# Raport

Przemysław Lis 229940

Wojciech Majchrzak 229947

## 1. Zasada działania programu

Nasz algorytm tworzy obiekt przyjmujący wartości na podstawie których generuje rój, w którym każda cząstka posiada swoją pozycję w przestrzeni rozwiązań i a jej kierunek w jakim się porusza zależy od pomiaru przystosowania. Na naszym obiekcie wykonujemy metodę find_local_minimum, w której literujemy algorytm x razy oraz a w pętli dodatkowo po wszystkich cząstkach roju aktualizując na bieżąco ich lokalizację w przestrzeni. W wyniku otrzymujemy przybliżone współrzędne punktu x i y które to definiują punkty lokalnego minimum zadanej wyżej w mainie funkcji.

## 2. Wybrane miejsca implementacji rozwiązania

```
def generate_population(self):
    for particle in range(self.population_size):
        self.population.append(
            Particle(random.uniform(-10, 10), random.uniform(-10, 10),
self.inersion, self.cognitive_constant,
                     self.social_constant, self.function))
```

Metoda generująca rój tworzy pustą listę i dodanie do niej cząstki o losowo generowanych koordynatach

```
def update_adaptation(self):
    self.adaptation = self.__calculate_adaptation()
    if self.adaptation < self.best_adaptation:
        self.best_adaptation = self.adaptation
        self.best_x = self.x
        self.best_y = self.y
```

Funkcja obliczająca przystosowanie cząstek sprawdza, czy nowe przystosowanie jest wyższe niż poprzednie najlepsze przystosowanie. Jeżeli tak to nadpisuję koordynaty do best_x i best_y

```
def get_best(self):
    best_adaptation = math.inf
    for particle in self.population:
        particle.update_adaptation()
        if particle.best_adaptation < best_adaptation:
            best_adaptation = particle.best_adaptation
            self.best_particle = particle
```

Funkcja do określania najlepszej cząstki w roju iteracyjnie sprawdza wszystkie cząstki aktualizuje ich przystosowanie na podstawie nowych pozycji i znajduję cząstkę o najniższej wartości.

```
def find_local_minimum(self):
    for i in range(self.iteration):
        self.get_best()
        for particle in self.population:
            particle.update_coordinates()
```

Funkcja która jest odpowiedzialna za literowania po wszystkich cząstkach roju i znalezienia lokalnego minimum aktualizując ich współrzędne

## 3. Wyniki

Funkcję na których zostały przeprowadzone obliczenia metodą optymalizacji przedstawione są poniżej:

```
def booth_function(x, y):
    return pow(x + 2 * y - 7, 2) + pow(2 * x + y - 5, 2)


def matyas_function(x, y):
    return 0.26 * (x ** 2 + y ** 2) - 0.48 * x * y
```

Lokalne minimum funkcji booth_function wynosi: (1, 3)
Natomiast funkcji matyas_function wynosi: (0, 0)

Wartości startowe:

| Inercja | Stała poznawcza | Stała społeczna | Liczba iteracji | Liczba cząstek |
|---|---|---|---|---|
| 0.2 | 0.35 | 0.45 | 100 | 100 |

Wynik matyas:

| x | y |
|---|---|
| 0.14642632186886573 | 0.0720975529714103 |

Wynik booth:

| x | y |
|---|---|
| 0.6212689641736109 | 3.3038906958865315 |

##########################################################################################

Zmieniamy inercje:

| Inercja | Stała poznawcza | Stała społeczna | Liczba iteracji | Liczba cząstek |
|---|---|---|---|---|
| 0.1 | 0.35 | 0.45 | 100 | 100 |

Wynik matyas:

| x | y |
|---|---|
| -2.135492352805617 | -1.9088222837015003 |

Wynik booth:

| x | y |
|---|---|
| 1.2488893234936498 | 2.8435512687645055 |

Zmieniamy inercje:

| Inercja | Stała poznawcza | Stała społeczna | Liczba iteracji | Liczba cząstek |
|---|---|---|---|---|
| 0.4 | 0.35 | 0.45 | 100 | 100 |

Wynik matyas:

| x | y |
|---|---|
| 0.5364550579027956 | 0.7150895491469775 |

Wynik booth:

| x | y |
|---|---|
| 0.8900231811215349 | 2.6607908848255875 |

##########################################################################################

Zmieniamy stałą poznawczą:

| Inercja | Stała poznawcza | Stała społeczna | Liczba iteracji | Liczba cząstek |
|---|---|---|---|---|
| 0.2 | 0.65 | 0.45 | 100 | 100 |

