำหนดขนาดหน้าต่างนั้นแตกต่างกันไปในแต่ละปัญหา โดยส่วนมากแล้วขนาดหน้าต่างจะอยู่ในช่วงที่ 1 ถึง 10 รุ่นที่ผ่านมา

ข. แบบเชิงเส้น (Linear) เป็นวิธีการปรับค่าเหมาะสมโดยกำหนดให้โครโมโซมที่มีค่าความเหมาะสมเฉลี่ยจะต้องมีจำนวนที่คาดหวังว่าจะสุ่มได้เป็นต้นแบบเท่ากับจำนวนที่คาดหวังว่าจะสุ่มได้ของค่าข้อมูลดิบเฉลี่ย และควบคุมค่าคาดหวังว่าที่จะสุ่มได้โครโมโซมที่ดีที่สุดเป็นจำนวนเท่าของโครโมโซมที่มีค่าความเหมาะสมเฉลี่ย แล้วปรับค่าความเหมาะสมของโครโมโซมอื่นๆ ในลักษณะเชิงเส้นดังสมการที่ 3.20

$$F = af+b \tag{3.20}$$

โดยที่ a,b = ค่าคงที่คำนวณได้จากการกำหนดให้ $F_{avg} = f_{avg}$ และ $F_{best} = C_{mult} * f_{avg}$

f_{avg} คือค่าความข้อมูลดิบเฉลี่ย , f_{best} คือค่าข้อมูลดิบที่ดีที่สุด

F_{avg} คือค่าความเหมาะสมเฉลี่ย, F_{best} คือค่าความเหมาะสมที่ดีที่สุด และ

C_{mult} คือจำนวนเท่าของโครโมโซมที่มีค่าความเหมาะสมเฉลี่ย

การปรับสัดส่วนสำหรับการหาค่าสูงสุดของปัญหานั้นจะทำในรุ่นแรกๆ ถ้าโครโมโซมดีเกินไปจะปรับสัดส่วนให้มีค่าเหมาะสมลดลง และถ้าโครโมโซมที่ไม่ดีจะถูกปรับค่าความเหมาะสมให้สูงขึ้น ทำให้ไม่เกิดจำนวนที่คาดหวังว่าจะสุ่มได้แตกต่างกันมากนัก ค่า a และ b คำนวณจากสมการที่ 3.21 และสมการที่ 3.22 ตามลำดับ

$$a = \frac{(C_{mult} - 1) * f_{avg}}{f_{best} - f_{avg}} \tag{3.21}$$

$$b = \frac{f_{avg} * (f_{best} - C_{mult} f_{avg})}{f_{best} - f_{avg}} \tag{3.22}$$

ค. แบบตัดส่วนเบี่ยงเบนมาตรฐาน (Sigma Truncation) จากการปรับสัดส่วนแบบเชิงเส้นอาจทำให้เกิดค่าความเหมาะสมเป็นลบในรุ่นหลังๆ ของการดำเนินการ แต่สามารถแก้ไขได้โดยการตัดโครโมโซมที่แตกต่างจากโครโมโซมส่วนใหญ่ทิ้งไป พิจารณาจากค่าส่วนเบี่ยงเบนมาตราฐาน(σ) จากสมการที่ 3.23

$$F = \frac{f - (f_{avg} - cg) \quad for \quad MAX(f) \quad if(F < 0) \quad then \quad F = 0}{(f_{avg} + c\sigma) - f \quad for \quad MIN(f) \quad if(F < 0) \quad then \quad F = 0} \tag{3.23}$$

โดยที่ c = จำนวนเท่าของส่วนเบี่ยงเบนมาตรฐานที่กำหนด

σ = ส่วนเบี่ยงเบนมาตรฐานของประชากรในรุ่นนั้นๆ

$$= \sqrt{\sum_{i=1}^{popsize} (f_i - f_{avg})^2 / popsize} \quad (3.24)$$

วิธีการปรับสัดส่วนแบบตัดส่วนเบี่ยงเบนมาตรฐานเป็นการปรับค่าความเหมาะสมให้ลดลงเป็นระยะ c เท่าของส่วนเบี่ยงเบนมาตรฐานจากค่าความเหมาะสมเฉลี่ย นั้นคือการพิจารณาโครโมโซมที่ไม่ดีเฉพาะโครโมโซมที่ไม่แตกต่างจากโครโมโซมที่มีค่าความเหมาะสมเฉลี่ยเกินกว่าระยะ c เท่าของส่วนเบี่ยงเบนมาตรฐาน ซึ่งโครโมโซมที่ไม่ดีนี้จะถูกตัดทิ้งไป โดยการแทนค่าความเหมาะสมเป็น 0 ทำให้ไม่มีค่าเหมาะสมเป็นลบ การกำหนดค่า c นิยมให้อยู่ระหว่างค่า [1.0,3.0] และยังขึ้นกับการประยุกต์ใช้ในแต่ละปัญหาด้วย

3.2.4.2 การคัดเลือกแบบอ้างอิงลำดับ (Ranking-based)

เป็นการคัดเลือกโครโมโซมต้นแบบเพื่อสร้างกลุ่มกำหนดสายพันธุ์ (mating pool) ที่จะช่วยลดการเกิดจุดจบก่อนที่เป็นสาเหตุจากโครโมโซมที่ดีเกินไปโดยการควบคุมการจัดสรรจำนวนที่คาดหวังว่าจะสุ่มได้ของแต่ละโครโมโซม ที่ช่วยไม่ให้มีโครโมโซมใดถูกจัดสรรจำนวนที่คาดหวังว่าจะสุ่มได้ในจำนวนที่มากเกินไป วิธีนี้ไม่พิจารณาค่าความเหมาะสม แต่จะกำหนดให้แต่ละโครโมโซมมีค่าคาดหวังว่าจะสุ่มได้จากการพิจารณาลำดับความสำคัญของค่าความเหมาะสมของโครโมโซมในแต่ละรุ่น มีวิธีการคัดเลือกแบบอ้างอิงลำดับอยู่ 2 วิธีที่นิยมใช้ทั่วไป คือ แบบเชิงเส้น (Linear) และแบบไม่เชิงเส้น(Nonlinear)

3.2.4.3 วิธีการสุ่มต้นแบบ

ในการสร้างกลุ่มกำหนดสายพันธุ์ (mating pool) โดยการสุ่มเลือกโครโมโซมต้นแบบตามค่าความน่าจะเป็นที่จะสุ่มได้แต่ละโครโมโซม ซึ่งควรสุ่มได้จำนวนโครโมโซมต้นแบบให้ถูกต้องตามจำนวนที่คาดหวังว่าจะสุ่มได้แต่ละโครโมโซม วิธีที่นิยมใช้มีอยู่ 3 วิธี คือ การจำลองแบบการหมุนวงล้อ การสุ่มทศนิยมแบบคืนกลับ และการสุ่มทศนิยมแบบไม่คืนกลับ

3.2.4.4 เทคนิคการกำเนิดประชากรใหม่

การรีโพรดักชันเป็นกระบวนการเกิดประชากรรุ่นใหม่จากการถ่ายทอดโครโมโซมรุ่นเก่าที่ผ่านขั้นตอนต่างๆ มีหลายวิธีเช่น

ก. การกำเนิดประชากรใหม่แบบทั่วไป เป็นการถ่ายทอดโครโมโซมรุ่นเก่าที่ผ่านขั้นตอนต่างๆ ทั้งหมดกลายเป็นประชากรรุ่นใหม่ตามจำนวนประชากรแต่ละรุ่นที่กำหนด ซึ่งชุดโครโมโซมรุ่นใหม่อาจมีคำตอบที่ไม่ดีเท่าชุดโครโมโซมรุ่นเก่าเนื่องจากการดำเนินการของ GA ไม่มีการบันทึกโครโมโซมที่ดีไว้ ทำให้เกิดการสูญเสียโครโมโซมที่ดี

ข. การกำเนิดประชากรใหม่โดยรักษาโครโมโซมที่ดีที่สุด (Elitism) เป็นการปรับปรุงการกำเนิดประชากรใหม่แบบทั่วไปโดยการรักษาโครโมโซมที่ดีไว้ โดยการคัดลอกโครโมโซมที่ดีของรุ่นเก่ามาไว้ในรุ่นใหม่ด้วยการแทนที่โครโมโซมที่ไม่ดีของรุ่นใหม่

ค. การกำเนิดประชากรใหม่โดยรักษาสถานะคงที่แบบซ้ำ (Steady State with Duplicate Reproduction : SS) โดยการกำหนดอัตราการสร้างโครโมโซมรุ่นใหม่ โดยที่ $0 < G \leq 1$ (Generation Gap : G) สำหรับการกำหนดค่า G ที่เหมาะสมนั้นขึ้นอยู่กับแต่ละปัญหา ซึ่งจำนวนโครโมโซมที่จะถูกสร้างขึ้นใหม่โดยผ่านขั้นตอนต่างๆจะเท่ากับ popsize*G และโครโมโซมส่วนที่เหลือเกิดจากการคัดลอกกลุ่มโครโมโซมที่ดีในรุ่นเก่าตามความสำคัญของโครโมโซม ซึ่งจะทำให้โครโมโซมต่างๆ ที่ดีในรุ่นหนึ่งไปยังอีกรุ่นหนึ่งเพื่อเป็นโครโมโซมต้นแบบได้มาก

ง. การกำเนิดประชากรใหม่โดยรักษาสถานะคงที่แบบไม่ซ้ำ (Steady State without Duplicate Reproduction : SS W/O) จากการดำเนินการของ GA แต่ละโครโมโซมรุ่นใหม่ที่ได้มีโอกาสเป็นโครโมโซมที่เหมือนกันหรือซ้ำกับโครโมโซมรุ่นเก่าได้ นั่นคือจำนวนที่คาดหวังว่าจะสุ่มได้ของโครโมโซมนั้นจะเพิ่มขึ้นทวีคูณตามจำนวนที่เหมือนกันหรือซ้ำกัน ซึ่งทำให้ GA จะต้องใช้เวลาดำเนินการกับโครโมโซมที่ซ้ำกันเหล่านั้นตลอด ดังนั้นการการกำเนิดประชากรใหม่โดยรักษาสถานะคงที่แบบไม่ซ้ำจะช่วยให้การค้นหาคำตอบของ GA เกิดรูปแบบโครโมโซมต่างๆ มากขึ้น

3.3 สรุป

การนำวิธีการเจนนิติกและการจำลองอัลแนลลิ่งมาใช้แก้ปัญหาการหาคำตอบของการจัดกลุ่มวงจร ซึ่งทั้งสองขั้นตอนวิธีอาศัยหลักการสุ่มและการทำซ้ำเพื่อให้ได้คำตอบที่ใกล้เคียงคำตอบที่ดีที่สุด โดยอาศัยหลักการแบ่งแบบ Bisection เปรียบเทียบขั้นตอนการทำงานของทั้งสองวิธีได้ดังนี้

ตารางที่ 3.1 เปรียบเทียบขั้นตอนการทำงานของวิธีจำลองอัลแนลลิ่งกับวิธีเจนนิติก

หลักการของอัลแนลลิ่งโดยทั่วไป	หลักการของเจนนิติกโดยทั่วไป
• คำตอบแรกเลือกอย่างสุ่มเพียงคำตอบเดียว (Single Solution) • การยอมรับคำตอบเกิดจากการเปรียบเทียบ Gainใหม่กับ Gain เก่า จะยอมรับคำตอบที่ดีที่สุดจากการหาค่าต่ำสุดของ Cost Function ภายในระยะเวลาหนึ่ง • คำตอบถูกพัฒนาโดยวิธีการสุ่มย้ายและสลับที่ • ได้คำตอบใหม่เพียงคำตอบเดียวในแต่ละการกระทำซ้ำ	• ค้นหาคำตอบภายใต้โครงสร้างของปัญหาที่เกิดจากการเข้ารหัสรูปแบบโครงสร้างจากกลุ่มตัวแปรต่างๆ ของปัญหา • ค้นหาคำตอบจากการพิจารณาคำตอบจากกลุ่มคำตอบที่สนใจ ไม่ได้พิจารณาคำตอบใดคำตอบหนึ่ง • ค้นหาคำตอบจากผลลัพธ์ของกลุ่มค่าตัวแปรที่เป็นฟังก์ชันเป้าหมายของปัญหา • การปรับปรุงคำตอบในรุ่นถัดมาเกิดจากการ Crossover และ Mutation จะยอมรับ

ตารางที่ 3.1 (ต่อ) เปรียบเทียบขั้นตอนการทำงานของวิธีจำลองอัลแนลลิ่งกับวิธีเจนนิติก

• Average Case Time Complexity คือ $O\left(n^3 \ln n\right)$ โดยที่ n เป็นจำนวนของ Vertexes หรือในกรณี Worst Case Time Complexity คือ $O\left(n^4 \ln n\right)$	คำตอบในรุ่นสุดท้าย • ค้นหาคำตอบจากการอาศัยค่าถ่วงน้ำหนักความเหมาะสมแต่ละคำตอบในแต่ละรุ่น • การพิจารณาคำตอบเพื่อตัดสินว่าคำตอบใหม่ที่ได้ว่าดีขึ้นหรือใกล้เคียงคำตอบที่ต้องการหรือไม่จากฟังก์ชันเป้าหมาย • Time Complexity ขึ้นอยู่กับจำนวนรุ่นที่กำหนดนั้นเองจึงใช้เวลา และขึ้นกับความยาวของโครโมโซม

จากการที่ SA พิจารณาคำตอบเดียวในแต่ละการกระทำซ้ำ ทำให้ใช้เนื้อที่หน่วยความจำในการประมวลผลน้อย และการยอมรับสถานะใหม่ที่เกิดนั้นขึ้นอยู่ ΔGain ถ้าค่า ΔGain มากกว่าหรือเท่ากับศูนย์หรือค่าความน่าจะเป็นของ Boltzmann Distribution น้อยกว่าค่า R ที่มีค่าอยู่ระหว่างศูนย์ถึงหนึ่งเราจะยอมรับสถานะใหม่และจะสิ้นสุดการทำงานเมื่ออุณหภูมิเข้าสู่ศูนย์ ส่งผลให้การประมวลผลช้า เมื่อพิจารณาเจนนิติกมีการค้นหาคำตอบโดยอาศัยหลักการสุ่มหาคำตอบจากรุ่นหนึ่งไปรุ่นถัดไป โดยคำตอบที่ได้เกิดจากการสร้างความสัมพันธ์ของโครงสร้างต่างๆที่เหมาะสมในรุ่นก่อน จึงทำให้ได้คำตอบที่ดีขึ้นนั้นคือการคาดเดาคำตอบใหม่จากสถิติคำตอบเดิมที่ดี การยอมรับคำตอบในแต่ละชุดหรือแต่ละรุ่นนั้นเกิดจากการพิจารณาค่าความเหมาะสมสูงสุด เจนนิติกมีจุดเด่นคือสามารถพัฒนาโปรแกรมและเพิ่มเงื่อนไขต่างๆได้ง่าย แต่มีข้อเสียตรงที่ถ้าจำนวนบิตที่ใช้ในการหาคำตอบมากเกินไปทำให้ต้องใช้เวลาในการคำนวณมาก โดยอาศัยหลักการแบ่งแบบ Bisection ทำให้จำนวนคำตอบที่สามารถเกิดขึ้นได้มีค่าเท่ากับ N จากจำนวนวงจร n โนด ซึ่งหาได้ดังนี้

ถ้า n เป็นเลขคู่

$$N = \frac{n!}{\left(\frac{n}{2}\right)!\left(\frac{n}{2}\right)!} \tag{3.25}$$

ถ้า n เป็นเลขคี่

$$N = \frac{n!}{\left(\frac{n+1}{2}\right)!\left(\frac{n-1}{2}\right)!} \times 2 \tag{3.26}$$

วิธีการผสมเกิดจากการหาคำตอบจากการพิจารณาคำตอบจากกลุ่มคำตอบหนึ่งที่ได้จากการสุ่มหาคำตอบตามหลักการของวิธีเจนนิติก และเลือกพิจารณาหาคำตอบที่ดีที่สุดจากการพิจารณาคำตอบเพียงคำตอบเดียวโดยอาศัยขั้นตอนการยอมรับคำตอบแบบเดียวกับวิธีจำลองอัลแนลลิ่ง แต่มีการค้นคำตอบที่เกิดจากการมิวเตชั่น และมีการตัดคำตอบที่ซ้ำซ้อนออกเพื่อลดจำนวนคำตอบที่สนใจ รายละเอียดจะได้เสนอในบทที่ 4 ต่อไป

บทที่ 4

วิธีการผสม (Hybrid algorithm)

การใช้วิธีการจำลองอัลแนลลิ่งและวิธีเจนนิติกในแก้ปัญหาการจัดกลุ่มวงจรนั้น เป็นการแก้ปัญหาที่ต้องการคำตอบที่ดีที่สุด (Optimal Solution) วิธีการผสมเกิดจากการนำข้อดีของทั้งสองวิธีดังกล่าวมาใช้ร่วมกัน โดยมีจุดประสงค์เพื่อที่จะปรับปรุงขั้นตอนวิธีการแก้ปัญหาการจัดกลุ่มวงจรของขบวนการออกแบบวงจรรวมทางกายภาพแบบอัตโนมัติ (IC Physical Design Automation) ให้มีประสิทธิภาพดียิ่งขึ้น โดยคาดหวังผลลัพธ์ที่จะได้เป็นความสัมพันธ์ระหว่างกลุ่มที่จัดวาง เพื่อลดความสัมพันธ์ระหว่างกลุ่มที่เกิดขึ้นซึ่งจะส่งผลต่อการลดสายสัญญาณที่ใช้เชื่อมโยงและลดเวลาในขบวนการออกแบบวงจรรวมทางกายภาพแบบอัตโนมัติให้มีความรวดเร็วขึ้น

4.1 วิธีการผสม

วิธีการผสมแบ่งออกเป็น 2 ส่วนหลัก คือ ส่วนที่เป็นการวิเคราะห์ปัญหาและส่วนที่เป็นขั้นตอนการทำงาน เริ่มจากการวิเคราะห์ปัญหาเริ่มต้นจากการมองปัญหาให้อยู่ในรูปของกราฟและมองปัญหาเทียบเท่ากับโครโมโซมชนิดหนึ่งๆ กำหนดรูปแบบโครโมโซมให้อยู่ในรูปไบนารี โครโมโซมคือประกอบด้วยบิทที่มีค่าเป็น 0 หรือ 1 ซึ่งเป็นเลขฐานสองและมีความยาวของโครโมโซม (Chromosome Length) ที่ขึ้นอยู่กับการกำหนดของปัญหา การกำหนดฟังก์ชันความเหมาะสมของปัญหาอย่างเช่น การหาค่าสูงสุดหรือการหาค่าต่ำสุดของฟังก์ชันความเหมาะสม เพื่อให้ง่ายต่อขั้นตอนการทำงานถัดมาสำหรับการหาคำตอบ

ถัดมาเป็นส่วนขั้นตอนการทำงานจะประกอบด้วยหลายขั้นตอนย่อยๆ ดังนี้

ก. จากการมองปัญหาให้อยู่ในรูปของโครโมโซมจะพบว่ามีปัญหาซ้ำซ้อนเกิดขึ้นซึ่งเกิดจากการอินเวอร์ (invert) บิทกันในทุกตำแหน่งของโครโมโซม ทำให้ได้คำตอบที่สามารถเกิดขึ้นได้ 2 ชุด ดูจากรูปที่ 4.1 เพื่อเป็นการลดจำนวนคำตอบที่ใช้ในการสุ่มหาคำตอบที่ต้องการ จึงเลือกพิจารณาคำตอบเพียงชุดเดียวโดยการสุ่มเลือกคำตอบชุดแรกหรือประชากรรุ่นแรกออกมาโดยคำนึงถึงค่าบิทแรกว่าต้องเป็นบิทเดียวกัน นั้นคืออาจจะเลือกสุ่มหาค่าบิทแรกเป็น 0 ทั้งชุดหรือเป็น 1 ทั้งชุดก็ได้

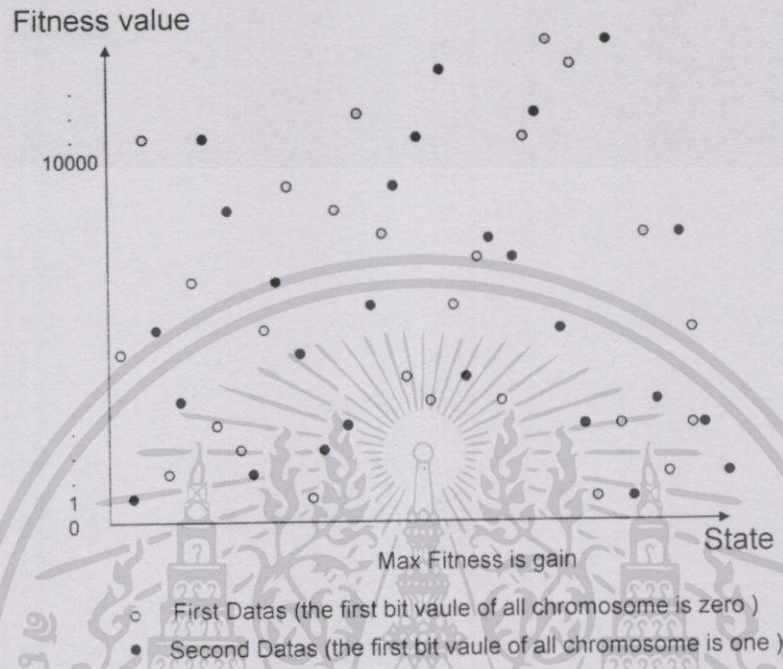
วิธีการนี้จะลดเซตของสถานะที่เป็นไปได้ทั้งหมด เป็นการลดสถานะรอบข้างของการสุ่มหาคำตอบ ส่งผลให้เกิดการยอมรับคำตอบเร็วขึ้น ดังนั้นจำนวนคำตอบทั้งหมดที่เกิดจากวิธีการนี้หาได้จาก

ถ้า n เป็นเลขคู่

$$N = \frac{n!}{\left(\frac{n}{2}\right)!\left(\frac{n}{2}\right)!\,2!} \qquad (4.1)$$

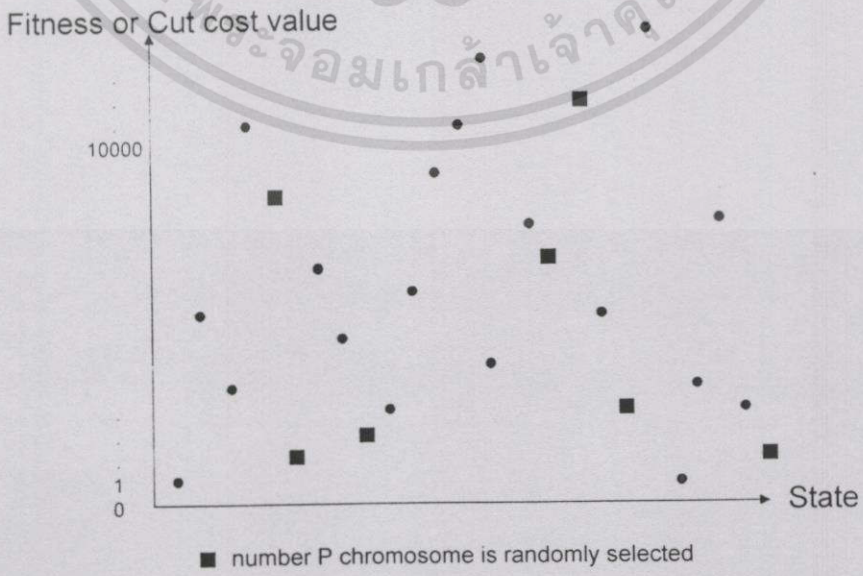
ถ้า n เป็นเลขคี่

$$N = \frac{n!}{\left(\frac{n+1}{2}\right)!\left(\frac{n-1}{2}\right)!} \quad (4.2)$$



รูปที่ 4.1 แสดงการจำลองประชากรทั้งหมดที่เกิดขึ้นจากการหาคำตอบที่สามารถเป็นไปได้

เช่นถ้าจำนวนวงจรเป็น n = 10 จำนวนคำตอบที่จะเกิดขึ้นทั้งหมด 2^{10}-2 = 1022 เมื่อแบ่งกลุ่มแบบ Bisection โดยคำนึงถึงลำดับการจัดวางสามารถสุ่มหาคำตอบได้จากกลุ่มคำตอบที่หาได้จากสมการที่ 3.25 ซึ่งได้ค่าเท่ากับ 252 คำตอบและถ้าสุ่มหาคำตอบโดยคำนึงถึงลำดับการจัดวางแล้วจำนวนคำตอบในกลุ่มที่สุ่มหาคำตอบจะลดลงเหลือ 126 คำตอบซึ่งหาได้จากสมการที่ 4.1



รูปที่ 4.2 แสดงการจำลองกลุ่มประชากรรุ่นแรกที่สุ่มหาจากชุดคำตอบที่ไม่มีค่าคำตอบซ้ำซ้อน

ข. คัดเลือกโครโมโซมที่ดีที่สุดจากข้อกำหนดความต้องการของฟังก์ชันความเหมาะสมมาเพียงหนึ่งโครโมโซม กำหนดค่านี้ให้เป็น Main Value คือค่าที่ใช้เป็นเกณฑ์ในการแบ่งคำตอบออกเป็น 2 กลุ่มคือกลุ่มที่มีค่าที่สูงกว่าค่า Main Value กับกลุ่มที่มีค่าน้อยกว่าค่า Main Value แล้วทำการค้นหาคำตอบในกลุ่มใดกลุ่มหนึ่งโดยประยุกต์ใช้วิธีการจำลองอัลแนลลิ่งเข้ามาช่วย เริ่มจากการกำหนดอุณหภูมิเริ่มต้น แล้วเข้าสู่ขั้นตอนการทำงานซ้ำดังวิธีการในรูปที่ 4.3 ภายในลูปทั้ง 3 จนกระทั่งถึงจุดที่พอใจ ซึ่งเราจะได้ Local Optimal Solution ที่ใกล้เคียงกับค่า Global Optimal Solution ที่ลูปในรอบสุดเกิดจากการมิเตชันโครโมโซม โดยการสุ่มหาตำแหน่งที่จะทำการกลับค่าบิตของ 1 และ 0 แต่จะยกเว้นตำแหน่งแรกของโครโมโซม จำนวนที่ทำการกลับค่าบิตไม่ควรเกินครึ่งหนึ่งของความยาวโครโมโซม เมื่อหาค่าฟังก์ชันความเหมาะสมของโครโมโซมออกแล้วให้ทำการเปรียบเทียบกับค่า Main Value เพื่อให้คำตอบที่ได้อยู่ในกลุ่มที่ต้องการ แล้วเข้าสู่การคัดเลือกคำตอบของสถานะใหม่กับคำตอบสถานะเดิม โดยให้ ΔGain เป็นค่าความแตกต่างระหว่างค่าฟังก์ชันเป้าหมายเดิมกับค่าฟังก์ชันเป้าหมายใหม่ ถ้า $\Delta\text{Gain} \leq 0$ แสดงว่า สถานะใหม่ดีกว่าสถานะเดิม แต่ถ้า $\Delta\text{Gain} > 0$ การยอมรับสถานะใหม่หรือไม่ขึ้นอยู่กับค่าความน่าจะเป็นของ Boltzman Distribution เช่นเดียวกับการยอมรับค่าใหม่ที่ได้จากวิธีจำลองอัลแนลลิ่ง

4.2 การจัดกลุ่มวงจรโดยใช้วิธีการผสม

การจัดกลุ่มวงจรเป็นปัญหาประเภทเอ็นพีค็อมพลีทดังที่ได้กล่าวมาแล้ว โดยจะอาศัยหลักการแบ่งกลุ่ม Two-Way Partitioning แบบ Bisection ข้อเสียคือถ้าจำนวนกลุ่มที่ต้องการแบ่งมากจำเป็นต้องใช้เวลาในการแบ่งมากขึ้นตาม การจัดกลุ่มโดยวิธีการผสมมีขั้นตอนการดำเนินงานตามแผนผังดังรูปที่ 4.3 โดยมีรายละเอียดดังต่อไปนี้

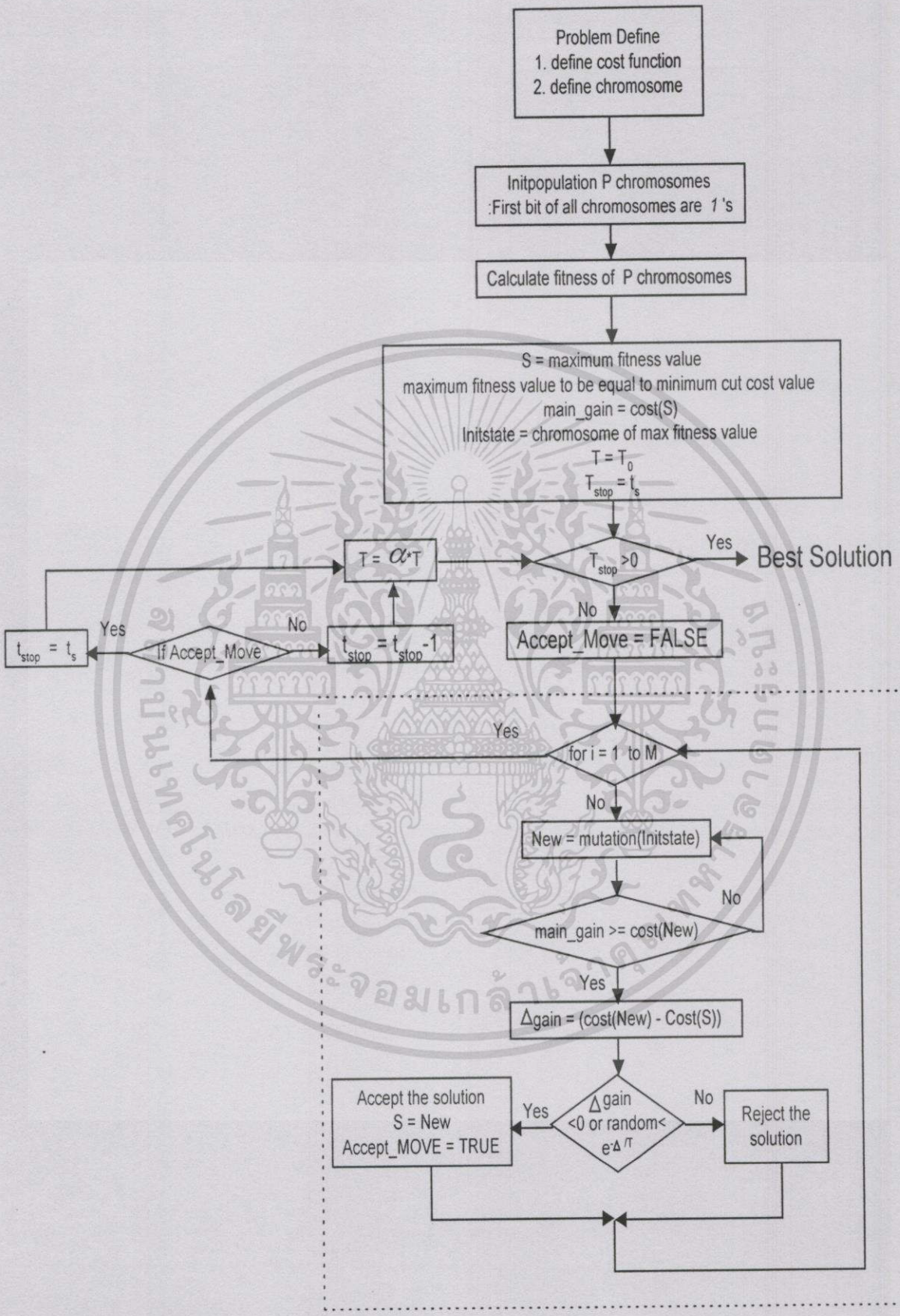
ขั้นตอนที่ 1 เริ่มต้นด้วยการแทน Circuit Network ด้วย Undirected Graph G(V,E) เมื่อ V เป็นเซ็ตของวงจร ดังรูปที่ 4.4

$$V_i \in V, V = \{V_1, V_2, V_3, ..., V_n\}$$

และ E เป็นเซ็ตของก้าน(edge)

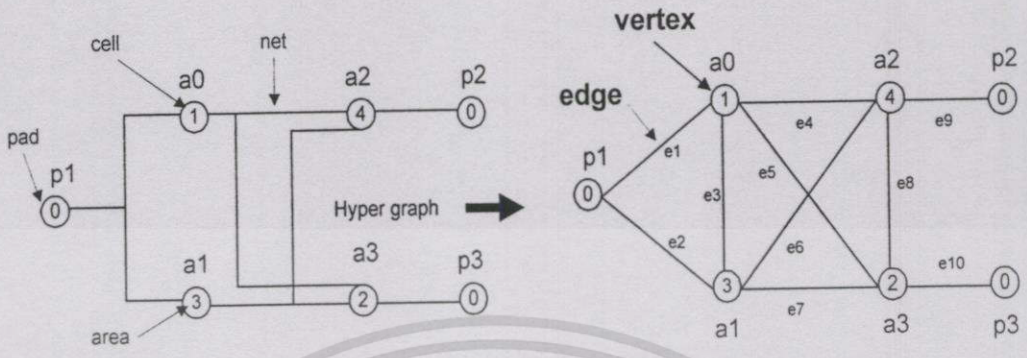
$$C_i \in E, E = \{e_1, e_2, e_3, ..., e_m\}$$

ให้ c_{ij} เป็นจำนวนก้าน (edge) ระหว่าง V_i กับ V_j และ $i \neq j$ กำหนดเซ็ตสถานะ



รูปที่ 4.3 แสดงขั้นตอนวิธีการผสมการทำงานระหว่างวิธีการจำลองอัลแนลลิ่งกับวิธีเจนนิติก

ให้ c_{ij} เป็นจำนวนก้าน(edge)ระหว่าง V_i กับ V_j และ $i \neq j$ กำหนดเซ็ตสถานะ



รูปที่ 4.4 แสดงการแทนปัญหาด้วยกราฟ

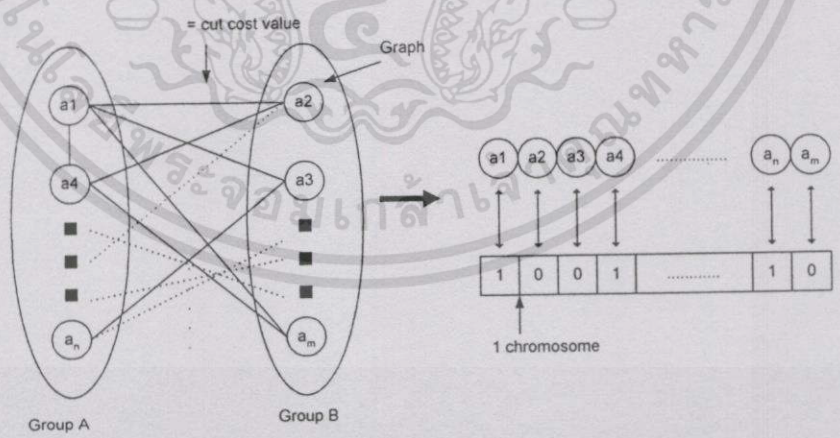
กำหนดฟังก์ชั่นเป้าหมายของการจัดกลุ่มวงจรเป็น

1. ความสมดุลย์ของกลุ่มที่แบ่ง (A,B) = ขนาดของกลุ่ม A - ขนาดของกลุ่ม B

$$= \sum_{v \in A} s(v) - \sum_{v \in B} s(v) \tag{4.3}$$

2. จำนวนความสัมพันธ์ระหว่างกลุ่มที่แบ่ง C_{AB}

$$C_{AB} = \sum_{i \in A} \sum_{j \in B} C_{ij} \tag{4.4}$$



รูปที่ 4.5 แสดงความสัมพันธ์ระหว่างฟังก์ชันเป้าหมายกับโครโมโซม

จากรูปที่ 4.5 การแบ่งกลุ่มวงจรออกเป็นสองกลุ่มโดยมีจำนวนกลุ่มที่แบ่งประมาณเท่ากัน Vertex ที่อยู่ทางซ้ายของเส้นแบ่งหรือ Vertex ที่อยู่ในกลุ่ม A มีค่าบิทเท่ากับ 1 ในโครโมโซม ส่วน Vertex ที่อยู่ทางขวาของเส้นแบ่งหรือ Vertex ที่อยู่ในกลุ่ม B มีค่าบิทเท่ากับ 0 ในโครโมโซมนั้นๆ

ขั้นตอนที่ 2 สุ่มเลือกประชากรมาจำนวน P โครโมโซมโดยคัดเลือกโครโมโซมที่มีค่าบิทแรกเป็นค่า 1 ทั้งจำนวน P ตัว หรือมีค่าบิทแรกเป็นค่า 0 ทั้งจำนวน P โครโมโซม เป็นการสุ่มตัวอย่างแบบสุ่มกลุ่ม ในที่นี้มี 2 กลุ่ม

ขั้นตอนที่ 3 จากนั้นทำการวิเคราะห์ค่าความเหมาะสมแต่ละโครโมโซมประชากรต้นกำเนิด ตรวจสอบเงื่อนไขความเหมาะสมต่างๆ โดยคำนวณหาค่าความเหมาะสม F_i สำหรับแต่ละโครโมโซมได้จาก

$$F_i = (C_w - C_i) + \frac{C_w - C_b}{3} \tag{4.5}$$

ขั้นตอนที่ 4 หาค่าความน่าจะเป็นที่จะถูกเลือกต่อการสุ่มเลือกแต่ละครั้ง (Probability of Selected Value : pselect)

