Cat Swarm Optimization

Prezentacja algorytmu.

Andrii Shekhovtsov,
Maciej Luberadzki

21 kwietnia 2020 roku

Algorytm Cat Swarm Optimization został przedstawiony w artykule "Cat Swarm Optimization" przez Shu-Chuan Chu z Cheng Shiu University i Pei-wei Tsai oraz Jeng-Shyang Pan z National Kaohsiung University of Applied Sciences w 2006 roku.

Metafora

Algorytm został zainspirowany wnioskami z obserwacji zachowania kotów. Pomimo, że istnieją około 32 gatunki kotów, żyjących w znacząco różniących się od siebie środowiskach, wszystkie łączy pewien zestaw zachowań. Autorzy algorytmu zwrócili uwagę na ogólne przystosowanie kotów do polowania.

Metafora

Podstawą dla algorytmu jest spostrzeżenie, że koty przez znaczną część czasu „odpoczywają”, a tylko w niektórych chwilach są aktywne. Nie oznacza to jednak, że w trakcie odpoczynku są całkowicie bezczynne. Przeciwnie, cały czas bacznie obserwują swoje otoczenie, wypatrując potencjalnych ofiar. W momencie, gdy pojawi się okazja, zaczynają śledzić swoją ofiarę. Właśnie te dwa stany – aktywny (tracing mode) i „odpoczywający” (seeking mode) – autorzy postanowili zamodelować w Cat Swarm Optimization.

Sposób działania

W algorytmie zastosowano model kota składający się z następujących składowych:

- Pozycji w M-wymiarowej przestrzeni optymalizowanego problemu,
- Wartości szybkości poruszania się kota w każdym z M wymiarów,
- Wartości funkcji celu,
- Flagi określającej, w którym z dwóch stanów znajduje się kot.

Ostatecznym rozwiązaniem jest pozycja kota.

Pętla główna algorytmu - pseudokod

```
cats = initialize_cats()
while not stop_condition():
    for cat in cats:
        cat.f_value = fitness_function(cat.position)

    best_cat = choose_best_cat(cats)

    seeking_cats, tracing_cats = split_cats(cats, MR)

    for cat in seeking_cats:
        cat = seeking_mode(cat)

    for cat in tracing_cats:
        cat = tracing_mode(cat)

return choose_best_cat(cats)
```

Seeking Mode

Seeking mode – stan, w którym kot nie tropi aktywnie ofiary, lecz wypatruje. Zdefiniowane zostały cztery czynniki:

- **SMP** [1, ∞] – seeking memory pool (ilość rozpatrywanych pozycji)
- **SRD** [0, 1] – seeking range of the selected dimension (jak mocno będą zmodyfikowane wybrane wymiary)
- **CDC** [0, 1] – count of dimensions to change (ile procentowo wymiarów zostanie zmienionych)
- **SPC** (bool) – self-position considering (czy obecna pozycja jest rozpatrywana jako następna)

Seeking Mode – pseudokod

```
def seeking_mode(cat):
    j = SMP
    if SPC:
        j = SMP - 1

    copies = make_cat_copies(cat, j)
    for copy in copies:
        modify_position(copy.position, CDC, SRD)
        copy.f_value = fitness_function(copy.position)

    if max(copies) == min(copies):
        return copies[0] # No matter what copy it will be
    elif SPC: # If true, also use current position in selection
        return roulette_selection(copies + cat)
    else: # If not, use only copies in selection
        return roulette_selection(copies)
```

Roulette selection

Prawdopodobieństwa dla selekcji roletkowej są wyliczane następującym wzorem:

$$\mathrm{Pi} = \frac{\left|\mathrm{FS}_i - \mathrm{FS}_b\right|}{\mathrm{FS}_{\max} - \mathrm{FS}_{\min}}, \quad \text{where } 0 < i < j.$$

W tym wzorze zamiast FS_b trzeba podstawić FS_{max}, jeżeli algorytm minimalizuje funkcje i FS_{min}, jeżeli algorytm maksymalizuje funkcje.

Tracing Mode

Stan, w którym kot podąża za "ofiarą", zgodnie z przypisanymi do kolejnych wymiarów prędkościami.

