บทที่ 4
ผลการวิจัย

4.1 ผลการออกแบบอัลกอริทึม

ผู้วิจัยได้มีการออกแบบอัลกอริทึมในสองส่วนด้วยกันคือ 1) อัลกอริทึมในส่วนของการหาคำตอบเริ่มต้นโดยกำหนดเลือกตำแหน่งที่จะเปิดโรงงาน และ 2) อัลกอริทึมในส่วนของการปรับปรุงคำตอบที่ได้ ซึ่งมีรายละเอียดดังนี้

4.1.1 การหาคำตอบเริ่มต้น

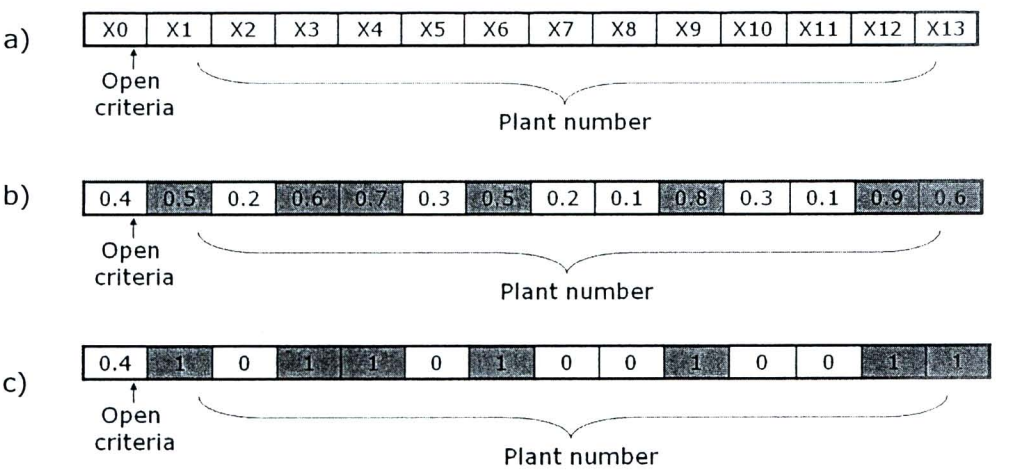
ผู้วิจัยได้ประยุกต์ใช้กระบวนการหาคำตอบเริ่มต้นซึ่งได้แรงบันดาลใจจากวิธีหาคำตอบของวิธีพาร์ติเคิลสวอร์มออพติไมเซชั่น (PSO) โดยการกำหนดให้มีจำนวนพาร์ติเคิลเท่ากับจำนวนของโรงงานน้ำตาลที่มีศักยภาพจะเปิดเป็นโรงงานเอทานอลบวกหนึ่งพาร์ติเคิล (n+1) ซึ่งพาร์ติเคิลที่บวกเพิ่มมานั้นจะทำหน้าที่เป็นตัวตัดสินใจว่าจะเปิดหรือไม่เปิดโรงงานใดโดยเปรียบเทียบค่าในพาร์ติเคิลที่เหลือ หากมีค่ามากกว่าให้กำหนดว่าโรงงานนั้นๆ “เปิด” เป็นโรงงานเอทานอล และเปลี่ยนค่าให้เป็น 1 ในทางกลับกันหากมีค่าน้อยกว่าให้กำหนดว่าโรงงานนั้นๆ ไม่เปิดเป็นโรงงานเอทานอล และเปลี่ยนค่าให้เป็น 0 นั่นเอง

เนื่องจากคำตอบเริ่มต้นที่ได้อยู่ในรูปของค่า 0 และ 1 ซึ่งสามารถนำไปประยุกต์ใช้อัลกอริทึมอื่นๆ ในการปรับปรุงคำตอบได้ง่ายขึ้น ผู้วิจัยจึงเรียกการหาคำตอบเริ่มต้นโดยวิธีนี้ว่า “การหาคำตอบเริ่มต้นแบบสุ่มโดยให้ค่าคำตอบเริ่มต้นเป็นไบนารี” (Randomized Binary Selection; RBS) โดยรหัสเทียมสำหรับอัลกอริทึมดังกล่าว แสดงในรูปที่ 4.1

```
Procedure RBS
    Generate 1 dimension array with n column
    Random number in each column (X0 – Xn)
        for i =2, i<n, do;
        if x_n<x_0 then x_n= 0, else =1;
        end for loop;
    end
```