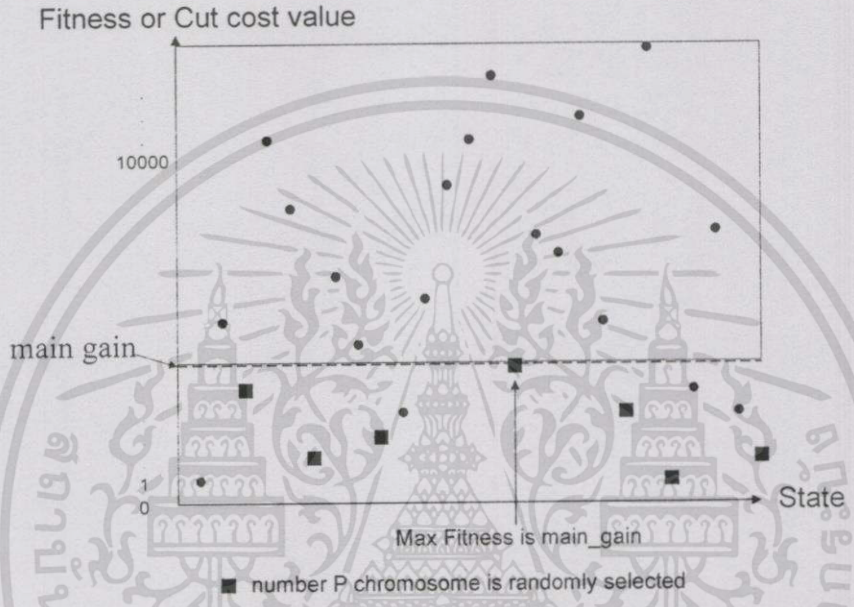
$$pselect = F_i / \Sigma F \tag{4.6}$$

เพื่อทำการคัดเลือกโครโมโซมประชากรที่ดีที่สุดหรือคำตอบที่เหมาะสมที่สุดมาหนึ่งคำตอบ กำหนดคำตอบนี้เป็นค่า Main_Gain ซึ่งจะเป็นค่าที่ใช้เป็นเกณฑ์คัดเลือกคำตอบอีกขั้นหนึ่งของขบวนการหาคำตอบ และส่งคำตอบแรกที่ได้พร้อมทั้งโครโมโซมของคำตอบไปเป็นคำตอบแรกของขบวนการหาคำตอบโดยอาศัยหลักของวิธีการจำลองอัลแนลลิ่ง ซึ่งวิธีการคัดเลือกคำตอบแรกที่อยู่ในกลุ่มคำตอบที่เข้าใกล้คำตอบที่ดีที่สุดนี้จะส่งผลให้ความแตกต่างระหว่างค่าฟังก์ชันเป้าเดิมกับค่าฟังก์ชันเป้าหมายใหม่มีค่าน้อย โอกาสการยอมรับคำตอบมีสูงเป็นผลให้การทำงานของขั้นตอนวิธี วิ่งเข้าหาคำตอบที่เหมาะสมได้เร็วขึ้น ดังรูปที่ 4.6

```
Hybrid algorithm (Fmaxi,Fchrom,To,alpha,Maxtime,M)
Fmaxi  is maximum fitness
Fchrom is string of maximum fitness chromosome
To  is the initial temperature
alpha is cooling rate
Maxtime is the total allowed time for the annealing process
M represents the time until the next parameter update
begin   Selection to random P chromosome
        Fmaxi = Find to maximum fitness of P chromosome
        T=To
        S=Calculate_Gain(Fmaxi)
        arrayS = Fchrom
        Time =0
        do{     Metropolis(S,arrayS,T,M)
                Time = Time*M
                T = alpha * T
        }until(Time>= Maxtime)
        output best solution found
end
```

รูปที่ 4.6 แสดงขั้นตอนวิธีผสม

ขั้นตอนที่ 5 และขั้นตอนที่ 6 ขั้นตอนการทำซ้ำของลูปใน Metropolis Algorithm เริ่มต้นด้วยการเปลี่ยนสถานะใหม่โดยวิธีการมิวเตชันโครโมโซมโดยยกเว้นค่าบิทแรกของโครโมโซม และนำค่าความเหมาะสมหรือคำตอบที่ได้ไปทำการเปรียบเทียบกับค่า Main Gain เพื่อคัดเลือกคำตอบที่ได้ว่าควรยอมรับได้หรือไม่ ค่า Main Gain ถือว่าเป็นเกณฑ์ที่จะส่งผลให้ไปช่วยลดจำนวนคำตอบหรือลดสภาวะแวดล้อมรอบๆข้างของการหาคำตอบอีกขั้นหนึ่ง กล่าวได้ว่าทำให้เกิดการตัดคำตอบที่เลวออกไป เหลือแต่คำตอบที่ควรสนใจ ส่งผลให้การทำงานเร็วขึ้น แสดงดังรูปที่ 4.7

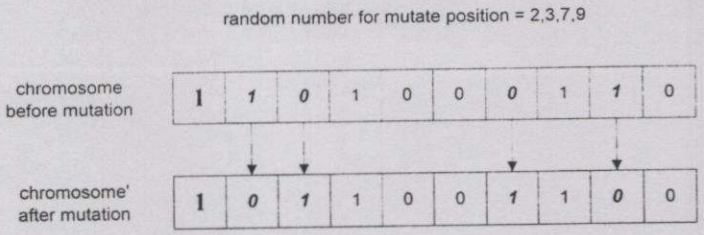


รูปที่ 4.7 แสดงคำตอบที่ถูกสุ่มเลือกมาในการหาประชากรรุ่นแรก

```
mutation(array_chromosome)
begin  random(position_1,position_0 ) //discharge first position
       newchrom= mutation(array_chromosome(position_1,position_0 ))
       return  newchrom
end
```

รูปที่ 4.8 แสดงขั้นตอนการมิวเตชัน

การมิวเตชันเป็นการกลับค่าบิตจาก 1 เป็น 0 หรือจาก 0 เป็น 1 จะทำการกลับบิตเป็นคู่ตามจำนวนคู่ที่สุ่มได้จากค่าที่อยู่ระหว่าง 1 ถึงค่าครึ่งหนึ่งของความยาวของโครโมโซมนั้นๆ



รูปที่ 4.9 แสดงการมิวเตชันของโครโมโซม

ขั้นตอนที่ 7 เป็นการเปรียบเทียบค่าฟังก์ชันเป้าหมายหรือค่าความเหมาะสมใหม่กับค่าความเหมาะสมเดิม โดยพิจารณา Δgain เช่นเดียวกับวิธีการจำลองอัลแนลลิ่ง คือการยอมรับสถานะใหม่ก็ต่อเมื่อ Δgain มีค่าน้อยกว่าหรือเท่ากับศูนย์ หรือค่าความน่าจะเป็นของ Boltzmann Distribution น้อยกว่าค่า R ดังในหัวข้อที่ 3.1

ขั้นตอนสุดท้าย เป็นขั้นตอนที่ภายหลังจากการทำซ้ำทั้ง 3 ลูป โดยลูปนอกสุดมีอุณหภูมิเป็นตัวควบคุม T ซึ่งจะเป็นตัวกำหนดการลดลงของแฟกเตอร์ α ที่ $0 < \alpha < 1$ ขั้นตอนวิธีการจะหยุดเมื่อ T_0 ลดลงจนเข้าสู่ศูนย์ ค่า α ที่ใช้จะอยู่ระหว่าง $0.8 < \alpha < 0.99$ ดังสมการนี้

$$T_k = \alpha T_{k-1} = \alpha^k T_0 \qquad (4.7)$$

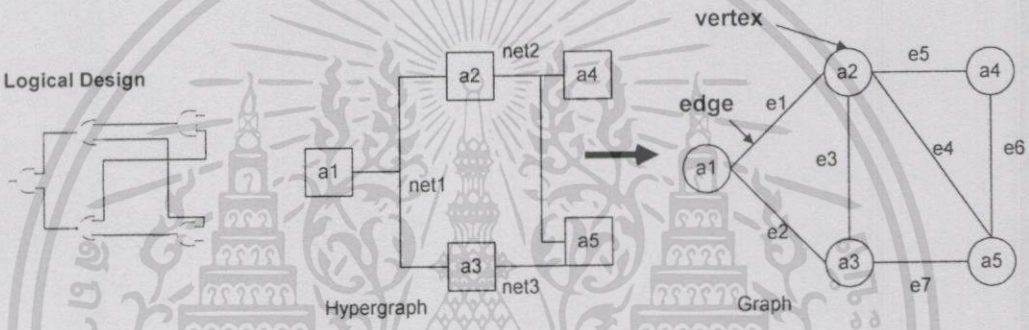
ตารางที่ 4.1 แสดงจำนวนคำตอบที่เกิดขึ้นได้ทั้งหมดในแต่ละวงจรตัวอย่างจากวิธีจำลองอัลแนลลิ่ง วิธีเจนนิติก และวิธีการผสม

จำนวน Vertex ของ วงจรตัวอย่าง	จำนวนคำตอบที่เกิดขึ้นในแต่ละวิธี โดย Two Way Partition แบบ Bisection		
	SA	GA	HA
6	20	20	10
8	70	70	37
9	252	252	126
10	252	252	126
12	924	924	462
15	12870	12870	6435
16	12870	12870	6435
82	92378	92378	46189

จากตารางที่ 4.1 วิธีการจำลองอัลแนลลิ่งและวิธีเจนนิติก นั้นสามารถหาคำตอบทั้งหมดที่สามารถเกิดขึ้นได้จากสมการที่ 3.25 และสมการที่ 3.26 ซึ่งเป็นวิธีการหาคำตอบโดยที่ไม่สนใจคำตอบที่เกิดซ้ำซ้อน ส่วนวิธีการผสม สามารถหาคำตอบทั้งหมดได้จากสมการที่ 4.1 และสมการที่ 4.2 จะเห็นได้ว่าจำนวนคำตอบที่สามารถเกิดขึ้นได้นั้นลดลงไปครึ่งหนึ่งจากการหาคำตอบที่ได้จากสมการที่ 3.25 และสมการที่ 3.26 นั้นคือสมการที่ 4.1 และสมการที่ 4.2 ได้จะตัดคำตอบซ้ำซ้อนออกไปเพื่อลดสภาวะแวดล้อมของการค้นหาคำตอบของวิธีการทำงานลง

แสดงตัวอย่างการวิเคราะห์การหาคำตอบที่จะนำมาพิจารณาหาค่าคำตอบที่ดีที่สุด

สมมุติให้จำนวน Node หรือ Vertex มีทั้งหมด 5 ตัว ดังรูปที่ 4.10 สามารถหาคำตอบที่นำมาพิจารณาได้ดังนี้ Two-Way Partitioning $N= 2^5-2 = 30$ คำตอบ



รูปที่ 4.10 แสดงตัวอย่างการแทนวงจรด้วยกราฟ

การแบ่งคำตอบแบบ Bisection ; n = 5 $N = \frac{n!}{\left(\frac{n+1}{2}\right)!\left(\frac{n-1}{2}\right)} \times 2$ = 20 คำตอบ

การแบ่งคำตอบแบบ Bisection โดย HA ; n = 5 $N = \frac{n!}{\left(\frac{n+1}{2}\right)!\left(\frac{n-1}{2}\right)}$ = 10 คำตอบ

จากการแบ่งกลุ่มโดยอาศัย Two-Way Partitioning แบบ Bisection และอยู่บนหลักการของการเจนนิติกโดยการเข้ารหัสตารางที่ 4.2 แสดงโครโมโซมหรือจำนวนคำตอบทั้งหมดที่เกิดขึ้นเมื่อมีจำนวน node เป็น 5 node

ตารางที่ 4.2 แสดงโครโมโซม 32 โครโมโซมที่เป็นคำตอบที่เกิดขึ้นทั้งหมด

	a1	a2	a3	a4	a 5		Cost
Chromosome 1	0	0	0	0	0	เป็นคำตอบที่ไม่สามารถเกิดขึ้นได้เลย	
Chromosome 2	0	0	0	0	1		
Chromosome 3	0	0	0	1	0		
Chromosome 4	0	0	0	1	1	* Bisection chromosome	3
Chromosome 5	0	0	1	0	0		
Chromosome 6	0	0	1	0	1	*	5
Chromosome 7	0	0	1	1	0	*	4
Chromosome 8	0	0	1	1	1	*	3
Chromosome 9	0	1	0	0	0		
Chromosome 10	0	1	0	0	1	*	4
Chromosome 11	0	1	0	1	0	*	5
Chromosome 12	0	1	0	1	1	*	4
Chromosome 13	0	1	1	0	0	*	5
Chromosome 14	0	1	1	0	1	*	4
Chromosome 15	0	1	1	1	0	*	5
Chromosome 16	0	1	1	1	1		
Chromosome 17	1	0	0	0	0		
Chromosome 18	1	0	0	0	1	*	5
Chromosome 19	1	0	0	1	0	*	4
Chromosome 20	1	0	0	1	1	*	5
Chromosome 21	1	0	1	0	0	*	4
Chromosome 22	1	0	1	0	1	*	5
Chromosome 23	1	0	1	1	0	*	4
Chromosome 24	1	0	1	1	1		
Chromosome 25	1	1	0	0	0	*	3
Chromosome 26	1	1	0	0	1	*	4
Chromosome 27	1	1	0	1	0	*	5
Chromosome 28	1	1	0	1	1		
Chromosome 29	1	1	1	0	0	*	3
Chromosome 30	1	1	1	0	1		
Chromosome 31	1	1	1	1	0		
Chromosome 32	1	1	1	1	1	เป็นคำตอบที่ไม่สามารถเกิดขึ้นได้เลย	

ในกรณีใช้วิธีการจำลองอัลแนลลิ่งและวิธีเจนนิติกนั้น ต้องสุ่มหาคำตอบจากจำนวนคำตอบ 20 คำตอบ ซึ่งในที่นี้จะมีคำตอบหนึ่งค่าที่เกิดจากโครโมโซม 2 ตัวที่กลับค่าบิทกัน แต่วิธีการผสมจะค้นหาคำตอบโดยที่แต่ละคำตอบเกิดจากโครโมโซมเพียงโครโมโซมเดียว ทำให้ลดจำนวนคำตอบที่สามารถเกิดขึ้นได้ถึงครึ่งหนึ่ง

4.3 สรุป

การใช้วิธีการจำลองอัลแนลลิ่งแก้ปัญหาการจัดกลุ่มวงจรนั้น คำตอบที่เกิดขึ้นจะมีเป็นจำนวนมาก การหาคำตอบที่ดีที่สุดจึงต้องอาศัยเวลามาก ในขณะเดียวกันถ้าใช้วิธีเจนนิติกแก้ปัญหาดังกล่าวจะส่งผลให้ต้องใช้หน่วยความจำในการประมวลผลมาก ดังนั้นขั้นตอนวิธีผสมที่นำเสนอจะช่วยลดเวลาและลดปัญหาเรื่องหน่วยความจำในการประมวลผลได้ในระดับหนึ่ง โดยการหาคำตอบเริ่มแรกจากวิธีเจนนิติกโดยการหาโครโมโซมที่ดีที่สุดในประชากรรุ่นแรก เพื่อกำหนดเป็นเกณฑ์การแบ่งกลุ่มคำตอบออกเป็นสองกลุ่มอย่างคร่าวๆ และพิจารณาเซตคำตอบที่อยู่เหนือหรือต่ำกว่าเกณฑ์นี้ขึ้นอยู่กับฟังก์ชันเป้าหมายที่กำหนด การหาคำตอบใหม่อาศัยแนวทางวิธีการจำลองอัลแนลลิ่งและวิธีเจนนิติก เป็นการลดสถานะที่เป็นไปได้ทั้งหมดให้แคบลง ทำให้จำนวนสถานะรอบข้างลดลงตาม ส่งผลให้ลดเวลาในการค้นหาคำตอบที่ดีที่สุด การกำหนดฟังก์ชันค่าเป้าหมายและการจัดการลดค่าอุณหภูมิ หรือจำนวนรอบของการทำซ้ำซึ่งต่างจากวิธีการที่ได้นำมาเปรียบเทียบที่จำเพาะกับปัญหาและการใช้หลักการพันธุศาสตร์เข้ามามีส่วนเกี่ยวข้องในการคัดเลือกคำตอบที่เหมาะสม นับได้ว่าเป็นการเพิ่มประสิทธิภาพและสร้างลักษณะเด่นให้กับวิธีการนี้

บทที่ 5
ผลการทดลองการจัดกลุ่มวงจรแบบอัตโนมัติโดยใช้ขั้นตอนวิธีการใหม่

5.1 ทดสอบกับ ISPD98 Circuit Benchmark Suite

ในการทดสอบประสิทธิภาพวิธีการผสมกับวิธีการจำลองอัลแนลลิ่งและวิธีการเจนนิติก บนการแก้ปัญหาการจัดกลุ่มวงจร โดยใช้วงจรทดสอบจาก ISPD98 Circuit Benchmark Suite ซึ่งเหมาะสำหรับการแก้ปัญหาการจัดกลุ่มวงจรแบบอัตโนมัติ[34] รายละเอียดของ ISPD98 Circuit Benchmark แสดงในตารางที่ 5.1

ตารางที่ 5.1 แสดงคุณสมบัติของวงจรที่ใช้ในการทดสอบ

Circuit (netlist.net)	# Cells	# Pads	# Modules	# Nets	# Pins
EX8	8	0	8	19	25
EX9	9	0	9	17	37
EX10	10	0	10	20	36
Qgr8	12	0	12	23	46
3k5	15	0	15	33	66
Clique	16	0	16	120	240
IC67	16	51	67	138	474
IC116	15	100	115	329	876
IC151	16	135	151	419	988
primary1	82	751	833	902	2908
primary2	108	2906	3014	3029	11219
test02	62	1601	1663	1720	6134
test03	58	1549	1607	1618	5807
test04	27	1488	1515	1658	5975
test05	56	2539	2595	2750	10076
test06	62	1690	1752	1641	6638

ตารางที่ 5.1 (ต่อ) แสดงคุณสมบัติของวงจรที่ใช้ในการทดสอบ

circuit (netlist.net)	# Cells	# Pads	# Modules	# Nets	# Pins
bm1	131	751	882	903	2910
19ks	161	2683	2844	3282	10547
Fract	149	0	149	147	462
Balu	801	0	801	735	2697

Netlist ประเภท a.net จะเริ่มต้นด้วย 5 บรรทัดแรกที่บ่งบอกคุณสมบัติของ Circuit Benchmark มีรายละเอียดดังนี้

ignored

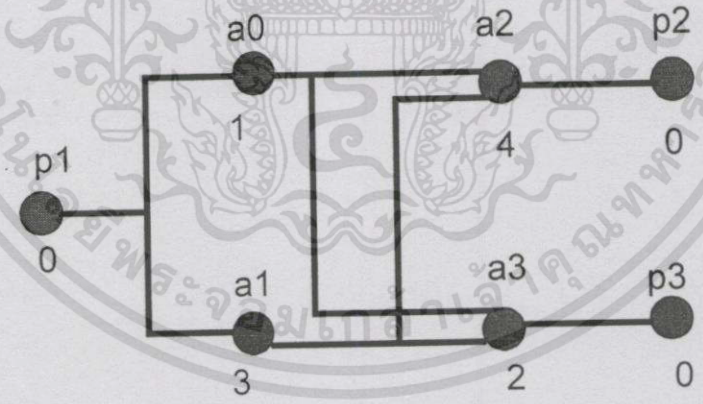
#Pins คือจำนวนของ pins ที่เกิดขึ้นทั้งหมดของ Circuit Benchmark นั้นๆ

#Nets คือจำนวนเส้นเชื่อมหรือเส้นแสดงความสัมพันธ์ทั้งหมดของ Circuit Benchmark นั้นๆ

#Modules คือจำนวนของวงจร cells และ pads ทั้งหมดรวมกัน โดยถูกแทนด้วยกราฟ

#Pad offset คือจำนวนของ pad ทั้งหมดของวงจรใน Circuit Benchmark นั้นๆ ถูกแทนด้วยกราฟ

#Cells คือจำนวนของวงจรทั้งหมดให้ Circuit Benchmark (หาได้จาก #Modules - #Pad offset)



รูปที่ 5.1 แสดงตัวอย่างวงจรที่ใช้ทดสอบโดยการแทนด้วยกราฟ

ใน circuit netlist นั้นความหมายของ "a" คือส่วนที่เป็นวงจร ความหมายของ "p" คือส่วนที่เป็น pad การเริ่มต้นความสัมพันธ์ของวงจรหนึ่งต่อวงจรอื่นในแต่ละ nets แสดงด้วย "s" สามารถดูรายละเอียดเพิ่มเติมของวงจรได้จากภาคผนวก ข.

ตัวอย่าง Netlist file ในภาพที่ 5.1 ประกอบด้วย 4 cells , 3 pads, 5 nets และ 13 pins Netfile นี้กำหนดได้ดังนี้

13

5

7

3

p1 s l

a0 l

a1 I

a0 s l

a2 l

a3 l

a1 s l

a2 l

a3 l

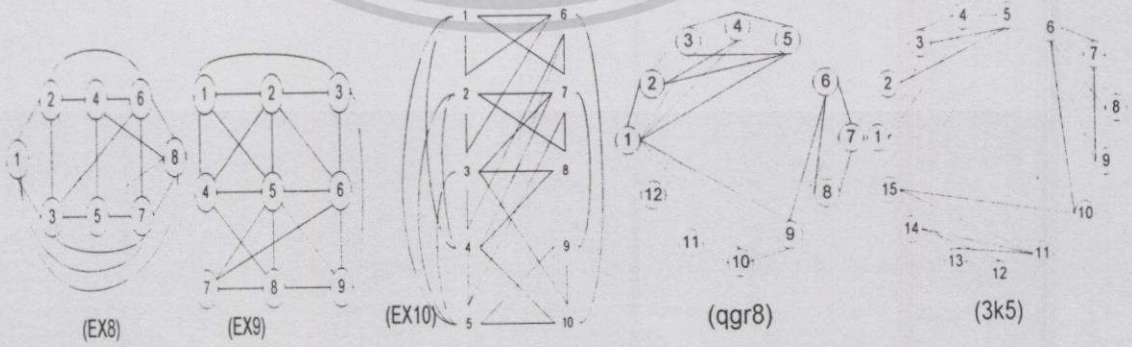
a2 s l

p2 l

a3 s l

p3 l

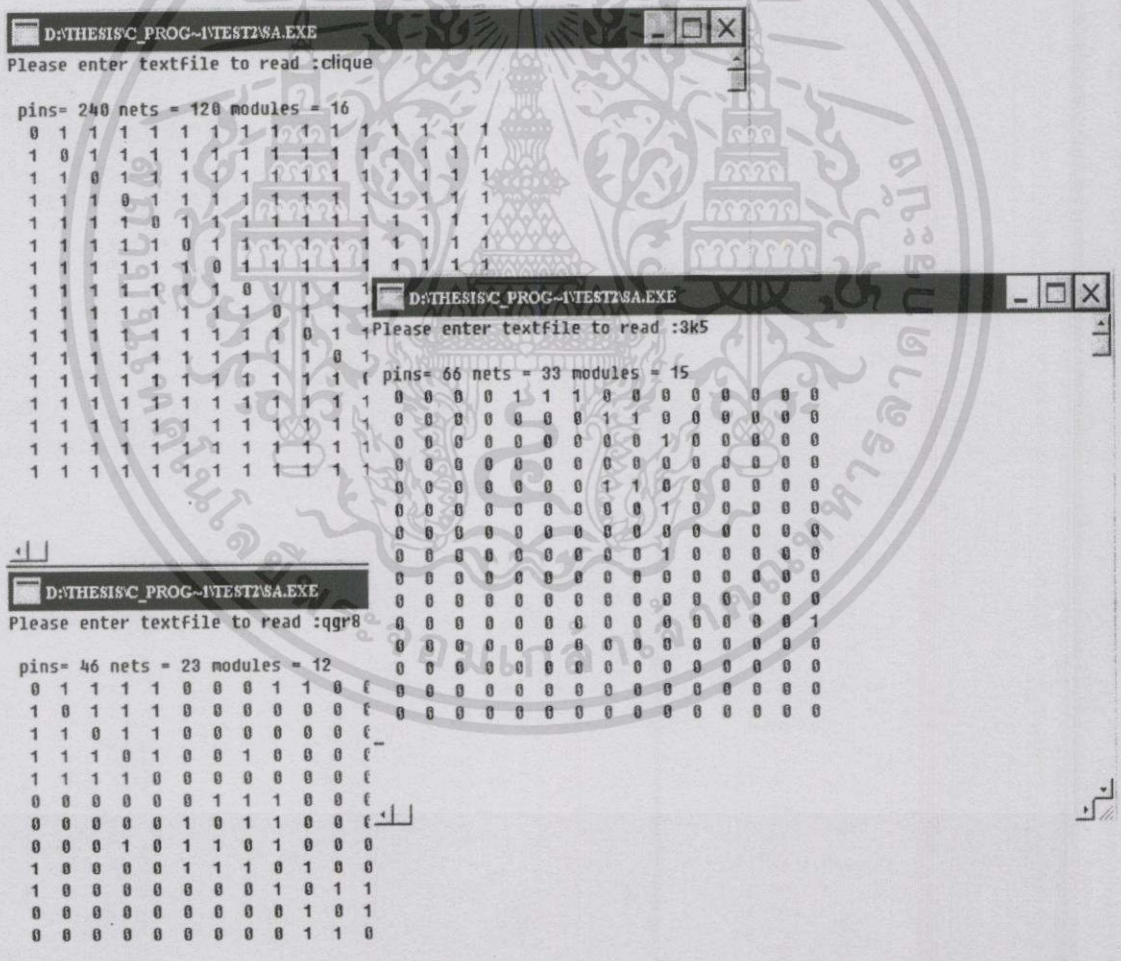
จากรูปที่ 5.2 แสดงให้เห็นถึงความสัมพันธ์ของแต่ละ node ที่เกิดในแต่ละวงจรตัวอย่างที่มีขนาดเล็ก แต่ในกรณีที่วงจรขนาดใหญ่นั้นเป็นการยากที่แสดงให้เห็นถึงความสัมพันธ์ระหว่าง node ได้อย่างชัดเจน



รูปที่ 5.2 แสดงความสัมพันธ์ของแต่ละ node ในวงจร EX8, EX9, EX10, qgr8 และ 3k5 ดังรายละเอียดใน netlist.net

5.2 ตัวอย่างการทำงานของโปรแกรม

โปรแกรมการจัดกลุ่มวงจรจะให้ใส่ชื่อ netlist.net ซึ่งเป็น Circuit Benchmark ที่ได้กล่าวไว้ในตารางที่ 5.1 เพื่อทำการหาค่าความสัมพันธ์ของแต่ละโนดวงจร โปรแกรมจะทำการอ่านข้อมูลมาแยกประเภทข้อมูลออกเพื่อทำการสร้างตารางความสัมพันธ์ (Adjacency table) ของแต่ละโนด วงจรที่เกิดเพื่อเก็บไว้ใช้คำนวณหาค่าความสัมพันธ์ระหว่างกลุ่มที่ถูกแบ่ง (Cut Cost หรือ Cut Size) และเวลาในการทำงานของแต่ละวงจรโดยแต่ละวิธีที่ได้กำหนดไว้นั้นคือ Simulated Annealing Algorithm (SA) , Genetic Algorithm (GA) และ Hybrid Algorithm (HA) เพื่อความเข้าใจได้แสดงการแยกคำตอบของการแก้ปัญหาการจัดกลุ่มวงจรของแต่ละวิธีออกเป็น text document ตามชื่อของวิธีการที่ใช้เช่น sa.txt, ga.txt, ha.txt และได้แสดงตัวอย่างการหาค่า Cut Cost และเวลาเฉลี่ยในวงจร qgr8, 3k5 และ Clique ไว้ในรูปที่ 5.3 จนถึงรูปที่5.12



รูปที่ 5.3 แสดงการหาค่าตารางความสัมพันธ์ของวงจร qgr8, 3k5 , clique

ตัวอย่าง qgr8.net ในรูปที่ 5.2 เมื่อสร้างตารางความสัมพันธ์ (Adjacency table) จะได้ดังตารางที่ 5.2 โดยที่ในแนวตั้งกำหนดให้เป็นสดมภ์ (Column) ของ node ของวงจร และในแนวนอนให้เป็นแถว (Row) ของ node ของวงจร เมื่ออ้างถึงความสัมพันธ์จะได้ว่า node2 ในแนวสดมภ์กับ node1 ในแนวแถวมีความสัมพันธ์กันโดยให้มีค่าเป็น 1 ส่วนในตารางที่มีค่าเป็น 0 หมายความว่า node นั้นๆในแนวสดมภ์กับ node อื่นๆ ในแนวแถวไม่มีความสัมพันธ์กัน

ตารางที่ 5.2 แสดงตารางค่าความสัมพันธ์ (Adjacency table) จากตัวอย่าง qgr8.net

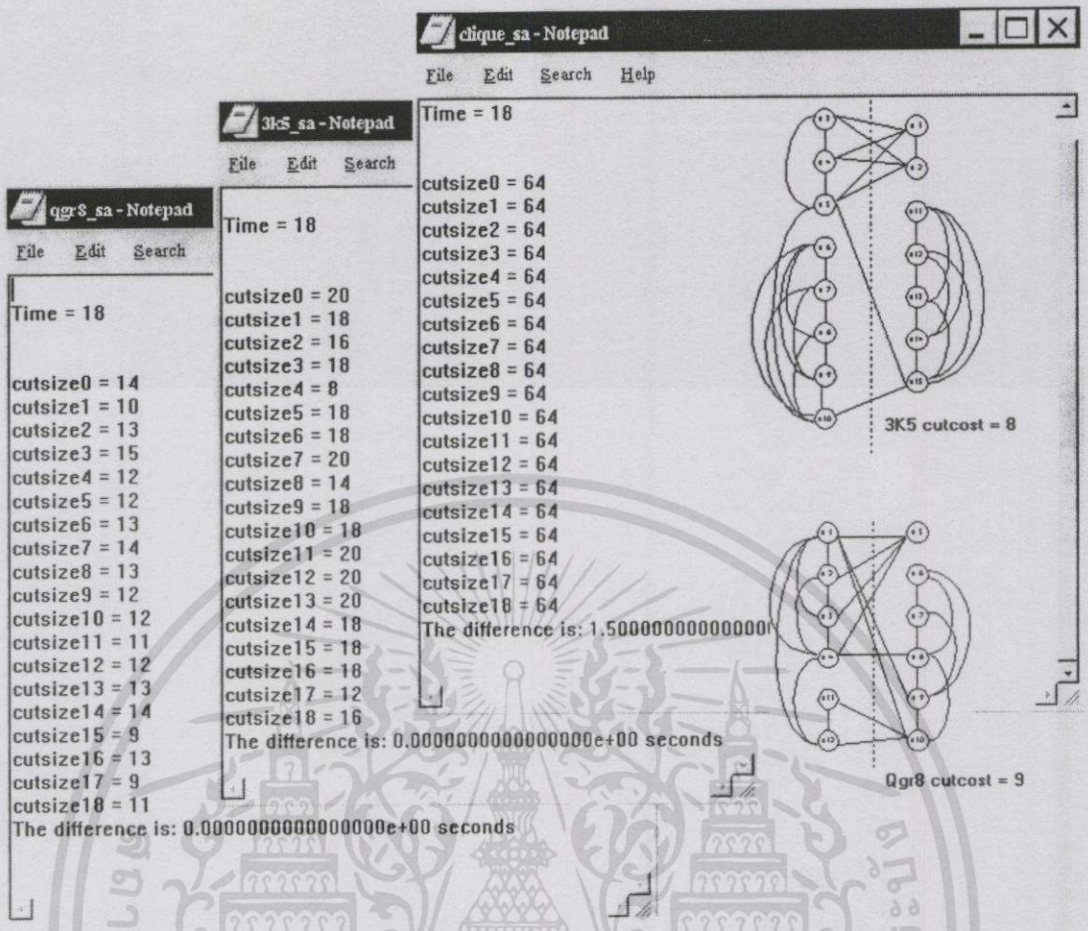
Column

Row Node	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	1	1	1	0	0	0	1	1	0	0
2	1	0	1	1	1	0	0	0	0	0	0	0
3	1	1	0	1	1	0	0	0	0	0	0	0
4	1	1	1	0	1	0	0	1	0	0	0	0
5	1	1	1	1	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1	1	1	0	0	0
7	0	0	0	0	0	1	0	1	1	0	0	0
8	0	0	0	1	0	1	1	0	1	0	0	0
9	1	0	0	0	0	1	1	1	0	1	0	0
10	1	0	0	0	0	0	0	0	1	0	1	1
11	0	0	0	0	0	0	0	0	0	1	0	1
12	0	0	0	0	0	0	0	0	0	1	1	0

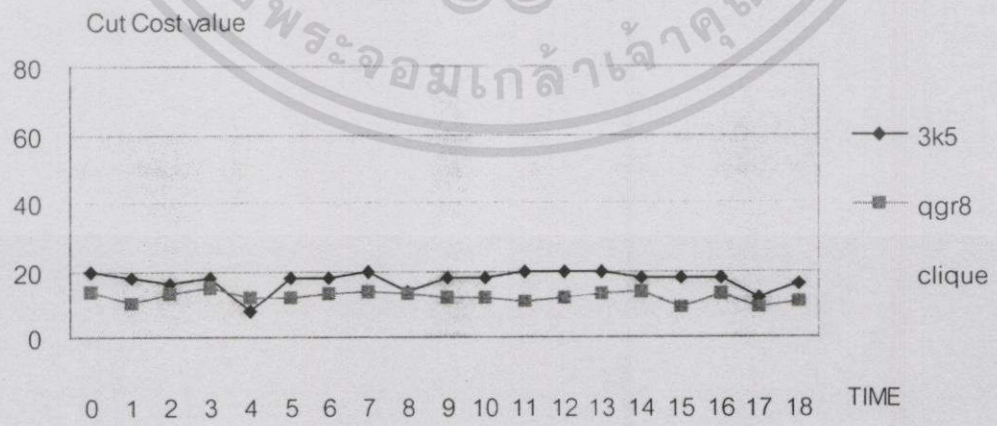
ในรูปที่ 5.4 แสดงค่าความสัมพันธ์ระหว่างกลุ่ม (Cut Cost) และเวลาเฉลี่ยในการทำงานจากวิธีการจำลองอัลแนลลิ่ง ของตัวอย่างวงจร qgr8, 3k5, clique และนำผลของค่าความสัมพันธ์ที่เกิดจากการจัดกลุ่มมาแสดงเป็นกราฟเส้นในรูปที่ 5.5

ในรูปที่ 5.6 และรูปที่ 5.7 แสดงการหาค่าความสัมพันธ์ระหว่างกลุ่ม (Cut Cost) และเวลาเฉลี่ยโดยการใช้วิธีการเจนนิติก ของวงจร qgr8, 3k5 ตั้งแต่การหาค่า Cut Cost รุ่นแรกไปถึงรุ่นสุดท้าย แล้วนำไปสร้างกราฟเปรียบเทียบค่าความสัมพันธ์ระหว่างกลุ่ม(Cut Cost) ในรูปที่ 5.8 ถึงรูปที่ 5.9 ตามลำดับ

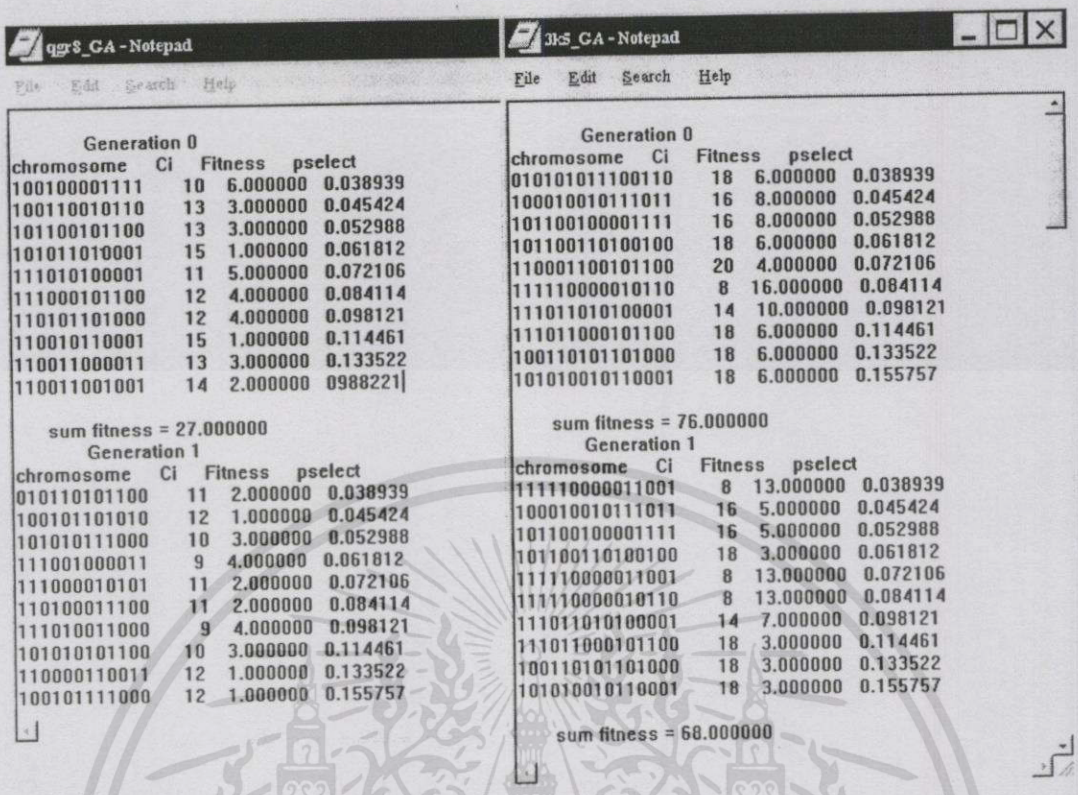
ในรูปที่ 5.10 และรูปที่ 5.11 แสดงการหาค่าความสัมพันธ์ระหว่างกลุ่มและเวลาเฉลี่ยโดยการใช้วิธีการผสมจากตัวอย่างวงจร qgr8 และ 3k5 ตามลำดับ นำผลลัพธ์ของการจัดกลุ่มวงจรมา สร้างกราฟเปรียบเทียบค่าความสัมพันธ์ระหว่างกลุ่ม (Cut Cost) ในรูปที่ 5.12