Wynik matyas:

| x | y |
|---|---|
| -0.10172865513523988 | -0.9023043050239394 |

Wynik booth:

| x | y |
|---|---|
| 1.1234952045586954 | 3.374133388788918 |

Zmieniamy stałą poznawczą:

| Inercja | Stała poznawcza | Stała społeczna | Liczba iteracji | Liczba cząstek |
|---|---|---|---|---|
| 0.2 | 0.15 | 0.45 | 100 | 100 |

Wynik matyas:

| x | y |
|---|---|
| -0.3353914849200006 | -0.03146511382280437 |

Wynik booth:

| x | y |
|---|---|
| 0.8178646421474962 | 3.6321718740712416 |

##########################################################################################

Zmieniamy stałą społeczną:

| Inercja | Stała poznawcza | Stała społeczna | Liczba iteracji | Liczba cząstek |
|---|---|---|---|---|
| 0.2 | 0.35 | 0.15 | 100 | 100 |

Wynik matyas:

| x | y |
|---|---|
| -1.4712016412396185 | -1.2731814755365356 |

Wynik booth:

| x | y |
|---|---|
| -0.4988782211329692 | 3.957349517837825 |

Zmieniamy stałą społeczną:

| Inercja | Stała poznawcza | Stała społeczna | Liczba iteracji | Liczba cząstek |
|---|---|---|---|---|
| 0.2 | 0.35 | 0.75 | 100 | 100 |

Wynik matyas:

| x | y |
|---|---|
| -2.45844319271177 | -2.8356030215405603 |

Wynik booth:

| x | y |
|---|---|
| 2.1593145875489643 | 0.9663813077503924 |

##########################################################################################

Zmieniamy ilość iteracji:

| Inercja | Stała poznawcza | Stała społeczna | Liczba iteracji | Liczba cząstek |
|---|---|---|---|---|
| 0.2 | 0.35 | 0.75 | 10 | 100 |

Wynik matyas:

| x | y |
|---|---|
| 1.4287933792338148 | 1.0944690965354802 |

Wynik booth:

| x | y |
|---|---|
| 2.0623576654084648 | 1.2082571732313045 |

Zmieniamy ilość iteracji:

| Inercja | Stała poznawcza | Stała społeczna | Liczba iteracji | Liczba cząstek |
|---|---|---|---|---|
| 0.2 | 0.35 | 0.75 | 1000 | 100 |

Wynik matyas:

| x | y |
|---|---|
| 0.4591914666164634 | 0.34275596150146725 |

Wynik booth:

| x | y |
|---|---|
| 1.242740591229003 | 2.7097584971283606 |

##########################################################################################

Zmieniamy liczbę czastek:

| Inercja | Stała poznawcza | Stała społeczna | Liczba iteracji | Liczba cząstek |
|---|---|---|---|---|
| 0.2 | 0.35 | 0.45 | 100 | 10 |

Wynik matyas:

| x | y |
|---|---|
| -1.1699633743091926 | -1.5431652795632917 |

Wynik booth:

| x | y |
|---|---|
| -1.01549882376802 | 5.027559287833586 |

Zmieniamy liczbę czastek:

| Inercja | Stała poznawcza | Stała społeczna | Liczba iteracji | Liczba cząstek |
|---|---|---|---|---|
| 0.2 | 0.35 | 0.45 | 100 | 1000 |

Wynik matyas:

| x | y |
|---|---|
| 0.19857237560221286 | 0.06410722262560675 |

Wynik booth:

| x | y |
|---|---|
| 1.6496663204403959 | 2.4171382653863844 |

## 4. Wnioski

- Zmniejszenie wartości inercji znacząco wpłynęło na dokładność wyniku w szczególności dla funkcji matyas – mocno zaniżyło współrzędne natomiast inkrementacje tego współczynnika zawyżyła wartości funkcji matyas. Jeżeli chodzi o funkcję booth to utrzymuje ona dość dobrą dokładność wraz ze zmieniającym się współczynnikiem inercji

- Przy zmianie stałej poznawczej nie zaobserwowaliśmy znacznego odchylenia wraz z dekrementowaną jak i inkrementowaną wartością tego współczynnika – wyniki są dość losowe ale nadal bliskie realnym wraz przy znaczących zmianach stałej poznawczej

- Wariacja wartością stałej społecznej inkrementacja jak i dekrementacja przy wielu próbach wykazała znaczące odchylenia od realnych wyników.

- Dekrementacja ilości iteracji algorytmy wykazała znaczne większe odchylenia wyników względem jej inkrementacji. Po zwiększeniu liczby wykonywania się algorytmu wyniki były znacząco zbliżone do tych realnych

- Liczba cząstek w naszym odczuciu to najważniejszy parametr wpływający na dokładność wyników. Na podstawie powyższych analiz widać, że zwiększenie liczby cząstek znacząco pozytywnie wpływa na otrzymane wyniki i ich dokładność. Zbyt mała liczba cząstek powoduje bardzo duże odchylenia