<u>อธิบายหลักการทำงานที่เลียนแบบการทำงานที่มีอยู่ในทางชีวภาพ</u>

ในทางชีวภาพของพืช

พืชส่วนใหญ่นั้น ก็มีแนวโน้มที่จะเคลื่อนที่ไปในทิศทางของดวงอาทิตย์ซึ่งชาวกรีกสมัยการเรียกปรากฏการนี้ว่า "Heliotropium" เพราะฉะนั้นเราจะเรียกพฤติกรรมนี้ว่า Heliotropium เช่น เดียว กับมนุษย์ (biological clock) พืชก็มีนาฬิกาชีวิตภาพ "Circadian Rhythm" ซึ่งการเต้นของหัวใจมนุษย์ทำให้เกิดการเปลี่ยนแปลง ทางสรีรวิทยาและทางเคมีหลายอย่างในขณะเดียวกันพืชก็มีการตอบสนองใน24ช.ม

ในทางชีวภาพของดอกทานตะวัน

ก่อนรุ่งสาง ดอกทานตะวันจะหันหน้าไปทางทิศตะวันออกในทางที่ดวงอาทิตย์ขึ้นและเมื่อดวงอาทิตย์เคลื่อนตัวไปยังทิศตะวันตกดอกทานตะวันก็จะหันเช่นกันและเมื่อตะวันตกดินดอกทานตะวันก็จะกลับไปยังทางทิศตะวันออกเพื่อเริ่มวนอีกครั้งในวันต่อๆไป นักวิจัยเชื่อว่า ดอกทานตะวัน แสดงให้ถึงปรากฏการณ์ Heliotropium เนื่องจากลำต้นของมันยืดออกในอัตราที่ต่างกัน ในแต่ละช่วงเวลาซึ่งในช่วงเวลาของดวงอาทิตย์ขึ้นในทิศตะวันออกไปทิศตะวันตก ด้านตะวันออกลำตัวของมันจะเติบโตเร็วกว่าข้างทิศตะวันตก เนื่องจากการเจริญเติบโตทั้ง2ข้างไม่เท่ากันดอกทาตะวันเลยโค้งงอไปในทิศของดวงอาทิตย์นั้นๆเพราะก้านของดอกทานตะวันจะเจริญเติบโตไม่เท่ากัน
เนื่องจาก AUXINS (ฮอร์โมนพืชที่กระตุ้นการเจริญเติบโต)ไม่เท่ากันทั้ง2ข้างและเมื่อดอกทานตะวันต้นนั้นแก่ลงการเจริญเติบโตก็จะลดลงไปเรื่อยๆและจะไม่หันไปอีก ดอกทานตะวันถ้าหันไปทางทิศตะวันออกจะดึงดูดผึ้งได้ดีให้ผสมเกศร

ในทางระบบหรืออัลกอริทึมที่เกี่ยวข้อง

เราจะเปรียบเทียบค่าความสว่าง (lux) กับอุณหภูมิที่ดอกทานตะวันได้รับมานั้นหมายถึง ยิ่งค่าความสว่างนั้นสูงอุณหภูมิก็จะยิ่งสูงตามไปด้วย ซึ่งสามารถเขียนเป็นสมการความสัมพันธ์ได้คือ

ความสว่าง $(\text{Lux: } \emptyset) \ \alpha$ อุณหภูมิ (Temperature : T)

ซึ่งในการศึกษาข้อมูลดังกล่าวทำให้เรานำ ***sunflower optimization (SFO) algorithm*** มาใช้ มีข้อมูลดังนี้

<u>sunflower optimization (SFO) algorithm</u>

1. พฤติกรรมทางชีวภาพและธรรมชาติ จะเคลื่อนที่เข้าหาแสงที่มีความเข็มแสงมากที่สุดโดยเราสามารถคำนวณได้โดย ใช้สูตรทางคณิตศาสตร์ ดังนี้

$$\mathrm{I} = \frac{P}{(4\pi d^2)}$$

เป็นสูตรในการหาความเข้มของแสงโดยเฉพาะ

โดยที่ I คือความเข้มของการแผ่รังสีดวงอาทิตย์

P คือพลังของดวงอาทิตย์

d คือระยะห่างระหว่างดวงอาทิตย์กับดอกทานตะวันแต่ละดวง

2. การปรับทิศทางดอกทานตะวัน ดอกทานตะวันแต่ละดอกจะปรับทิศทางของมันไปที่ ดวงอาทิตย์ดังแสดงในสมการต่อไปนี้

$$\mathrm{S(i)} = \frac{\mathrm{X^* - X_i}}{|\mathrm{X^* - X_i}|} \; ; \mathrm{i} = 1,2, \ldots \ldots, \mathrm{n}$$