รูปที่ 5.4 แสดงค่า Cut Cost และเวลาเฉลี่ยของวงจร qgr8, 3k5, clique โดยการจำลองอัลแนลลิ่ง และผลลัพธ์การจัดกลุ่มของวงจร 3k5 และ qgr8 โดยวิธี SA



รูปที่ 5.5 แสดงค่า Cut Cost ของวงจร qgr8, 3k5 และ clique ที่นำมาพล็อตเป็นกราฟโดยใช้วิธีการจำลองอัลแนลลิ่ง



qgr8_GA - Notepad

File Edit Search Help

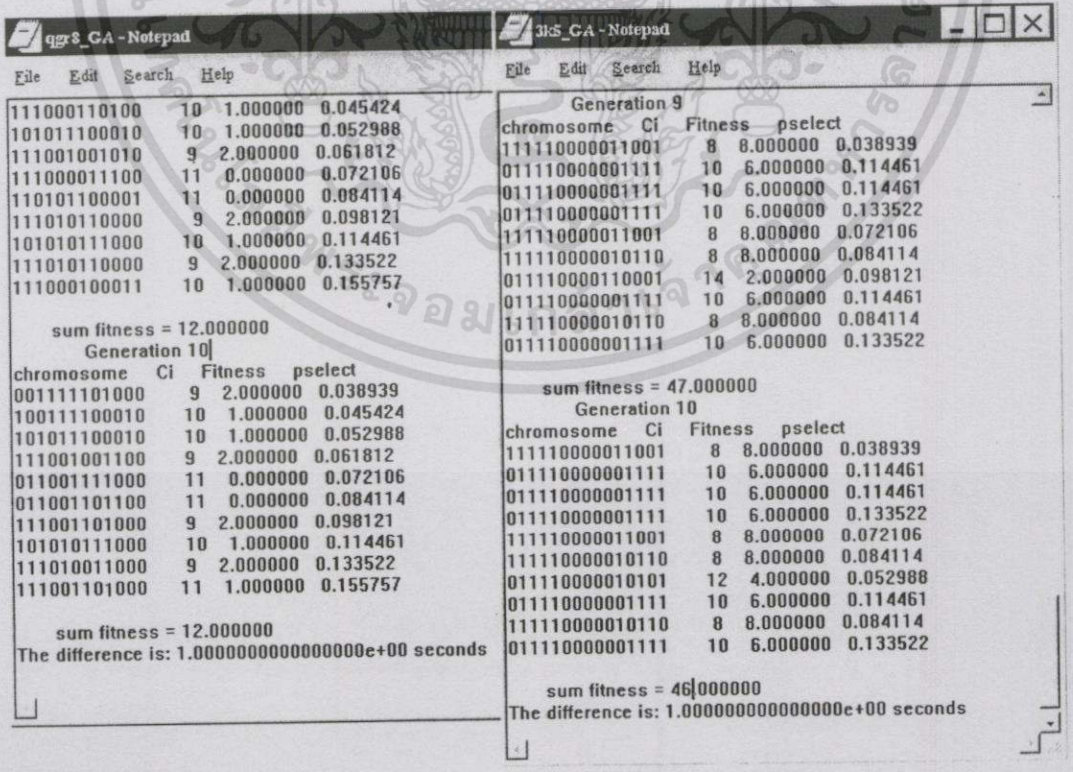
```
        Generation 0
chromosome    Ci    Fitness    pselect
100100001111     10   6.000000  0.038939
100110010110     13   3.000000  0.045424
101100101100     13   3.000000  0.052988
101011010001     15   1.000000  0.061812
111010100001     11   5.000000  0.072106
111000101100     12   4.000000  0.084114
110101101000     12   4.000000  0.098121
110010110001     15   1.000000  0.114461
110011000011     13   3.000000  0.133522
110011001001     14   2.000000  0988221

    sum fitness = 27.000000
        Generation 1
chromosome    Ci    Fitness    pselect
010110101100     11   2.000000  0.038939
100101101010     12   1.000000  0.045424
101010111000     10   3.000000  0.052988
111001000011      9   4.000000  0.061812
111000010101     11   2.000000  0.072106
110100011100     11   2.000000  0.084114
111010011000      9   4.000000  0.098121
101010101100     10   3.000000  0.114461
110000110011     12   1.000000  0.133522
100101111000     12   1.000000  0.155757
```

3k5_GA - Notepad

File Edit Search Help

```
        Generation 0
chromosome    Ci    Fitness    pselect
0101010111001l0     18   6.000000  0.038939
100010010111011     16   8.000000  0.045424
101100100001111     16   8.000000  0.052988
101100110100100     18   6.000000  0.061812
110001100101100     20   4.000000  0.072106
111110000010110      8   16.000000  0.084114
111011010100001     14   10.000000  0.098121
111011000101100     18   6.000000  0.114461
100110101101000     18   6.000000  0.133522
101010010110001     18   6.000000  0.155757

    sum fitness = 76.000000
        Generation 1
chromosome    Ci    Fitness    pselect
111110000011001      8   13.000000  0.038939
100010010111011     16   5.000000  0.045424
101100100001111     16   5.000000  0.052988
101100110100100     18   3.000000  0.061812
111110000011001      8   13.000000  0.072106
111110000010110      8   13.000000  0.084114
111011010100001     14   7.000000  0.098121
111011000101100     18   3.000000  0.114461
100110101101000     18   3.000000  0.133522
101010010110001     18   3.000000  0.155757

    sum fitness = 68.000000
```

รูปที่ 5.6 แสดงการทำงานของวิธีการเจนนิติกในการหาค่า Cut Cost วงจร ๆ qgr8, 3k5 ตั้งแต่รุ่นแรกเป็นต้นไป



qgr8_GA - Notepad

File Edit Search Help

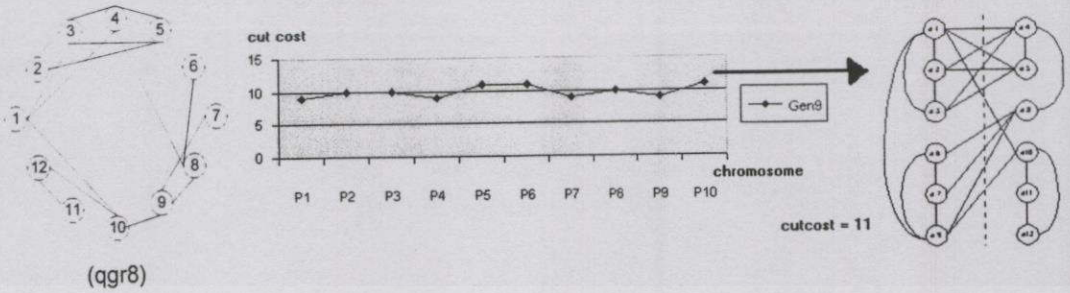
```
111000110100     10   1.000000  0.045424
101011100010     10   1.000000  0.052988
111001001010      9   2.000000  0.061812
111000011100     11   0.000000  0.072106
110101100001     11   0.000000  0.084114
111010110000      9   2.000000  0.098121
101010111000     10   1.000000  0.114461
111010110000      9   2.000000  0.133522
111000100011     10   1.000000  0.155757

    sum fitness = 12.000000
        Generation 10
chromosome    Ci    Fitness    pselect
001111101000      9   2.000000  0.038939
100111100010     10   1.000000  0.045424
101011100010     10   1.000000  0.052988
111001001100      9   2.000000  0.061812
011001111000     11   0.000000  0.072106
011001101100     11   0.000000  0.084114
111001101000      9   2.000000  0.098121
101010111000     10   1.000000  0.114461
111010011000      9   2.000000  0.133522
111001101000     11   1.000000  0.155757

    sum fitness = 12.000000
The difference is: 1.0000000000000000e+00 seconds
```

3k5_GA - Notepad

File Edit Search Help

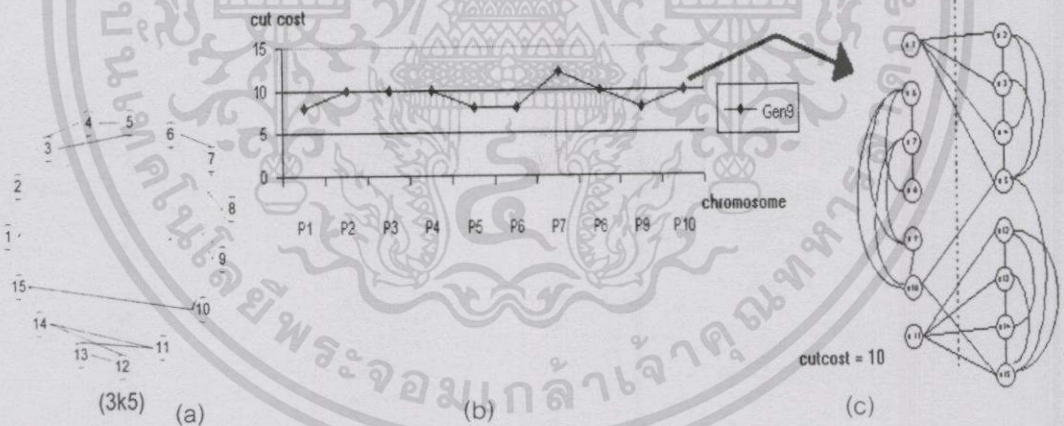
```
        Generation 9
chromosome    Ci    Fitness    pselect
111110000011001      8   8.000000  0.038939
011110000001111     10   6.000000  0.114461
011110000001111     10   6.000000  0.114461
011110000001111     10   6.000000  0.133522
111110000011001      8   8.000000  0.072106
111110000010110      8   8.000000  0.084114
011110000110001     14   2.000000  0.098121
011110000001111     10   6.000000  0.114461
111110000010110      8   8.000000  0.084114
011110000001111     10   6.000000  0.133522

    sum fitness = 47.000000
        Generation 10
chromosome    Ci    Fitness    pselect
111110000011001      8   8.000000  0.038939
011110000001111     10   6.000000  0.114461
011110000001111     10   6.000000  0.114461
011110000001111     10   6.000000  0.133522
111110000011001      8   8.000000  0.072106
111110000010110      8   8.000000  0.084114
011110000010101     12   4.000000  0.052988
011110000001111     10   6.000000  0.114461
111110000010110      8   8.000000  0.084114
011110000001111     10   6.000000  0.133522

    sum fitness = 46.000000
The difference is: 1.0000000000000000e+00 seconds
```

รูปที่ 5.7 แสดงการทำงานของวิธีการเจนนิติก ในการหาค่า Cut Cost ของวงจร qgr8, 3k5ในรุ่นสุดท้ายและแสดงเวลาที่ใช้ในการทำงานทั้งหมด



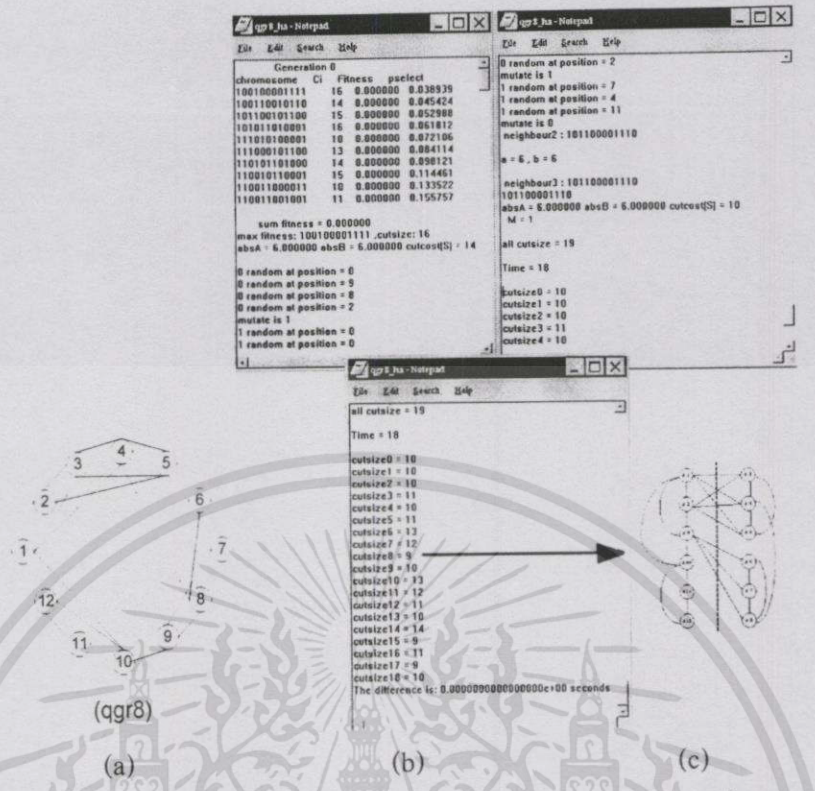
รูปที่ 5.8 แสดงกราฟของค่า Cut Cost ของวงจร qgr8 ที่เกิดขึ้นในรุ่นสุดท้ายของวิธีการเจนนิติก และผลลัพธ์การจัดกลุ่มโดยวิธี GA

จากรูปที่ 5.8 (a) และรูปที่ 5.9 (a) แสดงวงจรที่ยังไม่ได้จัดกลุ่มและจากรูปที่ 5.8 (b) กับรูปที่ 5.9 (b) แสดงกราฟของค่า Cut Cost ในรุ่นสุดท้ายและนำผลการจัดกลุ่มที่เกิดจากโครโมโซมตัวที่10 มาแสดงผลการจัดกลุ่มวงจรดังรูปที่ 5.8 (c) และรูปที่ 5.9 (c) โดยวิธีเจนนิติก

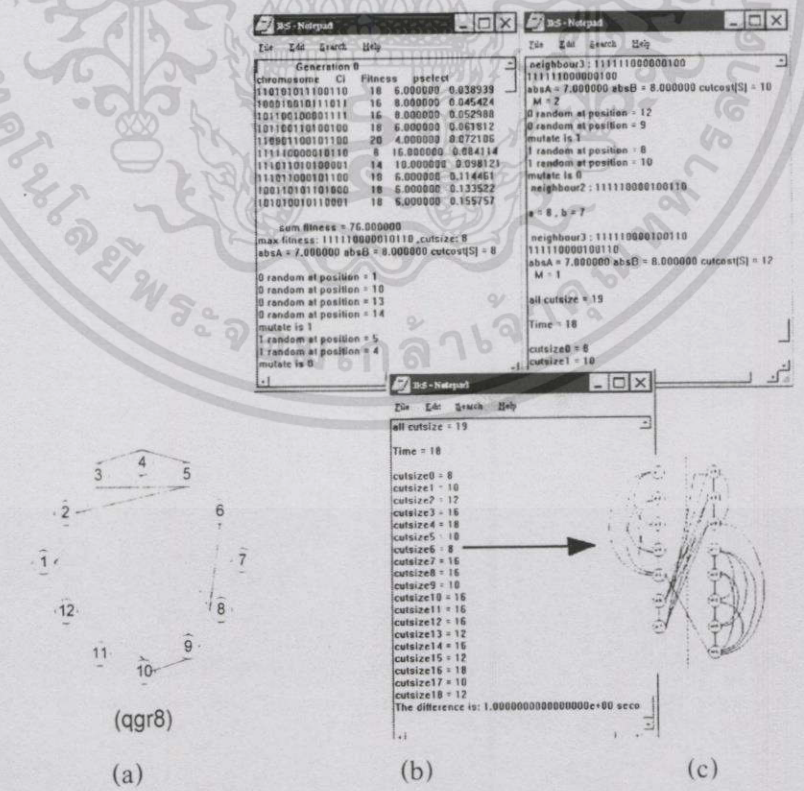


รูปที่ 5.9 แสดงกราฟของค่า Cut Cost ของวงจร 3k5 ที่เกิดขึ้นในรุ่นสุดท้ายของวิธีการเจนนิติก และผลลัพธ์การจัดกลุ่มโดยวิธี GA

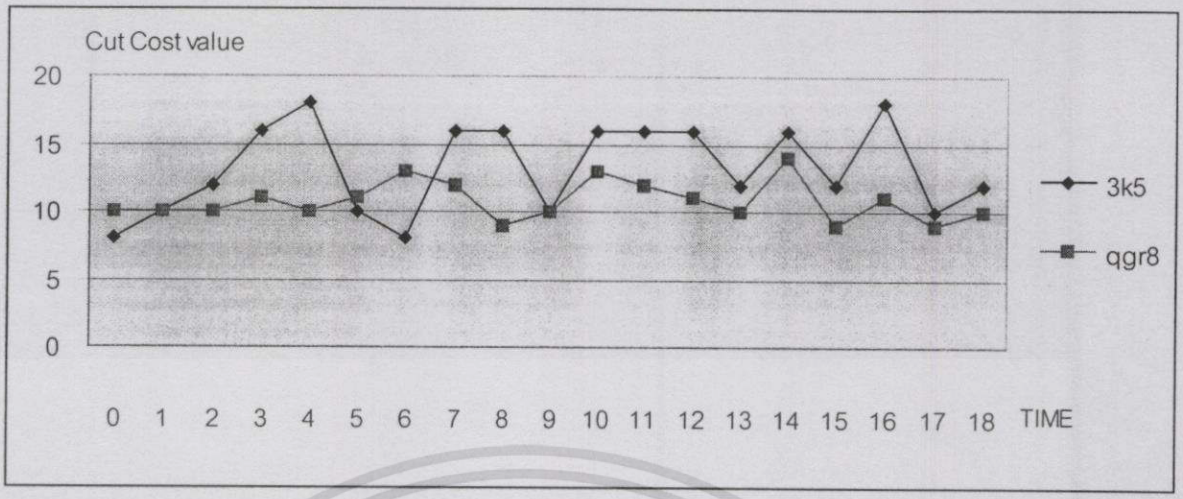
จากรูปที่ 5.10 (a) และรูปที่ 5.11 (a) แสดงวงจรที่ยังไม่ได้จัดกลุ่มและจากรูปที่ 5.10 (b) กับรูปที่ 5.11 (b) แสดงกราฟของค่า Cut Cost ที่ดีที่สุดที่เกิดจากวิธีผสมและนำผลการจัดกลุ่มที่เกิดมาแสดงผลการจัดกลุ่มวงจรดังรูปที่ 5.10 (c) และรูปที่ 5.11 (c)



รูปที่ 5.10 แสดงการทำงานของวิธีการผสม ในการหาค่า Cut Cost และเวลาเฉลี่ย ของวงจร qgr8 และผลลัพธ์การจัดกลุ่มโดยวิธี HA

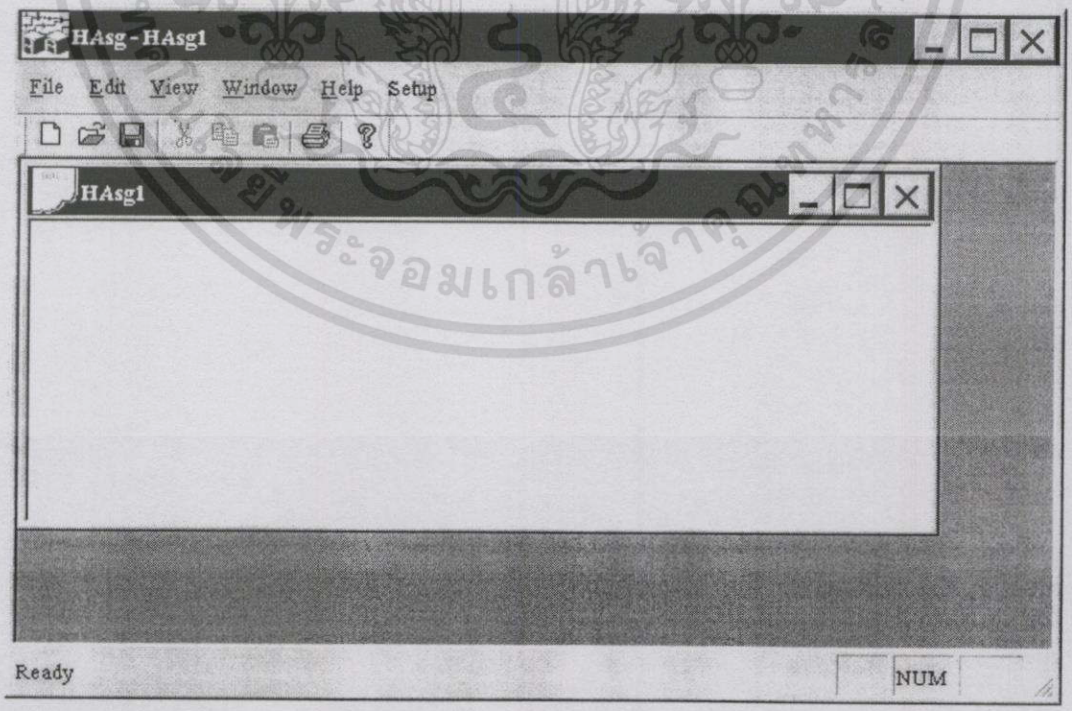


รูปที่ 5.11 แสดงการทำงานของวิธีการผสม ในการหาค่า Cut Cost และเวลาเฉลี่ย ของวงจร 3k5 และผลลัพธ์การจัดกลุ่มโดยวิธี HA

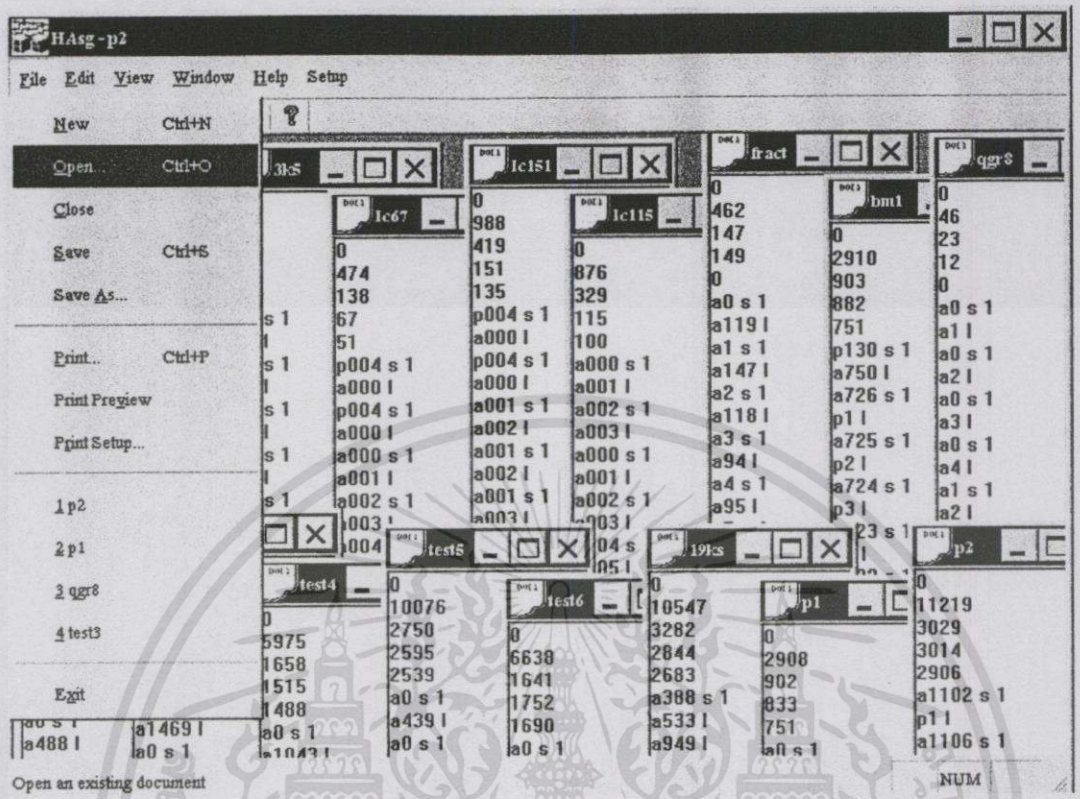


รูปที่ 5.12 แสดงกราฟค่า Cut Cost ของวงจร qgr8 และ 3k5 ที่ได้จากวิธีการผสม

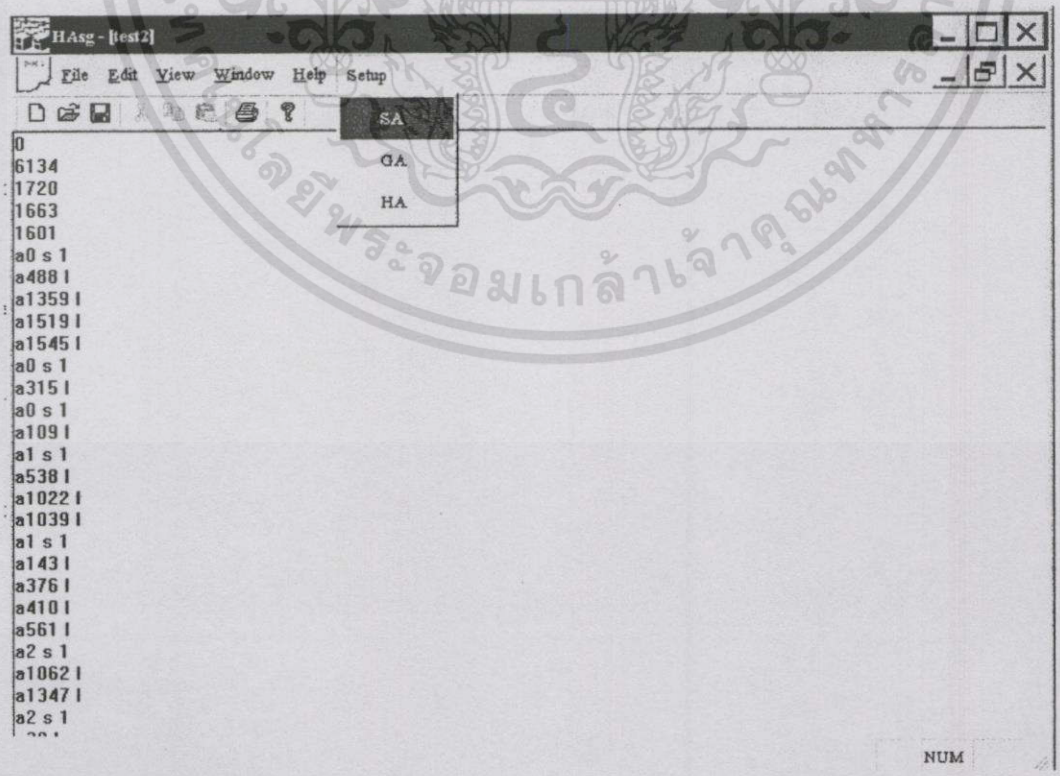
ในรูปที่ 5.13 แสดงโปรแกรมที่ใช้แก้ปัญหาการจัดกลุ่มวงจรเริ่มจากการเปิด netlist file แล้วเลือกวิธีการจัดกลุ่มจากเมนูตามต้องการดังรูปที่ 5.15 ผลลัพธ์ถูกแสดงอยู่ในรูปกราฟฟิกเป็นรูปวงกลม (ตัวแทนของวงจร) มี 2 สี สีแดงกับสีน้ำเงิน แต่ละสีแทนประเภทกลุ่มที่ถูกแบ่ง นั้นคือ 2 สี หมายถึง 2 กลุ่ม ดังแสดงในรูปที่ 15.16 15.17 และ15.18 คำตอบค่า Minimum Cut cost และ CPU Time แสดงอยู่ในส่วนของสเตตัสบาร์



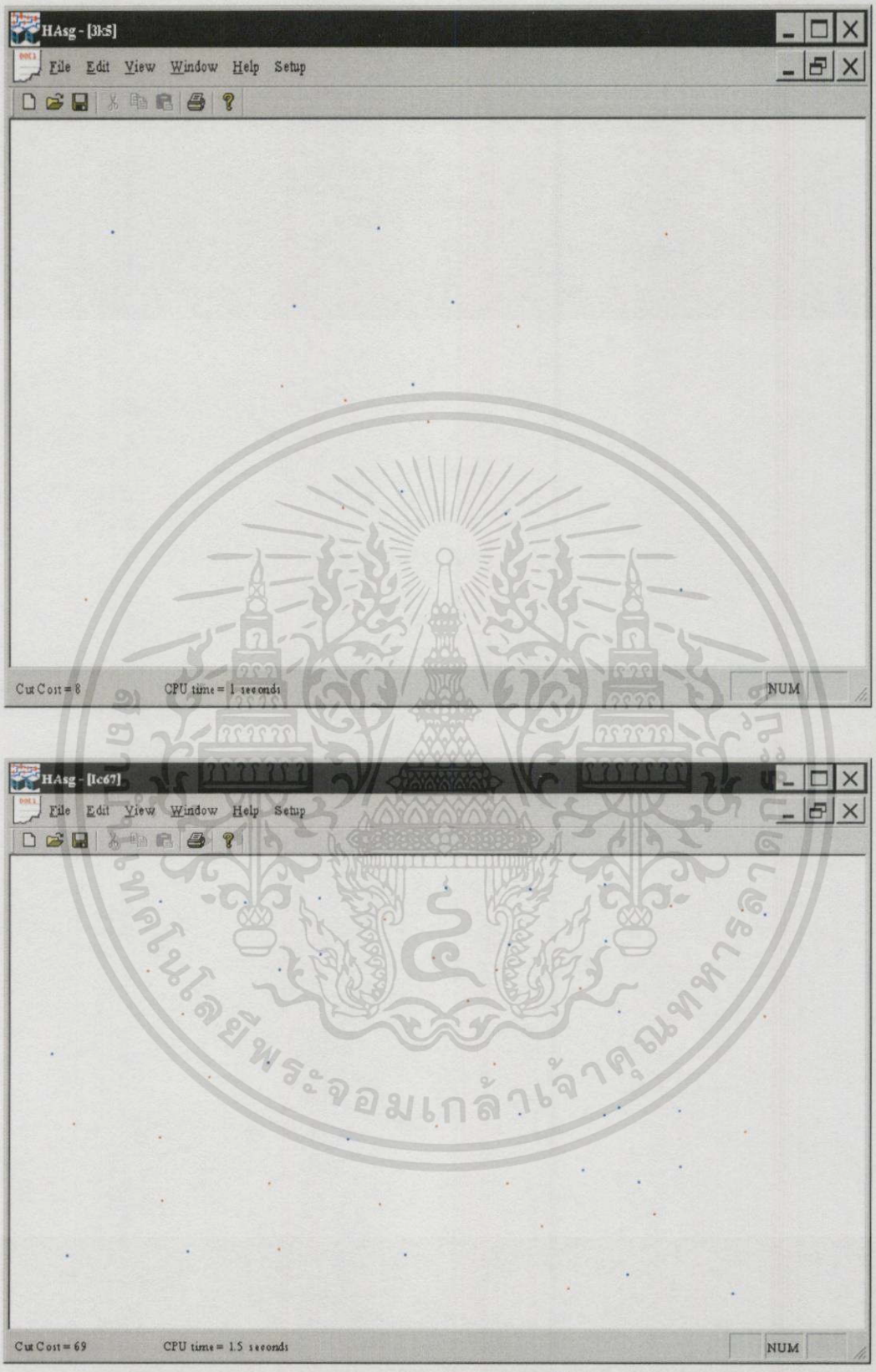
รูปที่ 5.13 แสดงโปรแกรมการจัดกลุ่ม



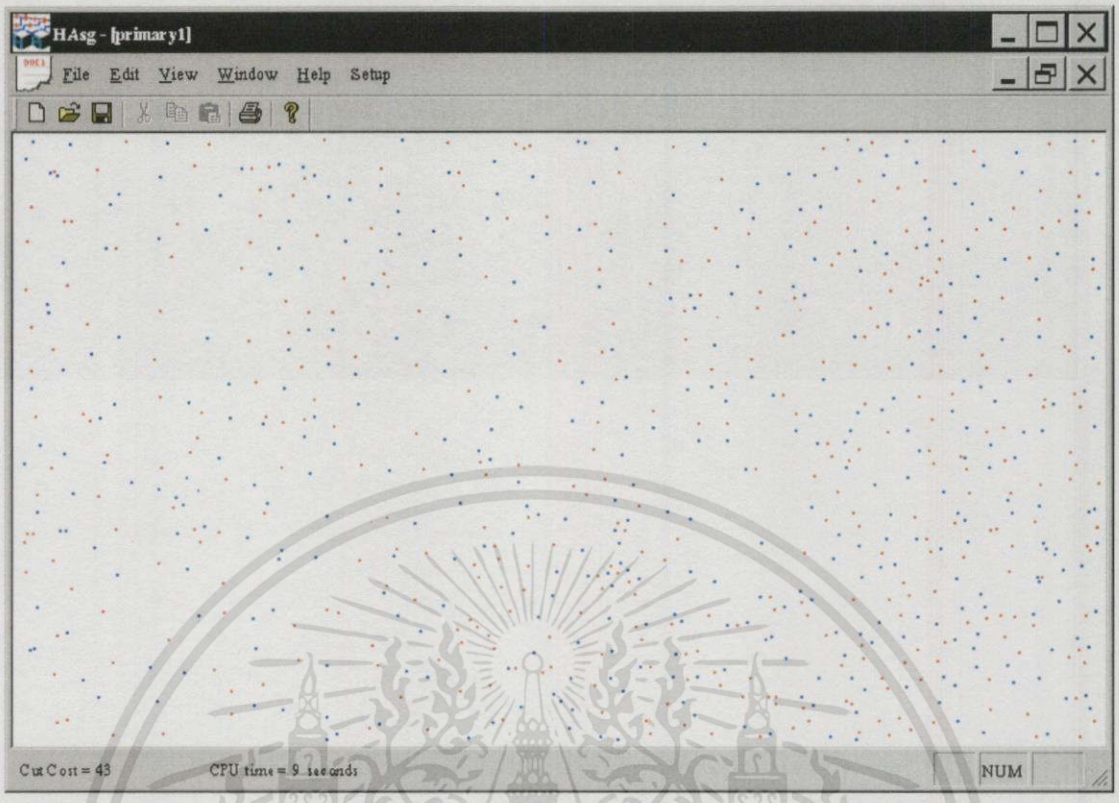
รูปที่ 5.14 แสดงเปิด netlist.net ของวงจรตัวอย่างที่นำมาทดสอบโปรแกรมการจัดกลุ่ม



