Ireneusz Bartoszek 251481
Szymon Kaźmierczak 251546

Poniedziałek [12:00 – 13:30]
Data zgłoszenia: 11.01.2026 r.

# Metaheurystyki i ich zastosowania

## Zadanie 5: Rój cząstek

## 1. Zasada działania algorytmu

Algorytm PSO (Particle Swarm Optimization) jest metaheurystyką inspirowaną zachowaniem stad/rojów w naturze, gdzie wiele osobników (cząstek) współpracuje w sposób rozproszony, aby znaleźć dobre rozwiązanie problemu optymalizacji.

### 2.1 Reprezentacja cząstki

Każda cząstka w roju posiada podstawowe atrybuty:

- Pozycję *x* (w tym zadaniu: [x, y]) – aktualny punkt w przestrzeni rozwiązań.
- Prędkość *v* (w tym zadaniu: [$v_x$, $v_y$]) – wektor określający kierunek i szybkosc przemieszczania.
- Najlepszą pozycję lokalną *pbest* – najlepsze rozwiązanie znalezione dotychczas przez daną cząstkę (na podstawie funkcji przystosowania).
- Najlepszą pozycję globalną *gbest* – najlepsze rozwiązanie znalezione dotychczas w całym roju (najlepsze spośród wszystkich pbest).

### 2.2 Funkcja przystosowania

Przystosowanie cząstki jest obliczane jako wartość funkcji celu w aktualnej pozycji cząstki. W implementacji przyjęto jednolitą postać porównań dla minimalizacji: mniejsza wartość jest lepsza. Natomiast maksymalizację realizuje się poprzez transformację celu do postaci minimalizacji.

### 2.3 Aktualizacja prędkości

W algorytmie PSO nowa prędkość cząstki jest sumą trzech wpływów:

1. Ruch inercyjny – utrzymuje część dotychczasowej prędkości, co odpowiada "rozpędowi" cząstki. Parametr $w \in [0, 1]$ kontroluje kompromis między eksploracją a eksploatacją (większe w sprzyja bardziej globalnej eksploracji).
2. Składnik kognitywny – tendencja do poprawiania własnego wyniku, czyli powrót w kierunku najlepszego osobistego rozwiązania *pbest*. Wpływ kontroluje *c1* oraz losowy współczynnik *r1*.
3. Składnik socjalny – tendencja do naśladowania najlepszego osobnika w roju, czyli ruch w kierunku *gbest*. Wpływ kontroluje *c2* oraz losowy współczynnik *r2*.

W ujęciu wektorowym (dla każdej składowej i):

$$v_i = wv_i + c_1 r_1 (pbest_i - x_i) + c_2 r_2 (gbest_i - x_i)$$

Następnie pozycja jest aktualizowana zgodnie z:

$$x_i = x_i + v_i$$

W implementacji ogranicza się dodatkowo prędkość do przedziału $[-v_{max}, v_{max}]$ oraz pilnuje, aby pozycja pozostała w zadanych granicach (bounds).

### 2.3 Warunek stopu

Podstawowym warunkiem stopu jest maksymalna liczba iteracji. Dodatkowo można zastosować kryterium stagnacji - brak poprawy najlepszego rozwiązania przez pewną liczbę iteracji.

## 2. Pseudokod algorytmu PSO

### 2.1 Inicjalizacja roju

1. Ustal liczbę cząstek, granice przestrzeni poszukiwań *bounds* oraz parametry: w, $c_1$, $c_2$, $v_{max}$.
2. Wygeneruj n_particles cząstek:
   - losuj pozycję w każdym wymiarze w zakresie bounds[i].
   - ustaw prędkość początkową na 0.
3. Dla każdej cząstki:
   - policz przystosowanie fitness(X),
   - ustaw pbest = position, pbest_value = fitness(position).
4. Wyznacz globalnie najlepszą cząstkę w roju:
   - gbest = najlepsze pbest spośród wszystkich cząstek.

### 2.2 Iteracja

Dla każdej cząstki p w roju:

1. Aktualizuj prędkość.
2. Ograniczenie prędkości do $[-v_{max}, v_{max}]$.
3. Aktualizuj pozycję
4. Ograniczenie pozycji do przedziału bounds[i].
5. Ocenń przystosowanie w nowej pozycji.

6. Aktualizacja pbest: jeśli nowe przystosowanie jest lepsze (mniejsze), zapisz nową pozycję jako pbest.
7. Aktualizacja gbest: jeśli pbest_value cząstki jest lepsze od gbest_value, ustaw gbest na pbest tej cząstki.

### 2.3 Warunek stopu

Algorytm kończy pracę, gdy:

- osiągnięto maksymalną liczbę iteracji, lub
- wystąpiła stagnacja (brak poprawy globalnego najlepszego rozwiązania przez patience iteracji).

Zwracany wynik to gbest_position oraz wartość funkcji celu w tym punkcie.

## 3. Opis implementacji

1) Reprezentacja cząstki: W kodzie cząstka przechowuje position (listę współrzędnych) oraz listę velocity (o tym samym wymiarze).

```
class Particle:
    def __init__(self, position, fitness, mm):
        self.position = position.copy() # kopia przekazanej listy pozycji
        dim = len(position)
        self.velocity = [0.0] * dim

        self.fitness = lambda X: mm * fitness(X)

        self.best_position = self.position.copy()
        self.best_value = self.fitness(self.position)
```

2) Funkcja przystosowania: W implementacji algorytm zawsze minimalizuje funkcję przystosowania. Maksymalizacja jest realizowana przez transformację:
- dla minimalizacji: fitness = f,
- dla maksymalizacji: fitness = -f.

```
self.mm = None
if minimization:
    self.mm = 1
else:
    self.mm = -1
```

```
self.fitness = lambda X: mm * fitness(X)
```

3) Aktualizacja prędkości i pozycji: W metodzie move() następuje wyliczenie nowej prędkości na podstawie trzech komponentów – inercji, poznawczy, społeczny. Następnie aktualizowana jest pozycja.

```
# aktualizacja predkosci
self.velocity[i] = (
    w * self.velocity[i]
    + c1 * r1 * (self.best_position[i] - self.position[i])
    + c2 * r2 * (gbest_position[i] - self.position[i])
)
```

4) Aktualizacja gbest i warunek stagnacji: W metodzie run() gbest jest aktualizowany, gdy jakaś cząstka poprawi najlepszy wynik, natomiast licznik no_improvement kontroluje stagnację i przerywa pętle ilości iteracji bez poprawy równej patience.

```
if p.best_value < self.gbest_value:
    self.gbest_value = p.best_value
    self.gbest_position = p.best_position.copy()
    improved = True
```

```
if no_improvement >= patience:
    break
```

# 4. Opis uruchomienia programu – instrukcja użytkownika

Program umożliwia uruchomienie algorytmu PSO dla dowolnej funkcji celu f(x,y) (z listy funkcji) w zadanych granicach przeszukiwania. Funkcja celu jest przekazywana jako funkcja Pythona przyjmująca wektor X=[x,y].

W pliku main.py użytkownik wybiera parametry algorytmu ustalając ich wartość w słowniku *parametry*.

```
parametry = {
    'objective_function': himmelblaus_function,
    'bounds': [(-1, 1), (-1, 1)], # this indicates in what dimension it will work
    'num_particles': 50,
    'num_iterations': 100,
    'w': 0.75, # inertia constant
    'c1': 1.0, # cognative constant
    'c2': 2.0, # social constant,
    'v_max': 1.0,
    'minimization': False
}
```

Program uruchamia się za pomocą metody run(), która przyjmuje parametry:

- patience – jest to ilość iteracji bez poprawy, domyślnie ustawiona na 20.
- plot – zmienna boolean informująca program czy generować wykresy podczas działania algorytmu, domyślnie ustawiona na False.
- pause – zmienna do modyfikowania tempa wyświetlania wykresu.

Program zwraca:

- best_pos – współrzędne najlepszego znalezionego punktu,
- best_val – wartość funkcji w tym punkcie,
- history – lista wszystkich znalezionych najlepszych rozwiązań z każdej iteracji.

# 5. Eksperymenty

## 5.1 Opis eksperymentu

Celem przeprowadzonych eksperymentów było zbadanie wpływu kluczowych parametrów algorytmu roju cząstek (PSO – Particle Swarm Optimization) na jakość znajdowanych rozwiązań oraz stabilność działania algorytmu podczas optymalizacji funkcji dwóch zmiennych.

