บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ขั้นตอนวิธีเชิงพันธุกรรม (Genetic algorithm)

ขั้นตอนวิธีเชิงพันธุกรรม (Genetic Algorithm หรือ GA) เป็นวิธีการในการหาคำตอบที่อยู่บนพื้นฐานของกระบวนการวิวัฒนาการที่เกิดขึ้นในธรรมชาติ จุดเริ่มต้นของการศึกษาวิธีการในการหาคำตอบที่ใช้หลักการตามธรรมชาตินี้เริ่มต้นจากหนังสือ [4] การค้นหาคำตอบของขั้นตอนวิธีเชิงพันธุกรรมอาศัยกลุ่มของประชากร โดยประชากรแต่ละตัวจะประกอบไปด้วยโครโมโซม (chromosome) โครโมโซมมักอยู่ในรูปของสายอักขระฐานสอง (binary string) หรือสายอักขระ (string) ซึ่งการแทนรหัสของโครโมโซมจะเป็นลักษณะเดียวกับคำตอบของปัญหา การทำงานของขั้นตอนวิธีเชิงพันธุกรรมทำให้เกิดการแข่งขันกันระหว่างประชากร ประชากรที่มีคุณภาพดีกว่าก็มีโอกาสสูงกว่าที่จะอยู่รอด และมีการวิวัฒนาการไปเป็นคำตอบที่ต้องการ

การทำงานของขั้นตอนวิธีเชิงพันธุกรรมจะเริ่มต้นด้วยการสร้างกลุ่มประชากรเริ่มต้นด้วยการสุ่ม จากนั้นประชากรแต่ละตัวจะผ่านการหาค่าความเหมาะสม (fitness value) โดยใช้ฟังก์ชันหาค่าความเหมาะสม (fitness function) ซึ่งค่าความเหมาะสมนี้จะแสดงว่าประชากรแต่ละตัวมีความใกล้เคียงกับคำตอบที่ต้องการเพียงใด หลังจากที่ประชากรแต่ละตัวได้รับการกำหนดค่าความเหมาะสมแล้ว การคัดเลือกประชากรเพื่อใช้ในการสร้างประชากรรุ่นต่อไปก็จะเริ่มขึ้น โอกาสที่ประชากรแต่ละตัวจะได้รับการคัดเลือกขึ้นอยู่กับค่าความเหมาะสม โดยประชากรที่มีค่าความเหมาะสมสูงก็จะมีความน่าจะเป็นที่จะได้รับเลือกสูง

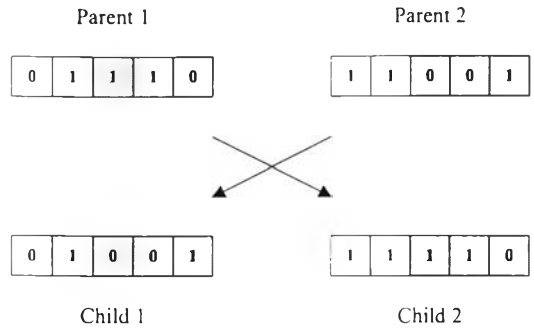
ประชากรที่ได้รับการคัดเลือกจะใช้ในการสร้างประชากรรุ่นใหม่ โดยอาศัยตัวดำเนินการทั้งหมด 3 แบบ คือ

1. **การทำซ้ำ (Reproduction)** ประชากรที่ถูกเลือกจะถูกกำหนดให้เป็นประชากรรุ่นใหม่โดยไม่มีการเปลี่ยนแปลง ประโยชน์ของตัวดำเนินการนี้เพื่อรักษาประชากรที่มีคุณภาพดีเอาไว้ และป้องกันไม่ให้ประชากรรุ่นใหม่มีคุณภาพแย่กว่าประชากรเดิม

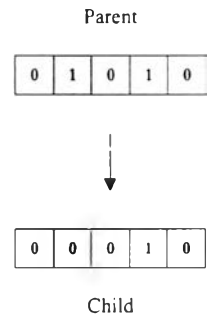
2. **การเปลี่ยนไขว้ (Crossover)** การทำงานของการเปลี่ยนไขว้ใช้ประชากรเดิมสองตัวเพื่อสร้างประชากรรุ่นใหม่สองตัว ประชากรรุ่นใหม่จะสร้างจากการผสมประชากรเดิมทั้งสองตัวเข้าด้วยกัน การเปลี่ยนไขว้สามารถทำได้หลายรูปแบบ รูปแบบที่ง่ายที่สุดคือ การเปลี่ยนไขว้แบบหนึ่งจุด (one-point crossover) ซึ่งทำงานโดยทำการเลือกจุดที่จะตัดโครโมโซมด้วยการสุ่ม ส่วนที่ถูกตัดของโครโมโซมทั้งสองจะถูกสับเปลี่ยนกันเพื่อสร้างประชากรใหม่สองตัว ลักษณะการทำงานแสดงในรูป 2.1

3. **การกลายพันธุ์ (Mutation)** การกลายพันธุ์เป็นการใส่การเปลี่ยนแปลงโดยการสุ่มลงไปในโครโมโซม โดยทั่วไปกระบวนการนี้จะมีโอกาสเกิดขึ้นไม่มากนัก ถ้าโครโมโซมอยู่ในรูปของสายอักขระฐานสอง การกลายพันธุ์จะทำโดยการสุ่มบิตที่จะมีการเปลี่ยนแปลง แล้วทำการเปลี่ยนบิตจาก 0 เป็น 1 หรือจาก 1 เป็น 0 ลักษณะการทำงานแสดงในรูป 2.2

หลังจากที่สร้างประชากรด้วยตัวดำเนินการทั้ง 3 แบบแล้ว ประชากรรุ่นใหม่ที่ได้จะแทนที่ประชากรเดิม การทำงานของขั้นตอนวิธีเชิงพันธุกรรมก็จะกลับไปสู่การหาค่าความเหมาะสม, การคัดเลือก และการ