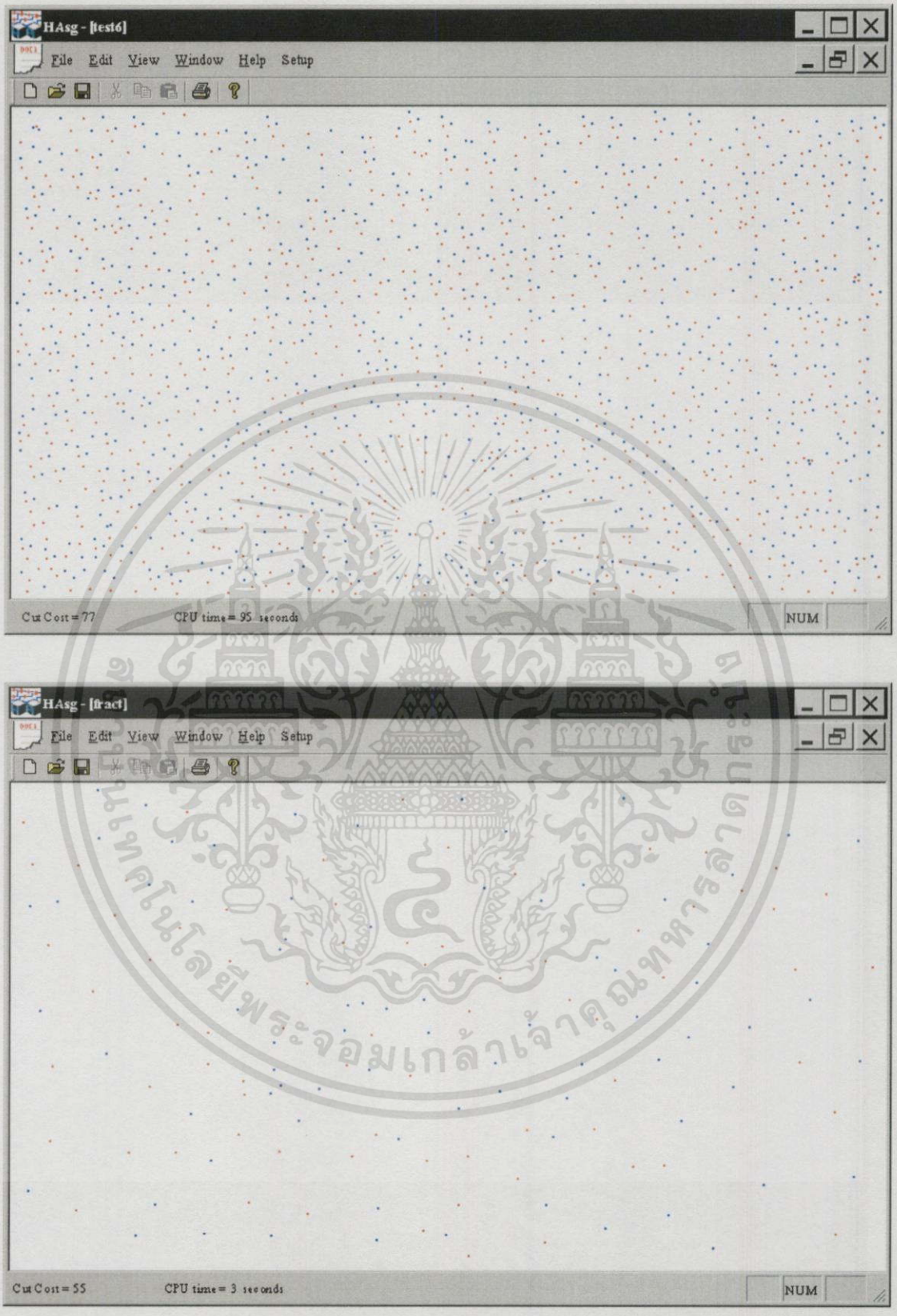
รูปที่ 5.15 แสดงตัวอย่างวงจร test2



รูปที่ 5.16 แสดง ผลลัพธ์ของวงจร 3k5 และ IC67 ที่นำมาจัดกลุ่มโดยวิธี SA



รูปที่ 5.17 แสดง ผลลัพธ์ของวงจร primary1และ test2 ที่นำมาจัดกลุ่มโดยวิธี GA



รูปที่ 5.18 แสดง ผลลัพธ์ของวงจร test6 และ fractที่นำมาจัดกลุ่มโดยวิธี HA

5.3 เปรียบเทียบผลการทดสอบการจัดกลุ่มวงจรที่ได้จากทั้งสามวิธี

การกำหนดค่าช่วงของพารามิเตอร์ต่างๆของทั้งสามวิธีมีรายละเอียดดังตารางที่ 5.3 ผลลัพธ์ของค่าความสัมพันธ์ของแต่ละวงจรที่ใช้ในการทดลองจัดกลุ่มวงจรในแต่ละวิธีได้แสดงในตารางที่ 5.4

ตารางที่ 5.3 แสดงค่าช่วงของพารามิเตอร์ใน GA , SA และ HA ที่ใช้ในงานวิจัย

Parameter of GA By Maxtime=20		**Parameter of SA by Maxtime=20**			**parameter of HA by Maxtime=20**			
P	**Gen**	**T_0**	**α**	**M**	**T_0**	**α**	**P**	**M**
10	5,10	1000	.9	5,10,15,18	1000	.9	10	5,10,15,18

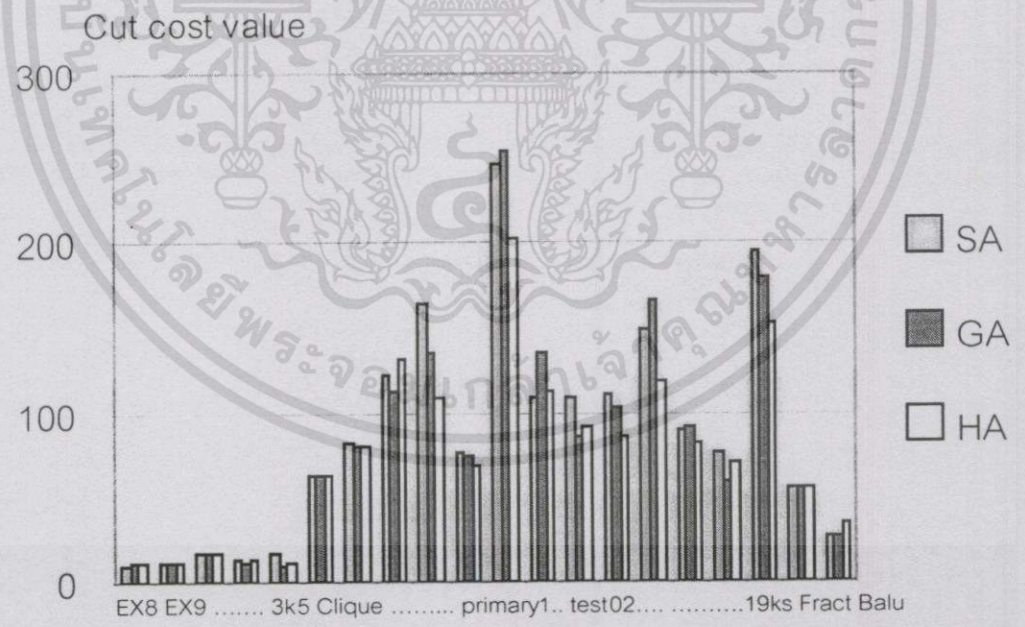
ตารางที่ 5.4 แสดงผลลัพธ์ค่า Avg Cut Cost และ Minimum Cut Cost แบบ Two Way Partitioning ทั้งสามวิธี

Circuit	**Avg Cut Cost**			**Min Cut Cost**		
	SA	**GA**	**HA**	**SA**	**GA**	**HA**
EX8	10.21	11.3	10.63	9	9	9
EX9	12.0	11.8	11.5	10	10	9
EX10	16.9	17	16.6	16	16	15
Qgr8	12.8	12	12.8	10	9	10
3k5	16.8	10.4	12.4	8	8	8
Clique	64	64	64	64	64	64
IC67	82.5	79.8	79.5	69	83	75
IC116	122	113.6	131	79	81	79
IC151	165.1	135.4	108	100	112	100
Primary1	76	75.2	69.2	54	39	54
Primary2	246	254.5	202.4	156	123	150
test02	109	136	113	105	92	94
test03	108	85.2	91	59	58	60
test04	111	104	86	82	52	56
test05	150	167	117.6	107	42	75
test06	89.9	91.1	83	73	66	62

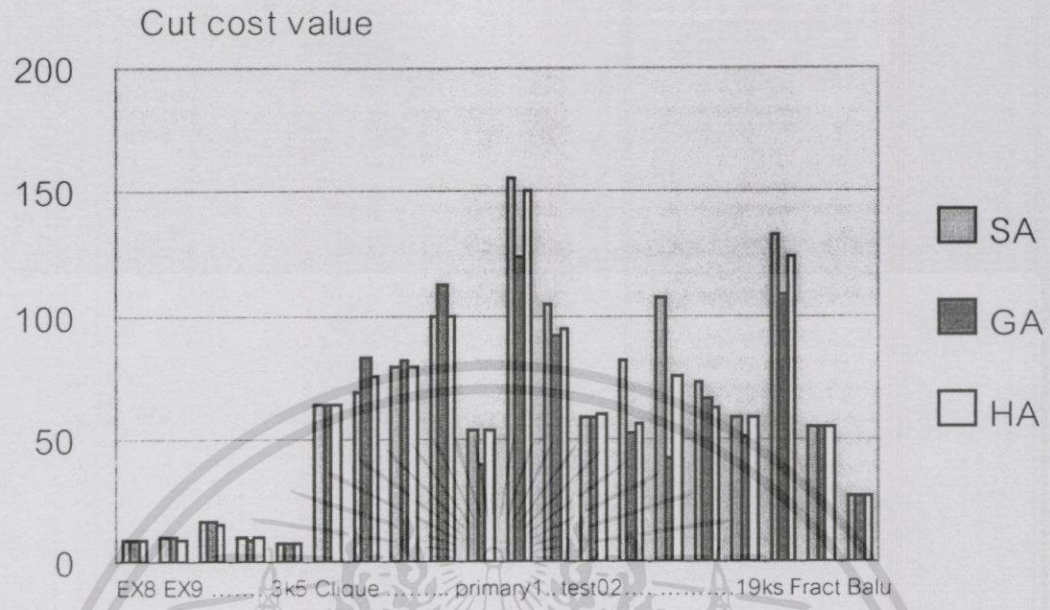
ตารางที่ 5.4 (ต่อ) แสดงผลลัพธ์ค่า Avg Cut Cost และ Minimum Cut Cost แบบ Two Way Partitioning ทั้งสามวิธี

Circuit	Average Cut Cost			Minimum Cut Cost		
	SA	GA	HA	SA	GA	HA
bm1	77	58.9	71	58	51	58
19ks	194	179	153.2	132	108	124
Fract	55	55	55	55	55	55
Balu	27	27	35.1	27	27	27

เมื่อนำข้อมูลที่ได้มาพล็อตกราฟเปรียบเทียบ ปรากฏว่าค่าเฉลี่ยของค่า Cut Cost และ minimum Cut Cost ที่ได้จากตัวอย่างการทดสอบนั้นแสดงให้เห็นว่า วิธีการผสม ให้ผลลัพธ์ดีกว่า หรือดีพอๆ กับวิธีการจำลองอัลแนลลิ่งและวิธีเจนนิติกโดยส่วนใหญ่ของคำตอบ ซึ่งแสดงในรูปที่ 5.19 และรูปที่ 5.20



รูปที่ 5.19 แสดงกราฟของค่าเฉลี่ยความสัมพันธ์ (Avg Cut Cost) ทั้งสามวิธีของทุกวงจรตัวอย่าง



รูปที่ 5.20 แสดงกราฟของค่าต่ำสุดของความสัมพันธ์ (Minimum Cut Cost) ทั้งสามวิธีของทุกวงจรตัวอย่าง

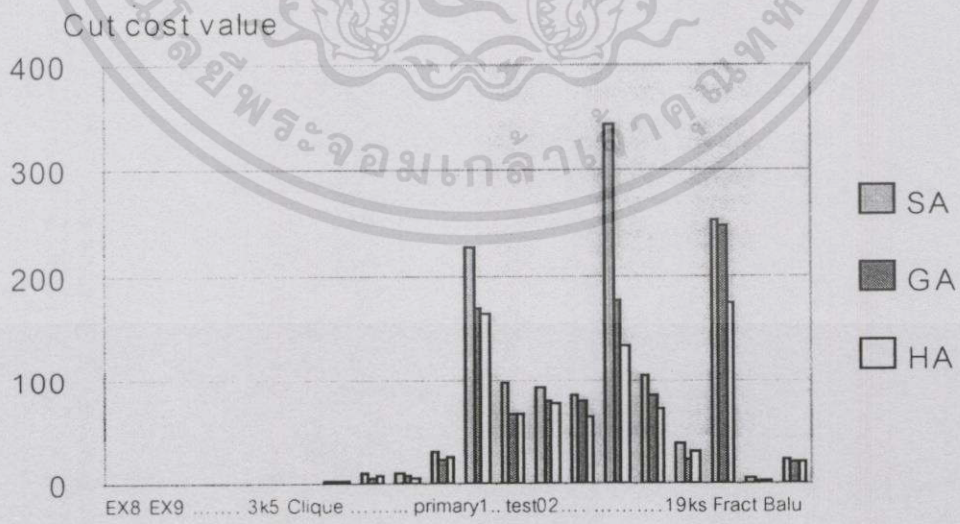
ตารางที่ 5.5 แสดงผลลัพธ์ค่าเวลาเฉลี่ย (Avg CPU) ของทั้งสามวิธี

Circuit	avg CPU(sec)		
	SA	GA	HA
EX8	N/a	N/a	N/a
EX9	N/a	N/a	N/a
EX10	N/a	N/a	N/a
Qgr8	N/a	N/a	N/a
3k5	1	1	0.96
Clique	0.9	0.8	0.8
IC67	2.75	2.4	2.5
IC116	10.3	5.8	7.4
IC151	11.2	7.4	5.8
primary1	31	24	25

ตารางที่ 5.5 (ต่อ) แสดงผลลัพธ์ค่าเวลาเฉลี่ย (Avg CPU) ของทั้งสามวิธี

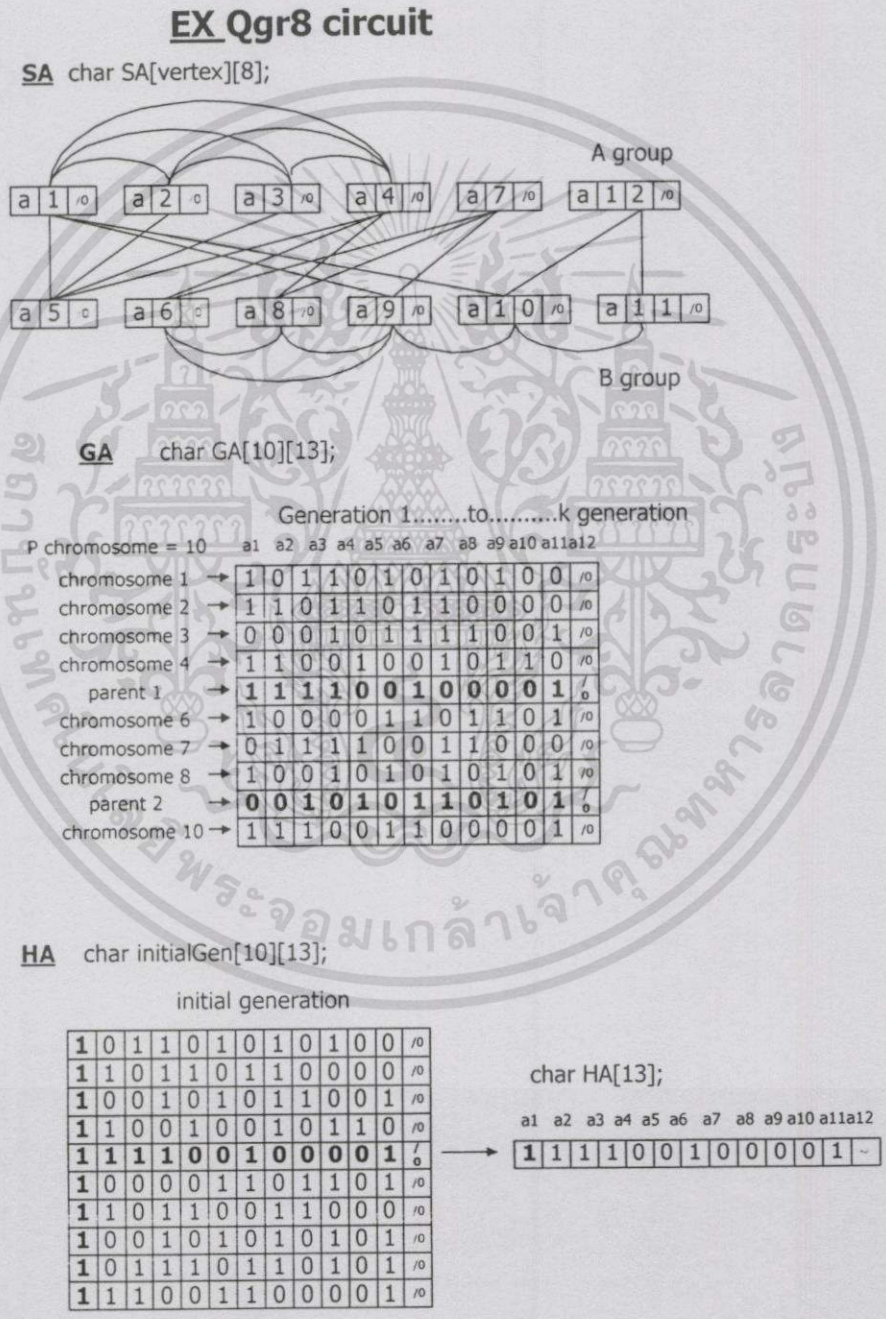
Circuit	avg CPU(sec)		
	SA	GA	HA
primary2	226	167	162
test02	98	65	67
test03	91	79	77
test04	85	79	64
test05	344	176	133
test06	105	84	71
bm1	39	24	31
19ks	251	247	174
fract	5	3	2.5
Balu	23	21	21

จากตารางที่ 5.5 เมื่อนำคำตอบมาพล็อตกราฟเส้นเปรียบเทียบเวลาที่ใช้ในการประมวลผลของทั้ง 3 วิธี ปรากฏว่าโดยเฉลี่ยแล้ววิธีการผสมจะใช้เวลาน้อยกว่าวิธีการจำลองอัลแนลลิ่งและวิธีเจนนิติก กราฟเปรียบเทียบค่าเฉลี่ยเวลาที่ใช้ (Avg CPU) แสดงในรูปที่ 5.21



รูปที่ 5.21 แสดงกราฟของค่าเฉลี่ยเวลาที่ใช้ (Avg CPU) ทั้งสามวิธีของทุกวงจรตัวอย่าง

จากการแก้ปัญหาการจัดกลุ่มวงจรด้วยวิธีจำลองอัลแนลลิ่งนั้น การหาคำตอบที่ดีที่สุดต้องอาศัยเวลามากพอสมควร เมื่อเทียบกับวิธีการเจนนิติก แต่การใช้วิธีการเจนนิติกนั้นต้องอาศัยเนื้อที่ในการประมวลผลมากกว่าหลายเท่าเมื่อเทียบกับวิธีจำลองอัลแนลลิ่งพิจราณาได้จากรูปที่ 5.22 เป็นการแสดงการเปรียบเทียบหน่วยความจำที่ใช้ในการประมวลของทั้งสามวิธีจากวงจรตัวอย่าง qgr8.net วิธีการผสมที่นำเสนอจะช่วยลดเวลาและลดปัญหาเรื่องหน่วยความจำในการประมวลผลได้ในระดับหนึ่ง



รูปที่ 5.22 แสดงการปรียบเทียบการจัดการหน่วยคำจำในการประมวลผลของวิธีจำลองอัลแนลลิ่ง วิธีเจนนิติก และวิธีผสม

5.4 การจัดกลุ่มวงจรที่ได้จากวิธีการอื่นๆ

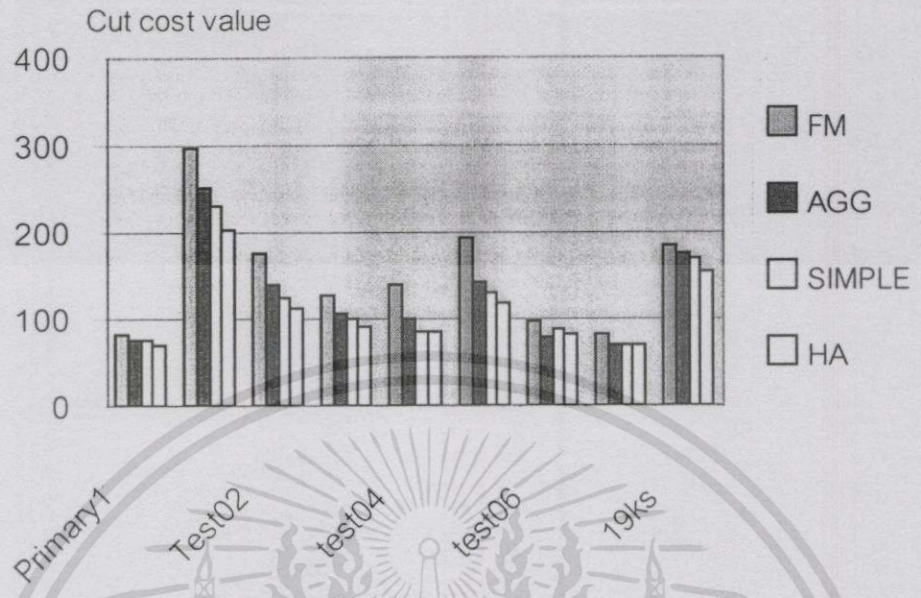
จากการค้นหาข้อมูลงานวิจัยทางด้านการจัดกลุ่มวงจรดังที่ได้กล่าวมานี้ พบว่ามีนักวิจัยหลายท่านได้ให้ความสนใจค้นคว้าหาวิธีการหาคำตอบที่เหมาะสมด้วยกันอยู่หลายวิธี และได้นำมาเปรียบเทียบกับวิธีผสมดังกล่าว แสดงผลอยู่ในรูปของกราฟแท่งเปรียบเทียบดังต่อไปนี้

ตารางที่ 5.6 แสดงการเปรียบเทียบผลลัพธ์ในการแก้ปัญหาการจัดกลุ่มวงจร โดยอาศัยวิธีการ FM. AGG และ SIMPLE Eigenvector clustering Algorithm. โดย 100 two-phase FM cuts แบบ bisection [35].

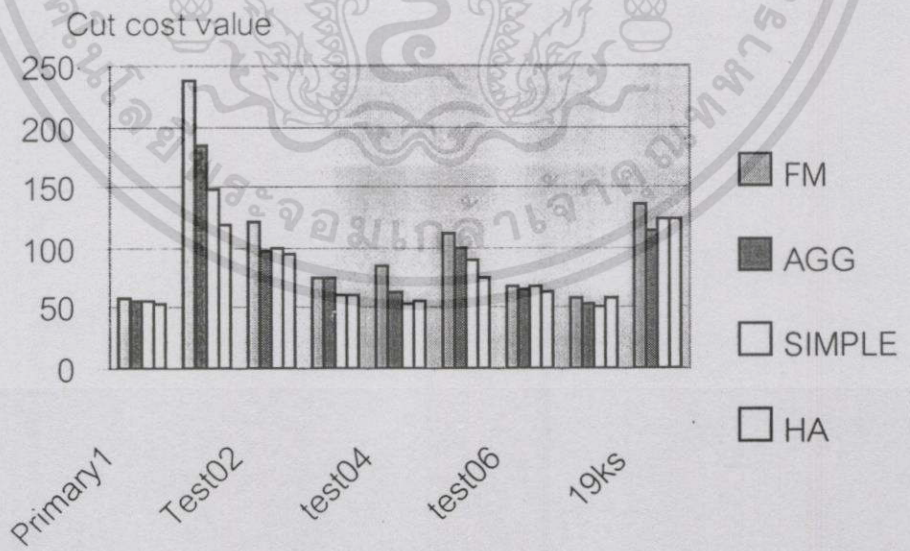
Circuit	Avg Cut Cost			Min Cut Cost		
	FM Algorithm	AGG Algorithm	SIMPLE Algorithm	FM Algorithm	AGG Algorithm	SIMPLE Algorithm
Primary1	83	75	80	59	56	55
Primary2	296	253	231	238	184	147
test02	177	140	124	122	97	99
test03	126	105	99	76	75	61
test04	140	99	84	86	64	53
test05	194	141	129	112	99	90
test06	96	78	88	69	65	69
bm1	83	70	71	58	54	52
19ks	184	177	169	137	115	123

จากตารางที่ 5.6 แสดงการเปรียบเทียบผลลัพธ์ในการแก้ปัญหาการจัดกลุ่มวงจร โดยอาศัยวิธีการ FM(Fiduccia-Mattheyses). AGG(เป็นวิธีการที่ได้จาก Geometric Embeddings for Faster and Better Multi-Way Netlist Partitioning ของ C.J. Alpert และ A.B. Hahng ในบทความ ACM/IEEE Design Automation Conf. ปี1993 และวิธี SIMPLE Eigenvector clustering Algorithm) [35]

ในรูปที่ 5.23 และรูปที่ 5.24 ได้นำผลการแก้ปัญหาการจัดกลุ่มวงจรจากตารางที่ 5.6 ทั้งค่า avg cut cost และค่า minimum cut cost มาสร้างกราฟเปรียบเทียบกับวิธีการผสมดังกล่าว วิธีผสมจัดว่าเป็นวิธีที่ยอมรับคำตอบได้ดี



รูปที่ 5.23 กราฟเปรียบเทียบค่าเฉลี่ยของ Cut Cost ที่ได้จากวิธี FM, AGG, SIMPLE และ HA



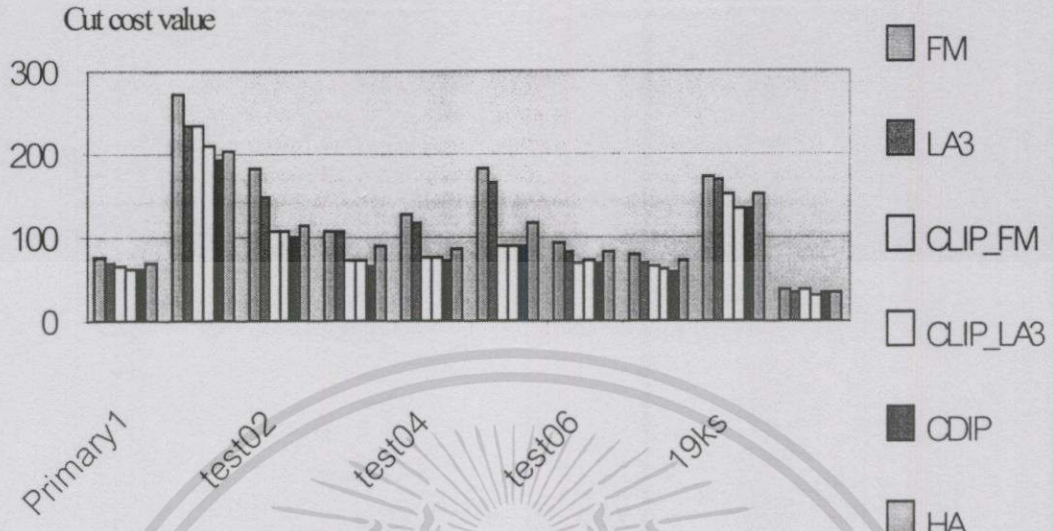
รูปที่ 5.24 กราฟเปรียบเทียบค่าต่ำสุดของ Cut Cost ที่ได้จากวิธี FM, AGG, SIMPLE และ HA.

ตารางที่ 5.7 การเปรียบเทียบผลลัพธ์ในการแก้ปัญหาการจัดกลุ่มวงจรโดยอาศัยวิธีการ FM, LA3, CLIP(FM.LA3) และ CDIP[36].

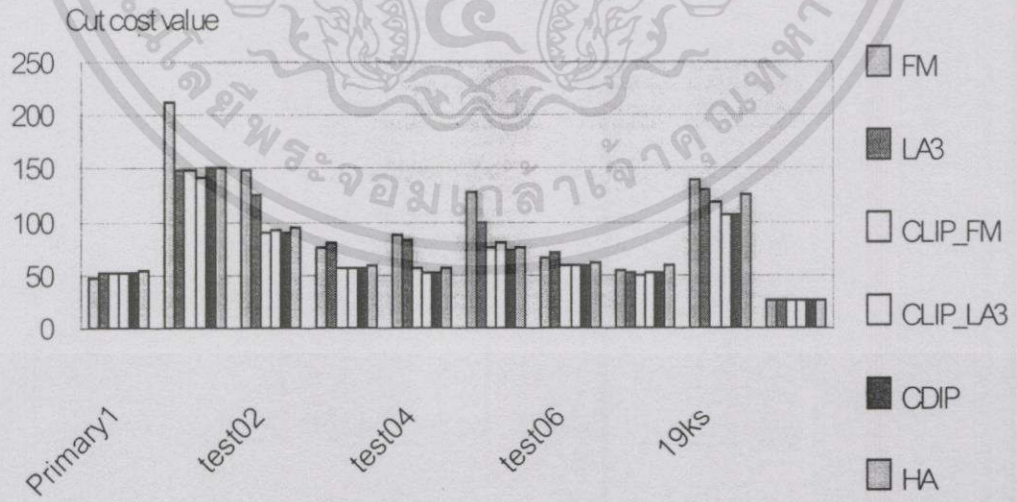
Circuit	Avg Cut Cost of 20 runs					Min Cut Cost of 20 runs				
	FM Alg	LA3 Alg	CLIP-FM	CLIP-LA3	CDIP	FM Alg	LA3 Alg	CLIP-FM	CLIP-LA3	CDIP
Primary1	74.9	68.5	65.8	61.8	61.5	47	52	52	52	52
Primary2	273.9	233.4	233.2	208.8	193.2	212	149	149	142	152
test02	182.1	148.1	105.2	106.8	101.0	149	126	89	92	90
test03	106.8	106.2	72.3	71.6	66.8	75	80	57	57	57
test04	129.3	117.2	77.0	72.9	71.7	87	82	56	51	52
test05	183.6	165.0	89.0	89	90.7	127	99	75	80	74
test06	94.2	84.4	70.1	70.1	71.8	67	70	60	60	60
bm1	79.5	67.5	65.0	60.4	59.8	54	53	49	52	52
19ks	171.7	169.0	150.3	107	105	140	130	119	136.1	134.6
Balu	38.1	36.1	38.9	32.5	33.9	27	27	27	27	27

จากตารางที่ 5.7 แสดงการเปรียบเทียบผลลัพธ์ในการแก้ปัญหาการจัดกลุ่มวงจรโดยอาศัยวิธีการ Fiduccia-Mattheyses(FM) ได้จากผลงานวิจัยของ C.M Fiduccia and R.M Mattheyses เป็นวิธีการที่อาศัยหลักการ linear-time heuristic for improving network partitioning ซึ่งถูกเสนอลงใน ACM/IEEE Design Automation ในปี 1982 , LA3 หรือ Krishnamurthy's Look-Ahead วิธีการนี้พัฒนามาจากวิธีการของ min-cut เพื่อ partitioning VLSI networks โดย B. Krishnamurthy และถูกเสนอผลงานในบทความของ IEEE Trans. On Comput [37] ในปี 1984 วิธี CLIP(Cluster-oriented Iterative-improvement Partitioner) วิธีการนี้สามารถนำไปประยุกต์ใช้หาคำตอบรวมกับวิธีอื่นๆได้ เช่น FM, LA3, PROP[37] (Recepient of the best-paper award) ทั้ง CLIP และ CDIP (Cluster-Detecting Iterative-improvement Partitioner) สามารถหาศึกษารายละเอียดเพิ่มเติมได้จาก [36]

ในรูปที่ 5.25 และรูปที่ 5.26 ได้นำผลการแก้ปัญหาการจัดกลุ่มวงจรจากตารางที่ 5.7 ทั้งค่า avg cut cost และค่า minimum cut cost มาสร้างกราฟเปรียบเทียบกับวิธีการผสมที่นำเสนอ ผลปรากฏว่า HA จัดได้ว่าการแก้ปัญหาโดยวิธีนี้อยู่ในเกณฑ์ที่สามารถยอมรับได้ ถึงแม้จะไม่ดีกว่าวิธีอื่นที่ให้เปรียบเทียบมากนัก



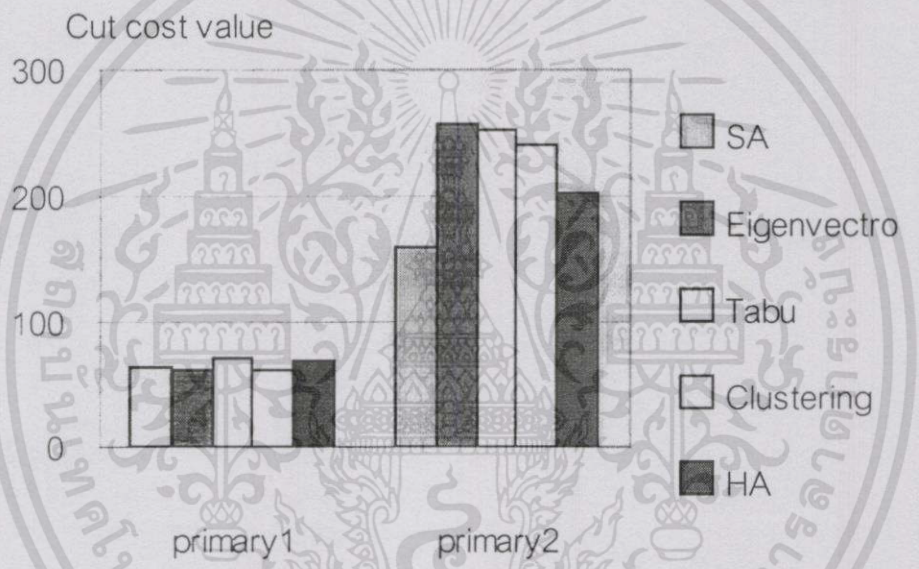
รูปที่ 5.25 กราฟเปรียบเทียบค่าเฉลี่ยของ Cut Cost ที่ได้จากวิธี LA3, CLIP-FM, CLIP-LA3, CDIP และ HA



รูปที่ 5.26 กราฟเปรียบเทียบค่าต่ำสุดของ Cut Cost ที่ได้จากวิธี LA3, CLIP-FM, CLIP-LA3, CDIP และ HA

ตารางที่ 5.8 แสดงการเปรียบเทียบผลลัพธ์ในการแก้ปัญหาการจัดกลุ่มวงจร โดยอาศัยวิธี SA, Eigenvectro, Tabu, Clustering [38].

Circuit	Avg Cut Cost			
	SA	Eigenvectro	Tabu	Clustering
Primary1	65	63	72	61
Primary2	160	257	252	240



รูปที่ 5.27 กราฟเปรียบเทียบค่าเฉลี่ยของ Cut Cost ที่ได้จากวิธี SA, Eigenvectro, Tabu, Clustering และ HA

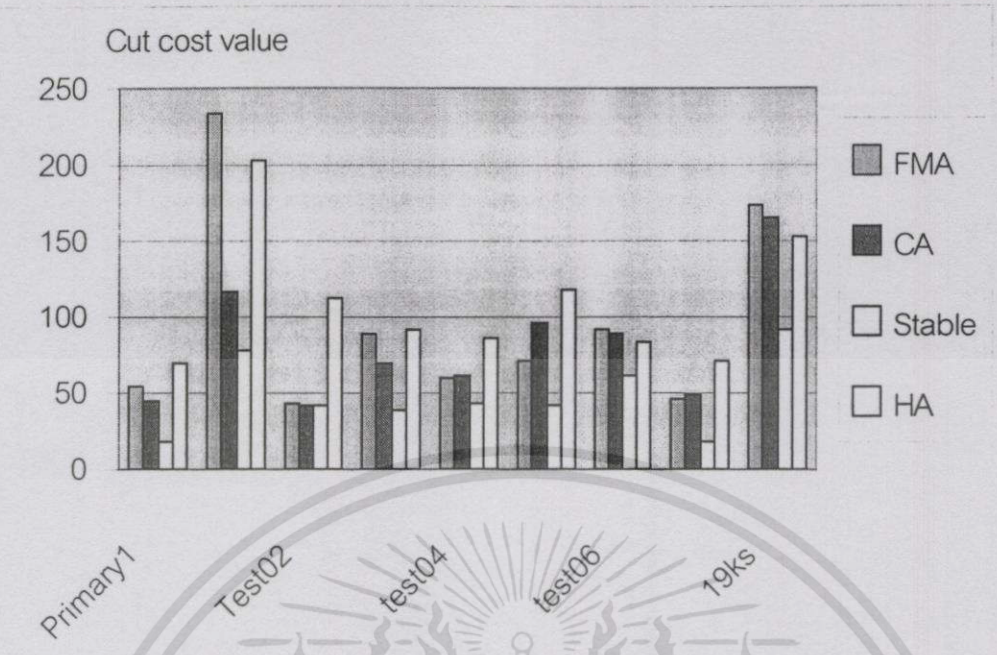
จากตารางที่ 5.8 แสดงการเปรียบเทียบผลลัพธ์ในการแก้ปัญหาการจัดกลุ่มวงจร โดยอาศัยวิธี SA, Eigenvectro, Tabu, Clustering [38] และ HA ได้นำผลการแก้ปัญหาการจัดกลุ่มวงจรจากตารางที่ 5.8 โดยค่า Avg Cut Cost มาสร้างกราฟเปรียบเทียบกับวิธีการผสมดังกล่าวแสดงดังรูปที่ 5.27 ผลที่ ได้รับคือ SA มีประสิทธิภาพในการหาคำตอบที่ดีที่สุดเพราะให้ค่า Cut Cost ต่ำกว่าวิธีอื่นๆที่นำมาเปรียบเทียบ ส่วนวิธี HA จัดอยู่ในเกณฑ์ที่ดีและยอมรับได้

ตารางที่ 5.9 การเปรียบเทียบผลลัพธ์การแก้ปัญหาการจัดกลุ่มวงจรโดยอาศัยวิธีการ Fiduccia-Mattheyses , Compaction Algorithm และ Stable Algorithm [39].