Tracing Mode – psudokod

```
def tracing_mode(cat):
    r1 = rand()
    c1 = 0.5 # Const
    cat.velocity = cat.velocity + r1 * c1
        * (best_cat.position - cat.position)
    cat.velocity[cat.velocity > self.MV] = self.MV
    cat.position = cat.position + cat.velocity
    return cat
```

Porównanie z innymi algorytmami

W celach badawczych przeprowadzono porównanie działania algorytmu Cat Swarm Optimization z działaniem algorytmu Strategie ewolucyjne porównywane będą implementacje wykonane przez autorów prezentacji.

Działanie algorytmów będzie porównywane dla funkcji Rastrigin (http://www.sfu.ca/~ssurjano/rastr.html), na kolejnym slajdzie podane są nastawienia obu algorytmów.

Nastawienia algorytmów

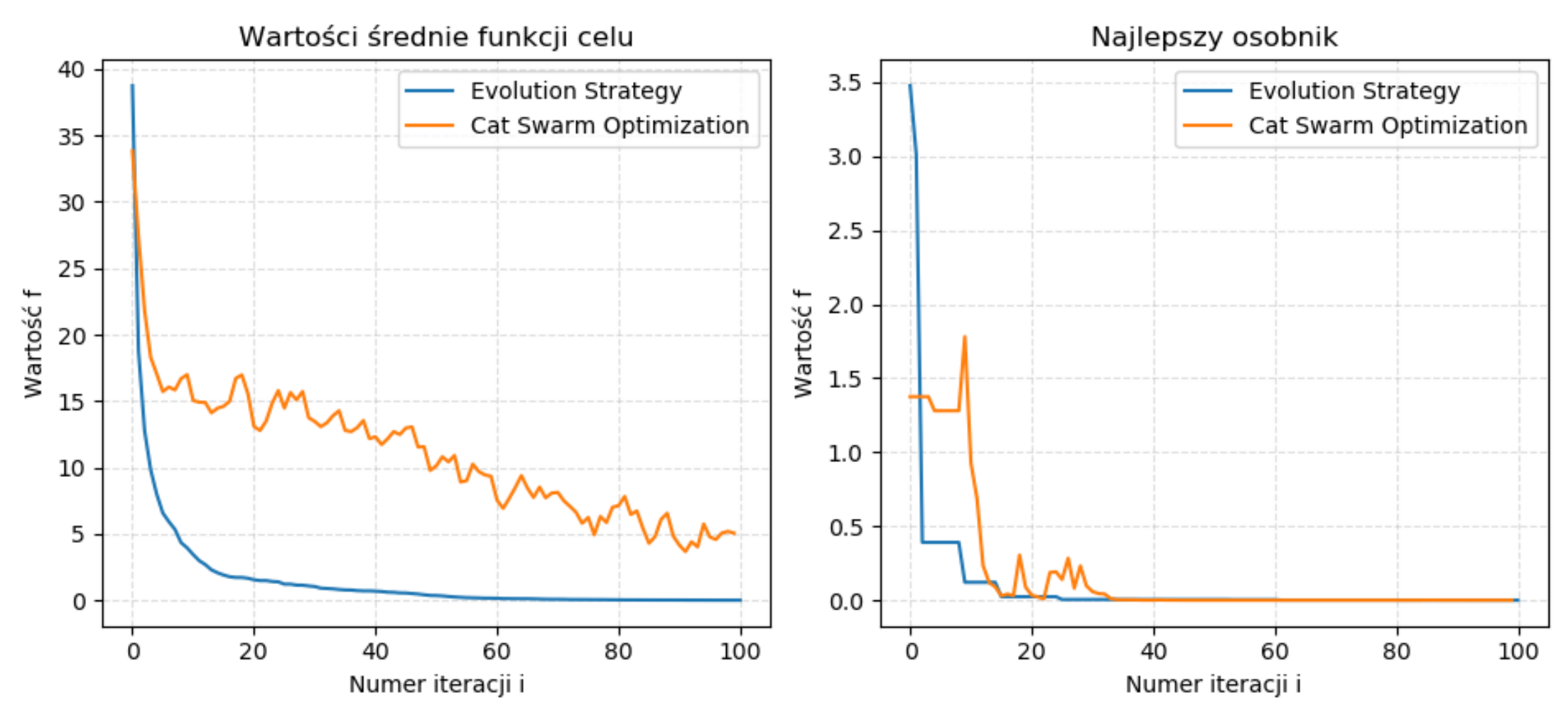
Strategie ewolucyjne

- $\mu = 50$
- $\rho = 10$
- $\boldsymbol{\lambda} = 70$
- Typ selekcji: "+"