รูปที่ 2.1 การเปลี่ยนไขว้แบบหนึ่งจุด



รูปที่ 2.2 การกลายพันธุ์

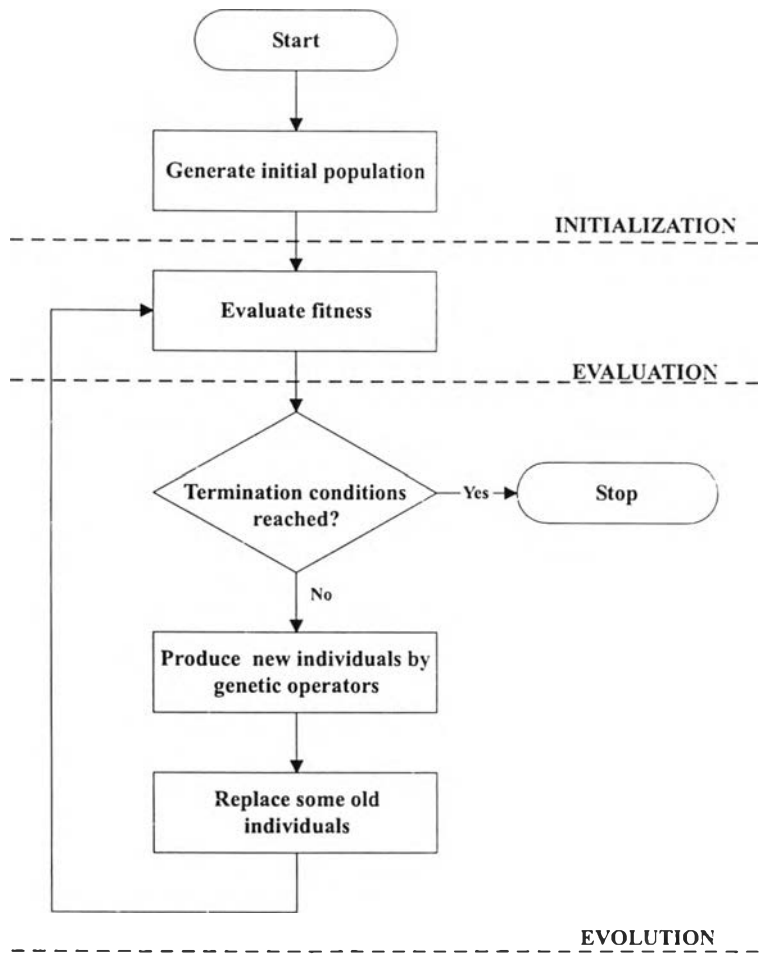
สร้างประชากรใหม่ด้วยตัวดำเนินการ แต่ละรอบของการทำซ้ำเรียกว่า รุ่น (generation) กระบวนการหาคำตอบของขั้นตอนวิธีเชิงพันธุกรรมจะดำเนินการไปจนกระทั่งเงื่อนไขในการหยุดทำงานเป็นจริง เงื่อนไขในการหยุดทำงานได้แก่ การพบคำตอบที่ต้องการ หรือจำนวนรุ่นเกินกว่าที่กำหนดเพื่อไม่ให้การหาคำตอบใช้เวลานานเกินไป ผังงาน (flowchart)ของลักษณะการทำงานของขั้นตอนวิธีเชิงพันธุกรรมแสดงในรูป 2.3

2.2 กำหนดการเชิงพันธุกรรม (Genetic programming)

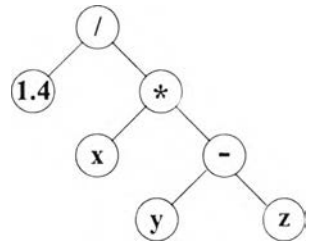
กำหนดการเชิงพันธุกรรม (Genetic Programming หรือ GP) [5] เป็นขั้นตอนวิธีในการหาคำตอบที่ดัดแปลงจากขั้นตอนวิธีเชิงพันธุกรรม ความแตกต่างของกำหนดการเชิงพันธุกรรมจากขั้นตอนวิธีเชิงพันธุกรรม คือ ลักษณะการแทนคำตอบของประชากร โดยในขั้นตอนวิธีเชิงพันธุกรรม ประชากรจะแทนด้วยสายอักขระที่มีขนาดคงที่ ในทางตรงกันข้ามประชากรของกำหนดการเชิงพันธุกรรมอยู่ในรูปโปรแกรมคอมพิวเตอร์ ซึ่งมักจะอยู่ในรูปโครงสร้างต้นไม้

ประชากรของกำหนดการเชิงพันธุกรรมอาจไม่ได้อยู่ในรูปโปรแกรมคอมพิวเตอร์ก็ได้ โดยอาจอยู่ในรูปของคำตอบที่มีขนาดและรูปร่างที่ไม่กำหนดตายตัว เนื่องจากปัญหาบางชนิดลักษณะของคำตอบไม่สามารถแทนด้วยสายอักขระของขั้นตอนวิธีเชิงพันธุกรรม ทำให้ปัญหาเหล่านี้จึงเหมาะกับการใช้กำหนดการเชิงพันธุกรรม

ขั้นตอนการทำงานของกำหนดการเชิงพันธุกรรมคล้ายคลึงกับขั้นตอนวิธีเชิงพันธุกรรม แต่เนื่องจากลักษณะของการแทนคำตอบที่แตกต่างกัน การทำงานบางอย่างจึงถูกดัดแปลงให้เหมาะสมกับลักษณะของการ



รูปที่ 2.3 ผังงานของลักษณะการทำงานของขั้นตอนวิธีเชิงพันธุกรรม



รูปที่ 2.4 ตัวอย่างของคำตอบในรูปโครงสร้างต้นไม้

แทนคำตอบ โดยทั่วไปประชากรของกำหนดการเชิงพันธุกรรมอยู่ในรูปโครงสร้างต้นไม้ ซึ่งประกอบไปด้วยชุดฟังก์ชัน (function set) และชุดเทอร์มินอล (terminal set) ที่เหมาะกับปัญหา

ฟังก์ชัน เป็นบัพภายใน (internal node) ที่มีอาร์กิวเมนต์ตั้งแต่ 1 ขึ้นไป ฟังก์ชันหนึ่งสามารถเป็นอาร์กิวเมนต์ของอีกฟังก์ชันหนึ่งได้ ดังนั้นฟังก์ชันหนึ่งสามารถส่งค่าคืนหลังจากการดำเนินการให้กับอีกฟังก์ชันหนึ่งได้ ตัวอย่างของฟังก์ชันได้แก่

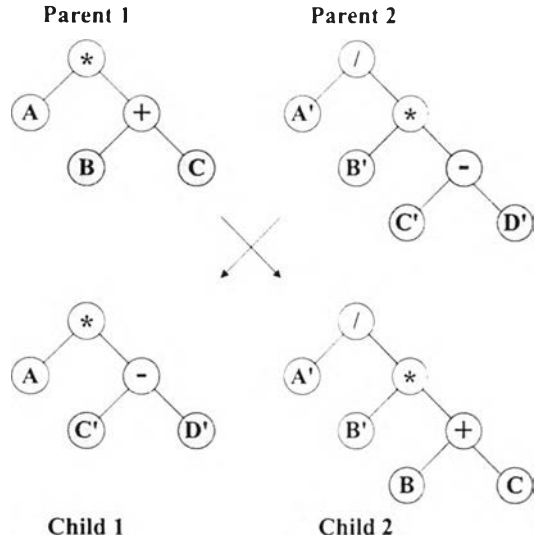
- ตัวดำเนินการคณิตศาสตร์ เช่น +, -, *, /
- ตัวดำเนินการเงื่อนไข เช่น `IF<=`, `IF<`, `IF=`, `IF-AND`, `IF-OR`, `IF-NOT`
- ตัวดำเนินการตรรกศาสตร์ เช่น `AND`, `OR`, `NOT`
- ตัวดำเนินการวนซ้ำ เช่น `DO-WHILE`
- ฟังก์ชันเฉพาะของปัญหานั้น เช่น `IF-FOOD-AHEAD`

เทอร์มินอล เป็นบัพปลายทาง (terminal node) ซึ่งไม่มีอาร์กิวเมนต์ ตัวอย่างของเทอร์มินอลได้แก่