รูปที่ 4.1 รหัสเทียมของ Randomized Binary Selection Algorithm

การทำงานของ RBS จะเริ่มจากการสร้าง array ซึ่งมีจำนวนคอลัมน์เท่ากับ n จากนั้นจึงทำการสุ่มค่าตั้งแต่ 0.0 – 1.0 ลงไปในแต่ละ array เมื่อได้ค่าของแต่ละ array แล้วจึงจะเริ่มทำการเปรียบเทียบค่ากับค่าที่อยู่ใน array ที่ 0 ซึ่งเป็นตัวตัดสินใจว่าจะเปิดหรือไม่เปิดโรงงานเอทานอลนั่นเอง โดยหากค่าใน array นั้นๆ มีค่าน้อยกว่าค่าใน array ตัดสินใจจะทำการเปลี่ยนค่านั้นให้เป็น 0 และหากค่านั้นมีค่ามากกว่าหรือเท่ากับก็จะเปลี่ยนค่าให้เป็น 1 ดังแสดงในรูปที่ 4.2



รูปที่ 4.2 การทำงานของ RBS Algorithm

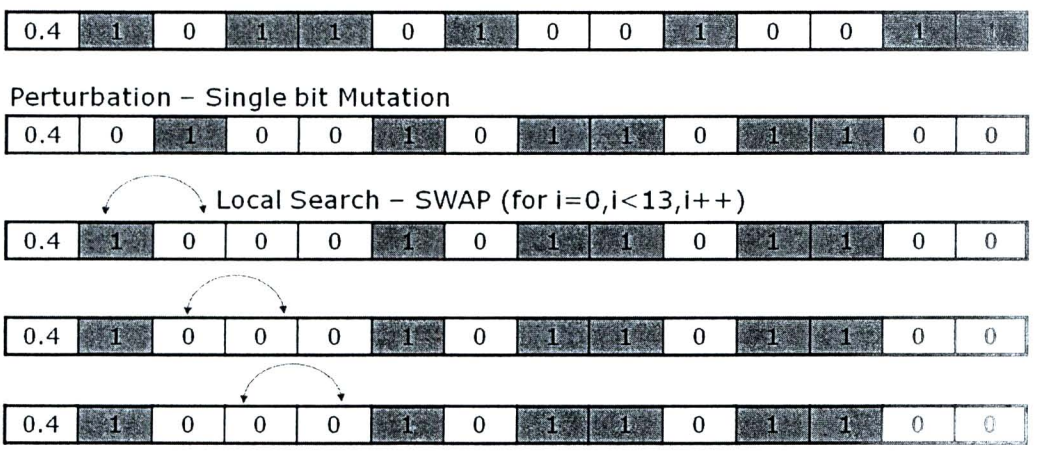
4.1.2 การปรับปรุงคำตอบที่ได้ด้วยวิธีการค้นหาคำตอบเฉพาะที่ซึ่งมีการทำงานแบบวนซ้ำ