Badania przeprowadzono dla dwóch wybranych funkcji testowych ze zbioru funkcji (funkcja Bootha oraz funkcja Himmelblaua). Dla każdej funkcji algorytm był uruchamiany w zadanym obszarze poszukiwań $[x_{min}, x_{max}] \times [y_{min}, y_{max}]$.

Dla każdej badanej konfiguracji parametrów algorytm został uruchomiony co najmniej 5 razy, z różnymi losowymi inicjalizacjami roju. Następnie obliczono statystyki: najlepszy wynik, najgorszy wynik, wartość średnią, medianę oraz odchylenie standardowe, co pozwoliło ocenić zarówno jakość rozwiązań, jak i stabilność algorytmu.

Jako konfigurację bazową (referencyjną) przyjęto następujące parametry, które były utrzymywane na stałym poziomie, o ile nie badano wpływu konkretnego z nich:

- Liczba cząstek w roju (n_particles): 50
- Maksymalna liczba iteracji (n_iterations): 200
- Współczynnik inercji (w): 0.75
- Współczynnik kognitywny (c1): 1.5
- Współczynnik socjalny (c2): 1.5
- Maksymalna prędkość (v_max): 1.0

## 5.2 Funkcja Himmelblau's

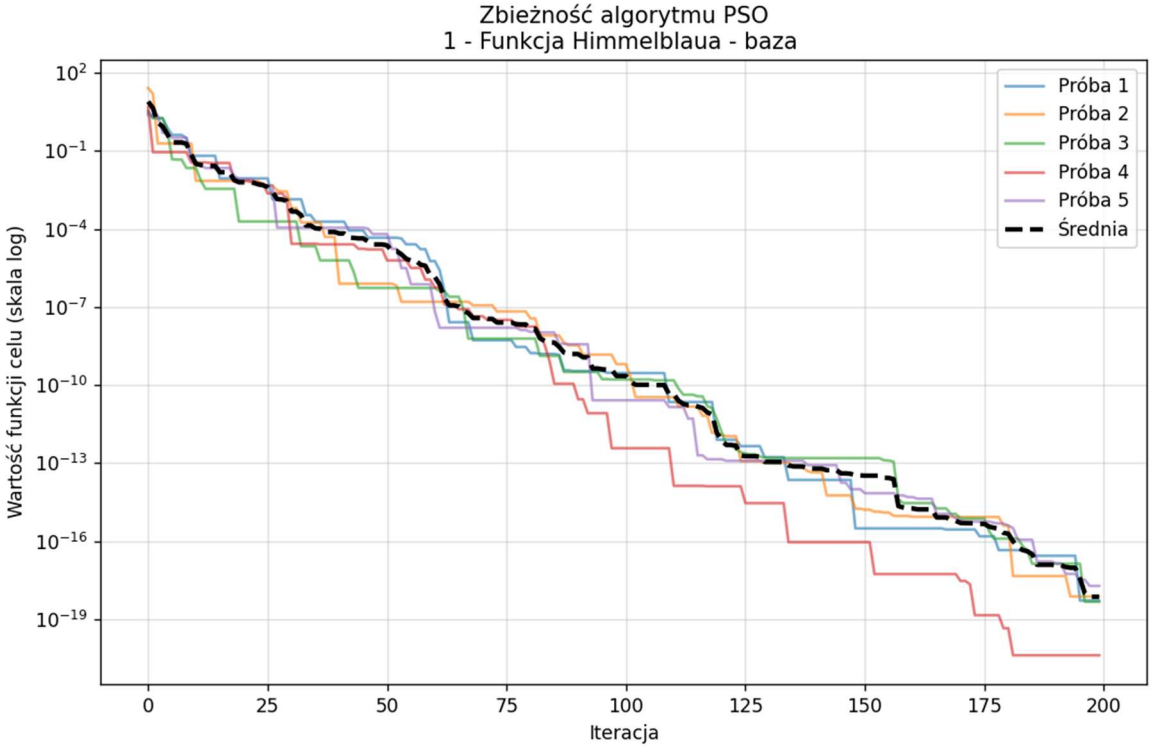
W eksperymencie będzie optymalizowane minimum globalne.

### 5.2.1 Wyniki bazowe

Wyniki dla konfiguracji bazowej eksperymentu.

| Próba | Wynik (f(x,y)) | x | y |
|---|---|---|---|
| 1 | 5.267544e−19 | −2.805118 | 3.131313 |
| 2 | 7.550880e−19 | −2.805118 | 3.131313 |
| 3 | 4.741015e−19 | 3.000000 | 2.000000 |
| 4 | 4.159552e−21 | 3.000000 | 2.000000 |
| 5 | 1.923857e−18 | 3.584428 | −1.848127 |

| Konfiguracja | Najlepszy wynik | Najgorszy wynik | Średnia | Mediana | Std (Odchylenie) |
|---|---|---|---|---|---|
| Baza | 4.159552e−21 | 1.923857e−18 | 7.367922e−19 | 5.267544e−19 | 6.417681e−19 |



