# บทที่ 2
# ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

## 2.1 เทคโนโลยีกลุ่ม (Group Technology)

เทคโนโลยีกลุ่ม (Group Technology) เป็นกระบวนการผลิตซึ่งนำเอาข้อมูลเกี่ยวกับลักษณะที่คล้ายคลึงกันของชิ้นงานและ/หรือเครื่องจักรมาดำเนินการจัดกลุ่มและใช้ข้อมูลเหล่านี้ให้เกิดประสิทธิภาพมากที่สุดสำหรับระบบการผลิต

ระบบการผลิตแบบเซลลูลาร์ เป็นการประยุกต์ใช้งานแบบหนึ่งของเทคโนโลยีกลุ่ม ซึ่งนำเอาชิ้นงานที่มีลักษณะการออกแบบและการผลิตที่คล้ายคลึงกันมาจัดกลุ่มเข้าด้วยกันเรียกว่า ครอบครัวชิ้นงาน (Part Family) และกลุ่มเครื่องจักร (Machine Group) ซึ่งจะรวมเรียกว่าเซล (Cell) โดยการสร้างเซลขึ้นมานั้นจะต้องให้ 1 ครอบครัวชิ้นงานหรือมากกว่า สามารถทำงานได้ในเซลเดียวกัน หรือเกิดการตัดกันระหว่างเซลน้อยที่สุด วิธีในการสร้างเซลแสดงดังรูปที่ 2.1

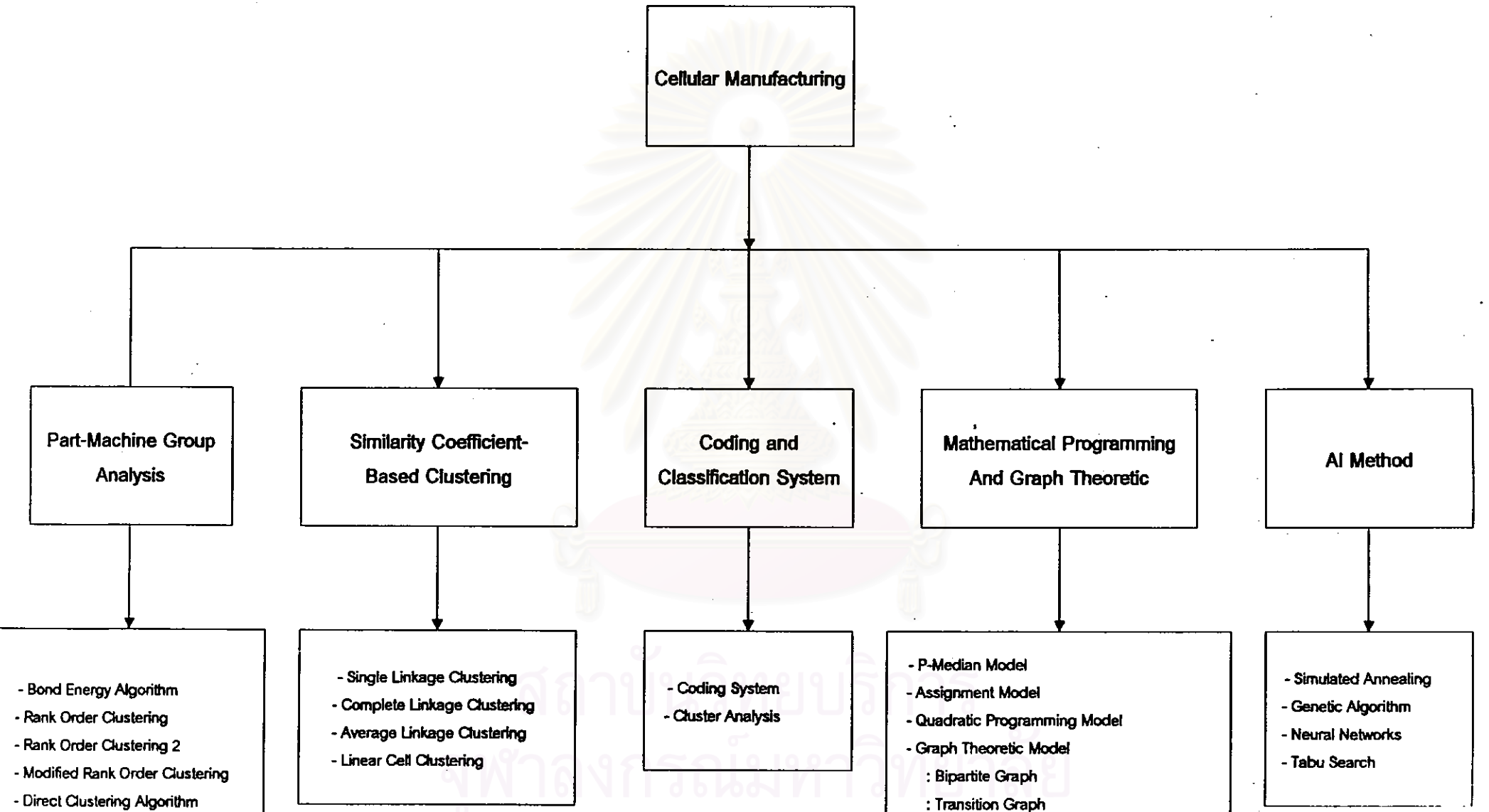
ข้อดีของเทคโนโลยีกลุ่ม เมื่อนำมาใช้กับระบบการผลิตแบบเซลลูลาร์คือ