ผู้วิจัยเลือกวิธีการค้นหาคำตอบเฉพาะที่ซึ่งมีการทำงานแบบวนซ้ำ (Iterated Local Search; ILS) เนื่องจากเป็นวิธีการที่ไม่ซับซ้อนและให้ผลเฉลยที่ยอมรับได้จากการทบทวนวรรณกรรมที่เกี่ยวข้อง โดยขั้นตอนของอัลกอริทึมดังกล่าวจะเริ่มต้นจากการสร้างคำตอบเริ่มต้น ซึ่งในที่นี้จะใช้อัลกอริทึมแบบ RBS ดังที่กล่าวไปแล้วในหัวข้อที่ผ่านมา จากนั้นจึงทำการค้นหาคำตอบแบบวนซ้ำและเก็บคำตอบที่ดีที่สุด (S*) ไว้ โดยหากไม่สามารถหาคำตอบที่ดีกว่าได้ในรอบการวนซ้ำนั้นๆ ก็จะทำการ perturbation ซึ่งในที่นี้จะใช้การสลับค่าในแต่ละพาร์ติเคิลหรือ array ซึ่งจากเดิมเป็น 0 หรือ 1 ให้กลายเป็นค่าตรงข้าม และทำการค้นหาคำตอบเฉพาะที่แบบวนซ้ำอีกครั้งจนกว่าจะได้คำตอบที่ดีกว่าเพื่อที่จะเก็บคำตอบนั้นไว้ และหาคำตอบไปเรื่อยๆ จนครบจำนวนรอบที่กำหนด

ด้วยเหตุที่มีการใช้การ perturbation แบบสลับค่านี่เอง ผู้วิจัยจึงเรียกวิธีการค้นหาคำตอบเฉพาะที่ซึ่งมีการทำงานแบบวนซ้ำแบบนี้ว่า "การค้นหาคำตอบเฉพาะที่ซึ่งมีการทำงานแบบวนซ้ำโดยวิการสลับตำแหน่งแบบง่ายและประยุกต์ใช้การกลายพันธุ์ในหน่วยเดี่ยว" (Iterated local search with simply swap and sing bit mutation; ILS-SS-SBM) ซึ่งได้แรงบันดาลใจจากวิธีการค้นหาคำตอบเชิงพันธุกรรม (Genetic Algorithm) ซึ่งจะมีการ Mutation และ Crossover เป็นแนวคิดหลัก โดยรหัสเทียมของอัลกอริทึม ILS-SS-SBM นี้แสดงในรูปที่ 4.3 และภาพแสดงการทำงานของอัลกอริทึมจะแสดงในรูปที่ 4.4

```
Procedure ILS-SS-SBM
    Generate Initial Solution (Random) ->S0
    LocalSearch (Simply SWAP) -> S*
    repeat
    Perturbation (Single Bit Mutation) -> S'
    LocalSearch (Simply SWAP) -> S*'
    Acceptance Criterion (if S*' < S* then S*' -> S*)
    until terminate condition met (no. of iterations)
end
```

รูปที่ 4.3 รหัสเทียมสำหรับอัลกอริทึม ILS-SS-SBM



รูปที่ 4.4 การทำงานของอัลกอริทึม ILS-SS-SBM

สำหรับรูปแบบการค้นหาเฉพาะที่ (Local search) นั้น ผู้วิจัยได้เลือกใช้การค้นหาเฉพาะที่แบบสลับตำแหน่งระหว่างจุดสองจุดหรือ 2-opt ซึ่งเป็นรูปแบบการค้นหาเฉพาะที่แบบที่พบได้ทั่วไปในอัลกอริทึมของงานวิจัยอื่นๆ ซึ่งจากงานวิจัยของ Hansen และ Mladenovic (2007) ได้ระบุว่าสำหรับการค้นหาเฉพาะที่ซึ่งใช้คำตอบเริ่มต้นจากการสุ่มนั้น การใช้ 2-opt แบบ first improvement จะให้คำตอบที่ดีกว่าและเร็วกว่าแบบ best improvement ดังนั้นผู้วิจัยจึงเลือกใช้การค้นหาเฉพาะแบบ 2-opt ในลักษณะของ first improvement หรือ greedy ascent นั่นเองซึ่งจะต่างจากการค้นหาด้วยวิธี steepest ซึ่งจะมีการใช้คำตอบที่ให้ค่าเป้าหมายวัตถุประสงค์ดีที่สุดมาเป็นคำตอบเริ่มต้น หรือที่เรียกอีกชื่อหนึ่งว่า best improvement