โดยที่

X* คือคำตอบที่ดีที่สุดโดยรวม

Xi คือคำตอบปัจจุบัน

n คือขนาดประชากร

3. องศาของดอกทานตะวันที่จะหันเข้าหาดวงอาทิตย์ มีสมการดังนี้

$$\mathrm{d_i} = \mathrm{Y(มุม)} \times \pi(||\mathrm{X_i} + \mathrm{X_{i-1}}||)$$

โดยจะมี

Y(*มุม)* = ความเฉื่อยของดอกทานตะวัน

$\boldsymbol{\pi}$(||Xi + Xi-*1*||) = ความน่าจะเป็นของการผสมเกสรที่ดอกทานตะวันแต่ละชนิดของดอกทานตะวันกับดอกทานตะวันที่อยู่ใกล้ที่สุดเพื่อสร้างดอกทานตะวันใหม่ในตำแหน่งใหม่โดยพิจารณาจากระยะห่างระหว่างดอกทานตะวันทั้งสองตัว

4. การกำหนดลิมิตของดอกทานตะวันแต่ละขั้นถูกจำกัดไม่ให้เกินค่าต่อไปนี้

ด้วยสูตร

$$d_{(max)} = \frac{||X_{(max)} - X_{(min)}||}{2n}$$

โดยที่

X(max) และ X(min) เป็นลิมิตที่มากที่สุดและลิมิตที่น้อยที่สุด

n คือ จำนวนประชากรของดอกทานตะวัน

5. อัพเดทตำแหน่งดอกทานตะวัน

ดอกทานตะวันแต่ละชนิด มีการเปลี่ยนตำแหน่งเพื่อผลิตดอกทานตะวันรุ่นใหม่ตามทิศทาง (ขึ้นตก *Sun)* และขั้นตอนของดอกทานตะวันที่มุ่งสู่ดวงอาทิตย์ (การหันหน้าเข้าหาดวงอาทิตย์)ดังนี้

โดยใช้สูตร

$$X(i+1) = X(i) + (d(i) \times S(i))$$

โดยที่

X(i+1) คือตำแหน่งของดอกทานตะวันที่สร้างขึ้นใหม่ (รายบุคคล)

<u>การนำมาใช้และการเปรียบเทียบกับ Solar-Tacker</u>

- การนำมาใช้
  1. ตั้งค่าเริ่มต้นสำหรับอัตราการตาย *m,* ขนาดประชากร *n,* อัตราการผสมเกสร *P* และจำนวนการวนซ้ำสูงสุด mazitr เช่น m = 1, n = 50, P = 20, mazitr = 80
  2. ตั้งค่า *t* = 0
  3. สร้างประชากร X(i) E (L,U] แบบสุ่ม i = *1,2,3*,...,n จะคล้ายกับในส่วนของ Genetic หรือเราสามารถปรับได้เป็น population = n
  4. การทำประเมิน Fitness Function รายต้นของดอกทานตะวัน ในประชากรที่เรากำหนดไว้ทั้งหมด ประเมินสมรรถภาพร่างกายของบุคคล (ทานตะวัน) ในประชากร 1,.. , n = *S(X)*
  5. เลือกดอกที่ดีที่สุดจากโดยรวมของประชากร X
  6. ปรับบุคคลทั้งหมด (ดอกทานตะวัน) ปรับตำแหน่งไปทางดวงอาทิตย์ ดังแสดงในสมการที่ 5
  7. *While Ture:*
     - คำนวณทิศทางการเคลื่อนที่ ของค่า Fitness เป็นรูปแบบเวกเตอร์ ให้กับทุกประชากร
     - ลบดอกไม้ที่มีค่าแย่ที่สุดออกไป ค่า m%
     - คำนวณขั้นตอนของบุคคลเข้าหาดวงอาทิตย์ดังในข้อที่ 6
     - ให้ปุ่ยดอกทานตะวันที่ดีที่สุดรอบดวงอาทิตย์
     - ตรวจสอบค่าที่มากที่สุด
     - อัพเดตข้อมูล
     - ประเมินบุคคลใหม่
     - ยอมรับบุคคลใหม่หากค่าความฟิตของดอกใหม่ดีกว่าค่าปัจจุบัน
     - ตั้ง t = t+*1*
  8. ถ้า t > mazitr ให้ออกจากลูป
  9. แสดงผลลัพธ์ที่ดีที่สุดออกมา

- **การเปรียบเทียบกับ Solar – Tacker**

โดยในการนำ *sunflower optimization (SFO) algorithm* มาใช้นั้นเราได้นำสมการต่างๆมาดัดแปลงเป็นฟังก์ชั่น ดังนี้

