|  |  |  |  |
| --- | --- | --- | --- |
| Obraz zawierający symbol, Grafika, logo, Prostokąt  Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna. | Politechnika Bydgoska im. J. J. Śniadeckich  Wydział Telekomunikacji,  Informatyki i Elektrotechniki  **Zakład Systemów Teleinformatycznych** | | Obraz zawierający symbol, logo, Grafika, Czcionka  Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna. |
| **Przedmiot** | Algorytmy genetyczne i sztuczne sieci neuronowe | | |
| **Prowadzący** | dr hab. inż. prof. PBŚ Tomasz Talaśka | | |
| **Temat** | **Zastosowanie algorytmu rojowego wieloryba (WOA)** | | |
| **Student** | Cezary Tytko | | |
| **Ocena** |  | **Data oddania spr.** |  |

1. Wstęp

Algorytmy rojowe to metaheurystyczne metody optymalizacyjne inspirowane zachowaniami zbiorowymi zwierząt. W niniejszym sprawozdaniu przedstawiono zastosowanie algorytmu rojowego wieloryba (Whale Optimization Algorithm), który symuluje technikę polowania delfinowatych (głównie humbaków), znaną jako bąbelkowa siatka spiralna (bubble-net feeding).

Celem eksperymentu było zaimplementowanie i przetestowanie WOA na wybranych funkcjach testowych o 3 i 10 wymiarach oraz ocena jego skuteczności.

**2. Opis algorytmu WOA**

Algorytm WOA opiera się na trzech głównych etapach:

**a) Zachowanie spiralne (eksploatacja):**

Wieloryby otaczają zdobycz i poruszają się po spirali w jej kierunku. Modelowane matematycznie jako:



Gdzie:











**b) Kurczenie okręgu (eksploatacja):**

Modelowane jako:



Gdzie:









**c) Losowe przeszukiwanie przestrzeni (eksploracja):**

Zachodzi, gdy ∣A∣>1, a nowe położenie wybierane jest w odniesieniu do losowego osobnika.



**3. Implementacja algorytmu rojowego wieloryba (WOA)**

W projekcie zaimplementowano algorytm optymalizacyjny inspirowany zachowaniem żerujących wielorybów, znany jako Whale Optimization Algorithm (WOA). Algorytm ten bazuje na trzech głównych mechanizmach: kurczeniu się okręgu, spiralnym ruchu wieloryba wokół ofiary oraz losowym przeszukiwaniu przestrzeni rozwiązań.

Główna logika algorytmu została zaimplementowana w klasie WhaleOptimizationAlgorithmRunner, która odpowiada za:

* Inicjalizację populacji kandydatów (wielorybów),
* Iteracyjne przeszukiwanie przestrzeni rozwiązań,
* Zastosowanie strategii eksploatacji i eksploracji,
* Monitorowanie i aktualizację najlepszego znalezionego rozwiązania.

**Parametry**

* PopulationSize – liczba osobników w populacji,
* MaxIterations – maksymalna liczba iteracji,
* Dimensions – liczba wymiarów funkcji celu,
* AMax – maksymalna wartość współczynnika a,
* B – stała kontrolująca kształt spirali.
* MinX, MaxX - zakres poszukiwań

**Opis działania**

1. **Inicjalizacja populacji**: Losowo generowana jest początkowa populacja wektorów decyzyjnych w zakresie [MinX, MaxX].
2. **Główna pętla optymalizacji**:
   * Dla każdej iteracji obliczana jest wartość współczynnika aaa, który maleje liniowo od AMax do 0.
   * Dla każdego osobnika w populacji losowana jest strategia ruchu:
     + Z prawdopodobieństwem 50% stosowany jest ruch **eksploatacyjny** — przyciąganie do najlepszego znanego osobnika (kurczenie się okręgu).
     + W przeciwnym wypadku wykonywany jest **ruch spiralny** wokół najlepszego osobnika.
   * Po każdej iteracji aktualizowane jest najlepsze znane rozwiązanie.
3. Zestaw dostępnych funkcji optymalizacyjnych

W celu przetestowania działania algorytmu rojowego wieloryba (WOA) zaimplementowano popularne benchmarkowe funkcje optymalizacyjne. Funkcje te różnią się krajobrazem przestrzeni rozwiązań – niektóre są wypukłe i gładkie, inne mają wiele lokalnych minimów. Każda z nich posiada znane globalne minimum.