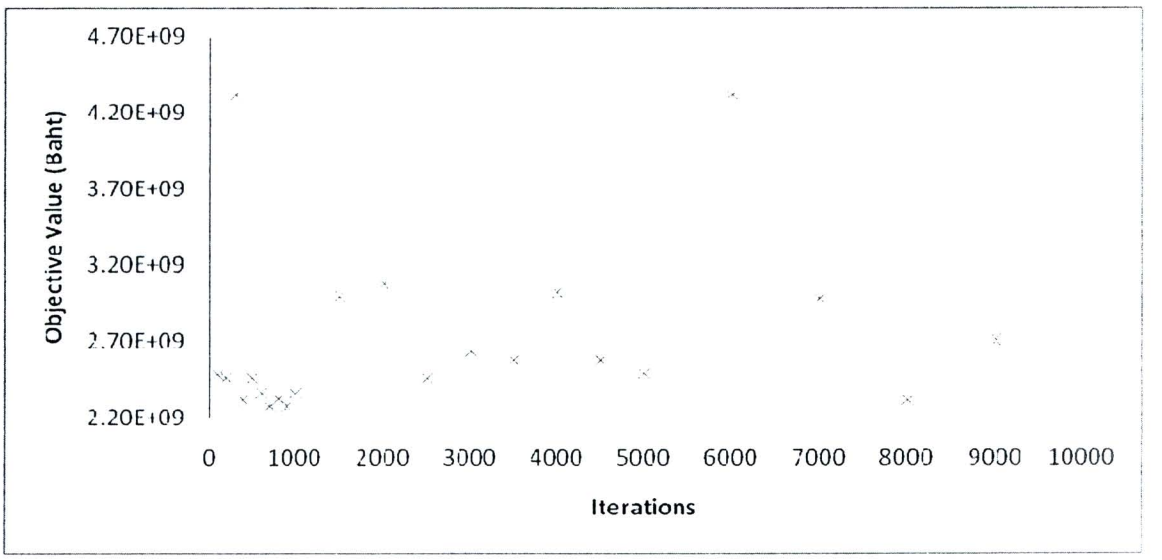
4.2 ผลการทดสอบอัลกอริทึม

ผู้วิจัยได้ทำการทดสอบอัลกอริทึมโดยออกแบบการทดลองให้มีการกำหนดรอบที่จะใช้ในการวนซ้ำไว้ที่รอบต่างๆ กันตั้งแต่ 100 รอบจนถึง 10,000 รอบ พบว่าผลที่ได้เป็นแสดงในตารางที่ 4.1

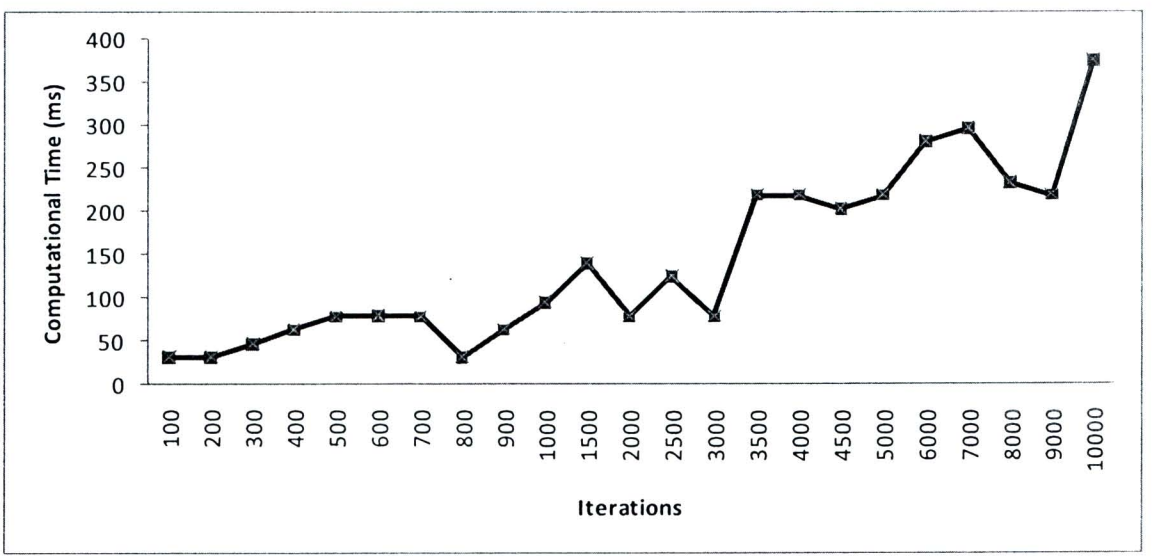
ตารางที่ 4.1 ผลการทดสอบอัลกอริทึม ILS-SS-SBM