1. ลดเวลาในการติดตั้งเครื่องจักร โดยการใช้เครื่องมือครอบครัวชิ้นงาน (Part Family Tooling) และการจัดลำดับการทำงานก่อนหลัง

2. ลดเวลาทำงานในกระบวนการ (Work - In - Process) โดยลดเวลาเคลื่อนที่ของชิ้นงาน การรอคอยสำหรับการเคลื่อนที่ และการใช้การส่งผ่านแบบขั้นๆ ภายใน Batch

3. ลดการทำงานซ้ำและเศษของวัตถุดิบ สินค้ามีประสิทธิภาพมากขึ้นเนื่องจากมีการจัดการให้มีความต่อเนื่องในการผลิต ทำให้ความผิดพลาดในการผลิตน้อยลง

4. ทำให้การทำงานต่าง ๆ เร็วขึ้นซึ่งจะช่วยลดปัญหาสินค้าคงคลัง และช่วยให้ตอบสนองกับภาวะของตลาดที่เปลี่ยนแปลงได้เร็วขึ้น




รูปที่ 2.1 วิธีการสร้างเซลของระบบการผลิตแบบเซลลูลาร์

## 2.2 การค้นหาคำตอบแบบทาบู (Tabu Search)

ในการหาคำตอบที่ดีที่สุด สำหรับปัญหา Combinatorial Optimization นั้น ได้มีการพัฒนาเทคนิคสำหรับแก้ไขปัญหาเหล่านี้อยู่หลายวิธี ลักษณะปัญหา Combinatorial Optimization นั้น เราจะไม่สามารถใช้วิธีการทางคณิตศาสตร์โดยตรงในการแก้ไขปัญหาได้ สำหรับปัญหาขนาดใหญ่ ดังนั้นจึงได้มีการคิดวิธีการอื่นที่จะนำมาใช้ในการแก้ปัญหาเหล่านี้ วิธีการที่นำมาใช้เรียกว่า ฮิวริสติกส์ (Heuristic) ในฮิวริสติกส์สามารถแบ่งเป็น Class เรียกว่า Metaheuristic ซึ่ง Metaheuristic จะมีวิธีการหาคำตอบที่ดีที่สุดสำหรับปัญหานั้น ๆ โดยมีการออกแบบให้เหมาะสมกับปัญหา ในที่นี้เราจะกล่าวถึง Metaheuristic แบบหนึ่งคือ การค้นหาคำตอบแบบทาบู (Tabu Search)

การค้นหาคำตอบแบบทาบู ถูกคิดค้นโดย Glover (1989) เป็นการออกแบบสำหรับหาคำตอบที่ดีที่สุดของปัญหาแบบ Combinatorial Optimization โดยกำหนดปัญหาที่จะทำการหาคำตอบ

กำหนดให้

X เป็นเซตของคำตอบที่เป็นไปได้ โดยมีคำตอบ s คำตอบ

f เป็นฟังก์ชันที่ให้ค่าน้อยที่สุด โดยแต่ละ s ใน X จะเป็นจำนวนจริง ซึ่งสามารถเขียนในรูป f(s)