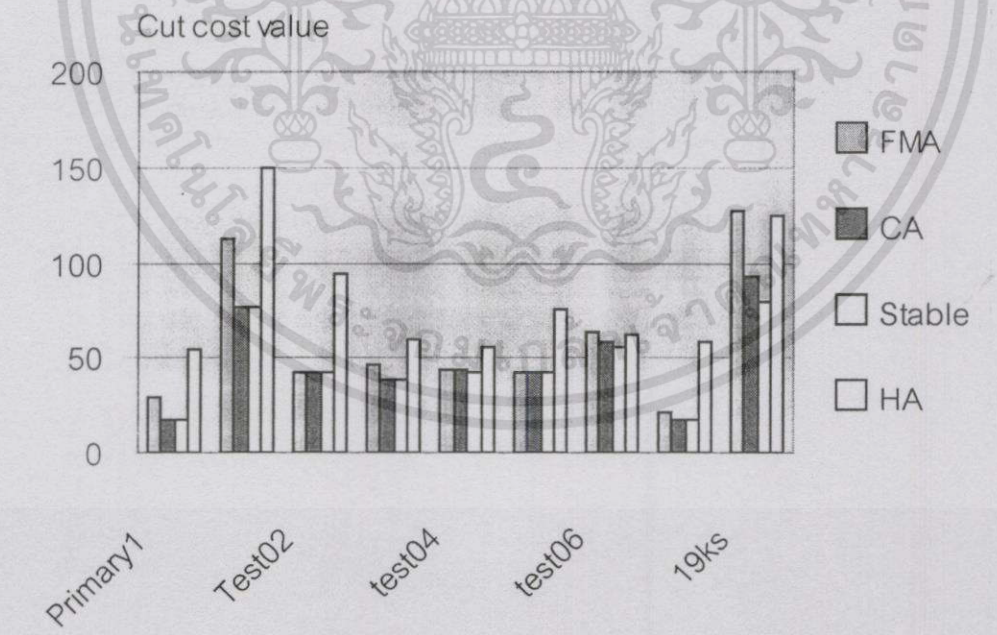
Circuit	Avg Cut Cost			Min Cut Cost / CPU		
	Fiduccia-Mattheyses Algorithm	Compaction Algorithm	Stable Algorithm	Fiduccia-Mattheyses Algorithm	Compaction Algorithm	Stable Algorithm
Prim1	54.45	43.95	18.20	29/3.4	17/7.3	17/29.7
Prim2	233.10	116.40	77.90	113/25.27	77/69.3	77/176.35
test02	42.8	42.15	42.0	42/20.85	42/22.85	42/138.7
test03	88.75	68.95	39.55	46/15.01	39/21.85	39/66.45
test04	60.3	60.70	43.4	44/12.15	44/24.5	42/67.65
test05	70.70	95.45	42.2	42/33.10	42/58.05	42/156.5
test06	91.90	88.70	60.85	64/20.65	58/33.3	55/142.25
bm1	46.40	48.35	17.95	21/5.2	17/12.05	17/33.85
19ks	173.45	165.15	91.0	127/44.45	93/75.95	80/228.8

จากตารางที่ 5.9 แสดงการเปรียบเทียบผลลัพธ์การแก้ปัญหาการจัดกลุ่มวงจรโดยอาศัยวิธีการ Fiduccia-Mattheyses. Compaction Algorithm (เป็นวิธีค้นหาคำตอบที่ให้ผลลัพธ์ที่ดีและเป็นวิธีที่นิยมใช้กันมากในปัจจุบัน ผู้ที่เสนอวิธีการนี้คือ Bui [40]) และ Stable Algorithm [39] จากค่าเฉลี่ยความสัมพันธ์ระหว่างกลุ่มซึ่งแสดงในรูปที่ 5.28 ค่าความสัมพันธ์ระหว่างกลุ่มน้อยสุดซึ่งแสดงในรูปที่ 5.29 นั้นจะเห็นได้ว่า Compaction Algorithm จะให้ผลลัพธ์ที่ดีกว่าวิธีการอื่นๆอย่างเห็นได้ชัดแต่วิธีการนี้มีข้อเสียในเรื่องของเวลาที่ใช้หาคำตอบซึ่งเมื่อดูจากรูปที่ 5.30 จะพบว่าวิธี Compaction Algorithm จะให้เวลานานกว่าวิธีอื่นๆที่นำมาใช้เปรียบเทียบ

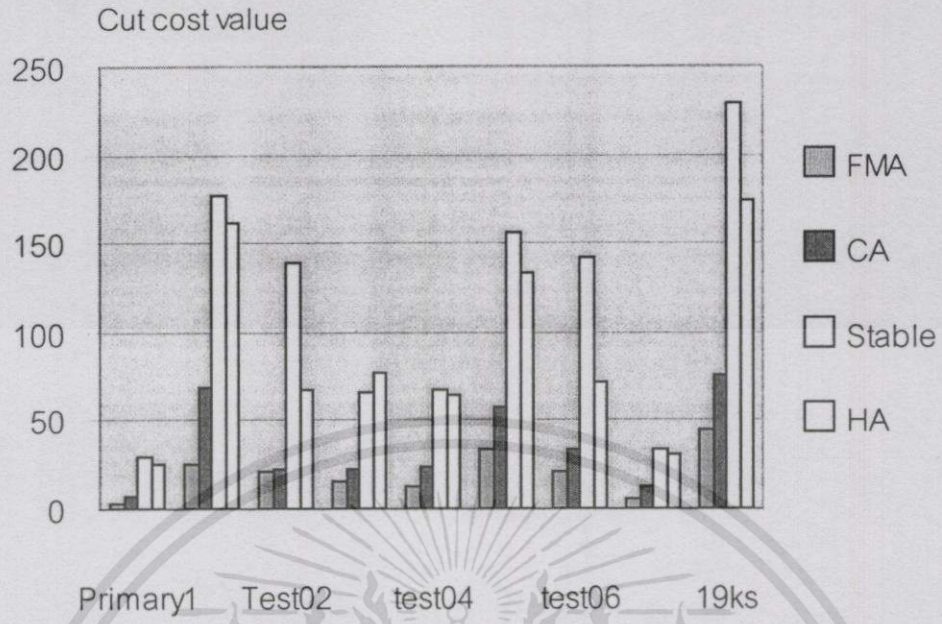
เวลาที่ใช้ในการทำงานของแต่ละวิธีในตารางที่ 5.9 ได้ทดสอบโดยการใช้ภาษา C เขียนทดสอบการทำงานบน SUN SPARC station สำหรับ 20 RUNs ส่วนวิธีการผสมนั้นทดสอบบนเครื่อง PC โดย Intel pentium 200MHz, 64 MB RAM



รูปที่ 5.28 กราฟเปรียบเทียบค่าเฉลี่ยของ Cut Cost ที่ได้จากวิธี FMA, CA, Stable และ HA.



รูปที่ 5.29 กราฟเปรียบเทียบค่าต่ำสุดของ Cut Cost ที่ได้จากวิธี FMA, CA, Stable และ HA.

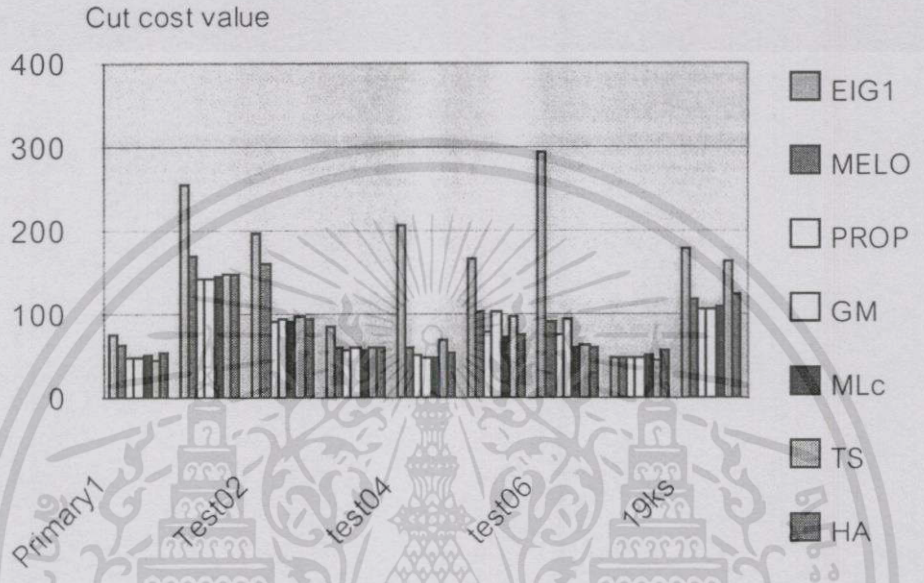


รูปที่ 5.30 กราฟเปรียบเทียบค่า CPU จากวิธี FMA, CA, Stable และ HA.

ตารางที่ 5.10 เปรียบเทียบวิธี TS (Tabu Search bisection heuristic) กับวิธี EIG1, MELO, PROP, GM และ MLc [41].

Circuit	Minimum Cut Cost					
	EIG1 Algorithm	MELO Algorithm	PROP Algorithm	GM Algorithm	MLc Algorithm	TS Algorithm
Primary1	75	64	47	47	52	45
Primary2	254	169	143	142	145	150
Test02	196	160	90	95	92	98
test03	85	60	59	62	58	60
test04	207	61	52	49	49	69
test05	167	102	79	104	72	97
test06	295	90	76	94	60	63
bm1	50	48	50	48	51	45
19ks	179	119	105	106	108	165

จากตารางที่ 5.10 แสดงการเปรียบเทียบผลลัพธ์การแก้ปัญหาการจัดกลุ่มวงจร โดยอาศัยวิธีการ TS (Tabu Search bisection heuristic), EIG1[42], MELO[43], PROP[44], GM[45] และ MLc[46] จากรูปที่ 5.31 แสดงกราฟเปรียบเทียบค่าความสัมพันธ์ระหว่างกลุ่มน้อยสุดของวิธีการต่างๆ ที่เสนอในตาราง 5.10 กับวิธี HA ซึ่งคำตอบที่ได้จากวิธี HA นับได้ว่าเป็นคำตอบที่ดี



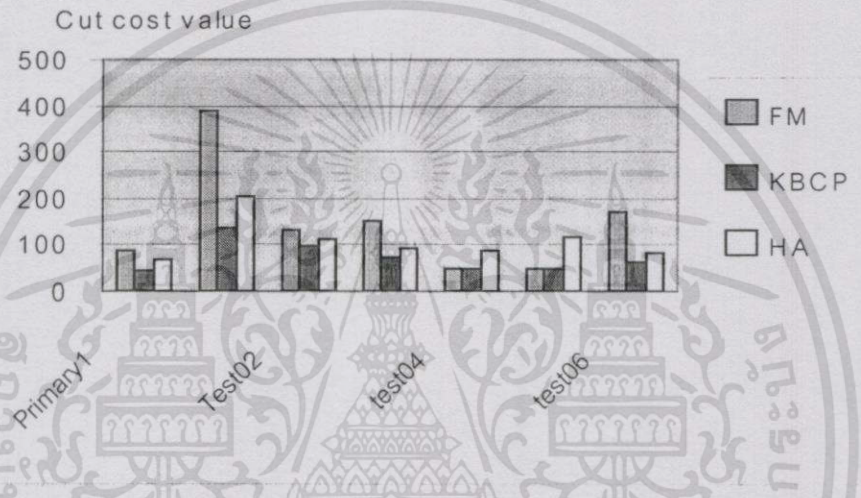
รูปที่ 5.31 กราฟเปรียบเทียบค่าต่ำสุดของ Cut Cost ที่ได้จากวิธี EIG1, MELO, PROP, GM, MLc, TS และ HA

ตารางที่ 5.11 เปรียบเทียบผลการแก้ปัญหาการจัดกลุ่มโดยวิธี FM กับ KBCP [47].

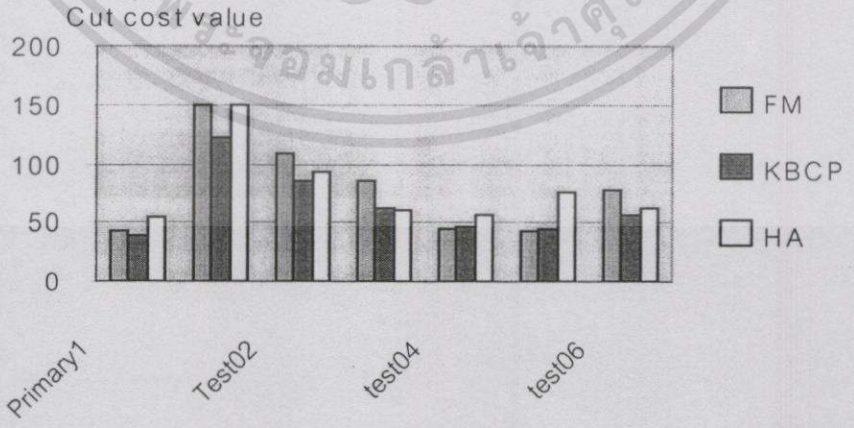
Circuit	Avg Cut Cost 20 run		Min Cut Cost 20 run	
	F&M Algorithm	KBCP Algorithm	F&M Algorithm	KBCP Algorithm
Primary1	89.3	45.6	43	39
Primary2	390.5	138.3	150	123
Test02	130.6	95.5	108	85
test03	148.8	71.7	85	63
test04	48.7	49.3	44	46
test05	49.2	48.3	43	45
test06	168.8	65.4	78	56

จากตารางที่ 5.11 แสดงการเปรียบเทียบผลลัพธ์การแก้ปัญหาการจัดกลุ่มวงจรโดยอาศัยวิธี FM, KBCP (k-way balance-driven circuit partitioning) เป็นวิธีการหาคำตอบโดยใช้หลักการของ Fuzzy สำหรับ K-way ในตารางที่ 5.11 ยกผลมาแค่ K=2 เท่านั้น การประมวลผลของทั้งสองวิธีทำบนเครื่อง SUN SPARC workstation ภายใต้ Berkeley 4.2 UNIX OS

จากรูปที่ 5.32 แสดงกราฟเปรียบเทียบค่าเฉลี่ยความสัมพันธ์ระหว่างของวิธี FM, KBCP เทียบกับวิธี HA และในรูปที่ 5.33 แสดงกราฟเปรียบเทียบค่าต่ำสุดของความสัมพันธ์ระหว่างของวิธี FM, KBCP เทียบกับวิธี HA



รูปที่ 5.32 กราฟเปรียบเทียบค่าเฉลี่ยของ Cut Cost ที่ได้จากวิธี KCCP, KBCP และ HA

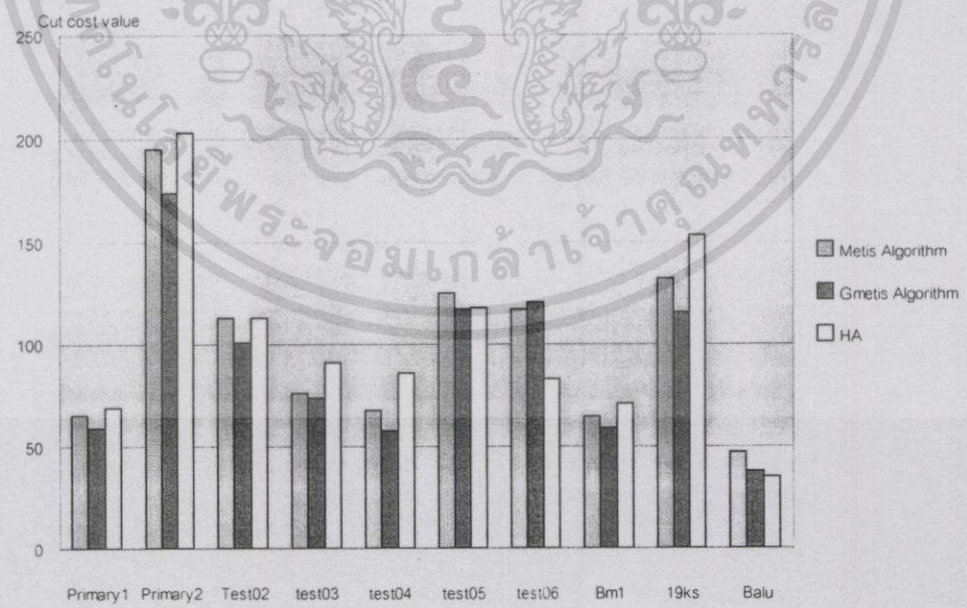


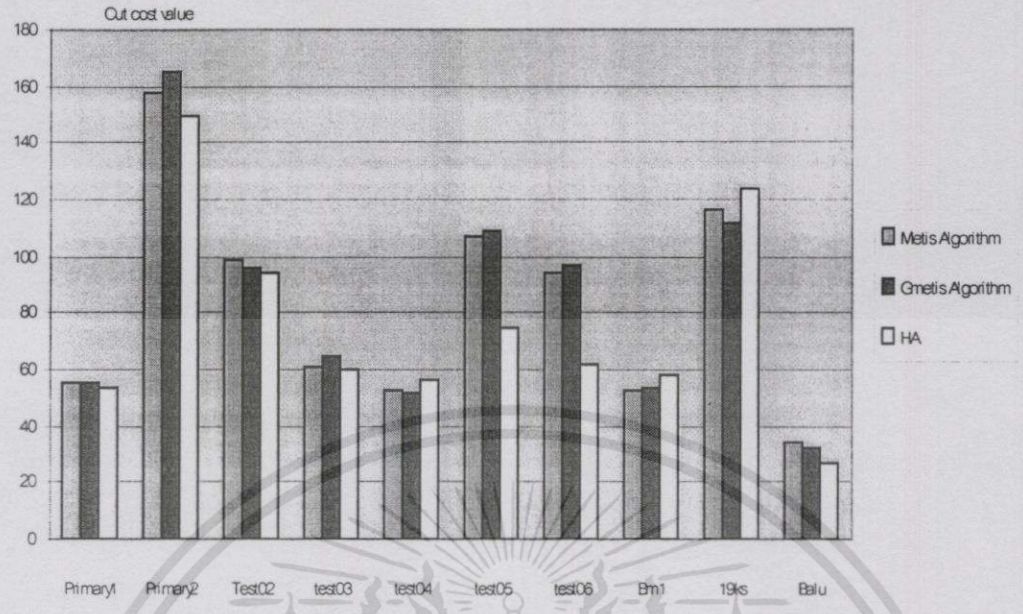
รูปที่ 5.33 กราฟเปรียบเทียบค่าต่ำสุดของ Cut Cost ที่ได้จากวิธี KCCP, KBCP และ HA

ตารางที่ 5.12 เปรียบเทียบผลการแก้ปัญหาการจัดกลุ่มโดยวิธี Metis กับ Genetic Metis ในส่วน Minimum Cut Cost, Average Cut Cost และ CPU time สำหรับ 100 runs ของแต่ละวิธี [48].

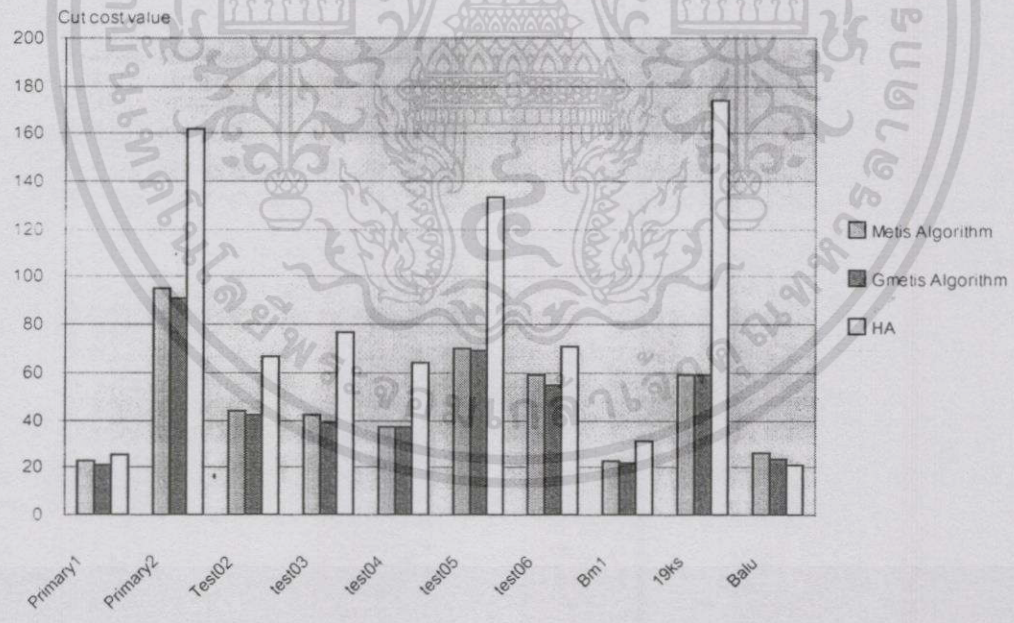
Circuit	Metis Algorithm			Gmetis Algorithm		
	Min	Avg	CPU	Min	Avg	CPU
Primary1	55	66	23	55	59	21
Primary2	158	195	95	165	174	91
Test02	99	113	44	96	101	42
test03	61	76	42	65	74	39
test04	53	68	37	52	58	37
test05	107	125	70	109	117	69
test06	94	117	59	97	121	55
Bm1	53	65	23	54	59	22
19ks	116	132	59	112	116	59
Balu	34	47	26	32	38	24

รูปที่ 5.34 กราฟเปรียบเทียบค่าเฉลี่ยของ Cut Cost ที่ได้จากวิธี Metis Alg, Gmetic Alg และ HA.





รูปที่ 5.35 กราฟเปรียบเทียบค่าต่ำสุดของ Cut Cost ที่ได้จากวิธี Metis Alg, Gmetic Alg และ HA



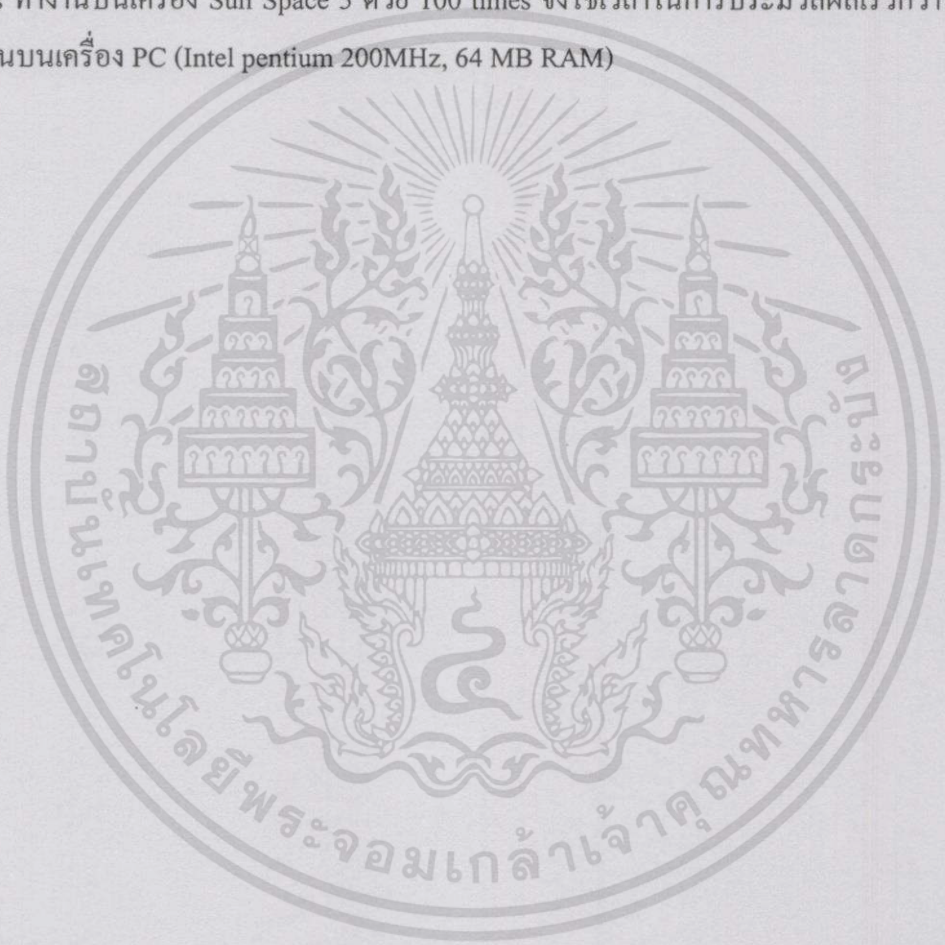
รูปที่ 5.36 กราฟเปรียบเทียบ CUP time ที่ได้จากวิธี Metis Alg, Gmetic Alg และ HA

จากตารางที่ 5.12 แสดงการเปรียบเทียบผลลัพธ์การแก้ปัญหาการจัดกลุ่มวงจรโดยอาศัยวิธี Metis วิธีนี้ถูกเสนอโดย G. Karypis และ V.Kumar เป็นการแก้ปัญหาโดย Unstructured Graph Partitioning and Sparse Matrix Ordering ดูข้อมูลเพิ่มเติมได้จาก http://www.cs.umn.edu/~kumar

ส่วนอีกวิธีคือ Ge7netic Metis (Gmetis) อาศัยหลักการของ Genetic รายละเอียดสามารถดูเพิ่มเติมได้จาก [48]

จากรูปที่ 5.34 แสดงกราฟเปรียบเทียบค่าเฉลี่ยความสัมพันธ์ระหว่างของวิธี Metis กับ Genetic Metis เทียบกับวิธี HA และในรูปที่ 5.35 แสดงกราฟเปรียบเทียบค่าต่ำสุดของความสัมพันธ์ระหว่างของแต่ละวิธีดังกล่าว ซึ่งวิธี HA ให้คำตอบที่จัดว่ายอมรับได้และถ้าเพิ่มจำนวน RUN อีกคาดว่าผลลัพธ์ยอมเป็นที่ยอมรับได้ดีในระดับหนึ่ง

ในรูปที่ 5.36 แสดงกราฟเปรียบเทียบ CPU time ของทั้งสามวิธี โดยที่วิธี Metis กับ Genetic Metis ทำงานบนเครื่อง Sun Space 5 ด้วย 100 times จึงใช้เวลาในการประมวลผลเร็วกว่าวิธี HA ที่ทำงานบนเครื่อง PC (Intel pentium 200MHz, 64 MB RAM)



บทที่6
บทสรุปและวิจารณ์

6.1 บทสรุปและวิจารณ์

จากการที่ใช้วิธีการจำลองอัลแนลลิ่งแก้ปัญหาการจัดกลุ่มวงจรนั้น คำตอบที่เกิดขึ้นจะมีเป็นจำนวนมาก การหาคำตอบที่ดีที่สุดจึงต้องอาศัยเวลามาก ในขณะเดียวกันถ้าใช้วิธีเจนนิติกแก้ปัญหาดังกล่าวจะหาคำตอบจากกลุ่มคำตอบจำนวน Pโครโมโซม จะมีวิธีการที่ซับซ้อนยุ่งยาก ดังนั้นขั้นตอนวิธีผสมที่นำเสนอจะช่วยลดเวลาและลดความซับซ้อนของการพิจารณาหาคำตอบ โดยที่วิธีการผสมแบ่งออกเป็น 2 ส่วนหลัก คือ ส่วนที่เป็นการวิเคราะห์ปัญหาและส่วนที่เป็นขั้นตอนการทำงาน เริ่มจากการวิเคราะห์ปัญหาเริ่มต้นจากการมองปัญหาให้อยู่ในรูปของกราฟและมองปัญหาเทียบเท่ากับโครโมโซมชนิดหนึ่งๆ กำหนดรูปแบบโครโมโซมให้อยู่ในรูปไบนารีโครโมโซมคือประกอบด้วยบิทที่มีค่าเป็น 0 หรือ 1 มีความยาวของโครโมโซมที่ขึ้นจำนวนวงจรทั้งหมด และให้สุ่มเลือกโครโมโซมมาเป็นจำนวน P โครโมโซมจากการที่ทุกตัวของโครโมโซมที่เลือกมามีค่าบิทแรกเหมือนกัน จากเหตุผลนี้จะเป็นการลดเซตของสถานะที่เป็นไปได้ทั้งหมด ทำให้ลดสถานะรอบข้างของการสุ่มหาคำตอบ ส่งผลให้เกิดการยอมรับคำตอบเร็วขึ้น

วิธีการผสมดังกล่าวสามารถนำไปใช้ได้ดีกับปัญหาการหาค่าเหมาะสมที่ใกล้เคียงคำตอบที่ดีที่สุดของการสับเปลี่ยนหาค่าการแบ่งส่วนเท่าโดยใช้หาคำตอบที่เป็นค่าต่ำสุดหรือค่าสูงสุด และสามารถแก้ปัญหาการติดอยู่กับที่ได้ดีในระดับหนึ่ง และการพัฒนาคำตอบเกิดจากการพัฒนาโครโมโซมด้วยวิธีมิวเตชัน

เมื่อนำวิธีการผสมมาพัฒนาวิธีการจัดกลุ่มวงจรแบบอัตโนมัติ ซึ่งเป็นขบวนการแรกของการออกแบบวงจรรวมในระดับกายภาพแบบอัตโนมัติ โดยอยู่ภายใต้เงื่อนไขความสัมพันธ์ระหว่างกลุ่มที่ถูกแบ่งน้อยที่สุด ส่งผลให้การออกแบบวงจรรวมทางกายภาพแบบอัตโนมัติในขั้นตอนถัดมาคือการจัดวางและการเชื่อมโยงวงจรมีความรวดเร็ว เนื่องจากไม่ต้องกลับทำการจัดวางและเชื่องโยงใหม่อยู่บ่อยๆ นอกจากนั้นผลจากการจัดกลุ่มวงจรที่ให้ค่าความสัมพันธ์ระหว่างกลุ่มน้อยที่สุดจะส่งผลต่อการลดค่าหน่วงสัญญาณของวงจรที่ออกแบบ ทำให้วงจรทำงานได้เร็วขึ้น

จากตารางที่ 6.1 แสดงผลลัพธ์การจัดกลุ่มวงจรโดยวิธีการผสมปรากฏว่าค่าความสัมพันธ์ระหว่างกลุ่มที่ได้มีค่าต่ำสุดในตัวอย่างวงจร EX9, EX10, IC67, IC151 และในตัวอย่างวงจร EX8, Qgr8, 3k5, Clique, Primary1, Primary2, test2, test3, test4, test5, tes6, bm1, 19ks, Fract, Balu ให้ค่าคำตอบจัดอยู่ใกล้ช่วงคำตอบที่ดีที่สุดมากกว่าในช่วงคำตอบที่แย่ที่สุด และมีเพียงตัวอย่างเดียวคือ IC116 ที่ค่าคำตอบแย่ที่สุดที่ได้จากวิธีการผสม

ตารางที่ 6.1 แสดงการเปรียบเทียบค่า Avg Cut Cost แบบ Two Way Partitioning ของวิธี HA กับวิธีการอื่นๆ * แสดงค่า cutcost ที่เกิดต่ำสุดในแต่ละตัวอย่างวงจร

Circuit	Avg Cut Cost								
	HA	SA	GA	FM	LA3	CDIP	Compaction	Gmetis	Metis
EX8	10.63	*10.21	11.3	-	-	-	-	-	-
EX9	*11.5	12.0	11.8	-	-	-	-	-	-
EX10	*16.6	16.9	17	-	-	-	-	-	-
Qgr8	*12.8	*12.8	12	-	-	-	-	-	-
3k5	12.4	16.8	*10.4	-	-	-	-	-	-
Clique	64	64	64	-	-	-	-	-	-
IC67	*79.5	82.5	79.8	-	-	-	-	-	-
IC116	131	122	*113.6	-	-	-	-	-	-
IC151	*108	165.1	135.4	-	-	-	-	-	-
Primary1	69.2	76	75.2	74.9	68.5	61.5	*43.95	59	66
Primary2	202.4	246	254.5	273.9	233.4	193.2	*116.40	174	195
test02	113	109	136	182.1	148.1	101.0	*42.15	101	113
test03	91	108	85.2	106.8	106.2	*66.8	68.95	74	76
test04	86	111	104	129.3	117.2	71.7	60.70	*58	68
test05	117.6	150	167	183.6	165.0	*90.7	95.45	117	125
test06	83	89.9	91.1	94.2	84.4	*71.8	88.70	121	117
bm1	71	77	58.9	79.5	67.5	59.8	*48.35	59	65
19ks	153.2	194	179	171.7	169.0	*105	165.15	116	132
Fract	55	55	55	-	-	-	-	-	-
Balu	35.1	*27	*27	38.1	36.1	33.9	-	38	47

จากรูปที่ 5.21 แสดงให้เห็นว่าวิธีการผสมใช้เวลาน้อยกว่าวิธีการจำลองอัลแนลลิ่งและวิธีเจนนิติก และจากการศึกษาถึงขั้นตอนวิธีการผสม กล่าวได้ว่าวิธีการผสมจัดเป็นเทคนิคการสุ่มตัวอย่างแบบหลายชั้น (Multi-Stage Sampling) [13]

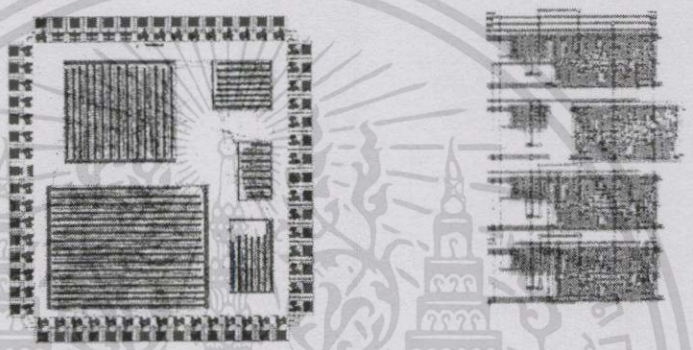
งานวิจัยนี้มีจุดมุ่งหมายเพื่อใช้ค้นหาวิธีการหาคำตอบที่เหมาะโดยกำหนดฟังก์ชันเป้าหมายเพื่อใช้ในการแบ่งกลุ่มวงจรดังที่ได้กล่าวมาแล้วจากเนื้อหาทั้งหมด โดยไม่ได้ทำการแสดงผลลงไปถึงการจัดวางและการเชื่อมโยงวงจร นอกจากนี้วิธีการผสมใช้อุณหภูมิเป็นตัวควบคุม T การยอมรับคำตอบและการสิ้นสุดการทำงาน ดังนั้นถ้าหาลองใช้เทคนิควิธีการควบคุมแบบอื่นๆ เช่น Fast Annealing, Very Fast Simulated Re-Annealing ก็อาจจะทำให้การค้นหาคำตอบเร็วยิ่งขึ้น

6.2 วิธีการผสมกับแนวทางการแก้ปัญหาอื่น

วิธีการผสมเป็นวิธีการแก้ปัญหาทางเลือกหรือวิธีการหาคำตอบที่ดีที่สุด จึงสามารถนำวิธีการผสมไปประยุกต์ใช้ในการแก้ปัญหาประเภทเดียวกันได้ โดยจะต้องมีการปรับรูปแบบการนำเสนอปัญหาและขั้นตอนที่จะแก้ปัญหานั้นๆ ซึ่งจะขึ้นอยู่กับลักษณะของปัญหาอาทิเช่น

6.2.1 ปัญหาการออกแบบวงจรรวมทางกายภาพ (VLSI Physical Design Automation)

จะเห็นได้ว่าวิธีผสมสามารถนำไปประยุกต์ใช้กับขั้นตอนการวางผัง (Cell Floorplanning) และการเชื่อมโยง (Routing) ซึ่งทั้งสองส่วนนี้เป็นปัญหาประเภทเอ็นพีค็อมพลีทเช่นเดียวกับปัญหาการจัดกลุ่มวงจรที่กล่าวมา [1][2][14][15][19]



รูปที่ 6.1 แสดงการวางจัดวางและการเชื่อมโยงของแสตนดาร์ดเซลล์

การวางผังวงจรสำหรับแสตนดาร์ดเซลล์จะแตกต่างจากการวางผังของ Building Blocks โดยสิ้นเชิง เนื่องจากคุณลักษณะของวงจร ในที่นี้จะขอพิจารณาการวางผังวงจรสำหรับแสตนดาร์ดเซลล์ ซึ่งทุกวงจรจัดว่าเป็นฟิคเซลล์ (วงจรที่มีขนาดความสูงเท่ากันแต่มีขนาดความกว้างของวงจรไม่จำกัด) และจัดเรียงเป็นแถว ความกว้างของวงจรขึ้นอยู่กับผลรวมของวงจรที่อยู่ในแถวที่ยาวที่สุด ความสูงของวงจรเลเอาต์ทั้งหมดขึ้นกับความสูงของวงจรกับความสูงของพื้นที่เชื่องโยง การวางผังที่ดีควรมีการปรับให้วงจรทุกแถวมีพื้นที่ว่างเปล่าน้อยสุด โดยกำหนดให้ R เป็นเซ็ตของกลุ่มวงจรในแต่ละแถว, $r_i \in R$, $R = \{r_1, r_2, r_3, ..., r_n\}$ และ M_{r_i} ซึ่งเป็นเซตของจำนวนวงจรในแถวใดๆ โดยที่ $m_i \in M_{r_i}$, $M_{r_i} = \{m_1, m_2, m_3, ..., m_n\}$ กำหนดให้ W เป็นเซ็ตของความกว้างของแถว $w_i \in W_{r_i}$ $W_{r_i} = \{w_1, w_2, w_3, ..., w_n\}$ วางวงจรลงบนแถวให้มีเนื้อที่ว่างเปล่าน้อยที่สุดโดยกำหนดโครโมโซม และเซ็ตที่เป็นไปได้ของสถานะเป็นเซ็ตของวงจรบนแถวที่เป็นไปได้ทั้งหมด เปลี่ยนสถานะใหม่โดยการสลับที่ของ วงจร สำหรับวงจรที่มีสายสัญญาณที่เกี่ยวเนื่องกับเส้นทางของสายสัญญาณที่มีการหน่วงเวลามากสุด (clitical path) จะไม่อนุญาตให้มีการสลับที่ ฟังก์ชันเป้าหมายกำหนดให้เป็น

$$f(n) = \sum_{i=1}^{r} s_i, f(n) = \text{เนื้อที่ว่างเปล่า} \quad \text{เมื่อ } s_i = \left| W_i - \overline{W} \right| \tag{6.1}$$

$W_i = \sum_{i=1}^{n} w_i$ เป็นความยาวรวมของวงจรที่อยู่แถวที่ i

$\overline{W} = \sum_{i=1}^{r} W_i$ เป็นความกว้างที่กำหนด ได้จากการประมาณ

การวางวงจรจะทำหลังจากการวางผังเรียบร้อยแล้ว การจัดวางวงจรเป็นการหาตำแหน่งที่ดีที่สุดบนเลเอาต์ที่ให้ความยาวโดยประมาณของสายสัญญาณทั้งหมดออกมาสั้นที่สุด และกำหนดให้ S เป็นเซ็ตของสายสัญญาณ $S_i \in S, S = \{s_1, s_2, s_3, ...s_k\}$ แต่ละวงจรประกอบด้วยเซ็ตของสายสัญญาณ $S_{m_i} \subseteq S$ แต่ละสายสัญญาณประกอบด้วยเซ็ตของวงจร $m_{s_i} \subseteq M$ วางวงจรลงบนเลเอาต์บนตำแหน่ง (x_i, y_i) โดยไม่มีการซ้อนทับระหว่างวงจร ซึ่งมีฟังก์ชั่นเป้าหมายดังนี้

$$f(n) = \sum_{i=1}^{k} w_i L_i, \quad f(n) = \text{ความยาวของสายสัญญาณ} \tag{6.2}$$

เมื่อ L_i = ความยาวของสายสัญญาณที่ i w_i = Weight ของสายสัญญาณที่ i ดังรูปที่ 6.1

ปัจจุบันการออกแบบวงจรรวมมีขนาดใหญ่ขึ้นเรื่อยๆ เช่น ULSI (Ultra Large Scale Integration :: Approximate number of transistors per chip in commercial products > 500,000) หรือ GSI (Giant Scale Integration :: Approximate numberof transistors per chip in commercial products > 1,000,000) การออกแบบ Tool เพื่อช่วยให้การออกแบบวงจรรวมมีความสะดวกรวดเร็วจึงเป็นการวิจัยที่สำคัญในยุคปัจจุบัน มีนักวิจัยมากมายได้ค้นคิดวิธีการใหม่ๆ ที่มีประสิทธิภาพทำให้ CAD tool ที่ใช้ในการออกแบบวงจรรวมมีประสิทธิภาพในการทำงานมากขึ้น งานวิจัยชิ้นนี้จึงเป็นการเสนอแนวทางใหม่แนวทางหนึ่งเพื่อพัฒนาซอฟต์แวร์ภายในประเทศต่อไป

เอกสารอ้างอิง

[1] M. Sarrafzadeh and C.K. Wong **An Introduction To VLSI Physical Design.** McGraw-Hill Book Companies, 1996.