Iteration	Plant Open	Objective Value	Computational Time (ms)
100	05,11,13	2.4835e9	32
200	04,05,13	2.4606e9	31
300	All plants	4.3119e9	47
400	02,13	2.3209e9	63
500	04,05,13	2.4606e9	78
600	05,13	2.3688e9	79
700	02	2.2809e9	78
800	01,02	2.3271e9	31
900	02	2.2809e9	63
1000	05,13	2.3688e9	94
1500	01,02,03,05,09,13	3.0008e9	140
2000	01,03,05,09,11,13	3.0764e9	78
2500	04,05,13	2.4606e9	125
3000	04,05,11,13	2.6335e9	78
3500	02,05,11,13	2.5823e9	219
4000	02,04,05,06,11,13	3.0246e9	219
4500	02,05,11,13	2.5823e9	203
5000	02,06,13	2.4947e9	219
6000	All plants	4.3119e9	281
7000	01,02,05,10,11,13	2.9849e9	297
8000	02,13	2.3209e9	234
9000	01,02,05,11,13	2.7098e9	219
10000	01,03,05,09,11,13	3.0764e9	375

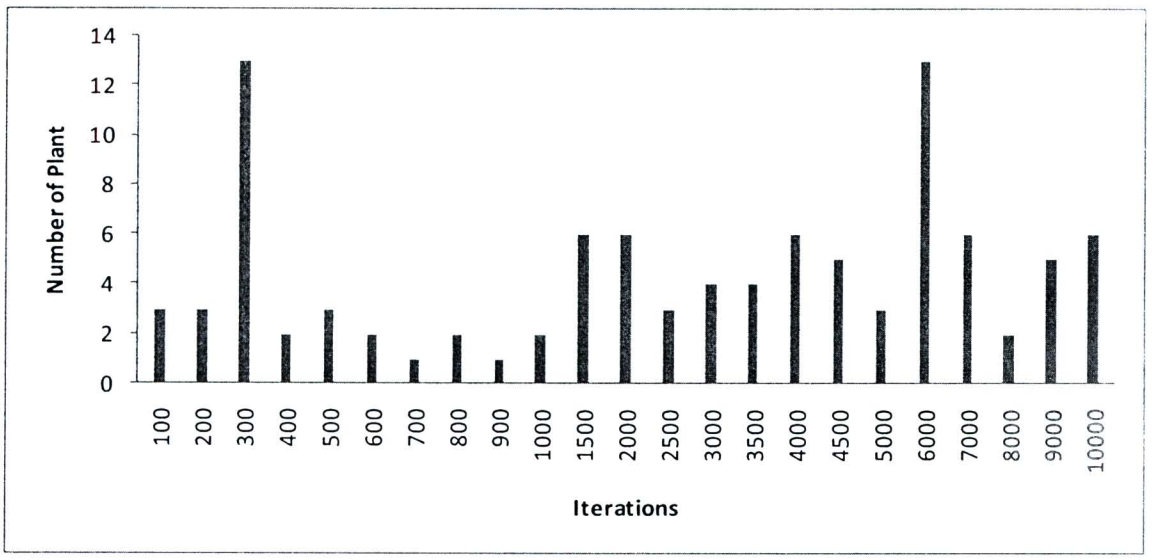
ผู้วิจัยได้ทำค่าฟังก์ชันวัตถุประสงค์ที่ได้ไปทำการแสดงผลในรูปของแผนภาพการกระจาย, นำเวลาในการคำนวณไปแสดงผลในรูปของกราฟเส้น และนำจำนวนโรงงานที่เปิดไปแสดงผลในรูปของกราฟแท่ง ดังแสดงในรูปที่ 4.5, 4.6 และ 4.7 ตามลำดับ