N(s) หรือ Neighborhood ถูกกำหนดสำหรับแต่ละคำตอบของ s ใน X

วิธีการค้นหาคำตอบแบบทาบูจะเริ่มต้นจาก

1. หาคำตอบที่เป็นไปได้ จากคำตอบ s
2. หลังจากนั้นจะทำการย้ายตามเงื่อนไขและข้อบังคับของ Neighborhood List
3. เมื่อไรก็ตามที่มีคำตอบที่เป็นไปได้ จากคำตอบ s N(s) จะถูกตรวจสอบ ถ้าคำตอบ s' ที่ได้ ในN(s) ทำให้ $f(s') < f(s)$ แล้ว ก็จะเลือกค่าที่ s' นั้นไว้ เราจะทำตามขั้นตอนเหล่านี้ไป เรื่อย ๆ จนกว่าจะถึงเงื่อนไขของการหยุด
4. เพื่อที่จะหลีกเลี่ยงไม่ให้เกิดการวนรอบ เราจะไม่ยอมให้มีการย้อนกลับไปยังคำตอบที่ผ่านมาแล้วใน Iteration k ที่ผ่านมา ซึ่งเราเก็บคำตอบเหล่านั้นใน Tabu List ขนาดของ Tabu List จะเป็นค่าคงที่ หรือค่าผันแปรก็ได้ ขึ้นอยู่กับปัญหานั้น ๆ ในที่นี้เรากำหนดให้เท่ากับ k ลำดับขั้นตอนในการทำงานจะเป็นดังนี้ คือ เมื่อมีการย้ายจากคำตอบ s $\rightarrow$ คำตอบ s' เราจะนำคำตอบ s ไปเก็บไว้ที่ปลายของ Tabu List แต่ถ้าเกินขนาดของ Tabu List (ในกรณีที่ขนาดของ Tabu List ถูกกำหนดไว้) แล้วเราจะตัดคำตอบที่เก่าที่สุดออก และนำคำตอบ s นั้นไปใส่

ใน Tabu List เริ่มต้น แต่ถ้าคำตอบ s อยู่ใน Tabu List การย้ายของคำตอบ s ที่จะไปยัง Tabu List นั้นจะถูกห้ามไว้ ซึ่งค่าของคำตอบ s จะถูกเรียกว่าคำตอบทาบู (Tabu Solution)