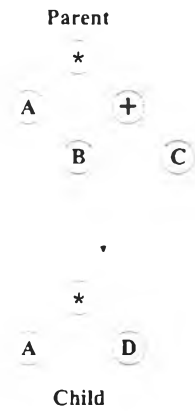
- ค่าคงที่ เช่น 1, 0 , 0.5
- ตัวแปร เช่น `x`, `y`, `z`
- คำสั่งที่มีการทำงานกับปัญหานั้น เช่น `Forward`, `TurnRight`

การเขียนสัญลักษณ์แทนคำตอบที่เป็นโปรแกรมซึ่งอยู่ในรูปโครงสร้างต้นไม้ มักจะเขียนในรูปของนิพจน์เชิงสัญลักษณ์ (symbolic expression) โดยตัวอย่างของคำตอบในรูป 2.4 สามารถเขียนเป็นนิพจน์เชิงสัญลักษณ์ได้คือ `(/ 1.4 (* x (- y z)))`

หลังจากการสร้างประชากรเริ่มต้นด้วยการสุ่ม เช่นเดียวกับการทำงานของขั้นตอนวิธีเชิงพันธุกรรม ขั้นต่อไปคือการหาค่าความเหมาะสมของประชากรแต่ละตัว ถ้าประชากรเป็นโปรแกรม การหาค่าความเหมาะสมก็คือการนำการดำเนินการตามโปรแกรมคอมพิวเตอร์แต่ละโปรแกรม (ประชากรแต่ละตัว) ไปทดลองแก้ปัญหาที่กำหนด จากนั้นจึงให้คะแนนเป็นค่าความเหมาะสมตามประสิทธิภาพการทำงานของโปรแกรมในการแก้ปัญหา การดำเนินการตามโปรแกรมซึ่งอยู่ในรูปโครงสร้างต้นไม้จะเริ่มทำงานจากตำแหน่งรากของต้นไม้ทำการประมวลผลไปจนถึงตำแหน่งเทอร์มินอล ซึ่งอาจเป็นคำสั่งการทำงานอย่างใดอย่างหนึ่ง การดำเนินจะกระทำไปเรื่อยๆ จนสุดต้นไม้จึงย้อนกลับไปที่ตำแหน่งรากอีกครั้ง จนกว่าเงื่อนไขจำกัดการวนซ้ำที่กำหนดไว้จะเป็นจริง



รูปที่ 2.5 การเปลี่ยนไขว้สำหรับกำหนดการเชิงพันธุกรรม



รูปที่ 2.6 การกลายพันธุ์สำหรับกำหนดการเชิงพันธุกรรม

หลังจากการหาค่าความเหมาะสม ประชากรจะได้รับเลือกเพื่อสร้างประชากรรุ่นใหม่ ตัวดำเนินการที่ใช้ในการสร้างประชากรรุ่นใหม่มี 3 แบบเช่นเดียวกับขั้นตอนวิธีเชิงพันธุกรรม การเปลี่ยนไขว้ และการกลายพันธุ์ จะมีลักษณะการทำงานดังรูป 2.5 และ 2.6 ตามลำดับ

2.3 การประมวลผลแบบขนาน

สถาปัตยกรรมการประมวลผลแบบขนานสามารถแบ่งแยกได้หลายแบบ Michael Flynn เป็นผู้ริเริ่มการจัดประเภท และได้รับการยอมรับอย่างแพร่หลาย การจัดประเภทแบบนี้อาศัยลักษณะของชุดคำสั่ง และชุดข้อมูล โดยสามารถแบ่งได้ออกเป็น 4 ประเภทดังนี้

1. **ทีละชุดคำสั่งและทีละชุดข้อมูล (Single-Instruction Single-Data หรือ SISD)** ลักษณะของคอมพิวเตอร์แบบนี้จะเป็นแบบพื้นฐานที่สุด คือมีหน่วยประมวลผลชุดเดียว ทำงานกับชุดข้อมูลทีละชุด ตัวอย่างของเครื่องคอมพิวเตอร์แบบนี้ได้แก่ เครื่องคอมพิวเตอร์ส่วนบุคคล (personal computer) แต่อย่างไรก็ตามเครื่องคอมพิวเตอร์แบบนี้ในปัจจุบันก็มีการทำงานแบบขนานอยู่ภายใน

ในระดับชุดคำสั่ง เช่น การทำงานแบบสายท่อ (pipeline) ซึ่งบรรจุอยู่ในฮาร์ดแวร์ เพื่อเพิ่มประสิทธิภาพในการทำงาน

2. **ทีละชุดคำสั่งและหลายชุดข้อมูล (Single-Instruction Multiple-Data หรือ SIMD)** จะประกอบด้วยหน่วยประมวลผลจำนวนมาก และแต่ละหน่วยประมวลผลจะมีหน่วยความจำเฉพาะที่ (local memory) เพื่อใช้เก็บข้อมูลขณะดำเนินการตามชุดคำสั่ง คำสั่งแต่ละคำสั่งจะถูกส่งไปยังกลุ่มของหน่วยประมวลผล หน่วยประมวลผลจะดำเนินการตามคำสั่งที่ส่งมาเป็นจังหวะ (lockstep) แบบประสานเวลา (synchronous) คือ ทุกหน่วยประมวลผลทำงานโดยใช้คำสั่งเหมือนกันและพร้อมกัน ทำให้สถาปัตยกรรมแบบนี้เหมาะกับข้อมูลเป็นจำนวนมาก และสามารถดำเนินการได้พร้อมกัน มิฉะนั้นจะทำให้หน่วยประมวลผลบางหน่วยจะอยู่ในช่วงที่ว่าง ไม่มีการดำเนินการตามคำสั่ง

3. **หลายชุดคำสั่งและทีละชุดข้อมูล (Multiple-Instruction Single-Data หรือ MISD)** ในประเภทนี้ไม่ค่อยได้รับความสนใจมากนัก เนื่องมาจากไม่เหมาะกับลักษณะของการเขียนโปรแกรม สถาปัตยกรรมแบบหนึ่งที่จัดอยู่ในประเภทนี้และมีการนำมาใช้คือ systolic arrays ซึ่งใช้ในงานเฉพาะทาง เช่น งานประมวลผลสัญญาณ

4. **หลายชุดคำสั่งและหลายชุดข้อมูล (Multiple-Instruction Multiple-Data หรือ MIMD)** จะประกอบด้วยกลุ่มของหน่วยประมวลผล โดยจะต่างจากแบบ SIMD ตรงที่แต่ละหน่วยประมวลผลจะทำงานตามโปรแกรมของตัวเอง และที่เวลาเดียวกันหน่วยประมวลผลแต่ละหน่วยไม่จำเป็นต้องทำงานกับคำสั่งเดียวกัน สถาปัตยกรรมแบบ MIMD สามารถแบ่งได้เป็นสองประเภทคือ ระบบที่ใช้หน่วยความจำร่วมกัน (shared-memory system) และระบบที่ใช้หน่วยความจำแยกกัน (distributed-memory system)