รูปที่ 4.5 ค่าของฟังก์ชันวัตถุประสงค์ที่ได้จากคำตอบที่การรันรอบต่างๆ ของ ILS-SS-SBM



รูปที่ 4.6 เวลาที่ใช้ในการรันรอบต่างๆ กันของ ILS-SS-SBM



รูปที่ 4.7 จำนวนโรงงานที่เปิดตามรอบที่รันด้วย ILS-SS-SBM

อย่างไรก็ตาม ผู้วิจัยพบว่าผลเฉลยที่ได้ยังไม่ดีนัก จึงได้ทำการปรับปรุงอัลกอริทึมโดยการเพิ่มขั้นตอนตัดสินใจให้มีการออกจากรอบการค้นหาเมื่อพบว่าค้นหาไปแล้วไม่ได้คำตอบที่ดีขึ้นมากกว่า 5 รอบขึ้นไป โดยเมื่อออกจากรอบการค้นหาปกติแล้วให้ไปเริ่มต้นสุ่มค่าใหม่ หรือไปเริ่มต้นที่ RBS Algorithm ใหม่นั่นเอง ซึ่งวิธีการดังกล่าวนี้จะทำให้เกิดการค้นหาในพื้นที่ของคำตอบใหม่ได้เร็วขึ้น ทำให้ไม่ติดอยู่ในวังวนของพื้นที่คำตอบเดิมซึ่งไม่ได้ทำให้คำตอบที่ได้มีค่าดีกว่าแต่อย่างใด โดยอัลกอริทึมที่ปรับปรุงนี้มีชื่อว่า ILS-SS-SBMRW ซึ่งมีรหัสเทียมดังแสดงในรูปที่ 4.8

```
Procedure ILS-SS-SBMRW
    Generate Initial Solution (Random) ->S0
    LocalSearch (Simply SWAP) -> S*
    repeat
    Perturbation (Single Bit Mutation) -> S'
    LocalSearch (Simply SWAP) -> S*'
    Acceptance Criterion (if S*' < S* then S*' -> S*)
    until terminate condition met (no. of iterations)
    if no update S* >5 times
    then regenerate initial solution
end
```

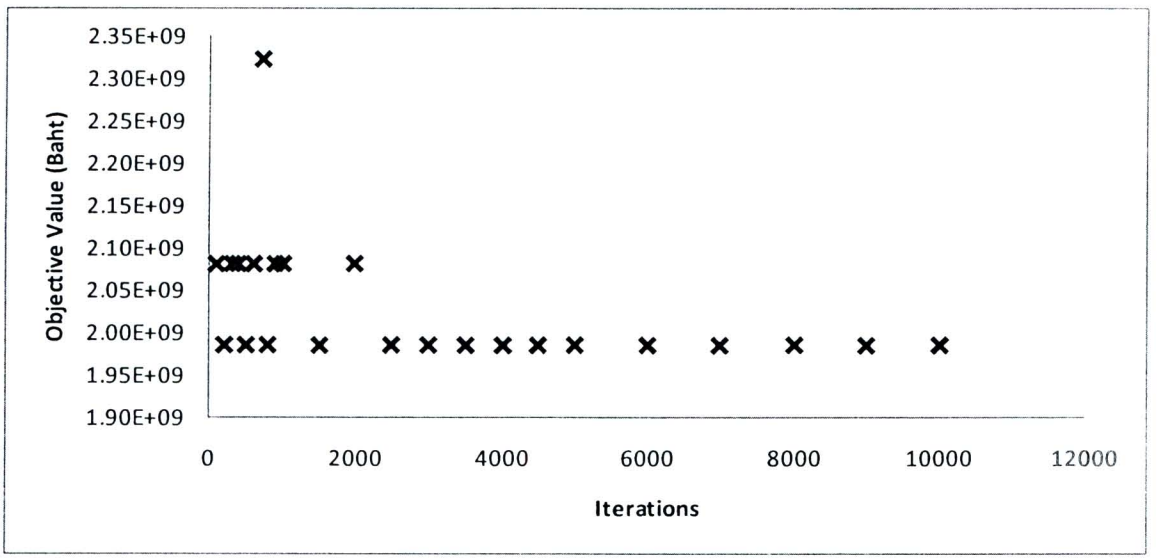
รูปที่ 4.8 รหัสเทียมของอัลกอริทึม ILS-SS-SBMRW