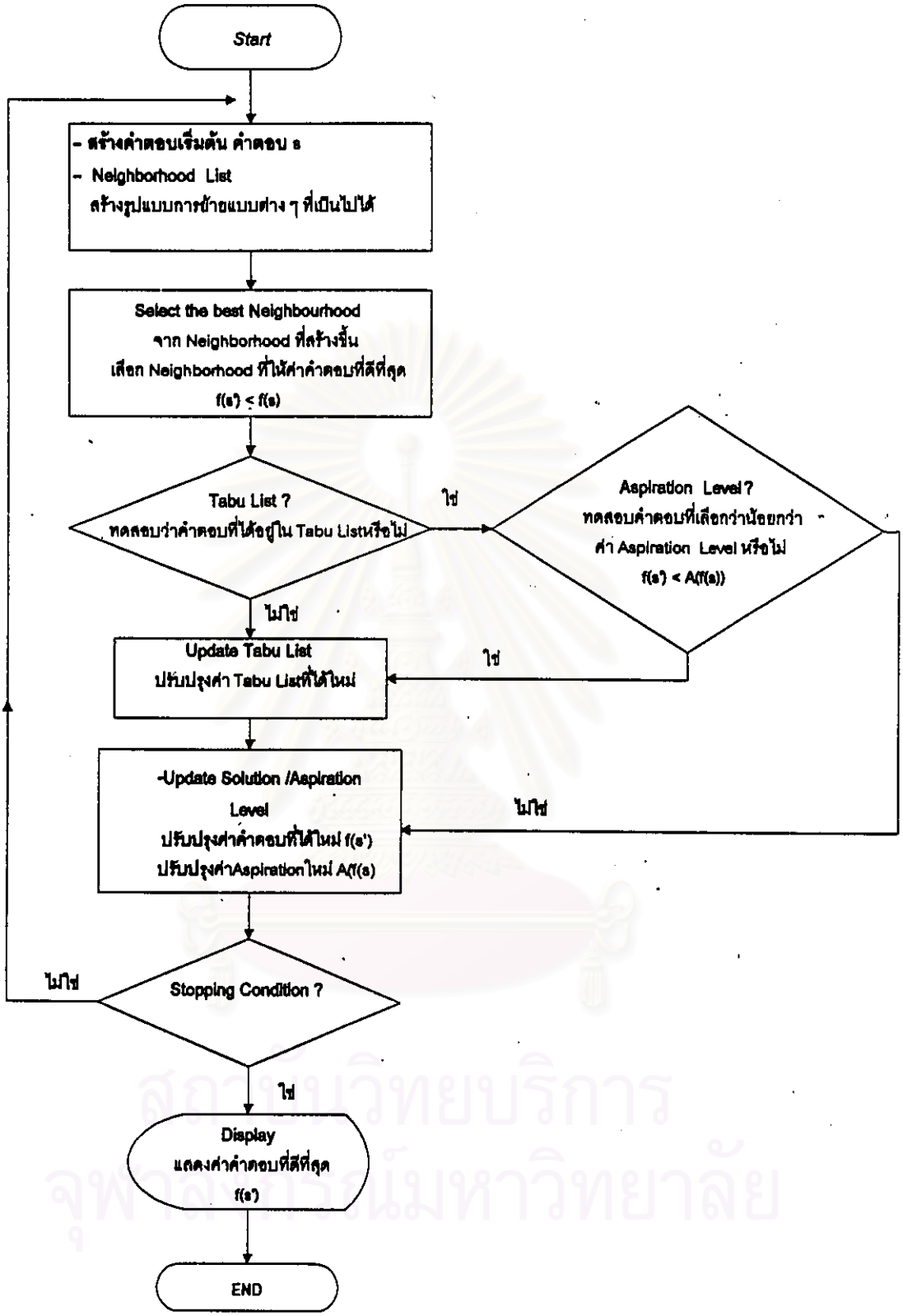
5. แต่อย่างไรก็ตาม การใช้คำตอบทาบูนั้น อาจมีข้อผิดพลาดได้ จึงได้มีการนำคุณสมบัติบางอย่างที่จะทำการยกเลิกสถานะทาบูใน Tabu List ของคำตอบ s ในกรณีที่เราต้องการ

สมมติว่า ฟังก์ชัน f เป็นค่าจำนวนเต็มบวก
เรากำหนดฟังก์ชัน Aspiration A(z) สำหรับทุก ๆ ค่าจำนวนเต็มบวก z
โดยกำหนดค่าเริ่มต้นที่ A(z) = z-1

ถ้าคำตอบ s' อยู่ใน Tabu list เราอาจจะหยุดสถานะทาบูไว้ แล้วตรวจสอบดูว่ามีการย้ายที่เป็นไปได้จากคำตอบ s ในกรณีที่ $f(s') \leq A(f(s))$ ซึ่งหมายความว่าการย้ายจะถูกยอมรับ ถ้าค่าของ f ที่ s' มีค่าน้อยกว่าค่าที่กำหนดไว้

6. หลังจากการย้ายและหาคำตอบตามเงื่อนไขของ Neighborhood List แล้ว จะทำการปรับปรุงค่าของ Tabu List และค่าคำตอบที่เป็นไปได้ใหม่

7. ในเงื่อนไขการหยุดการทำงานนั้นจะต้องถูกกำหนดไว้ โดยในกรณีแรก ถ้าค่าของ f ที่ได้มีค่าใกล้เคียงกับค่า Lower Bound, $\hat{f}$ (เป็นค่าการทำงานจริงที่เกิดขึ้น โดยไม่ได้คิดค่าเวลารอคอยต่าง ๆ) เราก็สามารถเลือกค่า f นั้นเป็นค่าที่ต้องการได้ ซึ่งโดยทั่วไปแล้ววิธีนี้จะไม่ให้ค่าที่ถูกต้องมากนัก ดังนั้นจึงได้มีการใช้วิธีการกำหนดจำนวนเต็มบวก nbmax ขึ้น เมื่อมีการหาค่าที่ Iteration nbmax แล้วปรากฏว่าไม่มีแนวโน้มของค่าคำตอบที่ดีที่สุดในทางที่ดีขึ้น เราก็จะหยุดการทำงาน นอกจากวิธีเหล่านี้แล้วเรายังอาจจะใช้วิธี CPU Time Limit คือถ้าเวลาในการประเมินผลทั้งหมดเกินเวลาทำงานของ CPU เราก็สามารถสั่งให้หยุดการทำงานได้ ผังการทำงานของการค้นหาคำตอบแบบทาบูแสดงในรูปที่ 2.2