   ระบบที่ใช้หน่วยความจำร่วมกันมักจะมีหน่วยความจำแคช (cache memory) สำหรับแต่ละหน่วยประมวลผล ซึ่งทำให้เกิดปัญหาว่าข้อมูลที่อยู่ในหน่วยความจำแคชตรงกับในหน่วยความจำหลักหรือไม่ เนื่องสถาปัตยกรรมแบบ MIMD หน่วยประมวลผลแต่ละหน่วยทำงานแบบไม่ประสานเวลา (asynchronous) ทำให้ค่าที่อยู่ในหน่วยความจำหลักอาจโดนแก้ไขโดยหน่วยประมวลผลหนึ่ง มีผลให้ค่าที่อยู่ในหน่วยความจำแคชของหน่วยประมวลผลอื่นเป็นค่าไม่ถูกต้อง

   ระบบที่ใช้หน่วยความจำแยกกันเป็นระบบหนึ่งที่ได้รับความนิยม และคอมพิวเตอร์แบบหนึ่งที่จัดอยู่ในประเภทนี้คือ สถานีงานแบบกลุ่ม (clustered workstations) [7] ซึ่งใช้ในวิทยานิพนธ์ฉบับนี้ ในบทความ [6] ได้แสดงความคิดเห็นว่าของคอมพิวเตอร์แบบนี้มีจุดน่าสนใจหลายอย่าง ประการแรกคือในด้านราคา คอมพิวเตอร์แบบนี้มีราคาถูก สามารถสร้างจากเครื่องคอมพิวเตอร์ที่ต่อเครือข่ายที่มีอยู่แล้ว ประการที่สองคือเครื่องคอมพิวเตอร์สามารถปรับเปลี่ยนไปตามยุคสมัย และใช้ซอฟต์แวร์ที่เป็นมาตรฐาน ทำให้ง่ายสำหรับการพัฒนาโปรแกรม จุดด้อยของคอมพิวเตอร์แบบนี้ก็คือประสิทธิภาพที่จำกัดเนื่องมาจากเวลาที่ใช้ในการติดต่อสื่อสาร อย่างไรก็ตามถ้าการคำนวณไม่จำเป็นต้องใช้การติดต่อสื่อสารมากนักก็เหมาะที่จะใช้กับคอมพิวเตอร์แบบนี้

ลักษณะการเขียนโปรแกรมแบบขนานสามารถแบ่งแยกได้หลายแบบ จากหนังสือ [8] แบ่งการเขียนโปรแกรมแบบขนานเป็น 5 แบบดังนี้

1. **แบบใช้ตัวแปรร่วมกัน (Shared-Variable Model)** เป็นลักษณะการเขียนโปรแกรมโดยใช้ตัวแปรร่วมกันสำหรับการสื่อสารระหว่างโปรเซส (Interprocess Communication หรือ IPC) การใช้ตัวแปรร่วมกันจำเป็นต้องมีการใช้หน่วยความจำร่วมกัน และการป้องกันการใช้พร้อมกัน (mutual exclusion) เมื่อมีความต้องการใช้ชุดตัวแปรเหมือนกันโดยโปรเซส (process) ตั้งแต่สองโปรเซสพร้อมกัน

2. **แบบส่งผ่านข้อความ (Message-Passing Model)** เป็นวิธีที่นิยมใช้สำหรับการเขียนโปรแกรมในสถาปัตยกรรมแบบ MIMD ที่ใช้หน่วยความจำแยกกัน ลักษณะการเขียนโปรแกรมอาศัยการส่งข้อความผ่านระบบเครือข่ายเพื่อสื่อสารระหว่างโปรเซส ข้อความอาจเป็นชุดคำสั่ง, ข้อมูล, การประสานเวลา หรือสัญญาณการขัดจังหวะ (interrupt signal) เวลาที่ใช้ในการส่งข้อความจะมีความล่าช้าสูงกว่าแบบที่ใช้ตัวแปรร่วมกัน

3. **แบบข้อมูลขนาน (Data-Parallel Model)** เป็นวิธีการที่ง่ายที่สุดวิธีหนึ่งสำหรับการเขียนโปรแกรมแบบขนาน โครงสร้างข้อมูลจะถูกแบ่งกระจายให้กับกลุ่มของโปรเซส แต่ละโปรเซสจะทำงานด้วยคำสั่งชุดเดียวกัน โดยใช้ข้อมูลในส่วนของโปรเซสตัวเอง การเขียนโปรแกรมแบบนี้เหมาะกับเครื่องในลักษณะ SIMD ภาษาที่ใช้ในการเขียนโปรแกรมจะมีการเพิ่มเติมส่วนขยายสำหรับการทำงานแบบขนาน ตัวอย่างของภาษาได้แก่ Fortran 90

4. **แบบเชิงวัตถุ (Object-Oriented Model)** เป็นการนำการเขียนโปรแกรมเชิงวัตถุมาใช้กับการเขียนโปรแกรมแบบขนาน การทำงานจะใช้การสื่อสารด้วยข้อความระหว่างวัตถุ

5. **แบบเชิงหน้าที่และเชิงตรรกะ (Functional and Logic Model)** เป็นการเขียนโปรแกรมแบบขนานโดยใช้ลักษณะของภาษา แบบแรกจะเป็นภาษาสำหรับการเขียนโปรแกรมเชิงหน้าที่ (functional programming language) แบบที่สองเป็นภาษาสำหรับการเขียนโปรแกรมเชิงตรรกะ (logic programming language) การเขียนโปรแกรมแบบขนานแบบนี้จะประยุกต์ใช้กับงานด้านปัญญาประดิษฐ์ (Artificial Intelligence หรือ AI)

2.4 การประมวลผลแบบขนานสำหรับขั้นตอนวิธีเชิงพันธุกรรม และกำหนดการเชิงพันธุกรรม

การทำงานของขั้นตอนวิธีเชิงพันธุกรรม และกำหนดการเชิงพันธุกรรม จะใช้กลุ่มของประชากรในกระบวนการหาคำตอบ ซึ่งกระบวนการในการจัดการกับประชากรแต่ละตัวค่อนข้างเป็นอิสระต่อกัน ทำให้ง่ายในการใช้การประมวลผลแบบขนานเพื่อลดเวลาที่ใช้ในการหาคำตอบ

จากงานวิจัย [9] แสดงให้เห็นว่าความสำเร็จในการหาคำตอบของขั้นตอนวิธีเชิงพันธุกรรมจะขึ้นกับการใช้ขนาดของประชากรให้เหมาะสมกับความยากของปัญหา ดังนั้นถ้าสามารถเพิ่มประสิทธิภาพด้วยการประมวลผลแบบขนาน ทำให้สามารถใช้ประชากรที่มีขนาดใหญ่ขึ้นกับขั้นตอนวิธีเชิงพันธุกรรม จะสามารถแก้ปัญหาที่มีความซับซ้อนมากขึ้นได้

การจัดแบ่งประเภทการประมวลผลแบบขนานสำหรับขั้นตอนวิธีเชิงพันธุกรรมจะแบ่งได้เป็น 3 ประเภท [1] ตามบทความ [10] ดังนี้

[1] กำหนดการเชิงพันธุกรรมใช้การจัดแบบเดียวกับขั้นตอนวิธีเชิงพันธุกรรม

2.4.1 แบบหัวหน้า-ผู้ช่วย (Master-slave model)