ผู้วิจัยได้ทำการทดสอบอัลกอริทึม ILS-SS-SBMRW โดยออกแบบการทดลองให้มีการกำหนดรอบที่จะใช้ในการวนซ้ำไว้ที่รอบต่างๆ กันตั้งแต่ 100 รอบจนถึง 10,000 รอบ เช่นเดียวกันกับอัลกอริทึม ILS-SS-SBM ที่ผ่านมา เพื่อสังเกตว่าจะมีการเปลี่ยนในแง่ของคำตอบที่ได้, การลู่เข้าสู่คำตอบที่ดีที่สุด, จำนวนโรงงานที่เปิด และเวลาในการคำนวณที่ใช้อย่างไรบ้าง ซึ่งจากการทดลองพบว่ามีผลแสดงดังตารางที่ 4.2

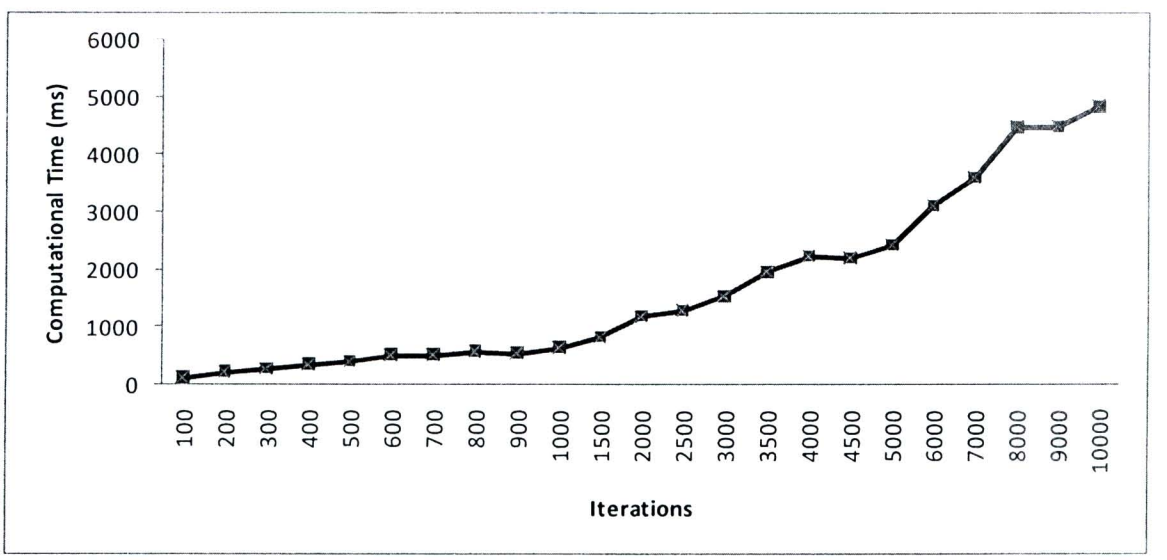
ตารางที่ 4.2 ผลการทดสอบอัลกอริทึม ILS-SS-SBMRW