**รูปที่ 2.2 ผังการทำงานของการค้นหาคำตอบแบบทาบู**

### 2.2.1 Neighborhood List

รูปแบบของการค้นหาคำตอบแบบทาบู จะเป็นการย้ายเพื่อหาคำตอบที่ดีที่สุดของปัญหาที่เราต้องการหาคำตอบ การใช้วิธี Neighborhood นั้นจะช่วยให้เราได้คำตอบที่มีประสิทธิภาพมากที่สุด แต่มันจะต้องใช้เวลาในการประมวลผลมาก ดังนั้นจึงได้มีการนำเอาการค้นหาคำตอบแบบทาบูมาใช้ร่วมกับโครงสร้างซึ่งแบ่งส่วนของ . Neighborhood ออก ในส่วนของ Neighborhood นั้นจะประกอบด้วยรูปแบบการย้ายที่ต้องการ ซึ่งมีโครงสร้างหลายรูปแบบได้แก่

- Single Adjacent Pairwise Interchange
- Swap Pairwise Interchange
- Insertion Interchange
- Reeves algorithm

- **Single Adjacent Pairwise Interchange**

วิธีนี้จะทำการแลกเปลี่ยนตำแหน่ง 2 ตำแหน่งที่อยู่ติดกันได้แก่ตำแหน่งที่ i และ i+1 การย้ายจะถูกกำหนดโดย i และขนาดของ Neighborhood ที่ได้คือ n-1 โดยที่ n เท่ากับขนาดของปัญหา ดังแสดงตังตัวอย่าง

*ปัญหาที่ต้องการ : {1,2,3,4,5,6}*
*วิธีการย้าย : {2,1,3,4,5,6}, {1,3,2,4,5,6}, {1,2,4,3,5,6}, {1,2,3,5,4,6,}*
*{1,2,3,4,6,5}*

วิธีนี้จะใช้การแลกเปลี่ยน 2 ตำแหน่งที่อยู่ติดกัน ยกตัวอย่างเช่นการเปลี่ยนตำแหน่งที่ 3 กับ 4 จะได้ว่า ตำแหน่งที่ i คือ 3 และตำแน่งที่ i+1 คือ 4 ดังนั้นค่าที่ตำแหน่งที่ 4 จะย้ายไปอยู่ตำแหน่งที่ 3 และค่าจากตำแหน่งที่ 3 จะถูกย้ายไปตำแหน่งที่ 4 ซึ่งเท่ากับ *{1,2,4,3,5,6}* ขนาดของ Neighborhood List สามารถคำนวณได้เท่ากับ 6-1 ซึ่งเท่ากับ 5

- **Swap Pairwise Interchange**

ในวิธีนี้จะทำการแลกเปลี่ยนที่ 2 ตำแหน่งคือ ตำแหน่งที่ i และตำแหน่งที่ k การย้ายถูกกำหนดโดย i และ k ขนาดของ Neighborhood ที่ได้คือ n(n-1)/2 ดังแสดงดังตัวอย่าง

*ปัญหาที่ต้องการ : {1,2,3,4,5,6}*

*วิธีการย้าย: {2,1,3,4,5,6}, {3,2,1,4,5,6}, {4,2,3,1,5,6}, {5,2,3,4,1,6},*
*{6,2,3,4,5,1}*
*{1,3,2,4,5,6}, {1,4,3,2,5,6}, {1,5,3,4,2,6}, {1,6,3,4,5,2}*
*{1,2,4,3,5,6}, {1,2,5,4,3,6}, {1,2,6,4,5,3}*
*{1,2,3,5,4,6}, {1,2,3,6,5,4}*
*{1,2,3,4,6,5}*