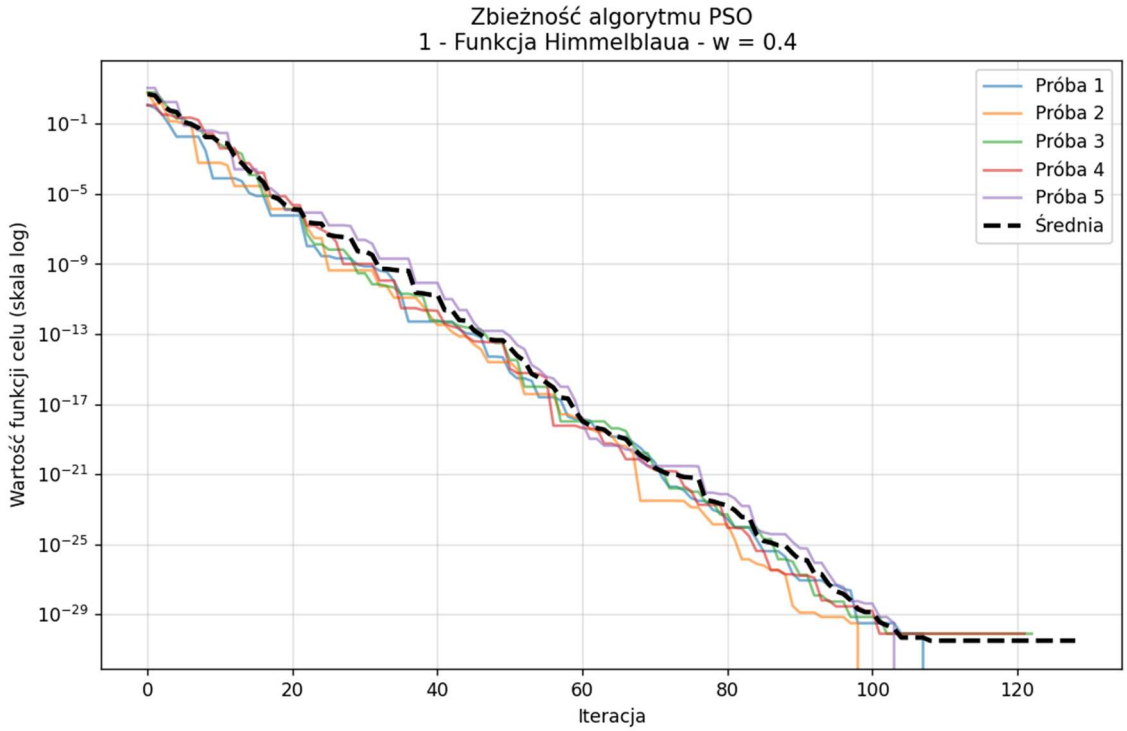

### 5.2.2 Badanie inercji

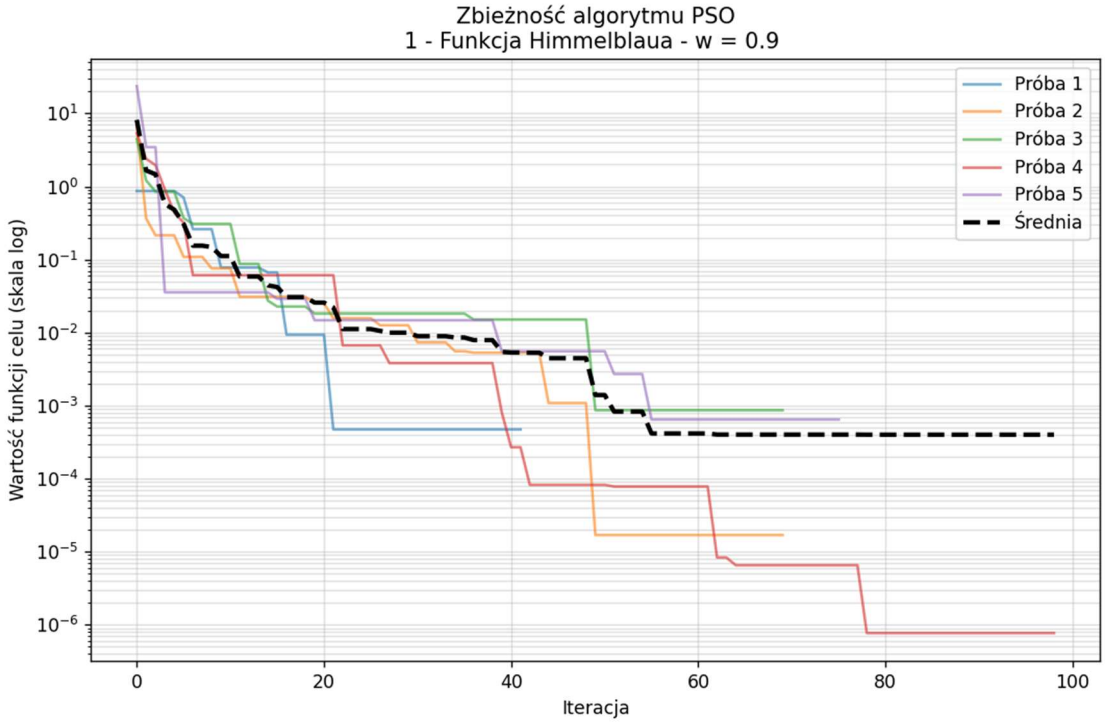
Współczynnik inercji (w) dla których przeprowadzone zostały eksperymenty to w = {0.4, 0.9}

| Inercja (w) | Próba | Wynik (f(x,y)) | x | y |
|---|---|---|---|---|
| 0.4 | 1 | 0.000000e+00 | 3.000000 | 2.000000 |
| | 2 | 0.000000e+00 | 3.584428 | −1.848127 |
| | 3 | 7.888609e−31 | −2.805118 | 3.131313 |
| | 4 | 7.888609e−31 | −2.805118 | 3.131313 |
| | 5 | 0.000000e+00 | 3.584428 | −1.848127 |
| 0.9 | 1 | 4.727153e−04 | 3.003235 | 1.995151 |
| | 2 | 1.696176e−05 | 3.000445 | 1.998941 |

| | | | | |
|---|---|---|---|---|
| | 3 | 8.673751e−04 | −3.781946 | −3.287376 |
| | 4 | 7.722547e−07 | 2.999850 | 2.000024 |
| | 5 | 6.462602e−04 | −2.807456 | 3.134762 |

| Konfiguracja | Najlepszy | Najgorszy | Średnia | Mediana | Std (Odchylenie) |
|---|---|---|---|---|---|
| w=0.4 | 0.000000e+00 | 7.888609e−31 | 3.155444e−31 | 0.000000e+00 | 3.864613e−31 |
| w=0.9 | 7.722547e−07 | 8.673751e−04 | 4.008169e−04 | 4.727153e−04 | 3.436477e−04 |



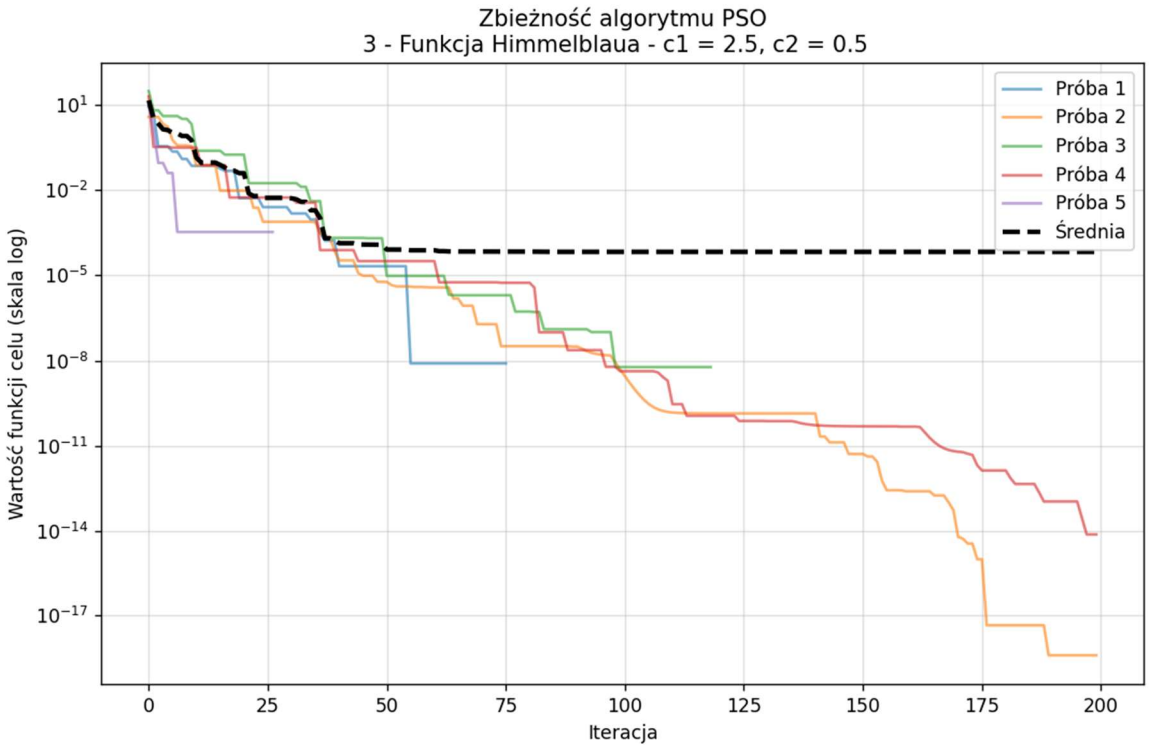


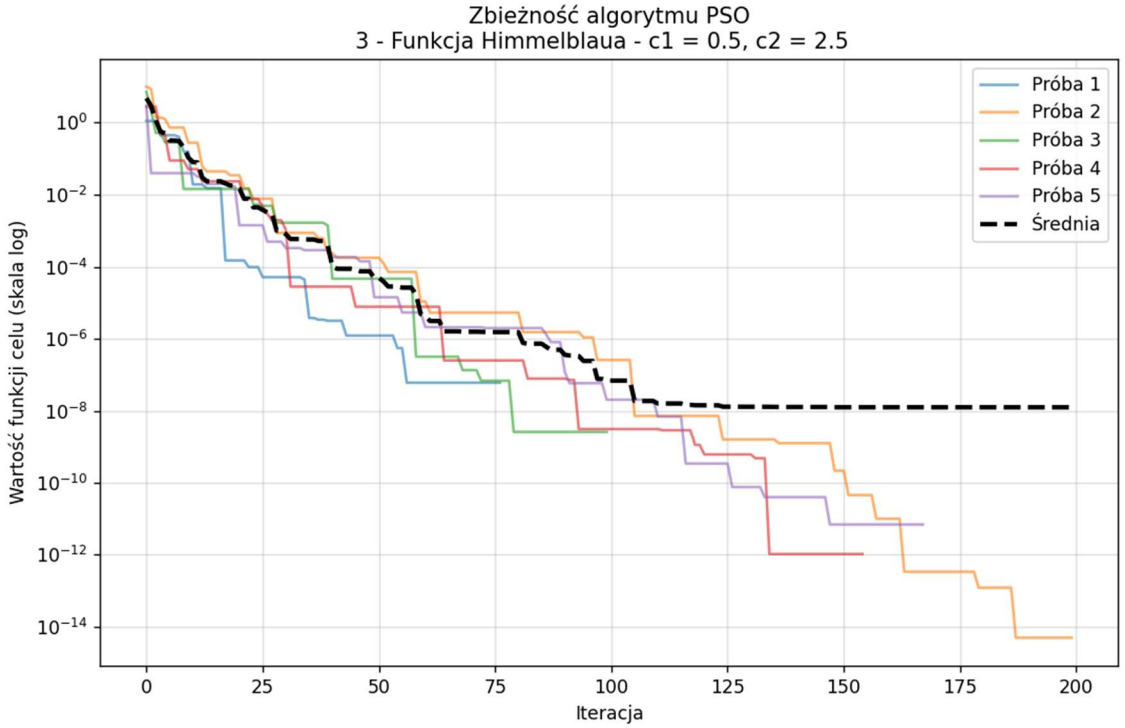

### 5.2.3 Psychologia Roju

W tym eksperymencie badane będą wyniki dla konfiguracji współczynnika kognitywnego i socjalnego. Zawierało to będzie wynik dla indywidualistów (wysoki kognitywny, niski socjalny) i naśladowców (niski kognitywny i wysoki socjalny). Badanie będzie dla c1 = 2.5, c2 = 0.5 i c1 = 0.5, c2 = 2.5.