1. Sphere Function

Funkcja wypukła, często używana jako podstawowy test:

Obraz zawierający tekst, Czcionka, biały, design

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Minimum globalne: f(0,…,0)=0f(0,…,0)=0

Charakterystyka: jedno minimum, brak lokalnych minimów

**2. Rastrigin Function**

Funkcja multimodalna o dużej liczbie lokalnych minimów:

Obraz zawierający Czcionka, biały, kaligrafia, tekst

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Minimum globalne: f(0,…,0)=0f(0,…,0)=0

Charakterystyka: wiele lokalnych minimów, trudniejsza do optymalizacji

**3. Rosenbrock Function**

Popularna funkcja o zakrzywionym, wąskim kanale:

Obraz zawierający Czcionka, tekst, biały, typografia

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Minimum globalne: f(1,…,1)=0f(1,…,1)=0

Charakterystyka: trudna do zbieżności, mimo jednego minimum

**4. Ackley Function**

Funkcja z wieloma minimami lokalnymi:

Obraz zawierający Czcionka, tekst, biały, pismo odręczne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Minimum globalne: f(0,…,0)=0f(0,…,0)=0

Charakterystyka: złożony krajobraz

**5. Griewank Function**

Funkcja z szerokim minimum globalnym:

Obraz zawierający Czcionka, tekst, biały, linia

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Minimum globalne: f(0,…,0)=0f(0,…,0)=0

Charakterystyka: liczne lokalne minima, ale mniej intensywne

**6. Schwefel Function**

Bardzo złożona funkcja o głębokich minimach:

Obraz zawierający Czcionka, tekst, biały, typografia

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Minimum globalne: f(420,9687,…,420,9687)≈0

Charakterystyka: głębokie lokalne minima i zmienne pole

**7. Zakharov Function**

Funkcja z dodatkowymi terminami kwadratowymi i potęgowymi:

Obraz zawierający Czcionka, biały, linia, pismo odręczne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Minimum globalne: f(0,…,0)=0f(0,…,0)=0

Charakterystyka: gładka, ale złożona funkcja celu

**4. Instrukcja użytkowania programu ConsoleApp.exe**

Program ConsoleApp.exe to aplikacja konsolowa służąca do testowania algorytmów optymalizacyjnych. Pozwala uruchomić symulację optymalizacji dla wybranej funkcji testowej w określonej przestrzeni wymiarowej.

.\ConsoleApp.exe [opcje]

| **Opcja** | **Typ** | **Opis** |
| --- | --- | --- |
| --population-size <Int32> | liczba | Liczba osobników w populacji (np. 30) |
| --dimensions <Int32> | liczba | Liczba wymiarów przestrzeni (np. 10) |
| --max-iterations <Int32> | liczba | Maksymalna liczba iteracji algorytmu |
| --a-max <Double> | liczba | Parametr maksymalny „a” algorytmu wielorybiego |
| --b <Double> | liczba | Parametr „b” sterujący spiralą algorytmu |
| --min-x <Double> | liczba | Minimalna wartość zmiennej decyzyjnej |
| --max-x <Double> | liczba | Maksymalna wartość zmiennej decyzyjnej |
| --optimization-function <OptimizationFunction> | tekst | Wybór funkcji optymalizacyjnej do testu |
| --help, -h | - | Wyświetlenie pomocy |
| --version | - | Wyświetlenie wersji aplikacji |

**.\ConsoleApp.exe --optimization-function Zakharov --dimensions 10 --population-size 30 --max-iterations 100 --a-max 2 --b 1 --min-x -10 --max-x 10**

**Dostępne funkcje optymalizacyjne**

* Sphere
* Rastrigin
* Rosenbrock
* Ackley
* Griewank
* Schwefel
* Zakharov

5. Wyniki

**Parametry wspólne:**

* --population-size 30
* --max-iterations 100
* --a-max 2
* --b 1
* --min-x -10.0
* --max-x 10.0

|  |  |  |
| --- | --- | --- |
| Funkcja optymalizacyjna | Liczba wymiarów | |
| 3 | 10 |
| Sphere | 5,8803317977835176E-27 | 4,1134637214998384E-18 |
| Rastrigin | 1,861054042416481E-15 | 1,4210854715202004E-14 |
| Rosenbrock | 0622848621955523102 | 8,895805636262764 |
| Ackley | 5,0182080713057076E-14 | 2,1156201235328354E-09 |
| Griewank | 1,146804348262381E-16 | 2,065522625796219E-15 |
| Schwefel \* | 1,0315150073372479 | 4,6531864394419244 |
| Zakharov | 1,0544400458642516E-10 | 306,5520085442147 |

Dla funkcji Schwefel zastosowano przedział -500 do 500