วิธีนี้จะเป็นการย้ายโดยการสลับตำแหน่งจากตำแหน่งที่ i ไปตำแหน่งที่ k ยกตัวอย่างเช่น ต้องการสลับค่าจากตำแหน่งที่ 3 ไปยังตำแหน่งที่ 5 จะได้ว่า i เท่ากับ 3 และ k เท่ากับ 5 ดังนั้นค่าในตำแหน่งที่ 3 จะโดนย้ายไปตำแหน่งที่ 5 และค่าในตำแหน่งที่ 5 จะถูกย้ายไปตำแหน่งที่ 3 ซึ่งจะได้ค่าดังนี้ *{1,2,5,4,3,6}* ขนาดของ Neighborhood List สามารถคำนวณได้เท่ากับ 6(6-1)/2 ซึ่งเท่ากับ 5

- **Insertion Interchange**

ในวิธีนี้จะเอาค่าในตำแหน่งที่ i ไปใส่ในตำแหน่งที่ k การย้ายถูกกำหนดโดย i และ k ขนาดของ Neighborhood ที่ได้คือ $(n-1)^2$ ดังแสดงตามตัวอย่าง

*ปัญหาที่ต้องการ : {1,2,3,4,5,6}*

*วิธีการย้าย: {2,1,3,4,5,6}, {2,3,1,4,5,6}, {2,3,4,1,5,6}, {2,3,4,5,1,6},*
*{2,3,4,5,6,1}*
*{1,3,2,4,5,6}, {1,3,4,2,5,6}, {1,3,4,5,2,6}, {1,3,4,5,6,2}*
*{3,1,2,4,5,6}, {1,2,4,3,5,6}, {1,2,4,5,3,6}, {1,2,4,5,6,3}*
*{4,1,2,3,5,6}, {1,4,2,3,5,6}, {1,2,3,5,4,6}, {1,2,3,5,6,4}*
*{5,1,2,3,4,6}, {1,5,2,3,4,6}, {1,2,5,3,4,6}, {1,2,3,4,6,5}*
*{6,1,2,3,4,5}, {1,6,2,3,4,5}, {1,2,6,3,4,5}, {1,2,3,6,4,5}*

วิธีนี้จะเป็นการนำเอาตำแหน่งที่ i ไปแทรกในตำแหน่งที่ k ยกตัวอย่างเช่น ต้องการย้ายตำแหน่งที่ 6 ไปตำแหน่งที่ 3 จะได้ว่าค่า i เท่ากับ 6 และค่า k เท่ากับ 3 ซึ่งจะได้คำตอบคือ *{1,2,6,3,4,5}* ขนาดของ Neighborhood List สามารถคำนวณได้เท่ากับ $(6-1)^2$ ซึ่งเท่ากับ 25

- **Reeves Algorithm**

ถ้าปัญหาที่มีอยู่ค่อนข้างใหญ่ ในวิธีข้างต้นจะทำให้มี Neighborhood ค่อนข้างมาก ทำให้เปลืองหน่วยความจำในการจัดเก็บ Reeves (1993) จึงได้พยายามลดขนาดของ Neighborhood ลง โดยเราสามารถตรวจสอบผลของการ Remove และ Reinsert ของแต่ละงาน เมื่อ $(j_1, j_2, \ldots\ldots, j_p)$ ที่ $p<n$ ย้ายเพื่อหาคำตอบที่ดีที่สุด แล้วย้าย $(j_{p+1}, j_{p+2}, \ldots\ldots, j_{2p})$ และอื่น ๆ ดังแสดงดังตัวอย่าง

*ปัญหาที่ต้องการ : {1,2,3,4,5,6}*

*วิธีการย้าย : กำหนด p = n/2*

*{2,1,3,4,5,6}, {2,3,1,4,5,6}, {2,3,4,1,5,6}, {2,3,4,5,1,6}, {2,3,4,5,6,1}*

*{1,3,2,4,5,6}, {1,3,4,2,5,6}, {1,3,4,5,2,6}, {1,3,4,5,6,2}*

*{3,1,2,4,5,6}, {1,2,4,3,5,6}, {1,2,4,5,3,6}, {1,2,4,5,6,3}*

และ

*{4,1,2,3,5,6}, {1,4,2,3,5,6}, {1,2,3,5,4,6}, {1,2,3,5,6,4}*

*{5,1,2,3,4,6}, {1,5,2,3,4,6}, {1,2,5,3,4,6}, {1,2,3,4,6,5}*

*{6,1,2,3,4,5}, {1,6,2,3,4,5}, {1,2,6,3,4,5}, {1,2,3,6,4,5}*

ในวิธีนี้จะมีวิธีการย้ายเหมือนกับวิธี Insertion Interchange ที่ได้กล่าวแล้วข้างต้น ส่วนที่แตกต่างกันก็คือ การย้ายจะถูกแบ่งออกเป็น 2 ส่วนโดยแบ่งจากขนาดของปัญหา หลังจากแบ่งแล้วจึงทำการย้าย ตามวิธีของ Neighborhood ต่าง ๆ