| Konfiguracja | Próba | Wynik (f(x,y)) | x | y |
|---|---|---|---|---|
| | 1 | 7.933850e−09 | 3.584437 | −1.848112 |
| | 2 | 3.884289e−19 | 3.584428 | −1.848127 |
| | 3 | 5.874578e−09 | −2.805106 | 3.131317 |
| | 4 | 7.560294e−15 | −3.779310 | −3.283186 |
| c1=2.5,c2=0.5 | 5 | 3.345343e−04 | 3.586702 | −1.846516 |
| | 1 | 6.133611e−08 | 3.584430 | −1.848192 |
| | 2 | 4.999112e−15 | −3.779310 | −3.283186 |
| | 3 | 2.662700e−09 | −3.779314 | −3.283181 |
| | 4 | 1.036792e−12 | 3.000000 | 2.000000 |
| c1=0.5,c2=2.5 | 5 | 6.815801e−12 | 3.000000 | 1.999999 |

| Parametry | Najlepszy (Best) | Najgorszy (Worst) | Średnia (Mean) | Mediana | Std (Odchylenie) |
|---|---|---|---|---|---|
| c1=2.5,c2=0.5 | 3.884289e−19 | 3.345343e−04 | 6.690962e−05 | 5.874578e−09 | 1.338123e−04 |
| c1=0.5,c2=2.5 | 4.999112e−15 | 6.133611e−08 | 1.280133e−08 | 6.815801e−12 | 2.428925e−08 |



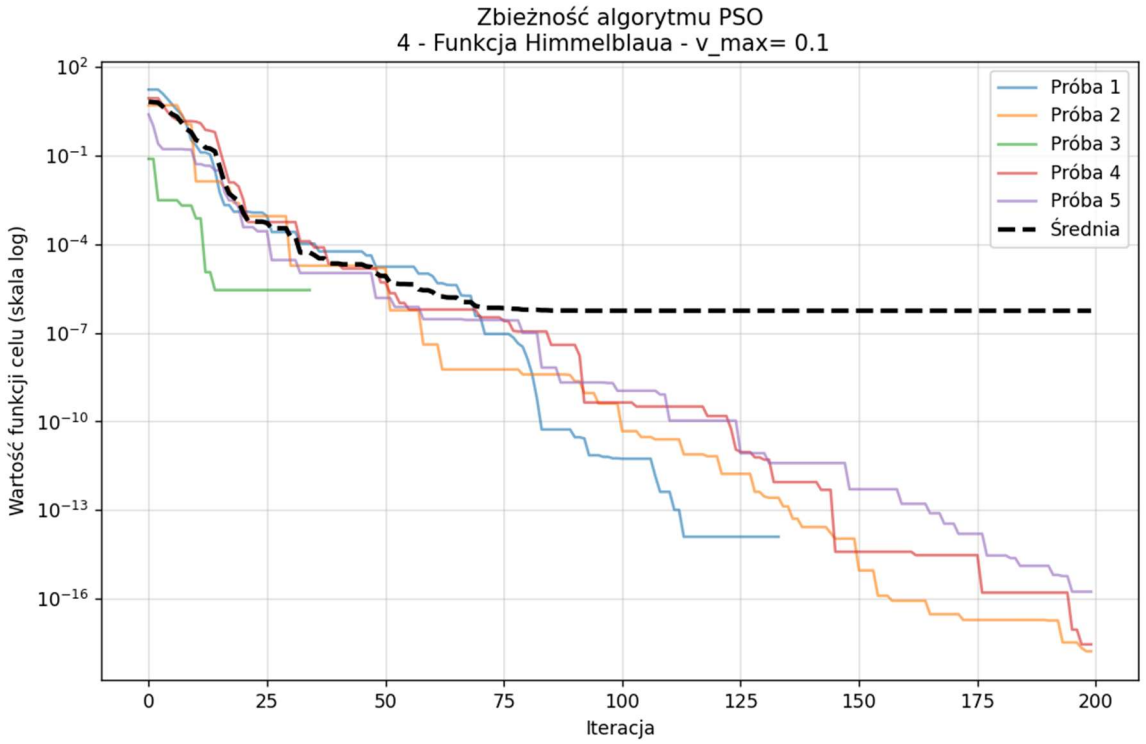

### 5.2.4 Eksperyment maksymalnej prędkości

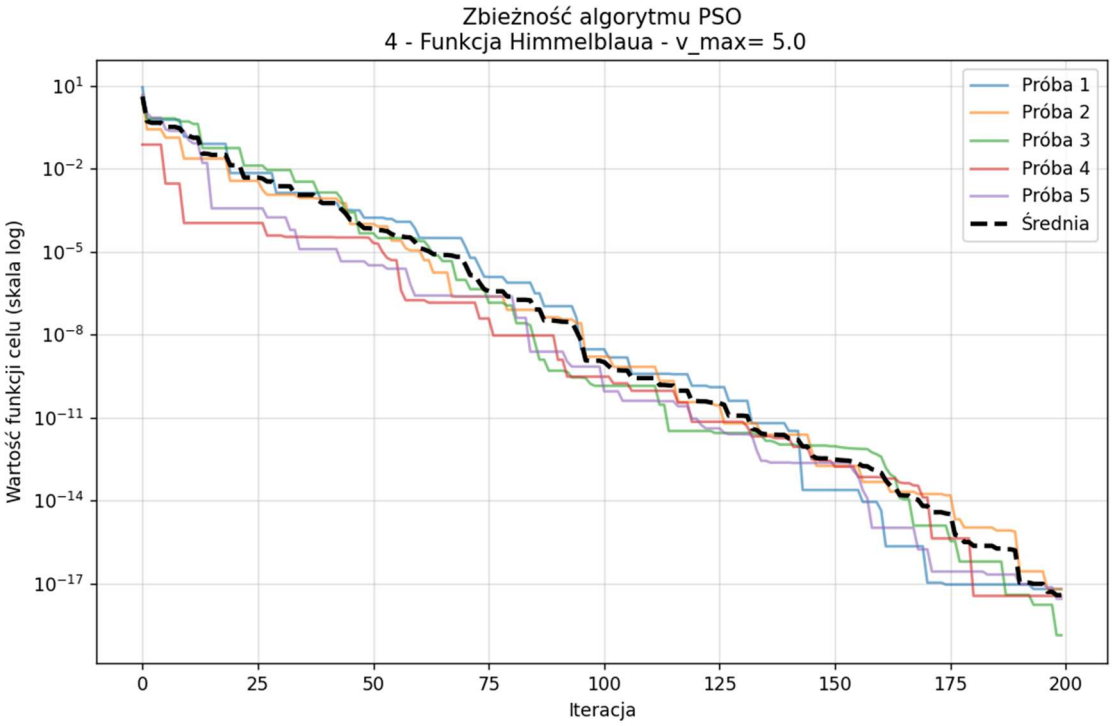
W tym eksperymencie testowana będzie prędkość v_max = {0.1, 5.0}.