การประมวลผลแบบขนานแบบนี้จะมีประชากรเพียงกลุ่มเดียว แต่ขั้นตอนในการหาค่าความเหมาะสมของประชากร ซึ่งเป็นขั้นตอนที่ใช้เวลานานจะทำแบบขนาน นอกจากนี้กระบวนการของขั้นตอนวิธีเชิงพันธุกรรมอื่นๆ ก็สามารถทำแบบขนานได้ เช่น การสร้างประชากรรุ่นใหม่

ลักษณะการประมวลผลแบบหัวหน้า-ผู้ช่วยแสดงในรูป 2.7 ในการทำงานจะมีหน่วยประมวลผลหลักอยู่หนึ่งหน่วยที่ทำหน้าที่เก็บประชากรทั้งหมด จากนั้นในขั้นตอนการหาค่าความเหมาะสม หน่วยประมวลผลหลักจะทำการแบ่งประชากร และทำการส่งไปยังกลุ่มของหน่วยประมวลผลที่ทำหน้าที่เป็นผู้ช่วย กลุ่มของหน่วยประมวลผลที่เป็นผู้ช่วยจะทำการหาค่าความเหมาะสม และส่งค่าความเหมาะสมคืนให้กับหน่วยประมวลผลหลัก

ในขั้นตอนการรับค่าความเหมาะสมจากหน่วยประมวลผลที่เป็นผู้ช่วย ถ้ามีการหยุดรอให้ได้ค่าความเหมาะสมจากหน่วยประมวลผลที่เป็นผู้ช่วยทุกหน่วยแล้วจึงดำเนินการต่อ การทำงานจะเหมือนกับขั้นตอนวิธีเชิงลำดับ และประสิทธิภาพที่เพิ่มขึ้นอาจไม่มากนัก โดยทั่วไปในการทำงานแบบหัวหน้า-ผู้ช่วยนิยมใช้การทำงานแบบประสานเวลาคือ มีการหยุดรอให้ได้ค่าความเหมาะสมจากหน่วยประมวลผลทุกหน่วย แต่ก็เป็นไปได้ที่จะใช้การทำงานแบบไม่ประสานเวลาคือ ไม่หยุดรอหน่วยประมวลผลที่ทำงานช้า แต่ลักษณะการทำงานจะไม่เหมือนกับขั้นตอนวิธีเชิงลำดับ

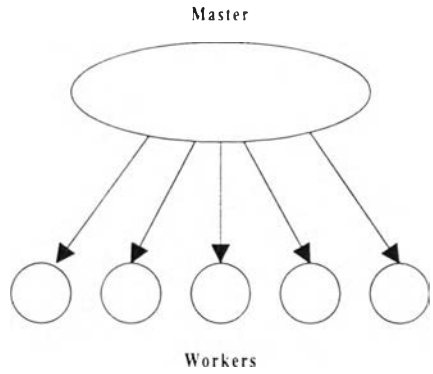
หลังจากที่หน่วยประมวลผลหลักได้รับค่าความเหมาะสมจากหน่วยประมวลผลที่เป็นผู้ช่วยแล้ว ก็จะทำการคัดเลือกประชากรเพื่อใช้ในการสร้างประชากรรุ่นใหม่ การสร้างประชากรรุ่นใหม่สามารถทำแบบขนานโดยแบ่งประชากรที่ได้รับการคัดเลือกแล้วส่งให้หน่วยประมวลผลที่เป็นผู้ช่วย เพื่อทำการสร้างประชากรรุ่นใหม่ แล้วส่งคืนให้กับหน่วยประมวลผลหลัก อย่างไรก็ตาม เนื่องจากการสร้างประชากรรุ่นใหม่มักจะใช้เวลาไม่นาน ผลจากการสร้างประชากรรุ่นใหม่แบบขนานอาจจะไม่คุ้มค่ากับเวลาที่ใช้ในการส่งข้อมูลจากหน่วยประมวลผลหลักไปยังหน่วยประมวลผลที่เป็นผู้ช่วย โปรแกรมบนหน่วยประมวลผลหลักสามารถเขียนเป็นรหัสเทียม (pseudo code) ดังรูป 2.8 และโปรแกรมบนหน่วยประมวลผลที่เป็นผู้ช่วยสามารถเขียนเป็นรหัสเทียมได้ดังรูป 2.9

ในการแบ่งประชากรเพื่อส่งให้หน่วยประมวลผลที่เป็นผู้ช่วยสำหรับกำหนดการเชิงพันธุกรรม ขนาดและความซับซ้อนอาจจะไม่เท่ากันเมื่อใช้จำนวนเท่ากัน ทำให้ใช้เวลาในการหาค่าความเหมาะสมไม่เท่ากัน เวลาที่ใช้ในการรอค่าความเหมาะสมก็จะเพิ่มขึ้นและไม่ได้ใช้ความสามารถของหน่วยประมวลผลอย่างเต็มที่ งานวิจัย [11] ได้เสนอการปรับภาระให้หน่วยประมวลผลแบบง่าย เพื่อให้ภาระของแต่ละหน่วยประมวลผลมีค่าใกล้เคียง ในขณะที่บทความ [6] ได้ให้ความเห็นว่ามีอีกวิธีที่จะแก้ปัญหานี้คือ ไม่ทำงานในลักษณะเป็นรุ่น แต่ใช้ลักษณะสถานะต่อเนื่อง (steady-state)

ข้อดีของการใช้การประมวลผลแบบนี้คือ การทำงานยังคงเหมือนกับการทำงานแบบเชิงลำดับ ทำให้เหมาะในการศึกษาการทำงาน และศึกษาผลกระทบของการปรับพารามิเตอร์ของขั้นตอนวิธีเชิงพันธุกรรมและกำหนดการเชิงพันธุกรรม

2.4.2 แบบหน่วยหยาบ (Coarse-grained model)

การประมวลผลแบบขนานแบบนี้จะมีการแบ่งประชากรออกเป็นกลุ่มประชากรย่อย (subpopulation หรือ deme) ที่มีขนาดค่อนข้างใหญ่ ในตอนเริ่มต้นหน่วยประมวลผลแต่ละหน่วยจะทำการสร้างประชากรแบบสุ่ม



รูปที่ 2.7 แบบหัวหน้า-ผู้ช่วย

```
produce the initial population
while not termination
begin
    send a fraction of the population to each of the processors
    wait to receive the fitness values
    select individuals
    produce new individuals
end
```

รูปที่ 2.8 รหัสเทียมของโปรแกรมบนหน่วยประมวลผลหลัก

โดยใช้ค่าเริ่มต้นการสุ่ม (random seed) ที่แตกต่างกัน เพื่อให้ประชากรย่อยแต่ละกลุ่มไม่เหมือนกัน จากนั้นหน่วยประมวลผลแต่ละหน่วยจะทำหน้าที่ดำเนินการกับกลุ่มประชากรย่อย กระบวนการวิวัฒนาการของประชากรย่อยแต่ละกลุ่มจะเป็นอิสระต่อกัน เมื่อถึงช่วงเวลาที่กำหนด ประชากรย่อยแต่ละกลุ่มจะมีการแลกเปลี่ยนประชากรบางส่วนกับประชากรย่อยกลุ่มอื่น การแลกเปลี่ยนประชากรนี้เรียกว่าการอพยพ (migration) สาเหตุของการมีการอพยพเพื่อเพิ่มความหลากหลายให้กับกลุ่มประชากร นอกจากนี้การอพยพช่วยให้ประชากรย่อยแต่ละกลุ่มทำการค้นหาในบริเวณค้นหาคำตอบ (search space) ในตำแหน่งอื่น