1. การหาค่าความสว่าง (I) โดยใช้ฟังก์ชั่นแทนสมการ คือ

```
//ค่าที่ใช้ในการแปลง LUX
float LDR = 0;
float Volt = 0;
float Lux = 0;

for (int i = 0; i < N ; i++) { //sizeof(sensor_LDR)
 if (i != 2)
 {
  LDR = analogRead(sensor_PIN[i]);
  Volt = (3.3 / 4096) * (LDR);
  Lux = 42.0 * pow(Volt, 3.15);
  sensor_LDR[i] = int(Lux);
 }
 else
 {
  LDR = analogRead(sensor_PIN[i]);
  Volt = (3.3 / 4096) * (LDR+500);
  Lux = 42.0 * pow(Volt, 3.15);
  sensor_LDR[i] = int(Lux);
 }
}
```

2. การปรับทิศทางดอกทานตะวัน โดยใช้ฟังก์ชั่นแทนสมการ คือ

```
int P = 1;
int N = 4;
int y_Update = 0;
int x_Update = 0;

int Count(int si_value[] , int length) {
  int count = 0;
  for (int i = 0; i < length ; i++) {
    if (si_value[i] == 0 ) {
      count += 1;
    }
  }
  return count;
}

for (int i = 0; i < N ; i++) {
  if (MaxVal == sensor_LDR[i]) {
    si_value[i] = int(0);
  }
  else {
    int si = ( MaxVal - sensor_LDR[i] / abs(MaxVal - sensor_LDR[i]) ) ;
    si_value[i] = int(si);
  }
  int di = ( rad * (P * ((sensor_LDR[i] + sensor_LDR[i - 1]) * abs(sensor_LDR[i] + sensor_LDR[i - 1])))) % 180;
  di_value[i] = int(di);
  int X = sensor_LDR[i] + di_value[i] * si_value[i];
  X_value[i] = int(X) % 180;
}
int num = Count(si_value, N);
if ((num > 2 && si_value[1] == 0) || si_value[1] == 0) {
  x_Update = ((abs(X_value[1] - X_value[2]) / 4) * (-1));
}
else {
  x_Update = ((abs(X_value[1] - X_value[2]) / 4));
}

if ((num > 2 && si_value[3] == 0) || si_value[3] == 0) {
  y_Update = ((abs(X_value[0] - X_value[3]) / 4) * (-1));
}
else {
  y_Update = ((abs(X_value[0] - X_value[3]) / 4));
}
```

3. องศาของดอกทานตะวันที่จะหันเข้าหาดวงอาทิตย์และการกำหนดลิมิตของดอกทานตะวัน โดยใช้ฟังก์ชั่นแทนสมการ คือ

```
int MaxVal = 0;
int MinVal = 0;
int rad = 0;

int MaxValue(int sensor_LDR[] , int length) {
  int Max = 0;
  for (int i = 0; i < length ; i++) {
    if (sensor_LDR[i] > Max ) {
      Max = int(sensor_LDR[i]);
    }
  }
  return Max;
}

int MinValue(int sensor_LDR[] , int length) {
  int Min = 10000;
  for (int i = 0; i < length ; i++) {
    if (sensor_LDR[i] < Min ) {
      Min = int(sensor_LDR[i]);
    }
  }
  return Min;
}

int radian(int MaxVal , int MinVal , int N) {
  int d_max = (abs((MaxVal * 1000 / 57296) + (MinVal * 1000 / 57296)) / (2 * N) );
  int rads = (d_max % 180) * 1000 / 57296;
  return rads;
}

MaxVal = MaxValue( sensor_LDR , N );
MinVal = MaxValue( sensor_LDR , N );
rad = radian( MaxVal, MinVal, N);
```

4. อัพเดทตำแหน่งดอกทานตะวัน โดยใช้ฟังก์ชั่นแทนสมการ คือ

```
int y_Update = 0;
int x_Update = 0;

int Count(int si_value[] , int length) {
  int count = 0;
  for (int i = 0; i < length ; i++) {
    if (si_value[i] == 0 ) {
      count += 1;
    }
  }
  return count;
}

int num = Count(si_value, N);
if ((num > 2 && si_value[1] == 0) || si_value[1] == 0) {
  x_Update = ((abs(X_value[1] - X_value[2]) / 4) * (-1));
}
else {
  x_Update = ((abs(X_value[1] - X_value[2]) / 4));
}

if ((num > 2 && si_value[3] == 0) || si_value[3] == 0) {
  y_Update = ((abs(X_value[0] - X_value[3]) / 4) * (-1));
}
else {
  y_Update = ((abs(X_value[0] - X_value[3]) / 4));
}

pos_x = pos_x+map(x_Update,-43,43,-2,2);
pos_y = pos_y+map(y_Update,-43,43,-2,2);

if(pos_x>180)
{
  pos_x = 180;
}
else if(pos_x<0)
{
  pos_x = 0;
}

if(pos_y>180)
{
  pos_y = 180;
}
else if(pos_y<0)
{
  pos_y = 0;
}
```