| Vmax | Próba | Wynik (f(x,y)) | x | y |
|---|---|---|---|---|
| 0.1 | 1 | 1.268815e−14 | −2.805118 | 3.131313 |
| | 2 | 1.698006e−18 | −3.779310 | −3.283186 |
| | 3 | 2.778227e−06 | 3.584319 | −1.848484 |

| | | | | |
|---|---|---|---|---|
| | 4 | 2.916908e−18 | −2.805118 | 3.131313 |
| | 5 | 1.764401e−16 | 3.584428 | −1.848127 |
| | 1 | 6.441863e−18 | 3.000000 | 2.000000 |
| | 2 | 6.353870e−18 | −2.805118 | 3.131313 |
| | 3 | 1.377357e−19 | −2.805118 | 3.131313 |
| | 4 | 3.638326e−18 | 3.000000 | 2.000000 |
| 5.0 | 5 | 2.865501e−18 | −3.779310 | −3.283186 |

| Parametry | Najlepszy (Best) | Najgorszy (Worst) | Średnia (Mean) | Mediana | Std (Odchylenie) |
|---|---|---|---|---|---|
| Vmax=0.1 | 1.698006e−18 | 2.778227e−06 | 5.556454e−07 | 1.764401e−16 | 1.111291e−06 |
| Vmax=5.0 | 1.377357e−19 | 6.441863e−18 | 3.887459e−18 | 3.638326e−18 | 2.356907e−18 |





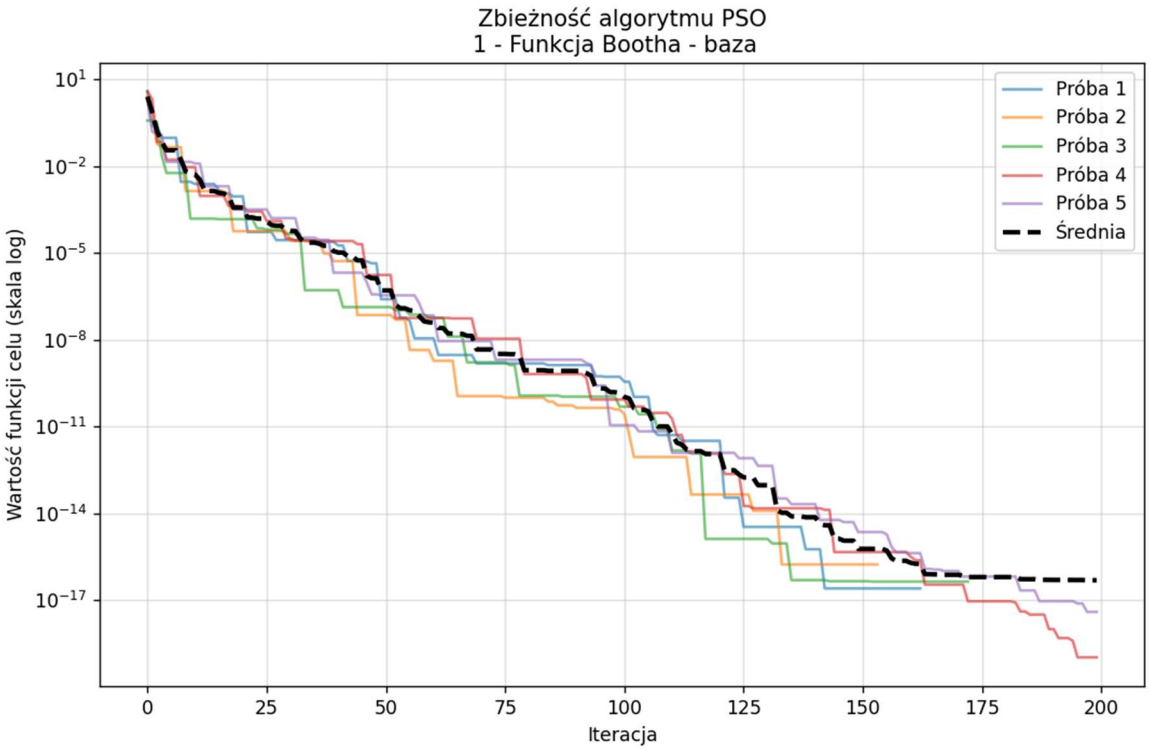

## 5.3 Funkcja Boola

W eksperymencie będzie optymalizowane minimum globalne.

### 5.3.1 Wyniki bazowe

| Próba | Wynik (f(x,y)) | x | y |
|---|---|---|---|
| 1 | 2.578017e−17 | 1.000000 | 3.000000 |
| 2 | 1.736558e−16 | 1.000000 | 3.000000 |
| 3 | 4.448126e−17 | 1.000000 | 3.000000 |
| 4 | 1.071226e−19 | 1.000000 | 3.000000 |
| 5 | 3.985131e−18 | 1.000000 | 3.000000 |

| Funkcja | Najlepszy | Najgorszy (Worst) | Średnia (Mean) | Mediana | Std (Odchylenie) |
|---|---|---|---|---|---|
| Bootha | 1.071226e−19 | 1.736558e−16 | 4.960189e−17 | 2.578017e−17 | 6.405260e−17 |