[2] Bryan Preas and Michael Lorenzetti **Physical Design Automation of VLSI Systems.** The Benjamin/ Cummings Publishing Company, 1988.

[3] Emile Aarts and Jan Korst **Simulated Annealing and Boltzmann Machines.** John Wiley & Sons Ltd ,1989.

[4] ทรงพล ใหม่สาลี **การปรับปรุงประสิทธิภาพการจัดวางและเชื่อมโยงของเซลเบส.** บัณฑิต วิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, พ.ศ. 2540.

[5] S.Kirkpatrick, S.,C.D. Gelattand and M.P. Vecchi "Optimization by Simulated Annealing" Science, Vol. 220, No.4598, May 1983, pp. 671-680.

[6] Zbigniew Michalewicz **Genetic Algorithms + Data Structures = Evolution Programs.** Springer-Verlag Berlin Heidelberg, 1992.

[7] David E. Goldberg **Genetic Algorithms in Search, Optimization, and Machine Learning.** Addison-Wesley Publishing Company, 1989.

[8] Thang Nguyen Bui and Byung Ro Moon "Genetic Algorithm and Graph Partitioning" IEEE Transactions on computers, Vol.45, No.7, JULY 1996, pp. 841-855.

[9] Lester Ingber "Very Fast Simulated Re-Annealing" Mathl. Comut. Modelling, Vol. 12, 1989, pp. 967-973.

[10] นวรัตน์ อนันต์ชื่น **ทฤษฎีกราฟ I Graph Theory I.** ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร วิทยาเขตพระราชวังสนามจันทร์ นครปฐม, มกราคม 2540.

[11] Michel Gondram, Michel Minoux and Steven Vajda **GRAPHS AND ALGORITHMS.** A Wiley-Interscience Publication JOHN WILEY & SONS, 1979, pp. 23-33.

[12] CHRISTOS H. PAPADIMITRIOU and KENNETH STEIGLITZ **Combinatorial Optimization Algorithms and Complexity.** PRENTICE-HALL, Englewood Cliffs, New Iersey, 1982, pp. 20-24.

[13] รศ. สุรินทร์ นิยมารกูร **เทคนิคการสุ่มตัวอย่าง SAMPLING TECHNIQUES.** พิมพ์ครั้งที่ 3 สำนักพิมพ์, มหาวิทยาลัยเกษตรศาสตร์, 2541.

[14] Naveed Sherwani **Algorithms For VLSI Physical Design Automation.** Kluwer Academic Publishers, 1995.

[15] SADIQ.M.SAIT and HABIB YOUSSEF **VLSI Physical Design Automation Theory and Practice.** McGraw-Hill Book Companies, 1995.

[16] Gregory Tumbush and Dinesh Bhatia "Partitioning Under Timing and Area Constraints" Design Automation Laboratory, Department of ECECS, University of Cincinnati, Cincinnati, OH 45221-0030.

[17] Gregory Tumbush and Dinesh Bhatia "K-Way Partitioning Under Timing, Pin, and Area Constraints" Design Automation Laboratory, Department of ECECS, University of Cincinnati, Cincinnati, OH 45221-0030.

[18] Jun'ichiro MINAMI, Tetsushi KOIDE and Shin'ichi WAKABAYASHI "A CIRCUIT PARTITIONING ALGORITHM UNDER PATH DELAY CONSTRAINTS" Faculty of Engineering, Hiroshima University, IEEE, 1998, pp. 113-116.

[19] Sabih H.Gerez "Algorithms for VLSI Design Automation" John Wiley&Sons Ltd. 1999.

[20] C. M. Fiduccia and R. M. Mattheyses "A linear-time heuristics for improving network partitions" Proceedings of the 19th Design Automation Conference, 1982, pp. 175-181.

[21] C. Kring and A. R. Newton. "A cell-replication approach to mincut-based circuit paritioning" Proceedings of IEEE International Conference on Computer-Aided Design, November 1991, pp. 2-5.

[22] Y.Wei and C.Cheng "Towards efficient hierarchical designs by ratio cut partitioning" Proceedings of IEEE International Conference on Computer-Aided Design, 1989, pp. 298-301.

[23] Sandeep K.Lodha and Dinesh Bhatia "Bipartitioning Circuits using TABU Search" Design Automation Laboratory.

[24] Jin-Tai Yan and Pei-Yung Hsiao "A New Fuzzy-Clustering-Based Approach for Two Way Circuit Partitioning" Department of Computer and Information Science National Chiao Tung University Hsinchu, Taiwan, R.O.C.

[25] S.Kirkpatrick, S.,C.D. Gelattand and M.P. Vecchi "Optimization by Simulated Annealing" Science, Vol. 220, May 1983, pp. 671-680.

[26] Theodore W. Manikas, James T. Cain "Genetic Algorithms vs. Simulated Annealing : A Comparison of Approaches for Solving the Circuit Partitioning Problem" Technical Report

CS-93-07, Dept.of Electrical Engineering, May 1996.

[27] Wei, Y. and Cheng, C. "Ratio Cut Partitioning for Hierachical Designs" IEEE Trans. On Computer-Aided Design of Integrated Circuits, Vol.10 No.7, July 1991, pp. 911-921.

[28] Vesa Starck "IMPLEMENTATION OF SIMULATED ANNEALING OPTIMIZATION METHOD FOR APLAC CIRCUIT SIMULATOR" Helsinki university of technology, October 29, 1996.

[29] วิชัย บุญแสง และคณะ **ลายพิมพ์ดีเอ็นเอ..จากสารพันธุกรรมสู่เทคโนโลยีพิสูจน์บุคคล.** พิมพ์ครั้งที่1, สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ, ธันวาคม 2541, หน้า 11-20.

[30] กาญจนี วงศ์วิภาพร **การจัดตารางสอนของโรงเรียนแบบอัตโนมัติโดยจีเนติก อัลกอริทึม.** บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, พ.ศ. 2541.

[31] Davis, L. **Handbook of Genetic Algorithms.** Van Nostrand Reinhold, New York, 1991.

[32] Thang Nguyen Bui and Byung Ro Moon "Genetic Algorithm and Graph Partitioning" IEEE Transactions on computers, VOL.45, NO.7, JULY 1996, pp. 841-855.

[33] Bui, T. and Moon B. "Genetic Algorithms for Graph Bisection" Technical Report CS-93-07. Pennsylvania, State University, Dept. of Computer Science, April 1993.

[34] Charles Alpert's Home Page. Benchmark Characteristics. **http://vlsicad.cs.ucla.edu/~cheese/bench marks.html.**

[35] Charles J. Alpert and Andrew B.Kahng **SIMPLE EIGENVECTOR-BASED CIRCUIT CLUSTERING CAN BE EFFECTIVE.** UCLA Computer Science Department, Los Angeles, CA.

[36] Shantanu Dutt and Wenyong Deng "VLSI Circuit partitioning by Cluster-Removal Using Iterative Improvement" Department of Electrical Engineering, University of Minnesota, Minneapolis, IEEE, 1996.

[37] S.Dutt and W.Deng "A Probability-Based Approach to VLSI Circuit Partitioning" Proc. ACM/IEEE Design Automation Conf., June 1996. pp.100-105 (Recepient of the best-paper award).

[38] M.W. Allam, A. Vannelli and M.I. Elmasry "A CLUSTERING ALGORITHM FOR CIRCUIT PARTITIONING" Department of Electrical and Computer Engineering University of Waterloo.

[39] Chung-Kuan Cheng, Member, IEEE and Yen-Chuen A, Wei, Member IEEE "AN

Improved Two-way Partitioning Algorithm with Stable Performance" IEEE Transactions on comouter-aided design ,VOL. 12 , December 1991.

[40] T. Bui, C. Heigham, C. Jones and T. Leighton "Improving the performance of the Kernighan-Lin and simulated annealing graph bisection algorithms" in Proc. ACM/IEEE 26th Design Automation. Conf., 1989, pp. 775-778.

[41] Sandeep K. Lodha and Dinesh Bhatia "Bipartitioning Circuits Using TABU Search" Design Automation Laboratory, University of Cincinnati Cincinnati.

[42] L. Hagen and A. B. Kahng "Fast Spectral Methods for Ratio Cut Partitioning and Clustering" Proc. IEEE Intel. Conf. On Computer Aided Design, 1991, pp. 10-13.

[43] C. J. Alpert and S. Yao. "Spectral Partitioning: The More Eigen Vectore, The Better" Proc. ACM/IEEE Design Automation Conference, 1995, pp. 195-200.

[44] S. Dutt and W. Laguna **VLSI circuit partitioning by cluster-removal using iterative improvement techniques.** Proc. Physical Design Workshop, 1996.

[45] C. J. Alpert, L. W. Hagen and A. B. Kahng **A Hybrid Multilevel/Genetic approch for Circuit Partitioning.** Physical Design Workshop, 1996, pp. 100-105.

[46] Charles J. Alpert, J. Huang and A. Kahng "Multilevel Circuit Partitioning" Proc. ACM/IEEE Design Automation Conference, 1997, pp. 530-533.

[47] Jin-Tai Yan "Connection-Oriented Net Model and Fuzzy Clustering Techniques for K-Way Circuit Partitioning" Department of Computer and Information Science National ChiaoTung University Hsinchu. Taiwan.

[48] Charles J. Alpert, Lars W. Hagen, Andrew B. Kahng "A Hybrid Multilevel/Genetic Approach for circuit partitioning" UCLA computer Science Department, Los Angeles, CA.

ภาคผนวก ก.
ผลงานวิจัยที่ได้ตีพิมพ์ไปแล้ว

1. A. Yodtean, S. Choomchuay "An Efficient Algorithm for Circuit Partitioning" ISIC-99 8th International Symposium on Integrated Circuits, Devices & Systems, Singapore, Setemper 1999, pp. 359-362.
2. A. Yodtean, S. Choomchuay and Yasoji Suzuki "An Improved Simulated Annealing Algorithm for Circuit Partitioning" 1999 IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS'99), Puket Thailand, December 1999, pp. 473-476.

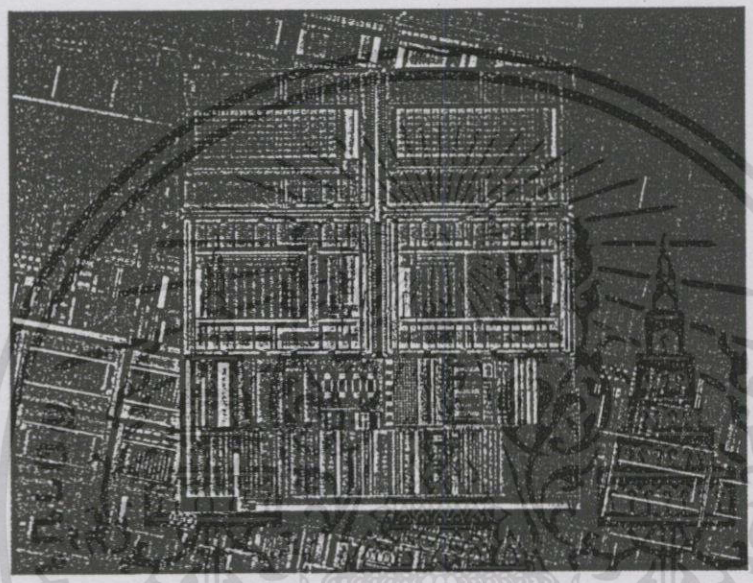
ISIC-99

8th International Symposium on
Integrated Circuits, Devices & Systems

8-10 September 1999
Grand Hyatt, Singapore

IEEE
Networking the World™
IEEE Singapore Section

PROCEEDINGS



Organised by:
Nanyang Technological University
School of Electrical and Electronic Engineering

And
IEEE Singapore Section

Sponsored by:
Cadence Design Systems

cadence

Supported by:
IEE Singapore Centre

IEE
IEE Singapore Centre

An Efficient Algorithm for Circuit Partitioning

Apiradee Yodtean*, Somsak Choomchuay*'**
*Department of Electronics, Faculty of Engineering, ** Research Center for Communications and Information Technology (ReCCIT), King Mongkut's Institute of Technology Ladkrabang (KMITL) Bangkok 10520 Thailand

Yasoji Suzuki
Department of Communication Eng., School of Eng., TOKAI Univ.,1117 Kita-kaname, City Kanagawa, 259-1292 Japan.

Abstract: This paper describes an improved simulated annealing algorithm applied for graph balanced bipartitioning problem, in particular for circuit partitioning problem. The new algorithm has incorporated the idea and procedure required in a genetic algorithm Experiment results on industrial benchmark circuits show that the proposed algorithm is superior to a conventional simulated annealing algorithm in both cut cost and run time.

1. INTRODUCTION

The partitioning problem is a NP-complete problem [1]; this means it is unlikely that a polynomial-time algorithm exists for solving the problem. One must use heuristic techniques for generating approximate solutions. One such heuristic is the widely used simulated annealing but abundant time of process. The circuit partitioning problem is, in fact, a two-way partitioning problem which is tied closely to a graph bisection model. Several optimization techniques can be applied to find the best solution in order to reduce the number of interconnections between groups and to minimize and economize computational work load [2].

In this paper, we propose a new simulated annealing algorithm (NSA) based on a conventional Simulated Annealing algorithm (SA). To improve an overall performance, an idea of Genetic Algorithm (GA) is incorporated in some procedures. Trying to reduce the solution set, NSA starts to reduce the solution set by half. In any half, a small set of solution is randomly picked up. GA is applied to find a best solution in such a set. The obtained solution is consequently used as an initial condition of the SA. A newer solution generated by mutation of a chromosome requires small storage in each iterative processing.

In the next section, we outline problem formulation and generic algorithms applied for partitioning problem. An improved simulated annealing algorithm is elaborated in section 3. Experiment and results obtained when the NSA is compared to the conventional SA are detailed before the paper conclusion.

2. PROBLEM FORMULATION AND ALGORITHMS

As a circuit partitioning problem can be treated as a graph partitioning problem, then input of a VLSI physical design can be viewed as a hypergraph. A given

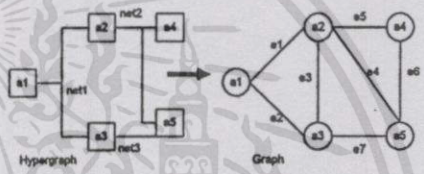


Figure 1 Hypergraph and Graph

circuit hypergraph is first translated into an actual graph before the application of optimization algorithms [3]. Let graph G = (V,E), where vertex $V=\{a_1,a_2,\ldots a_{n-1}\}$ and E is the set of edges. As shown in Figure 1, when a circuit is mapped into this graph model, cells are mapped into vertices and nets are mapped into edges.

2.1 Simulated Annealing (SA)

SA is quite a standard tool for optimization problems. It has an ability to jump out of poor local maxima because of the allowing of an occasional uphill move. A candidate solution is updated, continuously, until a termination condition is reached. In dept; starting at a point in the search space, a random move is made. If this move introduces a better point, such a move is accepted. This is called a positive move. On the other hand, if it introduces a worse point, the move is still accepted but with a decreasing probability over time. This is called a negative move. The negative move does allow SA to escape from a local optimum, however it is possible that SA could also miss a high-quality local optimum. A solution is driven toward a settle point by the application of a cool down temperature. To assure a really good solution, the optimization process may take a long time [4].

In the following subsection, a simulated annealing algorithm for circuit partition is implemented. The procedure is outlined in Figure 2. Start at the temperature of T_0 the algorithm iteratively compute the gain to determine its move.

$$Gain = \frac{cutsize}{|A| \cdot |B|}$$

cutsize is the number of interconnections between groups, whilst $|A|$ and $|B|$ are the number of vertices in Group A and B respectively. M is the number of move states per iteration. A vertex is randomly selected to move from its original group to another group. The acceptance of move is refused if such a move result in an unbalanced partition. In contrast, such a move is accepted if the attempt has improved cur cost. Otherwise, a move is accepted, with Boltzmann probability of acceptance which depend on the system temperature T. This is to keep the solution from converging to a local optimum. The algorithm will stop when t_s converge to zero value.

```
Simulated annealing Algorithm
Begin
   T = T0
   tstop = ts
   Current_Gain = Calculate_Gain()
   While tstop > 0 do
      Accept_Move = FALSE
      For I=1 to M do
         Randomly select vertex V to move from one partition to another
         New_Gain = Calculate_Gain()
         Δ Gain = New_Gain - Current_Gain
         If Accept_Gain_Change( Δ Gain,T) then
               Current_Gain = New_Gain
               Accept_Move = TRUE
         Else return V to original partition
      If Accept_Move then tstop = ts
      Else tstop = ts -1
      T = T * α
End
```

Figure 2 Simulated Annealing algorithm

2.2 Genetic Algorithm (GA)

GA is also an iterative procedure applied to several optimization problems. The algorithm begins with the first generation with population of P chromosomes. The following generations can be improved via the process of crossover and mutation. A chromosome can be encoded into a binary string of C genes. Each gene represents a component, and the allele represents a group (0 or 1). Each chromosome has a fitness value [5] which is considered for parents selection. Parents are chosen and combined through a crossover operation to produce an offspring. This offspring is modified by a mutation process (complement : zeroes become ones, ones become zeros). The replacement scheme is used to select a member of the population and replace them with the new offspring. A new population and the evolution process are repeated until certain condition is met. To store population record, a an amount of memory which varies with the size of the problem is required. For general application, the algorithm requires a huge storage to store all possible solution.

```
Genetic Algorithm
Begin  create initial population of size P
          Repeat
             Select parent1 and parent2 from the population
             Offspring = crossover(parent1,parent2)
             Mutation(offspring)
             Update_population
          Until  stopping criteria met
          Report the best solution
End
```

Figure 3 Genetic Algorithm

3. A NEW SIMULATED ANNEALING

A major drawback of a SA is its time consuming when one wants to ensure a neatly good solution. Not much better choice, a huge storage is sacrificed when one turns to a genetic optimization. We, thus, look for an algorithm that takes reasonable computation time, not so huge resource consuming while offering a nearly optimal solution. The proposing NSA is based on a conventional SA but takes some good points offered by GA. The following subsection is a rough basic concept of NSA.

Step 1: starts with a set of P population. P chromosomes of which the first bit is fixed at 1 or 0 are randomly selected. As shown in Figure 4, the first group comprises of chromosomes with the first bit value of 0 (represented by light color circles) and the second group comprises all chromosomes with the first bit value of 1 (represented by dark circles).

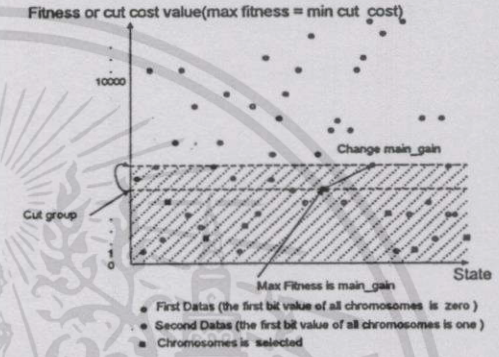


Figure 4 Physical demonstration of NSA

The cut cost value of each circle is obtained by bisection method. It should be noted that a chromosome length is corresponding to the number of circuits. A vertex in the graph is represented by a gene of a chromosome. The gene has a value of 1 if the corresponding vertex is on the left-hand side of the bisection and it has a value 0 if the vertex is on the right hand side of the bisection. This is illustrated in Figure 5 below)

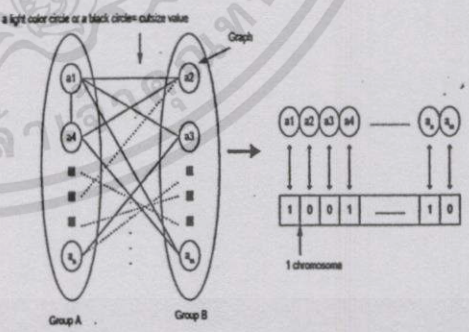


Figure 5 meaning of white circle and black circle

Step 2: For a group being considered (the second group, in this case), the first bit of all chromosomes are fixed at the same value. The number of all possible answers can then be reduced by half; i.e.

If n is an even
$$N = \frac{n!}{\left(\frac{n}{2}\right)!\left(\frac{n}{2}\right)!\times 2} \qquad (1)$$

If n is an odd

$$N = \frac{n!}{\left(\frac{n+1}{2}\right)!\left(\frac{n-1}{2}\right)!} \qquad (2)$$

Refer to Figure 4, consider P chromosomes located in the shaded area. Each chromosome has a fitness value, a measure of the quality of the solution. The maximum fitness of chromosomes can be selected and assigned as a main_gain value (generate cut group; Figure 4). The main_gain value divides the answer set into two groups. Cut cost value of chromosomes in the first group is less compared to the main_gain value while that of the second group is greater.

Step3: The main_gain can be improved iteratively by mutation of many positions at a time.

Step4: Taken into account the main_gain, a conventional procedure for SA-like is performed. T is a cooling factor α, where $0 < \alpha < 1$. The algorithm stops if there have been no changes to the solution after Time equals Maxtime.

```
NSA Algorithm(T0,a,M,Maxtime)
(* T0 is the initial temperature
   α (alpha) is the cooling rate
   M represents the time until the next parameter update
   Maxtime is the total allowed time for the annealing process *)
/* Fchrom is string of minimum cut cost chromosome */
Begin
   Create initial population of fixed size P and fixed first bit (1 or 0)
   Choose a Fchrom form P chromosomes
   Main_Gain = cut cost value of Fchrom
   Current_Gain = Main_Gain
   T = T0
   Time = 0
   Do Accept_Move = FALSE
         Metropolis(Current_Gain,T,M,Fchrom)
         Time = Time+M
         T = T * α
   Until(Time >= Maxtime)
   Output Best solution found
End
```

Figure 6: New Simulated Annealing (NSA)

As shown in Figure 7, chromosome mutation is invoked in order to change the cut cost value. The nearly obtained cost function is then compared to the main_gain. If a new cost function higher than the main_gain, the second group is considered and the main_gain is replaced by the new cost function. Next step is to examine whether a new cost solution is better than the current cost solution. If so, a new solution is accepted. Metropolis accepts the new solution on a probabilistic basis of Boltzmann distribution. A random number is generated in the range of 0 to 1. If this number is smaller than $e^{-\Delta C/T}$, where ΔC is the difference in cost, the inferior solution is accepted. Notice that in the mutation process, the first bit of a chromosome is exempted (not reverse). This method effects the fast accepted answer since the redundant set of answer is ignored.

```
Algorithm Metropolis (Current_Gain,T,M,Fchrom)
Begin k=0
      For I=1 to M do
        do
           New_chrom = Mutations(Fchrom)
           New_Gain = Calculate_cutcost(New_chrom)
           If New_Gain > Main_Gain Then
              K=1
              ΔGain = New_Gain-Current_Gain
              If Accept_Gain_Change(ΔGain,T) Then
                    Current_Gain = New_Gain
                    Accept_Move = TRUE
           Else  K=0
        While( K != 0 )
End
```

Figure 7: Metropolis of NSA

4. EXPERIMENT RESULTS

Experiment was carried out by using a subset of the benchmarks obtained from the ACM/SIGDA suite given in Table I [6] where the numbers of cells, Pads, Modules, and Nets for each circuits are detailed. The algorithm is implemented in C programming language and tested on a 486-66 based PC with 16 MB of RAM. Parameters for SA and NSA are summarized in Table II. The obtained average results given in Table III are taken of twenty runs. Table III shows the results of average cut cost, mincut and average CPU time obtained from SA and NSA.

Table I Circuit specification

Circuit	# Cells	#Pads	#Modules	# Nets
Qgr8	12	0	12	23
3k5	15	0	15	33
Clique16	16	0	16	120
IC67	16	51	67	138

Table II Parameters used in SA and NSA

Parameters of SA			
T_0	α	M	T_s
1000	.9	5,10,15,18	5,10

Parameters of NSA				
P	T_0	α	M	Maxtime
10	1000	.9	5,10,15,18	5,10

Table III Average cut cost, Mincut and average CPU time of SA and NSA

Circuit	Avg cut cost		Mincut		CPU time(sec)	
	SA	NSA	SA	NSA	SA	NSA
Qgr8	10.3	9.5	5	5	n/a	n/a
3k5	13.7	12.6	8	8	n/a	n/a
Clique16	64	64	64	64	0.9	0.75
IC67	82.5	73.3	69	37	2.75	2.55

Shown in Table III, it is obvious that the total quality (averages cut cost) of partitions obtained by the New Simulated Annealing Algorithm is better that obtained from a conventional Simulated Annealing Algorithm. *mincut* of partitions obtained form both algorithms are comparable. The New Simulated Annealing Algorithm also shows its superior in term of computing performance (n/a = very small and not notable).

5. CONCLUSIONS

An improved two-way partitioning algorithm is proposed. Its behavior is very stable for all circuits tested. The new algorithm which is based on both simulated annealing algorithm and genetic algorithms yields a comparable or better performance compared to the conventional SA. Experiment results show that our algorithm runs faster and produces better cut cost.

REFERENCES

[1] Christos H. Papadimitriou, Kenneth Steiglitz, "Combinatorial Optimization: Algorithms And Complexity", Prentice-Hall Inc., 1982, pp. 342-401.

[2] Sadiq.M.Sait, Habib Youssef, "VLSI Physical Design Automation Theory And Practice", McGraw-Hill Companies Inc., 1995, pp.37-79.

[3] M. Sarrafzadeh, C. K. Wong, "An Introduction To VLSI Physical Design", McGraw-Hill Companies Inc., 1996, pp. 33-34.

[4] Theodore W. Manikas, James T. Cain, "Genetic Algorithms vs Simulated Annealing: A Comparison of Approaches for Solving the Circuit Partitioning Problem", Technical Report, May 1996, pp. 96-101.

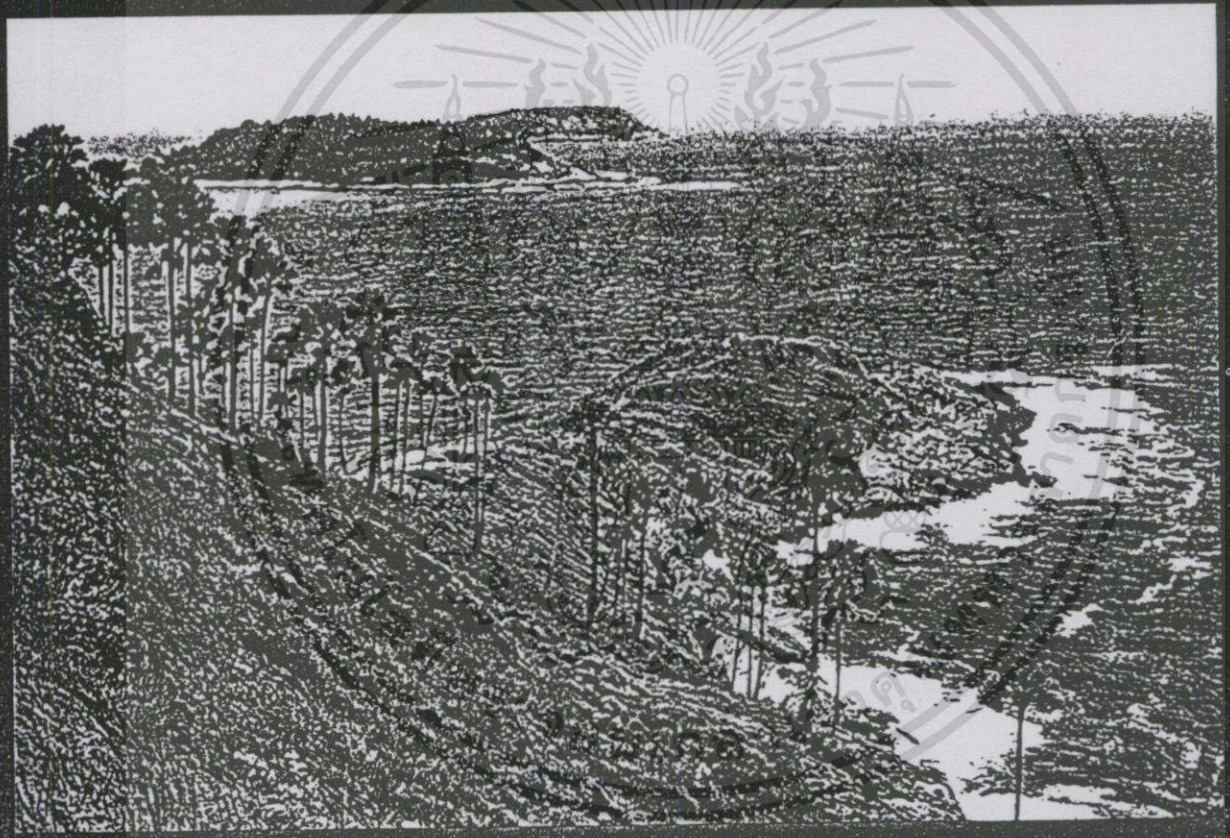
[5] Thang Nguyen Bui and Byung Ro Moon, "Genetic Algorithm and Graph Partitioning", IEEE Transactions on computers, Vol.45, No.7, July 1996, pp. 841-855.

[6] Charles Alpert's Home page, "Benchmark Characteristics", http://vlsicad.cs.ucla.edu/~cheese/ benchmarks.html

PROCEEDINGS

ISPACS'99

1999 IEEE International Symposium on Intelligent Signal Processing and Communication Systems



Signal Processing and Communications Beyond 2000

December 8-10, 1999

An Improved Simulated Annealing Algorithm for Circuit Partitioning

A. Yodtean*, S. Choomchuay*,** and Yasoji Suzuki***

* Faculty of Engineering and ** Research Center for Communications and Information technology King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok 10520 Thailand. Phone: (662)326-9968 Ext.114, Fax(662) 739-2384, E-mail: s9061049@kmitl.ac.th, skcxch@erc.kmitl.ac.th

*** Department of Communication Eng., School of Eng., TOKAI Univ., 1117 Kita-kaname, City Kanagawa, 259-1292 Japan.

Abstract

This paper presents an hybrid algorithm (HA) based on conventional simulated annealing algorithm (SA) and genetic algorithm (GA). The proposed algorithm can be applied to two-way circuit partitioning problem, in particular using bisection method. It provides better cut cost and computing performance. Experimental results on industrial benchmark circuits showed notable improvement over published algorithms.

1. Introduction

Circuit partitioning problem has become more important in VLSI design [1][2] as the whole circuit site is getting larger. Partitioned circuit can be slightly different since the problem belongs to the class of NP-completes. In order to obtain the better solution, rather than using only a conventional simulated annealing algorithm, this research has corporate some steps of another known optimization procedure, Genetic Algorithm. The objective is to reduce the number of interconnections between groups with minimal and economic CPU time [3]. Let G = (V,E) be a graph, where V is the set of vertices and E is the set of edges, when a circuit is mapped into this graph model, cells are mapped to vertices and nets are mapped to edges. The partitioning of G is a bisection problem for two-way partitioning. Let C_{ij} be the capacity of an edges connecting vertex i and vertex j. (A,B) denotes a cut that separates a set of vertices A from B = V-A [4]. The capacity of this cut is equal to $C_{AB} = \sum_{i \in A} \sum_{j \in B} C_{ij}$.

In section 2, we explain the general idea of using simulated annealing algorithm and genetic algorithm for partitioning problem. In section 3, a combination of both algorithms, hybrid algorithm, is introduced. In section 4 we report the results of our experiments and compare the performance of our algorithm with others. Finally, we conclude with a comparative discussion on our approach.

2. Algorithms for Circuit Partitioning

2.1 Naïve Simulated Annealing Algorithm

Simulated annealing algorithm is an iterative procedure that continuously updates one candidate solution until a termination condition is reached. A candidate solution is randomly generated and the procedure starts at a high temperature. Started at anywhere in the search space, a random move is made. If this move introduces a better point, it is accepted. This is called a positive move. If it introduces a worse point, it is still accepted, with a decreasing probability over time. This is called a negative move. The probability of negative move decreases as time goes by. The negative moves allow SA to escape from local optima and therefore it is possible that SA could also miss high-quality local optima. This optimization process may take a long time to reach the desired solution (good approximation solution) [3]. The generalize physical simulated annealing generated by a simulated annealing algorithm.

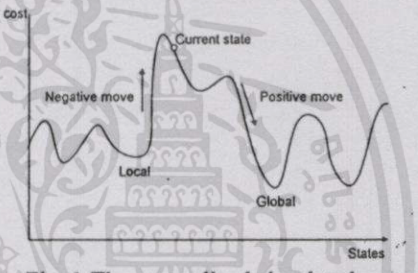


Fig. 1 The generalized simulated annealing.

2.2 Naïve Genetic Algorithm

GA is also an iterative procedure of which each iteration is called a generation. Set of solutions, population, is a binary string that corresponds to a bisection of the graph. Number of genes in the chromosome equals to *n*, the number of vertices in the graph. Each gene corresponds to a vertex in the graph. A gene has value 0 if the corresponding vertex is on say the left side of the bisection, and has value .1 otherwise. For each chromosome, the number of 0s and 1s is equal [3].

GA creates *p* solutions at random (the only constraint on a chromosome is that the number of 0s and 1s should be equal). Thus, numbers of all answers is N,and then number of n circuits is easy to see that;

if n is an even number; $N = \frac{n!}{\left(\frac{n}{2}\right)!\left(\frac{n}{2}\right)!}$ (1)

if n is an odd number; $N = \frac{n!}{\left(\frac{n+1}{2}\right)!\left(\frac{n-1}{2}\right)!} \times 2$ (2)

The fitness value F_i of a solution i can be calculated as followed

$$F_i = (C_w - C_i) + \frac{C_w - C_b}{3} \quad (3)$$

where, C_w is cut cost of the worst solution in the population, C_b is cut cost of the best solution in the population, C_i is cut cost of the solution i.

Each chromosome is selected as a parent with a probability that is proportional to its fitness value. According to the fitness definition it is easily seen that the probability that the best chromosome is chosen is four times as high as the probability of that worst chromosome. This is a very common parent selection scheme called proportional selection [3].

Let the probability factor be defined as;

Probability factor (r) $\quad r = 4^{\frac{1}{P-1}}$ (4)

Assume that the probabilities assigned to each individual is a geometric progression, where the sum of all these probabilities S is given by

$$S = 1 + r + r^2 + \ldots + r^{P-1} = \frac{1 - r^P}{1 - r} \quad (5)$$

The probability that the chromosome i has been selected, Pr i}, is found by

$$\Pr\{i\} = \frac{r^{i-1}}{S} \quad (6)$$

Crossover : A crossover operation creates new offspring chromosomes by combining parts of the two parent chromosomes. It randomly selects a cut point which is the same on both parent chromosome. (In Figure 2: *pos* is the cut point) The cut point divides the chromosome into two disjoint parts: the left part of parent 2 is copied to the same location of the offspring chromosome. Let this be offspring We devised one more crossover operator which is the same as the above except that it copies the complement values of the right part of parent 2 while the copies left part parent1 is unchanged. Let this be offspring 2. An example crossover is shown in Figure 2.