### 2.2.2 Tabu List

ในการหาคำตอบที่ดีที่สุดโดยใช้การค้นหาคำตอบแบบทาบูนั้น จะมีการย้ายเกิดขึ้นภายในโครงสร้างของ Neighborhood (ซึ่งมีอยู่หลายแบบได้แก่ Single Adjacent Pairwise Interchange, Swap Pairwise Interchange, Insertion Interchange, Reeves Algorithm ดังได้กล่าวมาแล้วข้างต้น) เพื่อที่จะป้องกันไม่ให้เกิดการย้ายที่ซ้ำซ้อนขึ้น จึงได้มีการกำหนดค่าที่เคยย้ายแล้วไว้ใน Tabu List ขนาดของ Tabu List (Tabu List Size) ที่เหมาะสมจะขึ้นอยู่กับปัญหา โดยทั่วไปจะกำหนดไว้เท่ากับ 7

ในบางครั้ง เมื่อทำการย้ายแล้วปรากฏว่า ค่าคำตอบที่ได้ของการย้ายไม่มีแนวโน้มในทางที่ดีขึ้น เราจึงได้มีการกำหนด Dynamic Tabu List ขึ้น เพื่อที่จะทำการลบข้อมูลใน Tabu List เดิมออก แล้วทำการกำหนดขนาดของ Tabu List ใหม่ที่เหมาะสม

### 2.2.3 Aspiration Criteria

จากการที่เรากำหนด Tabu List ขึ้นจะทำให้เราไม่สามารถย้ายกลับไปยังตำแหน่งเดิมได้ แต่ในบางครั้งข้อห้ามใน Tabu List อาจถูกยกเลิกได้ ถ้ามีเงื่อนไขที่ดีพอ ซึ่งเงื่อนไขเหล่านั้นจะต้องถูกกำหนดโดย Aspiration Level Condition

เรากำหนดฟังก์ชัน Aspiration A(z) สำหรับทุก ๆ ค่าจำนวนเต็มบวก z
โดยกำหนดค่าเริ่มต้นที่ A(z) = z-1

ถ้าคำตอบ s' อยู่ใน Tabu list เราอาจจะหยุดสถานะทามูไว้ แล้วตรวจสอบดูว่ามีการย้ายที่เป็นไปได้จากคำตอบ s ในกรณีที่ $f(s') \leq A(f(s))$ ซึ่งหมายความว่าการย้ายจะถูกยอมรับถ้าค่าของ f ที่ s' มีค่าน้อยกว่าค่าAspiration ที่กำหนดไว้

### 2.2.4 Stopping Condition

Glover (1989) ได้กำหนดเงื่อนไขต่าง ๆ ในการหยุดการย้ายไว้หลายวิธีได้แก่

- **Iteration Limit**
  จะเป็นการหยุดเนื่องจากจำนวนของ Iteration มากกว่าค่าที่กำหนดไว้
- **No-Improvement Criterion**
  จะเป็นการหยุด ถ้าไม่มีการเปลี่ยนแปลงของคำตอบในทางที่ดีขึ้น
- **CPU Time Limit**
  จะหยุด ถ้าเวลาในการประมวลผลเกินเวลาที่ CPU กำหนด
- **Low Bound Limit**
  จะหยุด ถ้าค่าคำตอบที่ได้มีค่าใกล้เคียงกับค่า Low Bound ที่กำหนดไว้

## 2.3 ต้นทุน

ในระบบการผลิตโดยทั่วไปแล้ว สิ่งที่สำคัญสิ่งหนึ่งที่ต้องพิจารณาถึงก็คือต้นทุนในการผลิต ซึ่งในระบบการผลิตแบบเซลลูลาร์นั้น ต้นทุนที่จะเกิดขึ้นได้จะมี 2 ส่วนคือ ต้นทุนที่เกิดจากเครื่องจักร และส่วนประกอบที่จะนำไปใช้ผลิตในเครื่องจักรนั้น ซึ่งในการศึกษานี้จะพิจารณาถึงรูปแบบของต้นทุน 2 แบบคือ