Iteration	Plant Open	Objective Value	Computational Time (ms)
100	02	2.0809e9	125
200	01,02	1.9851e9	234
300	02	2.0809e9	281
400	02	2.0809e9	359
500	01,02	1.9851e9	406
600	02	2.0809e9	516
700	02,13	2.3209e9	516
800	01,02	1.9851e9	578
900	02	2.0809e9	563
1000	02	2.0809e9	656
1500	01,02	1.9851e9	828
2000	02	2.0809e9	1,188
2500	01,02	1.9851e9	1,297
3000	01,02	1.9851e9	1,546
3500	01,02	1.9851e9	1,969
4000	01,02	1.9851e9	2,234
4500	01,02	1.9851e9	2,203
5000	01,02	1.9851e9	2,437
6000	01,02	1.9851e9	3,125
7000	01,02	1.9851e9	3,609
8000	01,02	1.9851e9	4,485
9000	01,02	1.9851e9	4,500
10000	01,02	1.9851e9	4,844

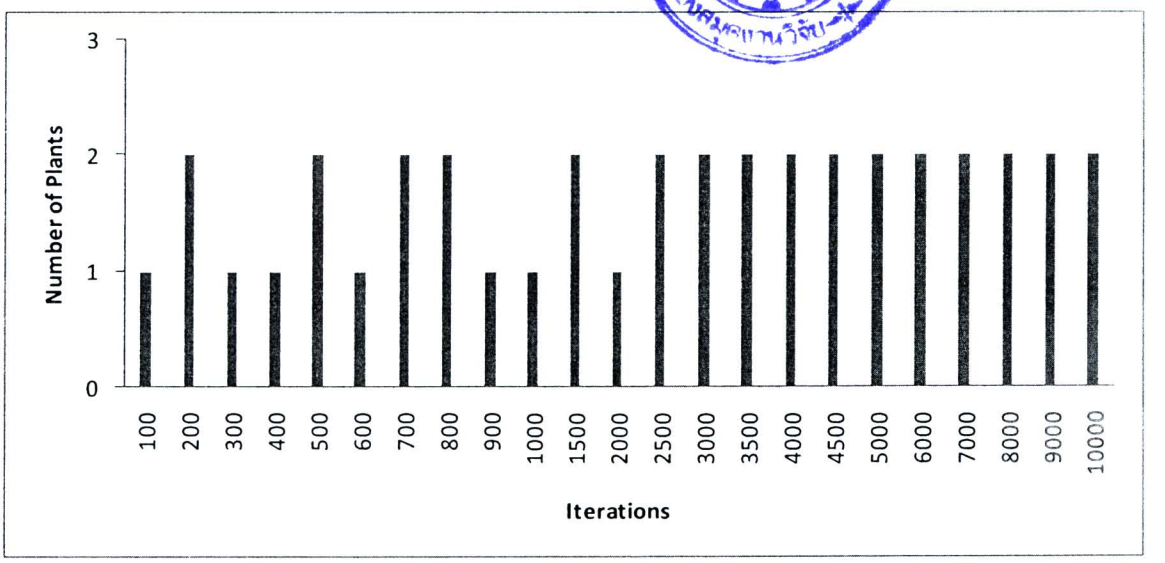
เช่นเดียวกันกับ ILS-SS-SBM ผู้วิจัยได้ทำค่าฟังก์ชันวัตถุประสงค์ที่ได้ไปทำการแสดงผลในรูปของแผนภาพการกระจาย, นำเวลาในการคำนวณไปแสดงผลในรูปของกราฟเส้น และนำจำนวนโรงงานที่เปิดไปแสดงผลในรูปของกราฟแท่ง ดังแสดงในรูปที่ 4.9, 4.10 และ 4.11 ตามลำดับ ซึ่งจะได้ทำการวิเคราะห์ผลที่ได้ในบทต่อไป



รูปที่ 4.9 ค่าของฟังก์ชันวัตถุประสงค์ที่ได้จากคำตอบที่การรันรอบต่างๆ ของ ILS-SS-SBMRW



รูปที่ 4.10 เวลาที่ใช้ในการรันรอบต่างๆ กันของ ILS-SS-SBMRW



รูปที่ 4.11 จำนวนโรงงานที่เปิดตามรอบที่รันด้วย ILS-SS-SBMRW