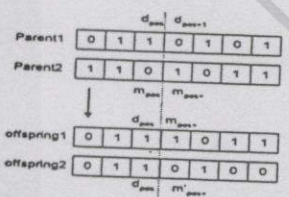


Fig. 2 Crossover example

Mutation : Each offspring much meet the same constraints as its parents (the number of ones and zeroes in the bit pattern should be nearly equal). However, the crossover operation may produce an offspring that does not meet this requirement. An offspring is altered via mutation, the operation that randomly adjusts bits in the offspring so that its bit pattern is valid. The mutation procedure determines the value *b*, which is the absolute value of the difference in the number of ones and zeroes. A bit location on the offspring is randomly selected, then starting at that location in offspring that represent valid partitions.

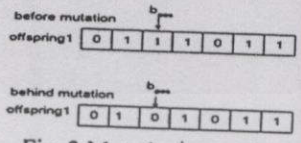


Fig. 3 Mutation example

Update population : The creation of two offsprings increases the size of the population to P+2. Since we want to maintain a constraint population size of P, Two individuals will need to be eliminated from the population. The goal of the algorithm is to converge to the best quality solution, thus the two individuals with the lowest fitness value are removed from the population

3. Hybrid Algorithm (HA)

HA is based on SA but with the attempt of trying to reduce CPU time. From reduced number of answers which have direct effect on fast acceptance of answer, the new answer is generated by mutation of a chromosome that requires a small storage for its each iteration processing.

Each solution to our problem is represented by a chromosome, which is a binary string. Each gene has a value from an alphabet S, say {0,1}. The number of vertices is often the size of the chromosome. The number of genes should be an equal number of 0s and 1s in a chromosome.

HA starts with a set of initial solutions is number of P chromosomes with the first bit of chromosome is fixed by *1* or *0*. (Figure 4 showed First data: the first bit value of all chromosomes is *0* and Second data: the first bit value of all chromosomes is *1*) We select to consider second group. The first bit of all chromosomes is fixed as valued. This has direct effect on the reduction of reiterate answers.

if n is an even number, $N = \dfrac{n!}{\left(\frac{n}{2}\right)!\left(\frac{n}{2}\right)!2!}$ (7),

if n is an odd number, $N = \dfrac{n!}{\left(\frac{n+1}{2}\right)!\left(\frac{n-1}{2}\right)!}$ (8)

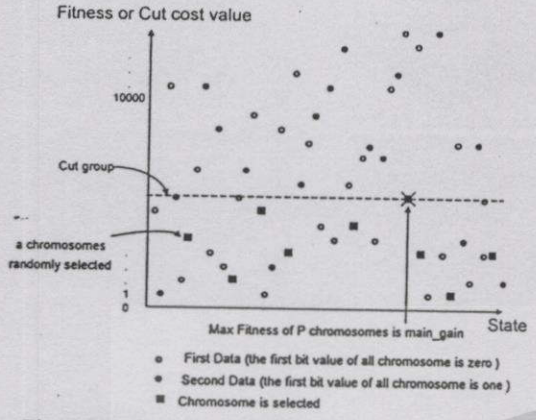


Fig. 4 Physical demonstration of a Hybrid Algorithm

Each individual has a fitness value, which is a measure of the quality of the solution represented by the individual. The fitness value F for individual i in Equation 3.

Maximum fitness of a chromosome is selected. It was called main_gain value (generate cut group shown in Figure 4), which separates the answer set into two groups. Cut cost value of each answer in the first group is less than main_gain value, while that of those in the second group is higher.

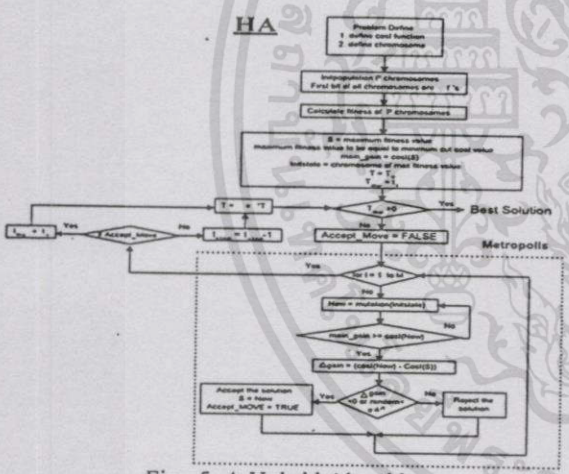


Fig. 5 A Hybrid Algorithm

The structure of the above discussed HA is given in Figure 5. The algorithm changes state with mutation of some bits of a chromosome. New cost function is compared to the main_gain value. If cost of the new cost function is higher than the main_gain, the second group is considered. Next step is to consider the cost of new solution whether it is better than the cost of the current solution. If it is then the new solution is acceptable. Metropolis will accept the new solution on a probabilistic basis of Boltzmann Distribution. A random number is generated in the range 0 to 1. If this random number is smaller than $e^{-\Delta gain/T}$, where Δgain is the difference in cost , the inferior solution is accepted.

The Mutations algorithm determines the value b, which is the absolute value of the difference in the number of ones and zeroes. A bit location on the chromosome is randomly selected, then starting at that location, b bits complemented (zeroes become ones, ones become zeroes). With an exemption that applied to the first bit of chromosome (not mutation), the method can reduce the number of repeat answers. Fast acceptance of answers is then made possible.

After each iteration, the temperature T is scaled down by a cooling factor α, where $0 < \alpha < 1$. The algorithm stops if there has been no changes to the solution after time equals Maxtime. HA can be considered as an improved version of NSA (New simulated annealing)[6].

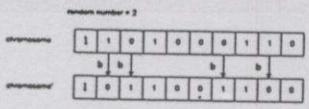


Fig. 6 Multi-mutation example

4. Experimental Results

We experimentally evaluated the quality of the bisections produced by our partitioning algorithm on a large number of hypergraphs that are part of widely used ACM/SIGDA circuit partitioning benchmark suite (available via the World Wide Web at http://www.cs.ucla.edu/~cheese). The characteristics of these hypergraphs are shown in Table1. Table 2 shows parameters applied to different algorithms; SA, GA, NSA and HA. All code was written in C and all benchmarks were tested on a 486 PC.

Table 1 Specification of the circuits

Circuit	# Cells	#Pads	#Modules	# Nets
Qgr8	12	0	12	23
3k5	15	0	15	33
Clique16	16	0	16	120
IC67	16	51	67	138

Table 2 Parameter of SA, GA, NSA and HA

Parameter of SA			Parameter of GA	
T_0	α	T_s	P	Generation
1000	.9	5,10,15,18	10	5,10

Parameter of NSA and HA, 1st Generation			
T_0	P	α	T_s
1000	10	.9	5,10,15,18

Average cut cost, minimum cut cost and average CPUt times of the applications of HA with SA, GA and NSA for Two-Way partitioning are given in Table 3, 4, 5 respectively.

HA was tested on 4 benchmark circuit graphs. Shown in table 3, for small circuit such as Qgr8 and 3k5 the algorithm yield the lowest cut cost. For larger circuit such as Clinique16 and IC67 it yields comparable results obtained from others.

In term of minimum cut cost, as detailed in Table 4, HA algorithm offers similar results compared with others.

Table 5 elaborate the average computing time. This is unobservable for small circuit size. For larger circuit size,

HA and NHA yield similar results that comparable to that btained from a conventional GA.

Table 3 Effect of average cut cost of SA, GA, NSA and HA

Circuit	Average cut cost			
	SA	GA	NSA	HA
Qgr8	10.3	9	9.5	8.5
3k5	13.7	8	12.6	8
Clique16	64	64	64	64
IC67	82.5	79.8	73.3	79.5

Table 4 Effect of minimum cut of SA, GA, NSA and HA

Circuit	Min cut			
	SA	GA	NSA	HA
Qgr8	8	8	8	8
3k5	8	8	8	8
Clique16	64	64	64	64
IC67	69	24	37	20

Table 5 Effect of average CPU of SA, GA, NSA and HA

Circuit	Average CPU (sec/run)			
	SA	GA	NSA	HA
Qgr8	N/A	N/A	N/A	N/A
3k5	N/A	N/A	N/A	N/A
Clique16	0.9	0.8	0.75	0.8
IC67	2.75	2.4	2.55	2.5

Discussion

An improved two-way partitioning algorithm is pposed. The hybrid algorithm is based on both simulated nealing and genetic algorithms. According to test applied benchmark circuits, HA yields comparable or better rformance compared to others. The obvious advantage of proposed algorithm is that it requires less storage mpared to the conventional GA and it is faster than a nventional SA. Applying of this algorithm to a very large cuit size is under investigation.

References

Sadiq M. SAIT and Habib Youssef, " VLSI PHYSICAL DESIGN AUTOMATION Theory and Practice", McGraw-Hill Book Company, pp 63-71, 1995.

Naveed Sherwani, "ALGORITHM FOR VLSI PHYSICAL DESIGN AUTOMATION Second Edition", Kluwer academic publishers, pp 141-171, 1995.

Thang Nguyen Bui and Byung Ro Moon, "Genetic Algorithm and Graph Partitioning " , IEEE Transactions on computers, Vol.45, No.7, pp. 841-855, JULY 1996.

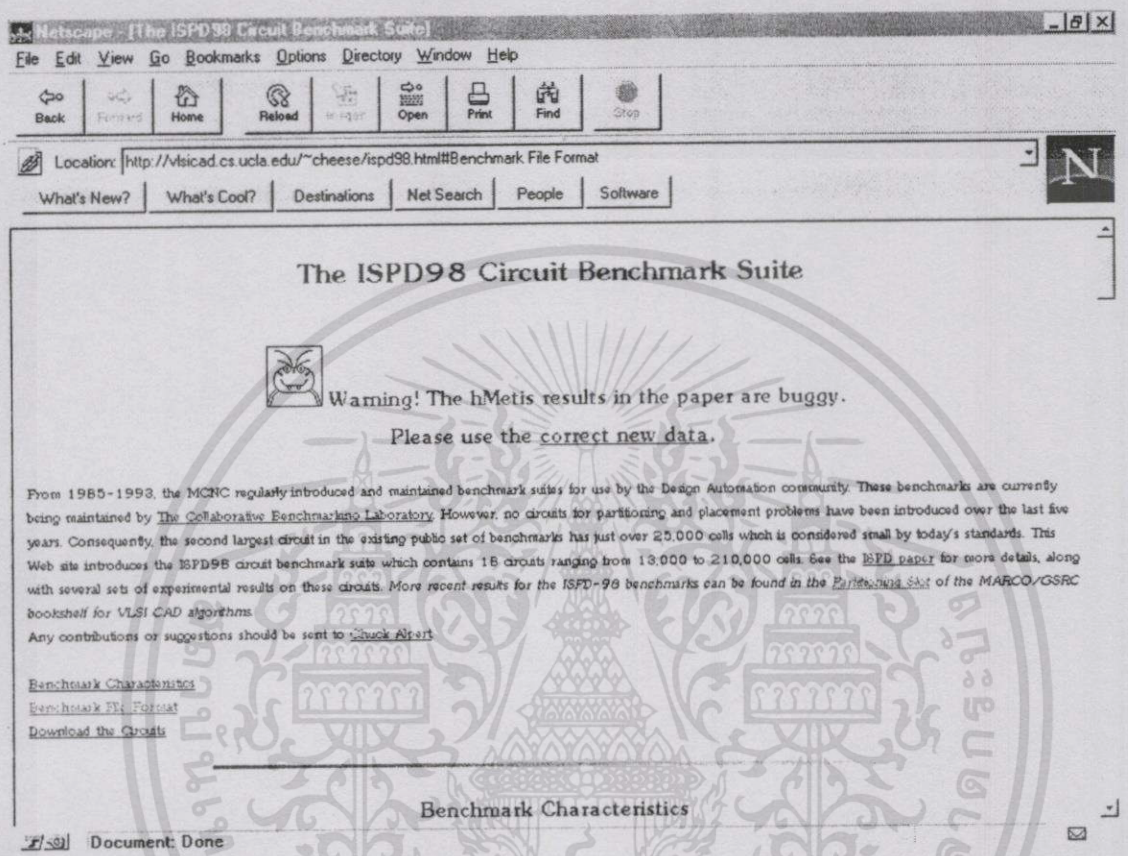
[4] Yen-Chuen Wei, Chung-Kuan Cheng and Member, "Ratio Cut Partitioning for Hierarchical Designs" , IEEE Transactions on computers-aided design , Vol.10, No.7, pp. 911-921, JULY 1991.

[5] Theodore W. Manikas, James T. Cain, " Genetic Algorithms vs. Simulated Annealing: A Comparison of Approaches for Solving the Circuit Partitioning Problem ", Technical Report 96-101, May 1996.

[6] A. Yodtean, S. Choomchuay, Yasoji Suzuki, "An Efficient Algorithm for Circuit Partitioning", ISIC-99 8th International Symposium on Integrated Circuits, Devices & Systems, pp 359-362, 1999.

ภาคผนวก ข.

Benchmark File Format



รูปแสดง web ที่ใช้ download benchmark file

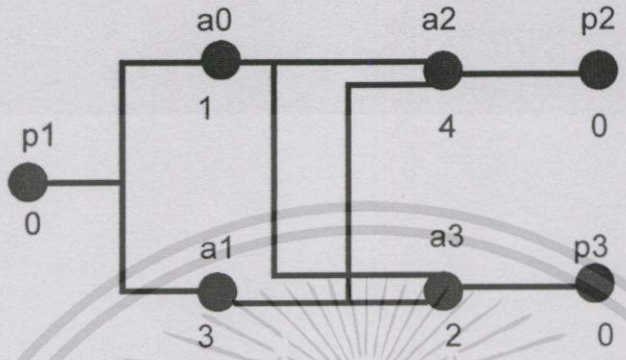
Benchmark file มีด้วยกัน 3 ประเภท คือ a.net , a.are, a.netD สองประเภทแรกเป็นที่รู้จักกันดี ในขณะที่ a.netD เป็นประเภทใหม่ ซึ่งคล้ายกับ .net format แต่มันประกอบด้วย signal direction

1. Netlist ประเภท a.net จะเริ่มต้นด้วย 5 บรรทัดแรกที่บ่งบอกคุณสมบัติของวงจรดังนี้

ignored	
#Pins	คือจำนวนของ pins ที่เกิดขึ้นทั้งหมดของ Circuit Benchmark นั้นๆ
#Nets	คือจำนวนเส้นเชื่อมหรือเส้นแสดงความสัมพันธ์ทั้งหมดของ Circuit Benchmark นั้นๆ
#Modules	คือจำนวนของวงจร cells และ pads ทั้งหมดรวมกัน โดยถูกแทนด้วยกราฟ
#Pad offset	คือจำนวนของ pad ทั้งหมดของวงจรใน Circuit Benchmark นั้นๆ ถูกแทนด้วยกราฟ
#Cells	คือจำนวนของวงจรทั้งหมดให้ Circuit Benchmark

ใน circuit netlist นั้นความหมายของ “a” คือส่วนที่เป็นวงจร ความหมายของ “p” คือส่วนที่เป็น pad การเริ่มต้นความสัมพันธ์ของวงจรหนึ่งต่อวงจรอื่นในแต่ละ nets แสดงด้วย “s”

ตัวอย่าง Netlist file หนึ่งแสดง 4 cells , 3 pads, 5 nets และ 13 pins Netfile นี้กำหนดได้ดังนี้



13

5

7

3

p1 s 1

a0 l

a1 I

a0 s 1

a2 l

a3 l

a1 s l

a2 l

a3 l

a2 s l

p2 l

a3 s l

p3 l

2. Netlist ประเภท a.are จะบอกขนาดของแต่ละ module ซึ่งจะบอกชื่อของ cells หรือ pads ตามด้วยขนาดเนื้อที่ของ cells หรือ pads นั้นๆ ตัวอย่างดังนี้

a0 1

a1 3

a2 4

a3 2

p1 0

p2 0

p3 0

3. Netlist ประเภท a.netD จะคล้ายกับ a.net จะต่างกันตรงที่ไฟล์ประเภทนี้บอกว่า cells ไหนเป็น input ด้วยสัญญาลักษณ์ (I) หรือ output ด้วยสัญญาลักษณ์ (O) หรือเป็นทั้งสองอย่างด้วยสัญญาลักษณ์ (B) สำหรับ nets นั้น I หมายถึงเป็น sink ของ net และถ้าเป็น O คือเป็น source ของ net ตัวอย่างแสดงดังนี้

0

13

4

7

3

p1 s O

a0 1 I

a1 1 I

a0 s O

a2 1 I

a3 1 I

a1 s O

a2 1 I

a3 1 I

a2 s O

p2 1 I

a3 s O

p3 1 I

ตัวอย่าง Benchmark File Format ของวงจร 3k5.net , qgr8.net

3k5.net

0	a6 1	a14 1	
66	a5 s 1	a11 s 1	
33	a7 1	a12 1	
15	a5 s 1	a11 s 1	
0	a8 1	a13 1	
a0 s 1	a5 s 1	a11 s 1	
a1 1	a9 1	a14 1	
a0 s 1	a6 s 1	a12 s 1	
a2 1	a7 1	a13 1	
a0 s 1	a6 s 1	a12 s 1	
a3 1	a8 1	a14 1	
a0 s 1	a6 s 1	a13 s 1	
a4 1	a9 1	a14 1	
a1 s 1	a7 s 1	a4 s 1	
a2 1	a8 1	a9 1	
a1 s 1	a7 s 1	a4 s 1	
a3 1	a9 1	a14 1	
a1 s 1	a8 s 1	a9 s 1	
a4 1	a9 1	a14 1	
a2 s 1	a10 s 1		
a3 1	a11 1		
a2 s 1	a10 s 1		
a4 1	a12 1		
a3 s 1	a10 s 1		
a4 1	a13 1		
a5 s 1	a10 s 1		

qgr8.net

0	a8 1		
46	a0 s 1		
23	a9 1		
12	a3 s 1		
0	a7 1		
a0 s 1	a5 s 1		
a1 1	a6 1		
a0 s 1	a5 s 1		
a2 1	a7 1		
a0 s 1	a5 s 1		
a3 1	a8 1		
a0 s 1	a6 s 1		
a4 1	a7 1		
a1 s 1	a6 s 1		
a2 1	a8 1		
a1 s 1	a7 s 1		
a3 1	a8 1		
a1 s 1	a8 s 1		
a4 1	a9 1		
a2 s 1	a9 s 1		
a3 1	a10 1		
a2 s 1	a9 s 1		
a4 1	a11 1		
a3 s 1	a10 s 1		
a4 1	a1		
a0 s 1			

ภาคผนวก ค.

โปรแกรมการทดลองเขียนบน Borland C Compiler

```
// main program for searce #nets,#pins,#modules
// and create AdjacencyTable of nodes
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <string.h>
#include <math.h>
#include <process.h>
#include <time.h>
/***************define**********************/
#define TR   1
#define MAXNODE 5000   // maximun node
#define TRUE  1
#define FALSE 0
#define NEWLINE 0x0a // Newline cose
#define SPACE 0x20 // Space code
/****************function*********************/
void readfile(void);
void SA(void);
void Display(void);
int  find(char name[50]);
void IniAdjacencyTable(void);
/******************declaration*****************/
char filename[50];
int pins,nets,modules;
int AdjacencyTable[MAXNODE][MAXNODE];//[v][e]
/*EX    Adjacency set          Adjacency table
    vertex   set      edges-->  0 1 2 3
                0    {1,2}          ----------
                1    {2,3}    vertexs--> 0 | F  T  T  F
                2    NULL             1 | F  F  T  T
                3    {0,1,2}          2 | F  F  F  F
                                      3 | T  T  T  F
/*******************************************************/
struct edge{
   char endpoint[30];
   struct edge *nexte;
   };
struct edge *firste[MAXNODE];
char graph[MAXNODE][30];
/********************** main ************************/
void main()
{
   readfile();
   IniAdjacencyTable();
   getche();
   while(TRUE)
   { printf("\n                    MENU \n");
    printf(" [1] Simulated annealing algorithm\n");
    printf(" [e] Exit\n");
    printf(" \n Enter choice ==> ");
    switch(getche())
    { case '1': SA();break;
              case 'e':
              case 'E': exit(0); break;
                    default : break;
     }//end cwitch
     getche();
    } //end while
}
/*******************************************************/
void readfile()
{
   char name[ 50],ch[50];
   int  j,i,r;
   int absA,absB;
   char temp[5][50];
   char *p,*p1, dam ='s';
   char *pp,nodename[30];

   FILE *textfile;
   struct edge *thise[MAXNODE];
   printf("Please enter textfile to read :");
   gets(name);
   strcat(name,".net");
   strcpy(filename,name);
   if((textfile=fopen(filename,"rt"))==NULL)
   {          printf("No file %s \n",filename);
              exit(1); //Send errorcode 1 to Dos
   }
   i=j=0;// number all line
   r=0;
   fgets(ch,50,textfile);
   while(!feof(textfile))
   {
    if(ferror(textfile))
    { printf("file error \n"); exit(1);     }
      //find edage
    if((ch[0]=='a')||(ch[0]=='p'))
    {
          p=strrchr(ch,dam);
          if(p)  //if have "s" in this line
          {  p1=strtok(ch," ");
          strcpy(graph[r],p1);
//        printf("%s \n",graph[r]); //created  all graph
          r=r+1;
          }
          else
          {
          pp = strtok(ch," ");
          if(pp)
                    { strcpy(nodename,pp);
                     thise[r]=(struct edge *)malloc(sizeof(struct edge));
    if(thise[r]==NULL)
    {  printf("Allocation error"); return;  }
    thise[r]->nexte = firste[r];
              firste[r] = thise[r];

                     strcpy(thise[r]->endpoint,nodename);
//                   printf(" edge [%d] is %s \n",r,thise[r]->endpoint);
                    }
```

```
                }
    }
    else //search number pins,nets,modules fo file.net
    { strcpy(temp[j],ch);
                j++;  // j=0,1,2,3,4
    }
    fgets(ch,50,textfile);
    i++;  // searce all lines
  }
  fclose(textfile);
  //display circuit benchmark "file.net" give #pins,#nets,#modules
  pins=atoi(temp[1]);
  nets=atoi(temp[2]);
  modules=atoi(temp[3]);
  printf("\n pins= %d nets = %d modules = %d",pins,nets,modules);
/*  int even=modules%2;
    if(even)
                absA=absB=modules/2;
              else
              {  absA=modules/2;
                 absB=absA+1;
              }*/
}
/*******************************************************/
void Display(void)
{  int r,i,j;
  struct edge *thise[MAXNODE];
                // Graph
  for(i=1;i<=modules;i++)
  //              printf(" Graph[%d] %s\n",i,graph[i-1]);
                // Edge
  //getche();
  for(r=1;r<=modules;r++)
  {  if(firste[r]==NULL)
              { printf("\n Empty list. \n"); return;}
              thise[r] = firste[r];
    do{
  //              printf(" %s",thise[r]->endpoint);
      thise[r] = thise[r]->nexte;
              }while(thise[r] != NULL);
  //              printf("\n");
  }
}
/*******************************************************/
void IniAdjacencyTable()
{  int i,j,r,k;
  int T=0;
  int cutsize;
  char nnode[50];
  struct edge *thise[MAXNODE];
  //set all 0
  printf("\n");
  for(i=1;i<=modules;i++)
  {  for(j=1;j<=modules;j++)
              {
                AdjacencyTable[i][j]=T;
               // printf("  %d", AdjacencyTable[i][j]);
              }
              //printf("\n");
  }
  // set 1 at join position
              for(r=1;r<=modules;r++)
              {  if(firste[r]==NULL)
                 { printf("\n Empty list. \n"); return;  }
                thise[r] = firste[r];
                do{
                  for(i=1;i<=modules;i++)
          {
                   if(strcmp(graph[i-1],thise[r]->endpoint)== 0)
                   {
                                        //              printf("[%d][%d]
",r,i);
                              AdjacencyTable[r][i]=1;
                   }
                  }
                  thise[r] = thise[r]->nexte;
                }while(thise[r] != NULL);
//                printf("\n");
              }
  // To show Adjacency nodes
  for(i=1;i<=modules;i++)
   {  for(j=1;j<=modules;j++)
              {
                printf("  %d", AdjacencyTable[i][j]);
              }
             printf("\n");
   }
}

/*********************************************************************/
/*********************Simulated Annealing Algorithm*******************/
/*********************************************************************/
void  Metropolis(int CGain,int T,int M);
void  randon(void);
int   neighbour();
int   initcost(void);
int   CALcut(int a,char setA[MAXNODE][30]);
int   cutsize[50]; // store cutsize value for plot graph
float Gain(int s);
int To=10000; //initial temperature
int M=18;  //represents the time until the next parameter update
double A=0.9,B=1.2; //alpha is the cooling rate
int So;   //is the initial solution
int Maxtime = 20; //Maxtimes is the total allowed time for the annealing process
int acut; //all cutsize
FILE *stream;
void SA()
//int initcost,int To=1000,double A=0.9,int M=10,int Maxtime=10)
{  int T,S,Time;
  int i;
  double c,r;
  /* open a file for update */
  stream = fopen("SA.FIL", "w+");
  time_t first, second;
  first = time(NULL);  /* Gets system time*/
  T=To;
  S=initcost();
  Time=0;
  do
  {             Metropolis(S,T,M);
                Time = Time+M;
                T = A*T;
                M = B*M;
                fprintf(stream,"\nTime = %d\n",Time);
  }while(Time>=Maxtime);

  fprintf(stream,"\n");
  for(i=0;i<acut;i++)
```

```
                fprintf(stream,"\ncutsize%d = %d",i,cutsize[i]);
  second = time(NULL); /* Gets system time again*/
  fprintf(stream,"\nThe difference is: %.16e seconds\n",difftime(second,first));
  fclose(stream);
  printf(" SA OK! ");
  getche();
}
/*************************Initcost************************/
int initcost()
{  int  cutcost;
  int  even,a=0,e=0,i,j;
  char setA[MAXNODE][30];
  char setB[MAXNODE][30];
  /********* initial bipartition graph *********/
  for(i=1;i<=modules;i++)
  {  even=i%2;
    if(even)
                { strcpy(setA[a],graph[i-1]);        a++;     }
               else
                { strcpy(setB[e],graph[i-1]);        e++;     }
  }
/*  fprintf(stream,"\nInitial partition\n a = %d  A Group : ",a);
  for(i=0;i<a;i++)
                fprintf(stream," %s :",setA[i]);   // group A
  fprintf(stream,"\n b = %d  B Group : ",e);
  for(i=0;i<e;i++)
                fprintf(stream," %s :",setB[i]);   // group B
*/ cutcost=CALcut(a,setA);
  return cutcost;
}
int CALcut(int a,char setA[MAXNODE][30])
/******* calculate cutsize from AdjacencyTable ********/
{  int pi[50],pj[50],k=0,ncut[50],ii=0;;  // defind max=50
  int cutcost;
  int i,j;
//  fprintf(stream,"\n");
  /**********fine i in AdjacencyTable[i][]**********/
  for(i=0;i<a;i++)
  {  for(j=0;j<modules;j++)
               {
                if(strcmp(setA[i],graph[j])==0)
                {  pi[i]=j;
                 //printf(" %s position is %d \n",setA[i],j);
  }              } }
  /**********fine j in AdjacencyTable[][j]**********/
  //printf("\n");
  for(i=0;i<a;i++)
  {  for(j=0;j<modules;j++)
               {  int l=pi[i];
                 if(AdjacencyTable[l+1][j+1]==1)
                  {
                    pj[k]=j;
//                    printf("i= %d j= %d ",l,pj[k]);
//                    printf(" val = %d \n",AdjacencyTable[l+1][(pj[k]+1)]);
                    k++; // all 1 in row of A group
                  }
                }
//               printf("\n");
  }
  for(j=0;j<k;j++)
               { for(i=0;i<a;i++)
                 if(pi[i] == pj[j])
                  { pj[j]=NULL;
                          ii++; // all 1 in column of B group
  } } }
  cutcost = k-ii;
//  printf("(1 of row A group) - (1 of column B group)");
//  printf("\n cutcost = %d \n",cutcost);
  return cutcost;
}
/******************** Metropolis**************************/
void Metropolis(int S,int T,int M)
{  int  NS,i=1,j=0;
  float CGain,NGain,diffcost,e,r;
  CGain = Gain(S);
  cutsize[0]=S;
  for(j=0;j<M;j++)
  { NS = neighbour();
   NGain = Gain(NS);
   diffcost = NGain-CGain;
   e=exp(-diffcost/T);
   r=(rand()%100)*.01;
//   fprintf(stream,"        gain = S/(absA*absB)\n");
//   fprintf(stream,"Newgain = %f  Currentgain = %f\n",NGain,CGain);
//   fprintf(stream,"diffcost= %f  e = %f  r = %f \n",diffcost,e,r);
   if((diffcost<0)||(r<e))
   {  CGain = NGain;
            cutsize[i] = NS;
    i++;
   }
   //accept the soulting
//   fprintf(stream,"  M = %d ",M-j);
   //getche();
  }
  acut=i;
//  fprintf(stream,"\n\nall cutsize = %d\n",i);
}
float Gain(int s)
{  int  x;
  float gain,absA,absB;
  x=modules%2;
  if(x==1)
  {  absA=modules/2;   absB=absA+1;  }
  else
  {          absA=absB=modules/2;       }
//  fprintf(stream,"absA = %f absB = %f cutcost(S) = %d\n",absA,absB,s);
  gain = s/(absA*absB);
  return gain;
}
int neighbour()
{
  int  cost,a,b,c,cc,i;
  char setA[MAXNODE][30]; //node[MAXNODE]:namenode[30]
  char setB[MAXNODE][30];
  unsigned long res,num;
  int  rem,n;
 // max at 2^32
 char
binary[1000]="00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
```

```
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000";
   int  buf[50];  // define only 50
   int  integ=pow(2,modules);
      do{// receive only bisection
            // Ten random numbers from 0 to integ(2^modules)
//          fprintf(stream,"\n Random graph \n");
            num = rand()%integ;
      /* keep the original value */
    res = num;
    n=0;
    while(res != 0)
    {  rem = res % 2;
      res = res / 2;
      if(rem == 1)   binary[50-n] = '1';
      else           binary[50-n] = '0';
      n++;
    }
   for(i=0;binary[i]=='0';i++)
       //NULL
/*     for(;i<51;++i)
    {  printf("%c",binary[i]);    }
            printf("binary \n");
    /******* number of all 1 and 0 *********/
    buf[modules];
    for(i=0;i<modules;i++)
    {  if(binary[50-i]=='1')   buf[modules-i]=1;
     else                  buf[modules-i]=0;
    }
/*   for(i=1;i<=modules;i++)
       printf("%d",buf[i]);
   printf("\n");*/
   a=b=0; //display binary code of num random
   for(i=1;i<=modules;i++)
   {
//          fprintf(stream,"%d",buf[i]);
            if(buf[i]==1)
            {  strcpy(setA[a],graph[i-1]);
              a++;
      }
            else
            {  strcpy(setB[b],graph[i-1]);
           b++;
            }
   }
   c=abs(a-b);
   if((c==0)|(c==1))  cc=TRUE;
   else           cc=FALSE;
   //checking bipartition of graph if c=1 or c=0 is biparition
  }while(cc!=TRUE);

/*   fprintf(stream,"\n a = %d  A Group : ",a);
  for(i=0;i<a;i++)
            fprintf(stream," %s :",setA[i]); // group A
  fprintf(stream,"\n b = %d  B Group : ",b);
  for(i=0;i<b;i++)
            fprintf(stream," %s :",setB[i]); // group B
*/  cost=CALcut(a,setA);
  return cost;
}
```

```
/*****************************************************************/
/************************ Genetic Algorithm **********************/
/*****************************************************************/
void ga(void);
int  CALcut(int a,char setA[MAXNODE][30]);
void Initialize(void);
void Init_Pop(void);
void Init_Fixalleleval(int i);
float Evaluate(float sum,int i);
void Selection(void);
int  pos(void);
int  sort1(void);
int  sort2(void);
int lookup1(int k);
int lookup2(int k,int j);
void Reproduct(int a,int b);
void Genpopu(int a1,int b1);
void mutation(void);
void GenNew(int gen);
void GenNewPop(void);
////////////////////INPUT/////////////////////
int  maxgen=10;  //Maximun generation
int  popsize=10; //Population size not over 2^modules
char Nchrom[MAXNODE][MAXNODE];
int  Cutsize1[10];
int  Cw,Cb;      //Cw is largest cutsize,Cb is smallest cutsize
float Fitness1[10];
float Pselect1[10];
char parent1[50],parent2[50];
int  Newpop[MAXNODE][MAXNODE];
struct GENpop{
     char  Gcode[51];
     int   Gcutsize;
     float Gfitness;
     float Gpselect;
     }Gen[10];
FILE *stream;
/*************************/
/******* GA process ******/
/*************************/
void ga()
{   int gen=0,a,a1,b,b1,i,j;
    /* open a file for update */
   stream = fopen("SGA.FIL", "w+");
   time_t first, second;
   first = time(NULL);  /* Gets system time */
   Nchrom[popsize][modules];
   Initialize();  //initialzation gen 0
   while(gen<10)  //stop (maxgen = 10)
   {          ++gen;

     // selecte parent1,2
            Selection();
            a=sort1(); // resive value most fitness
            a1=lookup1(a);
            b=sort2(); // resive value most fitness-1
//          printf("\na=%d  b=%d",a,b);
            b1=lookup2(b,a1);
//          printf("\nb1=%d  a1=%d\n",b1,a1);
            // create new pop
            Reproduct(a1,b1); // a,b is position of less fitness
```

```
                    Genpopu(a1,b1);
//                  getche();
                    /* display Newpop
      printf("Newpop \n");
                    for(i=0;i<10;i++)
                    {  for(j=0;j<modules;j++)
                           printf("%d",Newpop[i][j]);
        printf("\n");
                    } */
                    GenNew(gen);
                    getche();
    }//end of while
    second = time(NULL); /* Gets system time again*/
    fprintf(stream,"\nThe difference is: %.16e seconds\n",difftime(second,first));
    fclose(stream);
    printf(" SGA OK! ");
    getche();
}
/*************************/
/**** initialization *****/
/*************************/
void Initialize()
{   int i;
    float sumF=0;
    //Initialize population
    Init_Pop();           //define Nchrom value
    //Fix bit value
    for(i=0;i<popsize;i++)
                   Init_Fixalleleval(i);  //define FitnessI value
    for(i=0;i<popsize;i++)
       sumF= sumF+FitnessI[i];//sumation FitnessI value
                      // display Initial  chromosome data

    fprintf(stream,"\n          Generation 0\n");
    fprintf(stream,"chromosome    Ci    Fitness    pselect\n");
    for(i=0;i<popsize;i++)
       PselectI[i]=Evaluate(sumF,i);
    for(i=0;i<popsize;i++)
    { //define struct for generation GA
     strcpy(Gen[i].Gcode,Nchrom[i]);
     Gen[i].Gcutsize = CutsizeI[i];
     Gen[i].Gfitness = FitnessI[i];
     //define Probability of selected Value
     //PselectI[i]=Evaluate(sumF,i);
     Gen[i].Gpselect = PselectI[i];
     fprintf(stream,"%s  %7d   %.6f   %.6f\n",Nchrom[i],
     CutsizeI[i],FitnessI[i],PselectI[i]);
    }
    fprintf(stream,"\n       sum fitness = %f",sumF);
}
/*************** Initialzation GA population ****************/
void Init_Pop()
{   // random pop = initsize and select by bisection
    int a,b,c,cc,Ci,i,j,rem,n,l;
    char setA[MAXNODE][30],setB[MAXNODE][30];
    unsigned long res,num;
    char bit[51];                  // max at 2^32
    char
binary[1000]="0000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000";
    int buf[50]; // define only 50
    int min,max;
    int integ=pow(2,modules);
    for(i=0;i<51;i++)
                   bit[i]='\0';
    for(j=0;j<popsize;j++)
    { // check duplication of binary code! No duplication

      do{// receive only bisection
                   // Ten random numbers from 0 to integ(2^modules)
                   num = rand()%integ;
      res = num;
                   n=0;
                   // convert decimal to binary number
      while(res != 0)
      { rem = res % 2;
        res = res / 2;
        if(rem == 1)   binary[50-n] = '1';
        else           binary[50-n] = '0';
        n++;
      }
      /******* number of all 1 and 0 *********/
                   buf[modules];
      for(i=0;i<modules;i++)
      { if(binary[50-i]=='1')   buf[modules-i]=1;
        else                    buf[modules-i]=0;
                   }
                   a=b=0; //display binary code of num random
                   for(i=1;i<=modules;i++)
                   {   if(buf[i]==1)
                       {   strcpy(setA[a],graph[i-1]);
                           a++;
                       }
                       else
                       {   strcpy(setB[b],graph[i-1]);
                                b++;
                       }
                   }
      //check bipartition of graph if (c=1||c=0) is biparition
      c=abs(a-b);
                   if((c==0)||(c==1))  cc=TRUE;
                   else            cc=FALSE;
    }while(cc!=TRUE);
//  getche();
/*  printf("\n a = %d  A Group : ",a);
    for(i=0;i<a;i++)
    printf(" %s :",setA[i]); // group A
    printf("\n b = %d  B Group : ",b);
    for(i=0;i<b;i++)
    printf(" %s :",setB[i]); // group B*/
    // find each Cutsize value in all chromosome
      Ci=CALcut(a,setA);
      CutsizeI[j]=Ci;
/*  printf(" C[%d]=  %d  code is ",j,CutsizeI[j]);
    for(i=1;i<=modules;i++)
                   printf("%d",buf[i]);
```

```
*/  // convert binary number to decimal
      for(i=1;i<=modules;i++)
      {  if(buf[i]==1)   bit[i-1]='1';
         else            bit[i-1]='0';
      }
//   printf("bit string is %s\n",bit);
     // define bit string of chromosome
      strcpy(Nchrom[j],bit);
    }//end of for 10 loop

    // find largest (Cw) and smallest(Cb) cutsize value
    min=Cutsize1[0];
    for(i=1;i<popsize;i++)
    {  if(min>Cutsize1[i])
         min=Cutsize1[i];
    }
    Cb=min; //   printf("\n min= %d",min);
    max=Cutsize1[0];
    for(i=1;i<popsize;i++)
    {  if(max<Cutsize1[i])
         max=Cutsize1[i];
    }
    Cw=max; //   printf("\n max= %d",max);
/*  printf("\n");
    for(i=0;i<popsize;i++)
      printf("\nchromosome[%d] is %s",i,Nchrom[i]);   */
}
int CALcut(int a,char setA[MAXNODE][30])
/******* calculate cutsize from AdjacencyTable ********/
{  int pi[50],pj[50],k=0,ncut[50],ii=0;  // defind max=50
   int cutcost;
   int i,j;
   /**********fine i in AdjacencyTable[i][]**********/
   for(i=0;i<a;i++)
   {  for(j=0;j<modules;j++)
                {
                 if(strcmp(setA[i],graph[j])==0)
                 {  pi[i]=j;
//          printf(" %s position is %d \n",setA[i],j);
   }             } }
   /********** fine j in AdjacencyTable[][j] **********/
//  printf("\n");
   for(i=0;i<a;i++)
   {  for(j=0;j<modules;j++)
              {  int l=pi[i];
                 if(AdjacencyTable[l+1][j+1]==1)
                 {
                   pj[k]=j;
//                 printf("i= %d j= %d ",l,pj[k]);
//                 printf(" val = %d \n",AdjacencyTable[l+1][(pj[k]+1)]);
                   k++; // all 1 in row of A group
                 }
              }
//  printf("\n");
   }
   for(j=0;j<k;j++)
   { for(i=0;i<a;i++)
     {  if(pi[i] == pj[j])
              {  pj[j]=NULL;
                 ii++; // all 1 in column of B group
   } } }
   cutcost = k-ii;
//  printf("(1 of row A group) - (1 of column B group)");
//  printf("\n cutcost = %d \n",cutcost);
   return cutcost;
}
void Init_Fixalleleval(int i)
{   float F[10];
      F[i]=(Cw-Cutsize1[i])+((Cw-Cb)/3);
    Fitness1[i]=F[i];
 //   printf("\n fitness %d = %f",i+1,Fitness1[i]);
}
float Evaluate(float sum,int i)
{   float r,S,Ev;   // P=10 is all chromosome
    int p=10;      // r=pow(4,(1/p-1))
    r=pow(4,0.111111111);
    S=(1-pow(r,p))/(1-r);
    Ev=pow(r,i-1)/S;
//    Ev=Fitness1[i]/sum;
    return  Ev;
}
/*********************************/
/******** Next Generation ******/
/*********************************/
void Selection()
{  // only position parent
   int i,j,k1,k2;
   float temp,smax1,smax2;
   // find position dad form maximun fitness
   smax1 = Gen[0].Gfitness;
   for(i=1;i<popsize;i++)
   {  if(smax1<Gen[i].Gfitness)
      {   smax1=Gen[i].Gfitness;
          k1=i;
      }
   }
   // find position mom form random
   k2 = rand()%10;
//  printf("position of dad = %d \nposition of mom = %d",k1,k2);
   strcpy(parent1,Gen[k1].Gcode);
   strcpy(parent2,Gen[k2].Gcode);
//   printf("dad : %s   mom : %s \n",parent1,parent2);
}
int  pos()
{   int pos;
    pos = rand()%modules;
//   printf("find position in one chromosome = %d\n",pos);
    return pos;
}
//********** sorting ********//
int int_comp( const void *i, const void *j);
int int_comp( const void *i, const void *j)
{
  return((*(int *)i)-(*(int *)j) );
}
int sort1()
{  int i;
   qsort(Cutsize1, 10, sizeof(int), int_comp);
/*  printf("The array after qsort() function :\n\n");
   for (i=0;i<10;i++)
              printf("%d  ", Cutsize1[i]);
*/   return Cutsize1[9];
}
int sort2()
{  int i;
   qsort(Cutsize1, 10, sizeof(int), int_comp);
   return Cutsize1[8];
}
```

```
//**********searching*********//
int lookup1(int k)
{  int i,r;
   for(i=0;i<10;i++)
   {  if(Gen[i].Gcutsize==k)
                r= i; // return position of maximun fitness
   }
   return r;
}
int lookup2(int k,int j)//k is fitness,j is position of max fitness
{  int i;
   for(i=0;i<10;i++)
   { if(i!=j)
    {  if(Gen[i].Gcutsize==k)
                return i;
} } }
// Crossover and mutation process
void Reproduct(int a,int b)
{   char *ptr1,*ptr2;
   char part1[25],part2[25];
   int i,posi,var;
   int offs1[25],offs2[25];
   int one1=0,zero1=0,one2=0,zero2=0;
   int deta1,deta2,m,e,num,r1,r2;
   posi = pos(); // random position of crossover
   m=modules/2;  // m is chromosome/2
   e=modules%2;
   if(e==1)
     m=m+1;
//  printf("m= %d",m);
   /********** Crossover **********/
   /********** create offstring1 **********/
   for(i=0;i<=posi;i++)
   {  if(parent1[i]=='1') offs1[i]=1;
              else        offs1[i]=0;
   }
   for(i=posi+1;i<modules;i++)
   {  if(parent2[i]=='1') offs1[i]=1;
     else        offs1[i]=0;
   }
/*   printf("off1 = ");
   for(i=0;i<modules;i++)
              printf("%d",offs1[i]);*/
   /********** create offstring2 **********/
   for(i=0;i<=posi;i++)
   {  if(parent1[i]=='1') offs2[i]=1;
              else        offs2[i]=0;
   }
   for(i=posi+1;i<modules;i++)
   {  if(parent2[i]=='1') offs2[i]=0;
              else        offs2[i]=1;
   }
/*   printf("\noff2 = ");
   for(i=0;i<modules;i++)
              printf("%d",offs2[i]);*/
   /***** mutation offstring1,2 in nonbisection case *****/
   /****** count 1 and 0  in offstring 1 for mutate ******/
   for(i=0;i<modules;i++)
   {  if(offs1[i]==1)  one1++;
     else        zero1++;
   }
   deta1=abs(one1-zero1);
//   printf("\n deta1 = %d\n",deta1);
   if(one1!=zero1)
   if(deta1!=1)
   { if(one1>zero1)
          {  num=one1-m;  //num is number mutation
           i=0;
           do{   //do random 1 value for mutate
             r1=rand()%modules; // case num 1 > num 0
                  if(offs1[r1]==1)  // mutate bit 1 is 0
                  {  offs1[r1]=0;
       i++;
      }
          }while(i<num);
         }
         else
         {  num=zero1-m;
           i=0;
           do{
             r1=rand()%modules; // case num 1 < num 0
                  if(offs1[r1]==0)  // mutate bit 0 is 1
                  {  offs1[r1]=1;
        i++;
      }
          }while(i<num);
     }
   }
//   printf("\n %d : %d",one1,zero1);
//   printf("\n mutation off1 = ");
   for(i=0;i<modules;i++)
   {// printf("%d",offs1[i]);
          Newpop[a][i]=offs1[i]; //replace in Newpop[a][] position
   }
   /****** count 1 and 0 in offstring 2 for mutate ******/
   for(i=0;i<modules;i++)
   {  if(offs2[i]==1)   one2++;
     else        zero2++;
   }
   deta2=abs(one2-zero2);
   if(one2!=zero2)
    if(deta2!=1)
    {  if(one2>zero2)
          {  num=one2-m;  //num is number mutation
           i=0;
           do{ //do random 1 value for mutate
             r2=rand()%modules; // case num 1 > num 0
                  if(offs2[r2]==1)  // mutate bit 1 is 0
                  {  offs2[r2]=0;
       i++;
      }
          }while(i<num);
         }
         else
         {  num=zero2-m;
           i=0;
           do{
             r2=rand()%modules; //case num 0 < num 1
                  if(offs2[r2]==0)  //mutate bit 0 is 1
                  {  offs2[r2]=1;
        i++;
      }
          }while(i<num);
     }
   }
//   printf("\n %d : %d",one2,zero2);
//   printf("\n mutation off2 = ");
  for(i=0;i<modules;i++)
```

```
  {//printf("%d",offs2[i]);
    Newpop[b][i]=offs1[i]; //replace in Newpop[b][] position
  }
}
void Genpopu(int a1,int b1)
{  int  i,j,tempi[50];
  char tempc[50];

  for(i=0;i<10;i++) //check position of pop
  {              if((i!=a1)&&(i!=b1))
                 {  strcpy(tempc,Gen[i].Gcode);
                   for(j=0;j<modules;j++)
                   {  if(tempc[j]=='1')  Newpop[i][j]=1;
        else          Newpop[i][j]=0;
} }              } }
void GenNew(int gen)
{  int  i,j;
  float sumF=0;
  char Newp[50][50];
/*   for(j=0;j<popsize;j++)
     for(i=0;i<modules;i++)
       Newp[j][i]=NULL;       */
  GenNewPop();
//   getche();
  //Fix bit value
  for(i=0;i<popsize;i++)
    Init_Fixalleleval(i);  //define Fitnessl value
//   getche();
  for(i=0;i<popsize;i++)
    sumF=sumF+Fitnessl[i];//sumation Fitnessl value
//   getche();
       // display Initial  chromosome data
  fprintf(stream,"\n          Generation %d \n",gen);
  fprintf(stream,"chromosome    Ci    Fitness    pselect\n");
  for(i=0;i<popsize;i++)
    Pselectl[i]=Evaluate(sumF,i);
  // change int to char
  for(j=0;j<popsize;j++)
  { for(i=0;i<modules;i++)
    {  if(Newpop[j][i]==1)  Newp[j][i]='1';
                  else            Newp[j][i]='0';
    }
  }
  for(i=0;i<popsize;i++)
  { //define struct for generation GA
    strcpy(Gen[i].Gcode,Newp[i]);
    Gen[i].Gcutsize = Cutsizel[i];
    Gen[i].Gfitness = Fitnessl[i];
    //define Probability of selected Value
    //Pselectl[i]=Evaluate(sumF,i);
    Gen[i].Gpselect = Pselectl[i];
    fprintf(stream,"%s  %7d   %.6f  %.6f\n",Newp[i],
    Gen[i].Gcutsize,Gen[i].Gfitness,Gen[i].Gpselect);
  }
  fprintf(stream,"\n     sum fitness = %f",sumF);
}
void GenNewPop()
{
  int  a,b,c,Ci,i,j,rem,n,l;
  char setA[MAXNODE][30],setB[MAXNODE][30];
  char bit[51];                // max at 2^32
  int  buf[50]; // define only 50
  int  min,max;
  // set bit value
  for(i=0;i<51;i++)
    bit[i]='\0';
  for(j=0;j<popsize;j++)
  { /******* number of all 1 and 0 ********/
    a=b=0; //display binary code of num random
    for(i=0;i<modules;i++)
    {  if(Newpop[j][i]==1)
                 {  strcpy(setA[a],graph[i]);
                   a++;
      }
                 else
                 {  strcpy(setB[b],graph[i]);
                   b++;
                 }
    }
/*    getche();
    printf("\n a = %d  A Group : ",a);
    for(i=0;i<a;i++)
    printf(" %s :",setA[i]); // group A
    printf("\n b = %d  B Group : ",b);
    for(i=0;i<b;i++)
    printf(" %s :",setB[i]); // group B
*/    // find each Cutsize value in all chromosome
    Ci=CALcut(a,setA);
    Cutsizel[j]=Ci;
/*    printf(" C[%d]=  %d  code is ",j,Cutsizel[j]);
    for(i=0;i<modules;i++)
                 printf("%d",Newpop[j][i]);
*/                 // convert binary number to decimal
  }//end of for 10 loop
   // find largest (Cw) and smallest(Cb) cutsize value
  min=Cutsizel[0];
  for(i=1;i<popsize;i++)
  {  if(min>Cutsizel[i])
                 min=Cutsizel[i];
  }
  Cb=min;  // printf("\n min= %d",min);
  max=Cutsizel[0];
  for(i=1;i<popsize;i++)
  {  if(max<Cutsizel[i])
                 max=Cutsizel[i];
  }
  Cw=max;  // printf("\n max= %d",max);
}

/*********************************************************************/
/*************************Hybrid Algorithm*************************/
/*********************************************************************/
void Metropolis(int CGain,int T,int M,int k);
void randon(void);
int  neighbour(int k);
int  initcost(void);
int  CALcut(int a,char setA[MAXNODE][30]);
int  cutsize[50]; // store cutsize value for plot graph
float Gain(int s);
int To=1000; //initial temperature
int M=18;  //represents the time until the next parameter update
double A=0.9; //alpha is the cooling rate
int Maxtime = 20; //Maxtimes is the total allowed time for the annealing process
int acut; //all cutsize
////////////////////////////////////////////
void Initialize(void);
void Init_Pop(void);
void Init_Fixalleleval(int i);
```