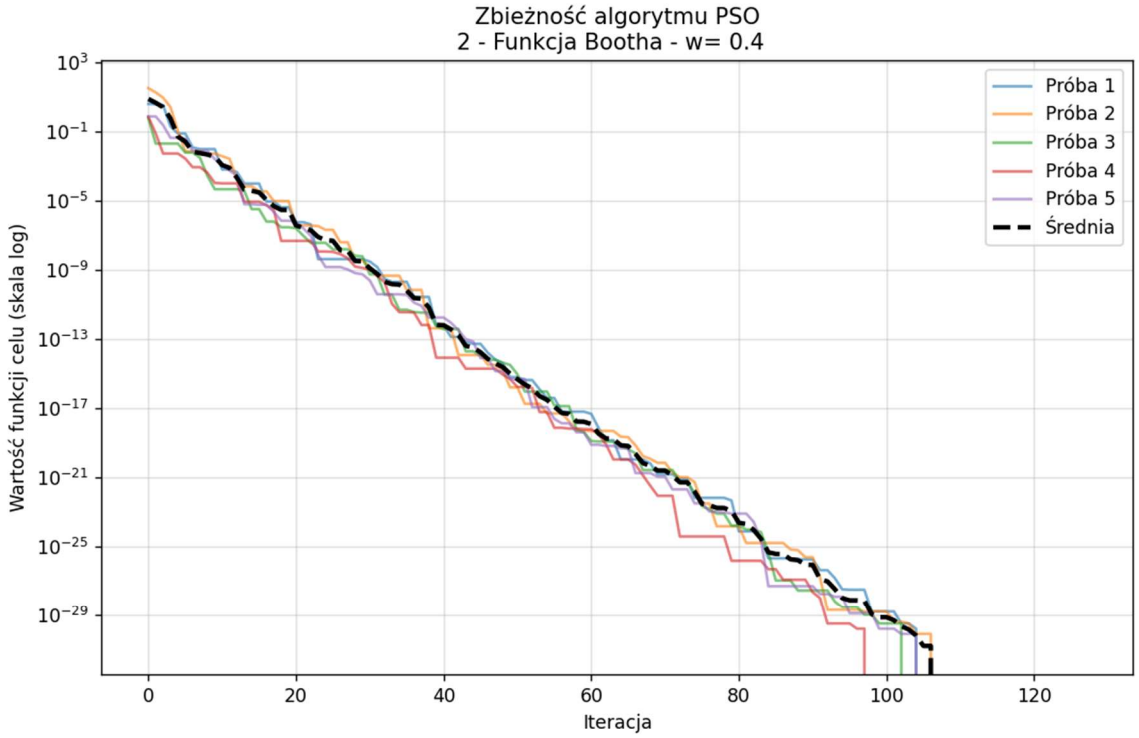


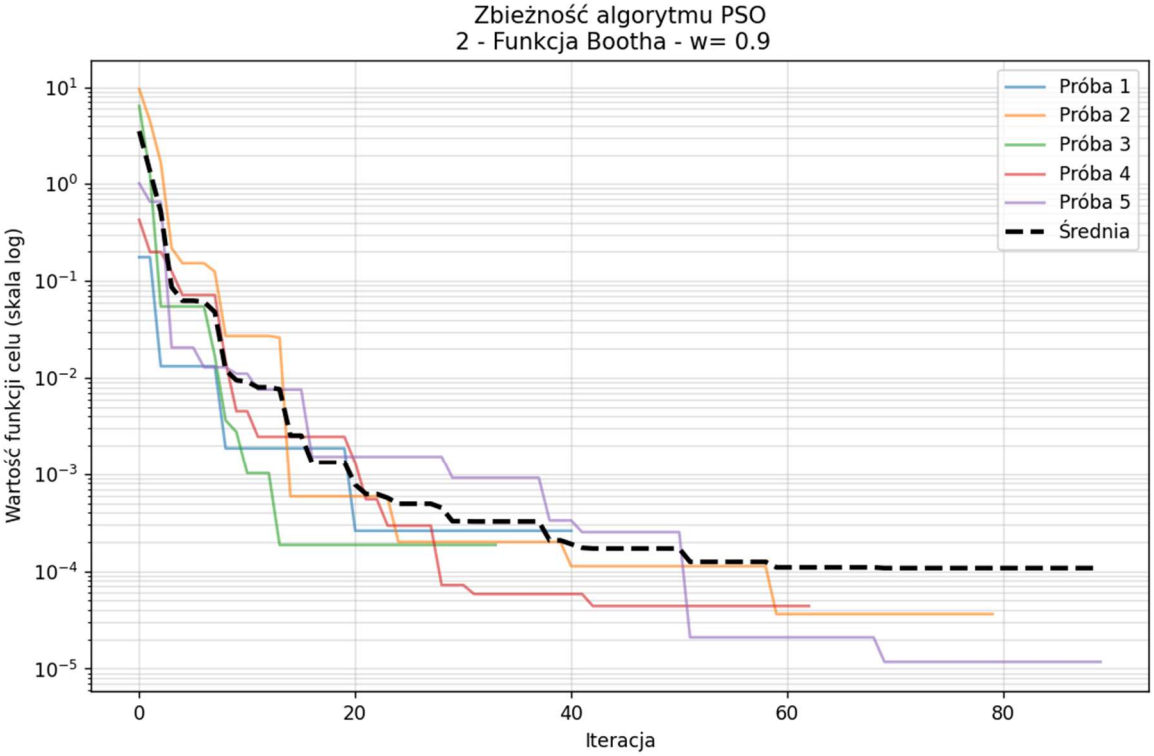

### 5.3.2 Badanie inercji

Współczynnik inercji (w) dla których przeprowadzone zostały eksperymenty to w = {0.4, 0.9}

| Inercja (w) | Próba | Wynik (f(x,y)) | x | y |
|---|---|---|---|---|
| 0.4 | 1-5 | 0.000000e+00 | 1.000000 | 3.000000 |
| 0.9 | 1 | 2.606259e−04 | 0.996672 | 2.995724 |
| | 2 | 3.639916e−05 | 1.003164 | 2.999386 |
| | 3 | 1.866543e−04 | 1.009494 | 2.994614 |
| | 4 | 4.400872e−05 | 1.002078 | 2.995645 |
| | 5 | 1.175160e−05 | 0.999411 | 3.001963 |

| Konfiguracja | Najlepszy (Best) | Najgorszy (Worst) | Średnia (Mean) | Mediana | Std (Odchylenie) |
|---|---|---|---|---|---|
| w=0.4 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 |
| w=0.9 | 1.175160e−05 | 2.606259e−04 | 1.078879e−04 | 4.400872e−05 | 9.794532e−05 |



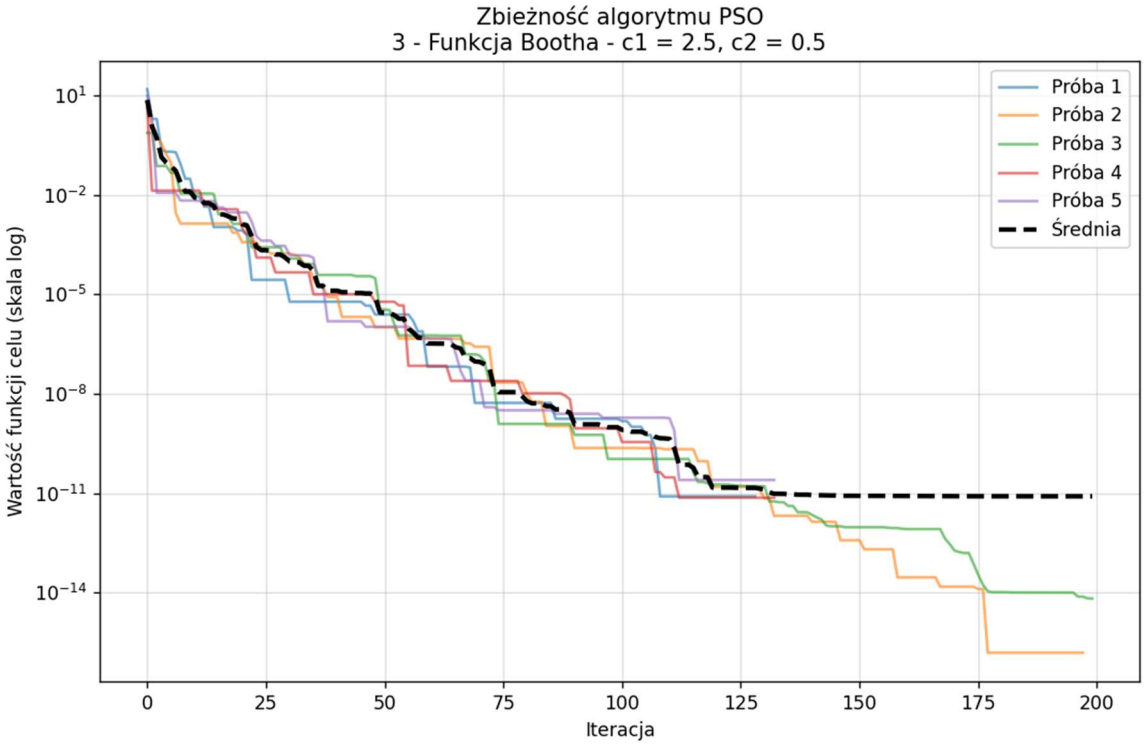

### 5.3.3 Psychologia Roju

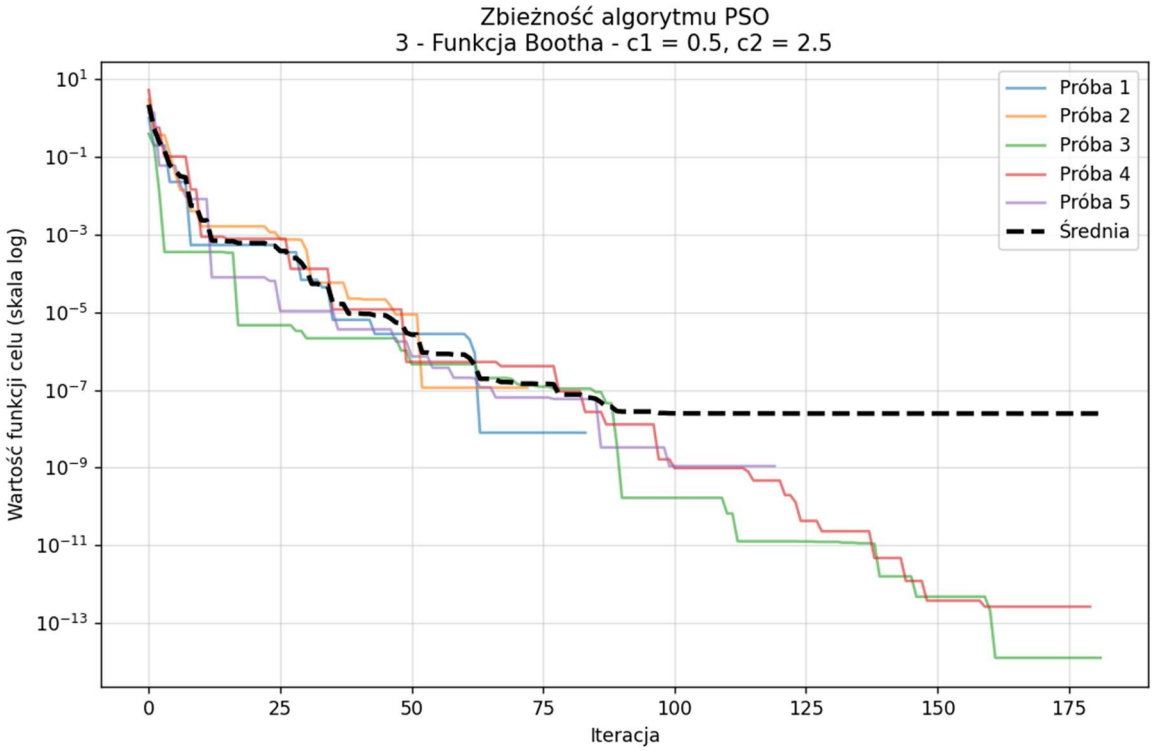
W tym eksperymencie badane będą wyniki dla konfiguracji współczynnika kognitywnego i socjalnego. Zawierało to będzie wynik dla indywidualistów (wysoki kognitywny, niski socjalny) i naśladowców (niski kognitywny i wysoki socjalny). Badanie będzie dla c1 = 2.5, c2 = 0.5 i c1 = 0.5, c2 = 2.5.