การประมวลผลแบบนี้เวลาที่เสียไปเนื่องจากการติดต่อสื่อสารจะน้อยกว่าแบบหัวหน้า-ผู้ช่วย เพราะการอพยพไม่จำเป็นต้องทำทุกรุ่นของการวิวัฒนาการ และประชากรที่ส่งเนื่องจากการอพยพจะมีจำนวนไม่มากนัก ในบางงานวิจัยจะเรียกการประมวลผลแบบนี้ว่า แบบเกาะ (island model) เนื่องจากหน่วยประมวลผลแต่ละหน่วยทำงานคล้ายกับเกาะที่มีประชากรอาศัยอยู่ และมีการอพยพประชากรข้ามระหว่างเกาะ

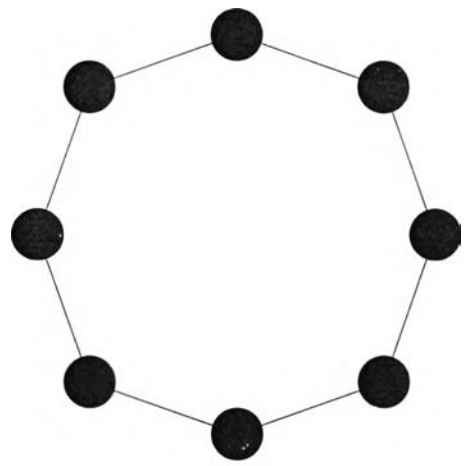
จากรูป 2.10 แสดงลักษณะการทำงานโดยใช้การเชื่อมต่อแบบวงแหวน วงกลมสีดำแทนกลุ่มประชากรย่อย เส้นตรงที่เชื่อมระหว่างวงกลมสีดำแทนเส้นทางการติดต่อสื่อสาร ซึ่งการอพยพจะกระทำผ่านเส้นทางนี้ โปรแกรมของหน่วยประมวลผลแต่ละหน่วยสามารถเขียนเป็นรหัสเทียมได้ดังรูป 2.11

การประมวลผลแบบนี้ได้รับความสนใจจากนักวิจัยมากที่สุด จากบทความ [10] ได้ให้ความเห็นว่าจุดเด่นของการประมวลผลแบบนี้ที่ทำให้ได้รับสนใจอย่างมากคือ

- การประมวลผลแบบนี้เป็นการเพิ่มส่วนขยายแบบง่ายลงไปในขั้นตอนวิธีเชิงลำดับ โดยนำขั้นตอนวิธีเชิงลำดับไปทำงานบนกลุ่มของหน่วยประมวลผล เมื่อถึงช่วงเวลาที่กำหนดก็มีการแลกเปลี่ยนประชา

while not termination
begin
    wait to receive a fraction of the population
    evaluate individuals
    send the fitness values to the master node
end

รูปที่ 2.9 รหัสเทียมของโปรแกรมบนหน่วยประมวลผลที่เป็นผู้ช่วย



รูปที่ 2.10 แบบหน่วยหยาบ

กรกัน

- ไม่ต้องใช้ความพยาพยามในการปรับโปรแกรมแบบเชิงลำดับมาเป็นแบบขนานสูงมากนัก ส่วนใหญ่ของโปรแกรมยังคงเป็นแบบเดิม โดยมีการเขียนโปรแกรมย่อยเพิ่มเดิมบางส่วน
- เครื่องคอมพิวเตอร์ที่ใช้หาได้ง่าย ถ้าไม่มีก็สามารถใช้เครื่องคอมพิวเตอร์ที่เชื่อมต่อด้วยระบบเครือข่าย และใช้โปรแกรมที่ไม่เสียค่าใช้จ่าย เช่น MPI [12] หรือ PVM [13]

ข้อเสียของการประมวลผลแบบนี้คือ เปลี่ยนลักษณะการทำงานของขั้นตอนวิธีเชิงลำดับ ทำให้ทฤษฎี และความรู้จากขั้นตอนวิธีเชิงลำดับ ไม่สามารถใช้กับการประมวลผลแบบนี้ นอกจากนี้พารามิเตอร์การทำงานยังเพิ่มขึ้นทำให้มีความซับซ้อนในการเลือกใช้ เช่น การอพยพจะถูกกำหนดด้วยพารามิเตอร์หลายอย่าง คือ ความถี่ของการอพยพ, จำนวนประชากรเมื่อมีการอพยพ, ลักษณะการส่งประชากร, การนำประชากรจากหน่วยประมวลผลอื่นมาแทนที่ประชากรเดิมจะทำอย่างไร จะใช้การสุ่ม หรือจะแทนที่ตัวที่แย่ที่สุด เป็นต้น ในปัจจุบันยังขาดความเข้าใจในพารามิเตอร์เหล่านี้ ดังนั้นยังคงต้องการการศึกษาเพิ่มเติมต่อไป

2.4.3 แบบหน่วยละเอียด (Fine-grained model)

ในการประมวลผลแบบนี้ ประชากรจะถูกแบ่งเป็นกลุ่มขนาดเล็ก และมีจำนวนมาก แต่ละกลุ่มอาจจะมีประชากรเพียงตัวเดียวกระจายอยู่บนหน่วยประมวลผลจำนวนมาก การหาค่าความเหมาะสมของแต่ละกลุ่มประชากรย่อยจะกระทำไปพร้อมกัน การคัดเลือกประชากร และการสร้างประชากรรุ่นใหม่จะกระทำโดยเลือก

```
produce the initial population
while not termination
begin
    evaluate individuals
    select individuals
    if (migration condition = True) then
    begin
        send n best individuals to a neighboring node
        receive n individuals from a neighboring node
        replace n individuals in the subpopulation
    end
    produce new individuals
end
```

รูปที่ 2.11 รหัสเทียมของการประมวลผลแบบหน่วยหยาบ

ประชากรจากหน่วยประมวลผลใกล้เคียง ลักษณะการทำงานจะคล้ายคลึงกับเซลลูลาร์ออโตมาตา (cellular automata) ดังนั้นการประมวลผลแบบนี้จึงมีอีกชื่อหนึ่งว่า แบบเซลลูลาร์ (cellular model)

จากบทความ [6] ให้ความเห็นว่าการประมวลผลแบบนี้เหมาะกับเครื่องที่มีสถาปัตยกรรมแบบ SIMD เนื่องจากความถี่ของการสื่อสารแบบท้องถิ่นมีเป็นจำนวนมาก แต่ไม่เหมาะในการนำการประมวลผลแบบนี้ไปใช้กับกำหนดการเชิงพันธุกรรม เนื่องจากประชากรแต่ละตัวมีขนาดและความซับซ้อนไม่เท่ากัน ทำให้หน่วยความจำท้องถิ่นที่ใช้ในการเก็บประชากรอาจไม่เพียงพอ นอกจากนี้การหาค่าความเหมาะสมของกำหนดการเชิงพันธุกรรมต้องใช้การดำเนินการตามประชากรที่เป็นโปรแกรม ซึ่งโปรแกรมแต่ละโปรแกรมบนแต่ละหน่วยประมวลผลจะไม่เหมือนกัน ดังนั้นถ้าใช้เครื่องคอมพิวเตอร์แบบ SIMD ที่การทำงานแต่ละครั้ง หน่วยประมวลผลทุกหน่วยจะทำงานเหมือนกัน จะทำให้หน่วยประมวลผลมีการว่าง ประสิทธิภาพจะไม่ดีเท่าที่ควร