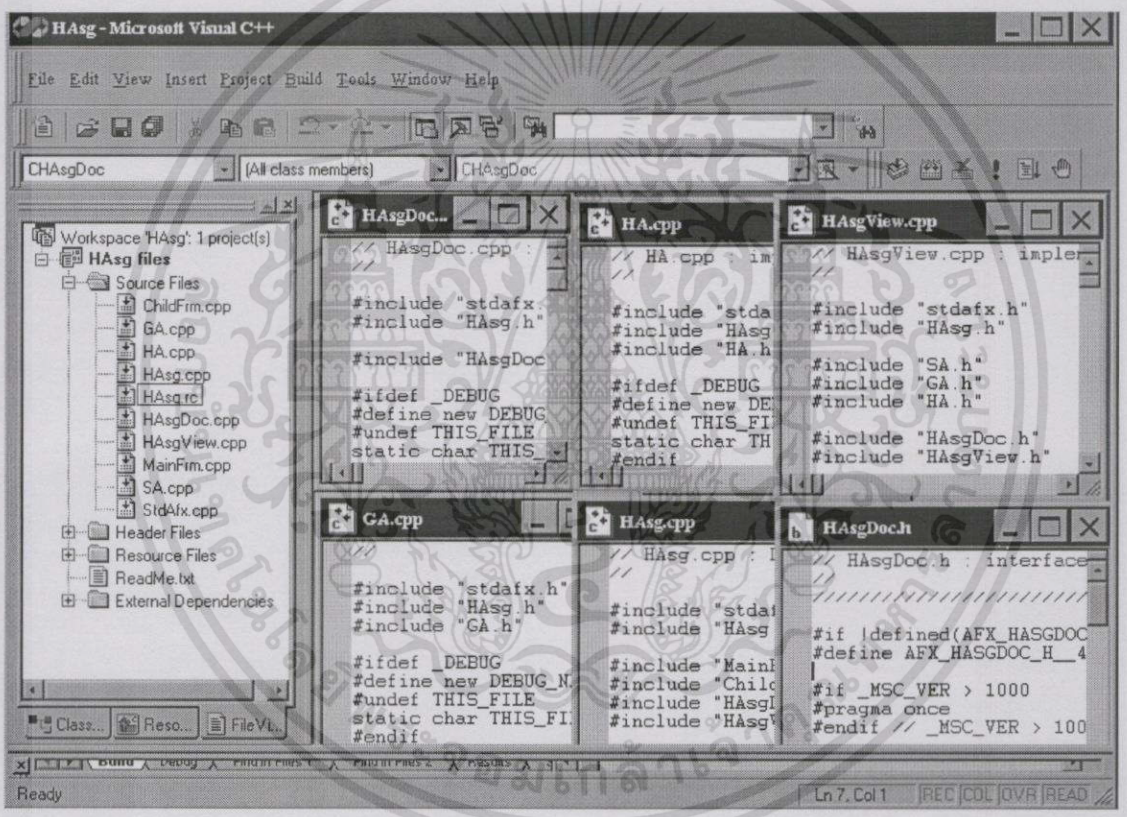
```
float Evaluate(float sum,int i);
int  Selection(void);
int  pos(void);
int  popsize=10; //Population size not over 2^modules
char Nchrom[MAXNODE][MAXNODE];
int  Cutsizel[10];
int  Cw,Cb;      //Cw is largest cutsize,Cb is smallest cutsize
float Fitnessl[10];
float Pselectl[10];
struct GENpop{
    char  Gcode[51];
    int   Gcutsize;
    float Gfitness;
    float Gpselect;
    }Gen[10];
FILE *stream;
/*******************************************************/
void ha()
//int initcost,int To=1000,double A=0.9,int M=10,int Maxtime=20)
{  int T,S,Time;
   int i,k1;
   double e,r;
   time_t first, second;
   stream = fopen("ha.FIL", "w+");
   first = time(NULL);  /* Gets system time*/
   T=To;
   Initialize();    // initialzation gen 0
   k1=Selection();  // position max fitness
   fprintf(stream,"\nmax fitness: %8 ,cutsize: %d\n",Gen[k1].Gcode,Gen[k1].Gcutsize);
   S=Gen[k1].Gcutsize;
   Time=0;
   do
   {            Metropolis(S,T,M,k1);
                Time = Time+M;
                T = A*T;
                fprintf(stream,"\nTime = %d\n",Time);
   }while(Time>=Maxtime);

   second = time(NULL); /* Gets system time again*/
   fprintf(stream,"\n");
   for(i=0;i<acut;i++)
            fprintf(stream,"\ncutsize%d = %d",i,cutsize[i]);

   fprintf(stream,"\nThe difference is: %.16e seconds\n",difftime(second,first));
   fclose(stream);
   printf(" HA OK! ");
   getche();
}
/*************************Initcost*************************/
/************************/
/**** initialization *****/
/************************/
void Initialize()
{   int i;
    float sumF=0;
    //Initialize population
    Init_Pop();          //define Nchrom value
    //Fix bit value
    for(i=0;i<popsize;i++)
            Init_Fixalleleval(i);  //define Fitnessl value
    for(i=0;i<popsize;i++)
      sumF= sumF+Fitnessl[i];//sumation Fitnessl value
            // display Initial  chromosome data
    fprintf(stream,"\n        Generation 0\n");
    fprintf(stream,"chromosome   Ci   Fitness   pselect\n");
    for(i=0;i<popsize;i++)
      Pselectl[i]=Evaluate(sumF,i);

    for(i=0;i<popsize;i++)
    { //define struct for generation GA
     strcpy(Gen[i].Gcode,Nchrom[i]);
     Gen[i].Gcutsize = Cutsizel[i];
     Gen[i].Gfitness = Fitnessl[i];
     //define Probability of selected Value
     Pselectl[i]=Evaluate(sumF,i);
     Gen[i].Gpselect = Pselectl[i];
     fprintf(stream,"%s  %7d   %.6f  %.6f\n",Gen[i].Gcode,
     Gen[i].Gcutsize,Gen[i].Gfitness,Gen[i].Gpselect);
    }
    fprintf(stream,"\n     sum fitness = %f",sumF);
}
/************** Initialzation GA population ****************/
void Init_Pop()
{   // random pop = initsize and select by bisection
    int  a,b,c,cc,Ci,i,j,rem,n,l;
    char setA[MAXNODE][30],setB[MAXNODE][30];
    unsigned long res,num;
    char bit[51];                // max at 2^32
    char
binary[1000]="00000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
00000";
    int  buf[50]; // define only 50
    int  min,max;
    int  integ=pow(2,modules);
    for(i=0;i<51;i++)
                bit[i]='\0';
    for(j=0;j<popsize;j++)
    { // check duplication of binary code! No duplication

      do{// receive only bisection
            // Ten random numbers from 0 to integ(2^modules)
            num = rand()%integ;
      res = num;
            n=0;
            // convert decimal to binary number
      while(res != 0)
      { rem = res % 2;
       res = res / 2;
       if(rem == 1)   binary[50-n] = '1';
       else           binary[50-n] = '0';
       n++;
      }
      /******* number of all 1 and 0 *********/
            buf[modules];
      for(i=0;i<modules;i++)
      { if(binary[50-i]=='1')   buf[modules-i]=1;
       else              buf[modules-i]=0;
            }
            a=b=0; //display binary code of num random
            for(i=1;i<=modules;i++)
            {   if(buf[i]==1)
```

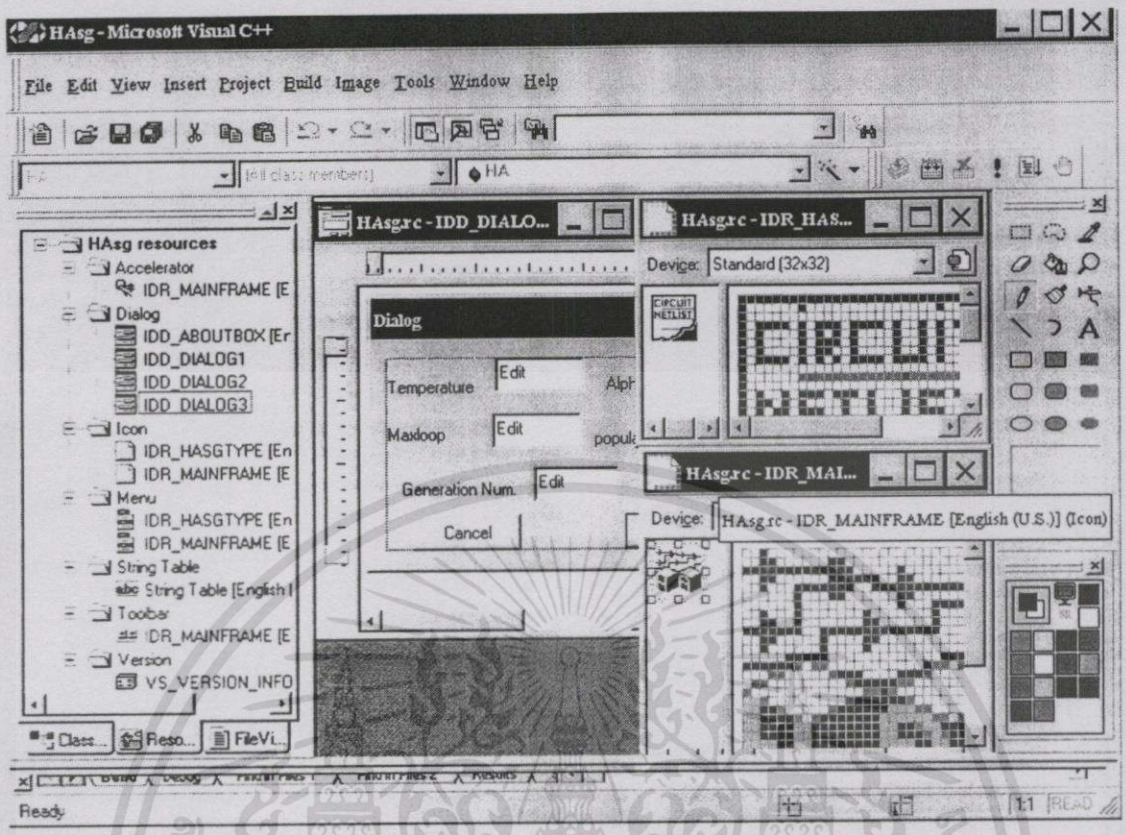
```
                {   strcpy(setA[a],graph[i-1]);
                    a++;
            }
                else
                {   strcpy(setB[b],graph[i-1]);
                        b++;
                }
              }
        //check bipartition of graph if (c=1||c=0) is biparition
        c=abs(a-b);
                if((c==0)||(c==1))  cc=TRUE;
                else            cc=FALSE;
      }while(cc!=TRUE);
//  getche();
/*  printf("\n a = %d  A Group : ",a);
    for(i=0;i<a;i++)
    printf(" %s :",setA[i]); // group A
    printf("\n b = %d  B Group : ",b);
    for(i=0;i<b;i++)
    printf(" %s :",setB[i]); // group B*/
    // find each Cutsize value in all chromosome
      Ci=CALcut(a,setA);
      CutsizeI[j]=Ci;
/*  printf(" C[%d]=  %d  code is ",j,CutsizeI[j]);
    for(i=1;i<=modules;i++)
                printf("%d",buf[i]);
*/  // convert binary number to decimal
      for(i=1;i<=modules;i++)
      {  if(buf[i]==1)   bit[i-1]='1';
         else            bit[i-1]='0';
      }
//  printf("bit string is %s\n",bit);
    // define bit string of chromosome
      strcpy(Nchrom[j],bit);
    }//end of for 10 loop
     // find largest (Cw) and smallest(Cb) cutsize value
    min=CutsizeI[0];
    for(i=1;i<popsize;i++)
    {  if(min>CutsizeI[i])
        min=CutsizeI[i];
    }
    Cb=min; //  printf("\n min= %d",min);
    max=CutsizeI[0];
    for(i=1;i<popsize;i++)
    {  if(max<CutsizeI[i])
        max=CutsizeI[i];
    }
    Cw=max; //  printf("\n max= %d",max);
}
void Init_Fixalleleval(int i)
{   float F[10];

    F[i]=(Cw-CutsizeI[i])+((Cw-Cb)/3);
    FitnessI[i]=F[i];
 //  printf("\n fitness %d = %f",i+1,FitnessI[i]);
}
float Evaluate(float sum,int i)
{   float r,S,Ev;  // P=10 is all chromosome
    int p=10;      // r=pow(4,(1/p-1))
    r=pow(4,0.111111111);
    S=(1-pow(r,p))/(1-r);
    Ev=pow(r,i-1)/S;
//    Ev=FitnessI[i]/sum;
    return  Ev;
}
int CALcut(int a,char setA[MAXNODE][30])
/******* calculate cutsize from AdjacencyTable ********/
{  int pi[50],pj[50],k=0,ncut[50],ii=0;;  // defind max=50
   int cutcost;
   int i,j;
   fprintf(stream,"\n");
   /**********fine i in AdjacencyTable[i][]**********/
   for(i=0;i<a;i++)
   {  for(j=0;j<modules;j++)
                {
                 if(strcmp(setA[i],graph[j])==0)
                 {  pi[i]=j;
                  //printf(" %s position is %d \n",setA[i],j);
   }             } }
   /**********fine j in AdjacencyTable[][j]**********/
   //printf("\n");
   for(i=0;i<a;i++)
   {  for(j=0;j<modules;j++)
                {  int l=pi[i];
                 if(AdjacencyTable[l+1][j+1]==1)
                 {
                  pj[k]=j;
//                printf("i= %d j= %d ",l,pj[k]);
//                printf(" val = %d \n",AdjacencyTable[l+1][(pj[k]+1)]);
                  k++; // all 1 in row of A group
                 }
                }
//              printf("\n");
   }
   for(j=0;j<k;j++)
                { for(i=0;i<a;i++)
                  {  if(pi[i] == pj[j])
                     {  pj[j]=NULL;
                                ii++; // all 1 in column of B group
                     } } }
   cutcost = k-ii;
//  printf("(1 of row A group) - (1 of column B group)");
//  printf("\n cutcost = %d \n",cutcost);
   return cutcost;
}
//*****************************************//
//******** Select one chromosome *********//
//*****************************************//
int Selection()
{  // only position parent
   int i,k1;
   float smax1;
   // find position dad form maximun fitness
   smax1 = Gen[0].Gfitness;
   for(i=1;i<popsize;i++)
   {  if(smax1<Gen[i].Gfitness)
     {  smax1=Gen[i].Gfitness;
        k1=i;
     }
   }
//   strcpy(parent1,Gen[k1].Gcode);
   return k1;// return position value of max fitness
}

//**********************//
//***** Metropolis *****//
//**********************//
void Metropolis(int S,int T,int M,int k)//k is position of max fitness
```

```
{   int  NS,i=1,j=0;
    float CGain,NGain,diffcost,e,r;
    CGain = Gain(S);// S is cutsize
    cutsize[0]=S;
    for(j=0;j<M;j++)
    {
     NS=neighbour(k);
     NGain = Gain(NS);
     diffcost = NGain-CGain;
     e=exp(-diffcost/T);
     r=(rand()%100)*.01;
//   printf("       gain = S/(absA*absB)\n");
//   printf("Newgain = %f  Currentgain = %f\n",NGain,CGain);
//   printf("diffcost= %f  e = %f  r = %f \n",diffcost,e,r);
     if((diffcost<0)||(r<e))
     {  CGain = NGain;
                cutsize[i] = NS;
       i++;
     }
     //accept the soulting
     fprintf(stream,"  M = %d ",M-j);
     //getche();
    }
    acut=i;
    fprintf(stream,"\n\nall cutsize = %d\n",i);
}
float Gain(int s)
{   int  x;
    float gain,absA,absB;
    x=modules%2;
    if(x==1)
    {  absA=modules/2;   absB=absA+1;  }
    else
    {              absA=absB=modules/2;       }
    fprintf(stream,"absA = %f absB = %f cutcost(S) = %d\n",absA,absB,s);
    gain = s/(absA*absB);
    return gain;
}
int  pos()
{   int pos;
    pos = rand()%modules;
//  printf("find position in one chromosome = %d\n",pos);
    return pos;
}
int neighbour(int k)//k is position of max fitness
{
    int  cost,a,b,c,cc,i;
    char setA[MAXNODE][30]; //node[MAXNODE]:namenode[30]
    char setB[MAXNODE][30];
    unsigned long res,num;
    int  buf[50];
    int  p0,p1,T,even;  // position of zero and one in chromosome
    char temp[51];
    strcpy(temp,Gen[k].Gcode);
    // fix the first gen and mutate
    // mutate 0 to 1 and 1 to 0 each one time
    T=0;
    do{ p0=pos();
                fprintf(stream,"\n0 random at position = %d",p0);
                if(temp[p0]=='0')
                {  if(p0!=0)
                  {  temp[p0]='1';
                            fprintf(stream,"\nmutate is %c",temp[p0]);
                            T=TRUE;
                  }
                }
    }while(T!=1);
    T=0;
    do{ p0=pos();
                fprintf(stream,"\n1 random at position = %d",p0);
                if(temp[p0]=='1')
                {  if(p0!=0)
                  {  temp[p0]='0';
                     fprintf(stream,"\nmutate is %c",temp[p0]);
                            T=TRUE;
                  }
                }
    }while(T!=1);
    fprintf(stream,"\n neighbour2 : %s \n",temp);
    a=b=0;
    for(i=0;i<modules;i++)
    {  //fprintf(stream,"%c",temp[i]);
      if(temp[i]=='1')
                a++;
      else
       b++;
    }
    fprintf(stream,"\na = %d , b = %d\n",a,b);
    c=abs(a-b);
    if(c==1) // if diff do
    {  p1=rand()%2; // if random is 0 do
                if(p1==0)
      {
       if(a>b)// num1 > num0
                { T=0;
        do{ p0=pos();
                            fprintf(stream,"\nposition random 1= %d",p0);
                  if(temp[p0]=='1')
                  {  if(p0!=0)
                    {  temp[p0]='0';
                       fprintf(stream,"\nmutate is %c",temp[p0]);
                                T=TRUE;
                    }
                  }
                 }while(T!=1);
       }
                  else  // num0 > num1
                  {  T=0;
                do{ p0=pos();
                    fprintf(stream,"\nposition random 0= %d",p0);
                    if(temp[p0]=='0')
                    {  if(p0!=0)
                      {  temp[p0]='1';
                                fprintf(stream,"\nmutate is %c",temp[p0]);
                                T=TRUE;
                      }
                    }
                   }while(T!=1);
      }
                }
    }
    fprintf(stream,"\n neighbour3 : %s \n",temp);
    // change char to int of chromosome for cal cutsize
    buf[modules];
    for(i=0;i<modules;i++)
    {  if(temp[i]=='1')   buf[i]=1;
                else          buf[i]=0;
    }
```

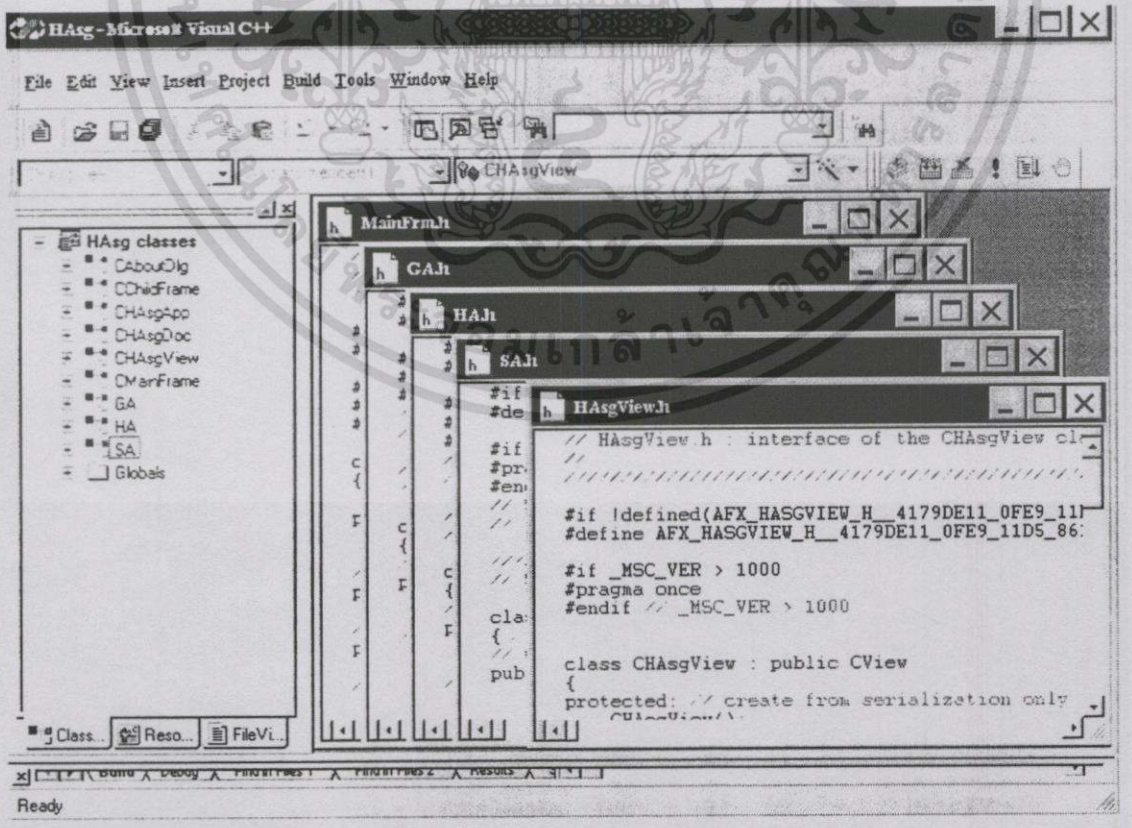
```
a=b=0;
for(i=0;i<modules;i++)
{ fprintf(stream,"%d",buf[i]);
 if(buf[i]==1)
        { strcpy(setA[a],graph[i]);
          a++;
 }
        else
        { strcpy(setB[b],graph[i]);
   b++;
        }
 }
/*  printf("\n a = %d  A Group : ",a);
   for(i=0;i<a;i++)
            printf(" %s :",setA[i]); // group A
   printf("\n b = %d  B Group : ",b);
   for(i=0;i<b;i++)
            printf(" %s :",setB[i]); // group B
*/  cost=CALcut(a,setA);
   return cost;
}
```

โปรแกรมภาพการจัดกลุ่มวงจรที่เขียนบน VISUAL C++ v.6 ในส่วนของ FILEVIEW





ในส่วนของ Resource



ในส่วนของ CLASS

ประวัติผู้เขียน

ชื่อผู้เขียน นางสาวอภิรดี ยอดเทียน

วันเดือนปีเกิด วันที่ 22 มกราคม 2512

สถานที่เกิด กรุงเทพมหานคร

การศึกษาปริญญาตรี วิทยาศาสตร์บัณฑิตสาขาวิชาสถิติ จากมหาวิทยาลัยศรีนครินทรวิโรฒ สงขลา ในปี 2536

ผลงานด้านวิชาการ

1. A. Yodtean, S. Choomchuay "An Efficient Algorithm for Circuit Partitioning" ISIC-99 8th International Symposium on Integrated Circuits, Devices & Systems, Singapore, Setemper 1999, pp. 359-362.
2. A. Yodtean. S. Choomchuay and Yasoji Suzuki "An Improved Simulated Annealing Algorithm for Circuit Partitioning" 1999 IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS'99), Puket Thailand, December 1999, pp. 473-476.

ความสนใจปัจจุบัน มีความสนใจงานวิจัยทางด้าน IC Physical Design Automation , IC Layout Design ทำงานในตำแหน่งผู้ช่วยนักวิจัยอยู่ที่ฝ่ายออกแบบวงจรรวม ศูนย์เทคโนโลยีไมโครอิเล็กทรอนิกส์ ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) มีหน้าที่รับผิดชอบในการออกแบบวงจรรวมทางกายภาพวงจรรวม (IC Layout Design with Tanner tools)