| Konfiguracja | Próba | Wynik (f(x,y)) | x | y |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| | 1 | 8.085533e−12 | 1.000002 | 2.999998 |
| | 2 | 1.559307e−16 | 1.000000 | 3.000000 |
| | 3 | 6.708110e−15 | 1.000000 | 3.000000 |
| | 4 | 7.394685e−12 | 1.000000 | 2.999999 |
| c1=2.5,c2=0.5 | 5 | 2.526740e−11 | 1.000003 | 2.999998 |
| | 1 | 7.930758e−09 | 1.000052 | 2.999983 |
| | 2 | 1.143738e−07 | 0.999996 | 3.000154 |
| | 3 | 1.257456e−14 | 1.000000 | 3.000000 |
| | 4 | 2.612295e−13 | 1.000000 | 3.000000 |
| c1=0.5,c2=2.5 | 5 | 1.089749e−09 | 0.999985 | 3.000024 |

| Parametry | Najlepszy (Best) | Najgorszy (Worst) | Średnia (Mean) | Mediana | Std (Odchylenie) |
|---|---|---|---|---|---|
| c1=2.5,c2=0.5 | 1.559307e−16 | 2.526740e−11 | 8.150897e−12 | 7.394685e−12 | 9.233778e−12 |
| c1=0.5,c2=2.5 | 1.257456e−14 | 1.143738e−07 | 2.467892e−08 | 1.089749e−09 | 4.494488e−08 |



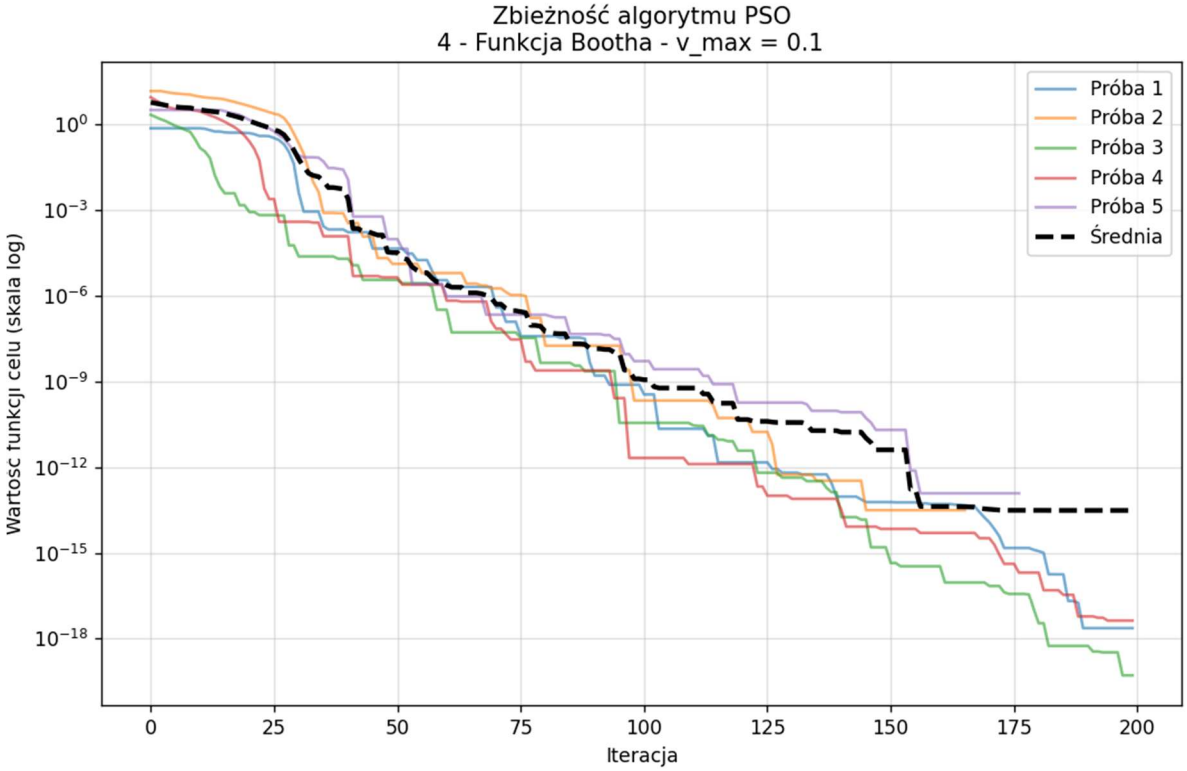


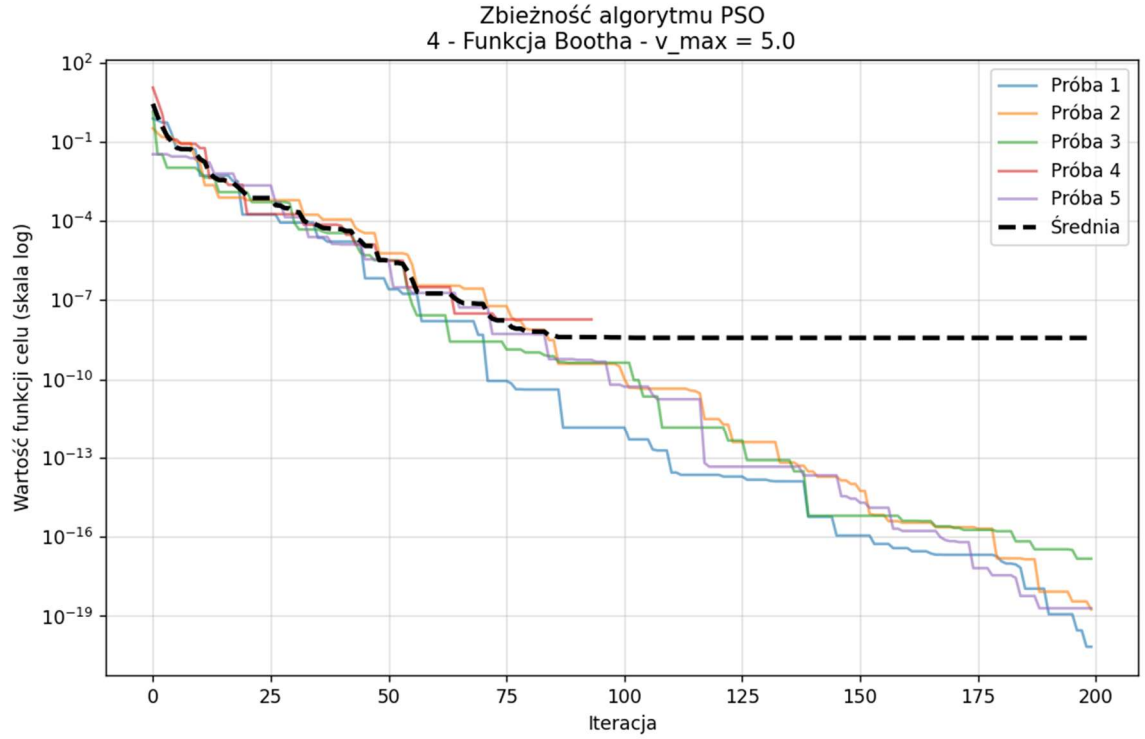

### 5.3.4 Eksperyment maksymalnej prędkości

W tym eksperymencie testowana będzie prędkość v_max = {0.1, 5.0}.