จากรูป 2.12 แทนลักษณะการเชื่อมต่อที่นิยมใช้กันคือ ตาราง 2 มิติ (2-Dimensional grid) โดยจุดสีดำแทนประชากรแต่ละตัวที่อยู่บนหน่วยประมวลผล และเส้นตรงแสดงการเชื่อมต่อของการติดต่อสื่อสาร ตัวอย่างวิธีการเลือกประชากรจากกลุ่มที่อยู่ใกล้เคียง 4 และ 8 กลุ่มเพื่อใช้ในการสร้างประชากรรุ่นใหม่แสดงในรูป 2.13 โปรแกรมของแต่ละหน่วยประมวลผลสามารถเขียนเป็นรหัสเทียมได้ดังรูป 2.14

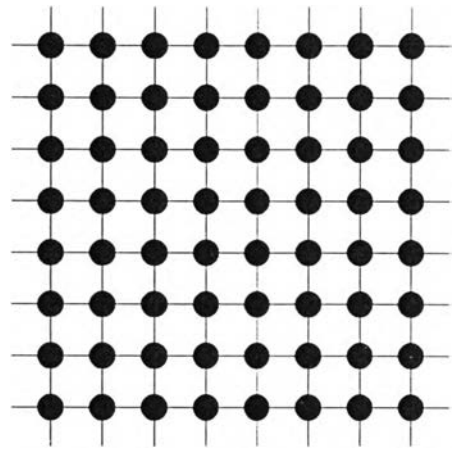
2.5 ข้อสังเกตเกี่ยวกับการประมวลผลแบบขนาน

2.5.1 ผลที่ได้นอกเหนือจากการลดเวลาในการทำงาน

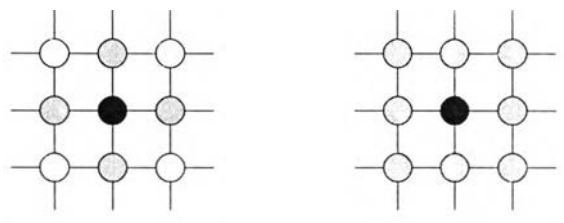
ในการใช้การประมวลผลแบบขนานกับขั้นตอนวิธีเชิงพันธุกรรม และกำหนดการเชิงพันธุกรรมโดยเฉพาะอย่างยิ่งแบบหน่วยหยาบ ประโยชน์ที่ได้นอกจากการแบ่งงานแล้วทำงานแบบขนานจะช่วยเพิ่มประสิทธิภาพในเรื่องของเวลา ยังมีประโยชน์อย่างอื่นอีกดังต่อไปนี้

2.5.1.1 การใช้หน่วยความจำ

เมื่อปัญหามีขนาดและความซับซ้อนมากขึ้น จึงจำเป็นที่ต้องเพิ่มขนาดของประชากรที่ใช้สำหรับการหาคำตอบ ทำให้ขนาดของการใช้หน่วยความจำในการเก็บประชากรมีขนาดใหญ่กว่าหน่วยความจำหลักของเครื่อง



รูปที่ 2.12 แบบหน่วยละเอียด



รูปที่ 2.13 วิธีการนับกลุ่มประชากรที่อยู่ใกล้เคียง

produce a random individual i
while not termination
begin
    evaluate the individual i
    select a neighboring individual j
    produce individuals from i and j
    assign one of the individuals to i
end

รูปที่ 2.14 รหัสเทียมของการประมวลผลแบบหน่วยละเอียด

คอมพิวเตอร์ โดยเฉพาะอย่างยิ่งกับกำหนดการเชิงพันธุกรรม การประมวลผลแบบขนานช่วยลดปัญหานี้ เพราะประชากรมีการกระจายไปยังหน่วยประมวลผลแต่ละหน่วย

2.5.1.2 การรักษาความหลากหลาย

ปัญหาหนึ่งที่พบในการใช้ขั้นตอนวิธีเชิงพันธุกรรม และกำหนดการเชิงพันธุกรรมคือ เมื่อการวิวัฒนาการของประชากรผ่านไปช่วงเวลาหนึ่ง ความหลากหลายของประชากรจะลดลง ทำให้ไม่ประสบความสำเร็จในการหาคำตอบ การแบ่งประชากรเป็นกลุ่ม โดยถูกจำกัดให้มีการคัดเลือกและการสร้างประชากรรุ่นใหม่เฉพาะในกลุ่มเดียวกันเท่านั้น การประมวลผลแบบนี้ช่วยรักษาความหลากหลายของการวิวัฒนาการได้

2.5.1.3 ความได้เปรียบในเรื่องโอกาสที่น่าจะเป็นไปได้

การใช้การประมวลผลแบบหน่วยหยาบจะทำงานเพื่อหาคำตอบน้อยกว่าแบบเชิงลำดับ ในงานวิจัย [14] ได้ให้ความคิดเห็นว่ายังไม่มีคำอธิบายที่แน่ชัด ซึ่งต้องการการศึกษาเพิ่มเติมเกี่ยวกับสาเหตุของการทำงานที่น้อยลง และได้อธิบายเหตุผลที่เป็นไปได้สองสาเหตุคือ

1. ประชากรแต่ละกลุ่มจะวิวัฒนาการอยู่คนละบริเวณในการค้นหาคำตอบ ดังนั้นการอพยพจึงเป็นการนำประชากรที่ต่างจากเดิมจากประชากรกลุ่มอื่นมารวมเข้าด้วยกัน ซึ่งจะช่วยเพิ่มความหลากหลายให้กับกลุ่มประชากร การเพิ่มความหลากหลายมีผลสองประการคือ ประการแรกการเพิ่มความหลากหลายช่วยยืดระยะเวลาการลู่เข้า (convergence) ของขั้นตอนวิธี ซึ่งจะทำให้มีเวลาเพียงพอที่การเปลี่ยนไขว้ทำการผสม ส่วนของคำตอบ (building block) เพื่อสร้างคำตอบที่มีคุณภาพสูงกว่าขั้นตอนวิธีเชิงลำดับ ดังนั้นเป็นไปได้ว่าประชากรรวมของทุกประชากรย่อยสามารถใช้ขนาดที่เล็กกว่าที่ใช้ในขั้นตอนวิธีเชิงลำดับ ประการที่สองก็คือเป็นไปได้ที่ประชากรแต่ละกลุ่มจะทำการสร้างคำตอบบางส่วน (partial solution)และการอพยพจะเป็นการนำคำตอบบางส่วนนั้นมารวมเข้าด้วยกัน
2. การลดลงของงานเกิดขึ้นเนื่องมาจากความกดดันในการคัดเลือก (selection pressure) ที่เกี่ยวเนื่องกับลักษณะของการอพยพ จากงานวิจัยที่มีการศึกษามาพบว่าถ้าใช้ความกดดันในการคัดเลือกสูงจะทำให้การลู่เข้าของขั้นตอนวิธีเป็นไปอย่างรวดเร็ว ในการอพยพสามารถกำหนดกฎของการคัดเลือกได้หลายรูปแบบ สำหรับการเลือกประชากรเพื่อส่งไปยังประชากรกลุ่มอื่น สามารถใช้การสุ่ม หรือการเลือกตัวที่ดีที่สุดตามค่าความเหมาะสม และเช่นกันการเลือกประชากรที่จะถูกแทนที่ด้วยประชากรที่ได้มาจากการอพยพสามารถใช้การสุ่ม หรือการเลือกตัวที่แย่ที่สุดตามค่าความเหมาะสม ถ้าการคัดเลือกอาศัยค่าความเหมาะสม จะทำให้ความกดดันในการคัดเลือกมีค่าสูงขึ้น มีผลให้ประชากรมีการลู่เข้าเร็วขึ้น