### 2.3.1 รูปแบบต้นทุนแบบที่ 1

- ต้นทุนหนี้สินของเครื่องจักร

  โดยต้นทุนในส่วนนี้ เราจะพิจารณาถึงต้นทุนหนี้สินของเครื่องจักรที่ใช้ในการผลิตชิ้นงานนั้น

- ต้นทุนการผลิตชิ้นงาน

  จะเป็นต้นทุนในส่วนการผลิตชิ้นงานนั้นขึ้นมา ซึ่งประกอบไปด้วย จำนวนความต้องการชิ้นงานต่อปี และต้นทุนที่ใช้ในการผลิตชิ้นงานหนึ่ง ๆ ตามแผนกระบวนการผลิตและเครื่องจักรที่กำหนดให้ผลิตชิ้นงานนั้น

### 2.3.2 รูปแบบต้นทุนแบบที่ 2

การพิจารณาต้นทุนในรูปแบบที่ 2 นี้ เราจะมีข้อกำหนดเพื่อง่ายต่อการพิจารณาดังนี้

- ชิ้นงานแต่ละชิ้น จะมีการกำหนดลำดับการผลิตด้วยเวลาในการติดตั้ง และขั้นตอนการผลิต
- Planning Horizon จะไม่มีที่สิ้นสุด และค่าความต้องการของตลาดจะเป็นค่าคงที่
- จะไม่มีการขาดแคลนของวัสดุเกิดขึ้น
- แต่ละรุ่นของชิ้นงานทั้งหมดต้องอยู่ที่สถานีการผลิต (Work Station) ก่อนการผลิต
- แต่ละชิ้นงานจะเป็นของ Initial Coding Family และต้นทุนในการติดตั้งภายใน ครอบครัวชิ้นงาน จะถูกกว่าต้นทุนในการติดตั้งระหว่างครอบครัว

ซึ่งรูปแบบต้นทุนนี้จะถูกแบ่งออกเป็น

- ต้นทุนคงที่ (Fix Cost) ประกอบด้วย
  - *ต้นทุนคงที่เครื่องจักร*
    จะขึ้นอยู่กับชนิดของเครื่องจักรที่ใช้ในการผลิตชิ้นงานนั้น ๆ

- ต้นทุนผันแปร (Variable Cost) ประกอบด้วย
  - *ต้นทุนพัสดุคงคลัง (Inventory Cost)*

    ค่าของต้นทุนพัสดุคงคลังของงานระหว่างทำ ขึ้นอยู่กับแผนการผลิต และเวลาในการผลิต ซึ่งจะประกอบด้วยค่า Appropriate Percentageซึ่งหาได้จากการเปรียบเทียบต้นทุนพัสดุคงคลังของงานระหว่างทำ ครั้งหลังที่เกิดขึ้นจริงกับปริมาณการผลิตที่วางแผนไว้ ต้นทุนต่อหน่วย และการการขนส่งต่อหน่วย
  - *ต้นทุนการผลิตผันแปร (Production Variable Cost)*

    ต้นทุนการผลิตผันแปรเกี่ยวข้องกับแต่ละเครื่องจักรคือแฟกเตอร์ต้นทุนการทำงานผันแปรของเครื่องจักร และ Utilization Time จะขึ้นอยู่กับเวลาในการทำงาน และอัตราความต้องการของชิ้นงาน
  - *ต้นทุนในการติดตั้ง (Set-Up Cost)*

    ค่าใช้จ่ายต้นทุนในการติดตั้ง สำหรับแต่ละชิ้นงานในกลุ่มหนึ่ง ถูกกำหนดโดยผลรวมของเครื่องจักรทั้งหมดที่ใช้ในการผลิตชิ้นงานนั้น โดยแบ่งเป็น :-
    - ต้นทุนการติดตั้งภายในครอบครัวของชิ้นงาน
    - ต้นทุนการออกแบบการติดตั้งครอบครัวของชิ้นงาน ซึ่งถูกแบ่งโดยจำนวนของชิ้นงานในครอบครัวเดียวกัน ซึ่งถูกผลิตบนเครื่องจักรนั้น