| Vmax | Próba | Wynik (f(x,y)) | x | y |
|---|---|---|---|---|
| 0.1 | 1 | 2.270904e−18 | 1.000000 | 3.000000 |
| | 2 | 3.253720e−14 | 1.000000 | 3.000000 |
| | 3 | 5.101795e−20 | 1.000000 | 3.000000 |
| | 4 | 4.158145e−18 | 1.000000 | 3.000000 |
| | 5 | 1.264859e−13 | 1.000000 | 3.000000 |
| 5.0 | 1 | 6.872845e−21 | 1.000000 | 3.000000 |
| | 2 | 1.783817e−19 | 1.000000 | 3.000000 |
| | 3 | 1.491694e−17 | 1.000000 | 3.000000 |
| | 4 | 1.857378e−08 | 0.999934 | 3.000006 |
| | 5 | 1.979623e−19 | 1.000000 | 3.000000 |

| Parametry | Najlepszy (Best) | Najgorszy (Worst) | Średnia (Mean) | Mediana | Std (Odchylenie) |
|---|---|---|---|---|---|
| Vmax=0.1 | 5.101795e−20 | 1.264859e−13 | 3.180592e−14 | 4.158145e−18 | 4.898832e−14 |
| Vmax=5.0 | 6.872845e−21 | 1.857378e−08 | 3.714755e−09 | 1.979623e−19 | 7.429510e−09 |




# 6. Analiza wyników

## 6.1 Jak wybrane parametry wpływają na jakość wyników?

Jakość rozwiązania była ściśle skorelowana z doborem współczynnika inercji (w) oraz balansem między parametrami kognitywnymi a socjalnymi.
**Wpływ inercji (w):** Zaobserwowano, że niska inercja (w=0.4) sprzyja eksploatacji (dokładnemu przeszukiwaniu otoczenia). Dla obu funkcji pozwoliła ona uzyskać wyniki bliskie idealnemu zeru (rzędu 10^-31) dla Himmelblaua i absolutne 0.0 dla Bootha. Wysoka inercja (w=0.9) znacząco pogorszyła precyzję (błąd rzędu 10^-3), powodując zjawisko "przestrzeliwania" minimum przez rozpędzone cząstki.
**Wpływ parametrów c1 i c2:**

Dla funkcji prostych (Booth), przewaga czynnika kognitywnego (c1 > c2) dała lepsze rezultaty, pozwalając cząstkom na niezależne i precyzyjne dostrojenie pozycji.
Dla funkcji trudniejszych (Himmelblau), przewaga czynnika socjalnego (c2 > c1) okazała się bezpieczniejsza, gwarantując stabilne zbieganie całego roju, podczas gdy strategia kognitywna była nieprzewidywalna (duży rozrzut wyników).

**Wpływ prędkości (V_max):** Restrykcyjne ograniczenie prędkości (V_max=0.1) niosło ryzyko nieosiągnięcia celu w zadanym limicie iteracji. Wysoka prędkość (V_max=5.0) pozwoliła na szybką eksplorację, choć wiązała się z ryzykiem drobnych fluktuacji w końcowej fazie.

### 6.2 Czy algorytm jest stabilny (analiza *std*)?

**Warianty stabilne:** Dla konfiguracji bazowej (w=0.75) oraz konfiguracji z niską inercją (w=0.4), odchylenie standardowe było minimalne (często bliskie zeru). Oznacza to, że algorytm jest deterministycznie powtarzalny w zakresie jakości wyniku.
**Warianty niestabilne:** Największą niestabilność odnotowano dla wysokiej inercji (w=0.9) oraz dla strategii kognitywnej (naśladowców) przy funkcji wielomodalnej (c1=2.5 na Himmelblau). W tych przypadkach wyniki poszczególnych prób różniły się od siebie nawet o 5 rzędów wielkości.

### 6.3 Czy zaobserwowano szybkie czy wolne zbieganie do rozwiązania?

Większość konfiguracji zbiegała do rozwiązania bardzo szybko, osiągając optymalny obszar już w pierwszych **20-30 iteracjach**.
Zastosowanie wysokiego parametru socjalnego (c2=2.5) powodowało niemal natychmiastowe "zapadanie się" roju w minimum (szybka zbieżność), co jednak niosło ryzyko przedwczesnej zbieżności.
Zbyt wolne zbieganie zaobserwowano jedynie przy drastycznym ograniczeniu prędkości (V_max=0.1), gdzie w jednej z prób cząstki nie zdążyły dotrzeć do minimum globalnego przed upływem 100 iteracji.

### 6.4 Czy pojawia się ryzyko utknięcia w minimum lokalnym?

**Funkcja Bootha:** Ze względu na unimodalny charakter funkcji (tylko jedno dno), ryzyko utknięcia w minimum lokalnym nie występowało.

**Funkcja Himmelblaua:** Funkcja ta posiada 4 minima globalne i lokalne wypłaszczenia. Algorytm wykazał się odpornością na utknięcie w złych rozwiązaniach, skutecznie odnajdując jedno z czterech minimów globalnych. Zaobserwowano jednak zjawisko, w którym przy różnych uruchomieniach algorytm zbiegał do **różnych** minimów globalnych. Jest to zachowanie pożądane, świadczące o dobrej eksploracji przestrzeni przeszukiwań.

### 6.5 Czy trudność obu funkcji była różna?

**Funkcja Bootha (Łatwa):** Algorytm osiągał tu wyniki perfekcyjne bez względu na drobne zmiany parametrów. Strategia niskiej inercji (w=0.4) pozwoliła na natychmiastowe rozwiązanie problemu z zerowym odchyleniem standardowym.
**Funkcja Himmelblaua (Trudniejsza):** Mimo że algorytm skutecznie znajdował minima, wyniki charakteryzowały się większą wrażliwością na dobór parametrów. Źle dobrane współczynniki (np. zbyt duży nacisk na indywidualizm c1) powodowały wyraźny spadek jakości i stabilności rozwiązania, czego nie obserwowano przy funkcji Bootha. Wymagała ona lepszego balansu między eksploracją a eksploatacją.

# 7. Wnioski

### 7.1. Które konfiguracje PSO okazały się najlepsze?

Na podstawie wyników można wskazać konfiguracje w zależności od charakteru problemu:

**Najwyższa precyzja:** Najlepsze wyniki liczbowe osiągnęła konfiguracja z **niską inercją (w=0.4)**.

Dla funkcji Bootha osiągnęła ona wynik idealny (0.0).
Dla funkcji Himmelblaua osiągnęła precyzję rzędu 10^-31.

**Największa stabilność (Bezpieczny Wybór):** Konfiguracja z przewagą **parametru socjalnego (c1=0.5, c2=2.5)**.

Konfiguracja ta zapewniała najmniejszy rozrzut wyników. Gwarantuje to, że algorytm nie zawiedzie w żadnym uruchomieniu.

**Najlepszy kompromis:** Konfiguracja **bazowa** (w=0.75, c1=1.5, c2=1.5) potwierdziła swoją użyteczność jako punkt wyjścia. Zapewniała bardzo dobre wyniki (10^-19) przy zachowaniu stabilności.

### 7.2. Jaki wpływ miały parametry algorytmu?

Eksperymenty pozwoliły na zidentyfikowanie kluczowych ról poszczególnych parametrów:

**Współczynnik inercji (w):** To najważniejszy parametr sterujący balansem między eksploracją a eksploatacją.

Niska wartość (0.4) sprzyja dokładnemu przeszukiwaniu lokalnemu (eksploatacji).

Wysoka wartość (0.9) sprzyja eksploracji, ale pogarsza precyzję (w eksperymencie różnica wyniosła aż 27 rzędów wielkości).

**Współczynniki c1 (kognitywny) i c2 (socjalny):**

Dominacja c1 pozwala na znalezienie wybitnych rozwiązań, ale jest ryzykowna i mniej stabilna.

Dominacja c2 wymusza spójność roju, zapobiegając błądzeniu pojedynczych cząstek.

**Prędkość maksymalna (V_max):**

Wysokie limity prędkości (V_max=5.0) okazały się korzystniejsze niż restrykcyjne (V_max=0.1). Możliwość szybkiego przemieszczania się po mapie jest kluczowa dla znalezienia obszaru minimum.

### 7.3. Czy algorytm skutecznie znalazł minimum dla obu funkcji?

**Funkcja Bootha (Unimodalna):**

Algorytm działał bezbłędnie. Dla optymalnych ustawień błąd wynosił absolutne 0.0, a dla ustawień bazowych był na poziomie granicy precyzji komputera (10^-17). Świadczy to o pełnym sukcesie optymalizacji.

**Funkcja Himmelblaua (Wielomodalna):**

Mimo istnienia czterech minimów globalnych i wielu pułapek, algorytm ani razu nie utknął w sposób trwały w minimum lokalnym (lokalnym wzniesieniu).

W każdej próbie odnajdywał jedno z czterech minimów globalnych z precyzją przynajmniej rzędu 10^-6).