2.5.2 การวัดประสิทธิภาพการประมวลผลแบบขนาน

วัตถุประสงค์หลักของการใช้การประมวลผลแบบขนานก็เพื่อลดเวลาในการทำงาน เพื่อที่จะบอกผลที่ได้รับจากการใช้การประมวลผลแบบขนานในการแก้ปัญหา การวัดประสิทธิภาพของการประมวลผลแบบขนานที่นิยมใช้กันทั่วไปคือ ค่าสปีดอัพ (speedup) ซึ่งเป็นอัตราส่วนของเวลาที่ใช้ในการประมวลผลแบบเชิงลำดับต่อเวลาที่ใช้ในการประมวลผลแบบขนาน

$$\text{Speedup} = \frac{\text{Serial Time}}{\text{Parallel Time}} \tag{2.1}$$

การอธิบายค่าสปีดอัพสามารถบอกในลักษณะของความสัมพันธ์ได้เช่น ค่าสปีดอัพเชิงเส้น (linear speedup) หมายถึง การลดลงแบบเชิงเส้นของเวลาในการประมวลผลแบบขนานเมื่อทำการเพิ่มหน่วยประมวลผล ค่าสปีดอัพมากกว่าเชิงเส้น (superlinear speedup) หมายถึง การลดลงของเวลามากกว่าเชิงเส้นในการประมวลผลแบบขนานเมื่อทำการเพิ่มหน่วยประมวลผล หรือกล่าวได้ว่าค่าสปีดอัพที่ได้มากกว่าจำนวนหน่วยประมวลผลที่ใช้

การที่จะถือว่าค่าสปีดอัพมากกว่าเชิงเส้นเป็นไปได้หรือไม่ขึ้นอยู่กับมุมมองของแต่ละคน หนังสือ [16] ได้แสดงความคิดเห็นของทั้งสองด้านดังนี้

- ค่าสปีดอัพมากกว่าเชิงเส้นเป็นไปไม่ได้ เหตุผลก็คือ คอมพิวเตอร์แบบเชิงลำดับสามารถจำลองการทำงานของคอมพิวเตอร์แบบขนาน ถ้าการประมวลผลแบบขนานสามารถทำงานเสร็จในเวลา T_p บนเครื่องคอมพิวเตอร์ที่มี p หน่วยประมวลผล ดังนั้นขั้นตอนวิธีแบบขนานนี้สามารถทำงานบนคอมพิวเตอร์เครื่องเดียวกันในเวลา $T_p \times p$ โดยใช้ 1 หน่วยประมวลผลด้วยการแบ่งเวลา (time slicing) ขั้นตอนวิธีแบบขนานมักจะต้องเสียเวลาไปบางส่วนมากกว่าขั้นตอนวิธีแบบเชิงลำดับ กล่าวคือสามารถหาขั้นตอนวิธีแบบเชิงลำดับที่ใช้เวลาในการทำงานที่น้อยกว่า $T_p \times p$ ดังนั้นค่าสปีดอัพไม่สามารถมากกว่าจำนวนหน่วยประมวลผลที่ใช้

- ค่าสปีดอัพมากกว่าเชิงเส้นเป็นไปได้ เหตุผลก็คือ การขาดความยุติธรรมในการเลือกขั้นตอนวิธีที่ใช้หลังจากการกำหนดชุดของปัญหา การวัดค่าสปีดอัพเป็นการสมมุติว่าใช้เวลาของขั้นตอนวิธีแบบเชิงลำดับที่ดีที่สุดหารด้วยเวลาของขั้นตอนวิธีแบบขนาน ซึ่งเป็นการยากที่จะเปลี่ยนข้อกำหนดของขั้นตอนวิธีที่ดีที่สุดเมื่อมีการเปลี่ยนชุดของปัญหา กล่าวอีกนัยหนึ่งคือ การเปรียบเทียบจะมีความยุติธรรมถ้ากำหนดขั้นตอนวิธีแบบเชิงลำดับที่ดีที่สุด และขั้นตอนวิธีแบบขนานก่อนที่กำหนดชุดของปัญหา ในกรณีนี้ถ้าค่าสปีดอัพมากกว่าเชิงเส้นก็จะเกิดจากการใช้หน่วยประมวลผลหลายชุดช่วยเพิ่มโอกาสที่เจอคำตอบเร็วขึ้น

งานวิจัย [14, 15] ได้แสดงความคิดเห็นเกี่ยวกับการโต้แย้งที่เกิดขึ้นในกลุ่มนักวิจัย เกี่ยวกับการที่เสนอผลการทดลองที่ค่าสปีดอัพมากกว่าเชิงเส้น งานวิจัย [15] อธิบายว่าสาเหตุของค่าสปีดอัพมากกว่าเชิงเส้นเนื่องมาจากขนาดของงานที่แตกต่างกันของการประมวลผลแบบเชิงลำดับ และการประมวลผลแบบขนาน ขนาดของงานนี้หมายถึงจำนวนฟังก์ชันที่ขั้นตอนวิธีใช้จนสามารถหาคำตอบได้ ดังที่ได้อธิบายในหัวข้อ 2.5.1.3 การประมวลผลแบบหน่วยหยาบช่วยให้จำนวนฟังก์ชันที่ใช้ในการหาคำตอบลดลง ดังนั้นการที่การประมวลผลแบบเชิงลำดับทำงานมากกว่าการประมวลผลแบบขนานจึงเป็นไปได้ที่ค่าสปีดอัพมากกว่าเชิงเส้น งานวิจัย [14] แสดงความคิดเห็นว่าบางครั้งการเสนอผลการทดลองที่ค่าสปีดอัพมากกว่าเชิงเส้นเกิดจากไม่ได้ทำการเปรียบเทียบคุณภาพคำตอบ บางครั้งการออกแบบการทดลองอาจจะดูเหมือนยุติธรรม โดยผลรวมของจำนวนประชากรทุกหน่วยประมวลผลเท่ากับจำนวนประชากรของการประมวลผลแบบเชิงลำดับ แต่คุณภาพของคำตอบที่ได้อาจแตกต่างกัน ดังนั้นการเปรียบเทียบประสิทธิภาพในการทำงานจึงต้องทำควบคู่กับการเปรียบเทียบคุณภาพคำตอบ