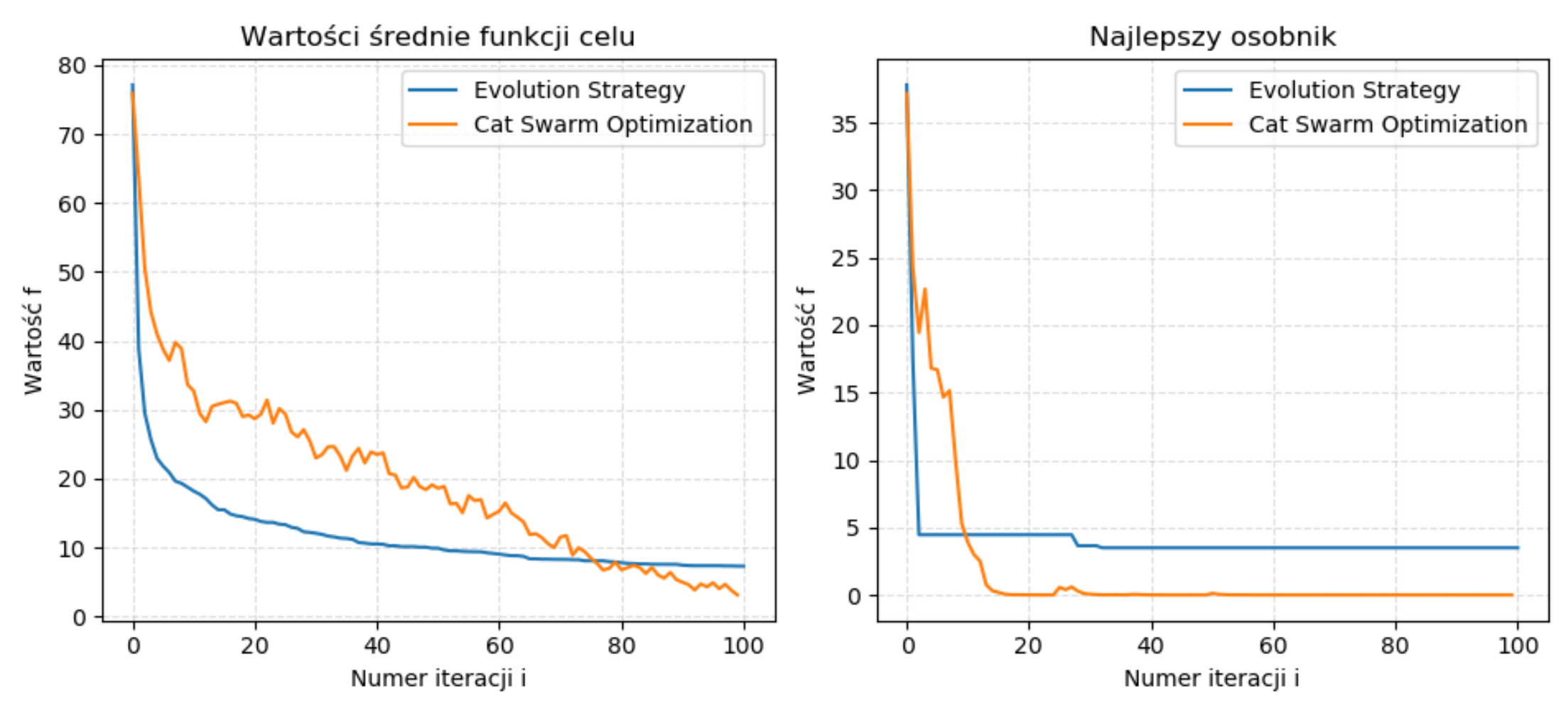
Cat Swarm optimization

- N = 50
- MR = 0.2
- MV = 2
- SMP = 5
- CRD = 0.5
- CDC = 0.5
- SPC = True

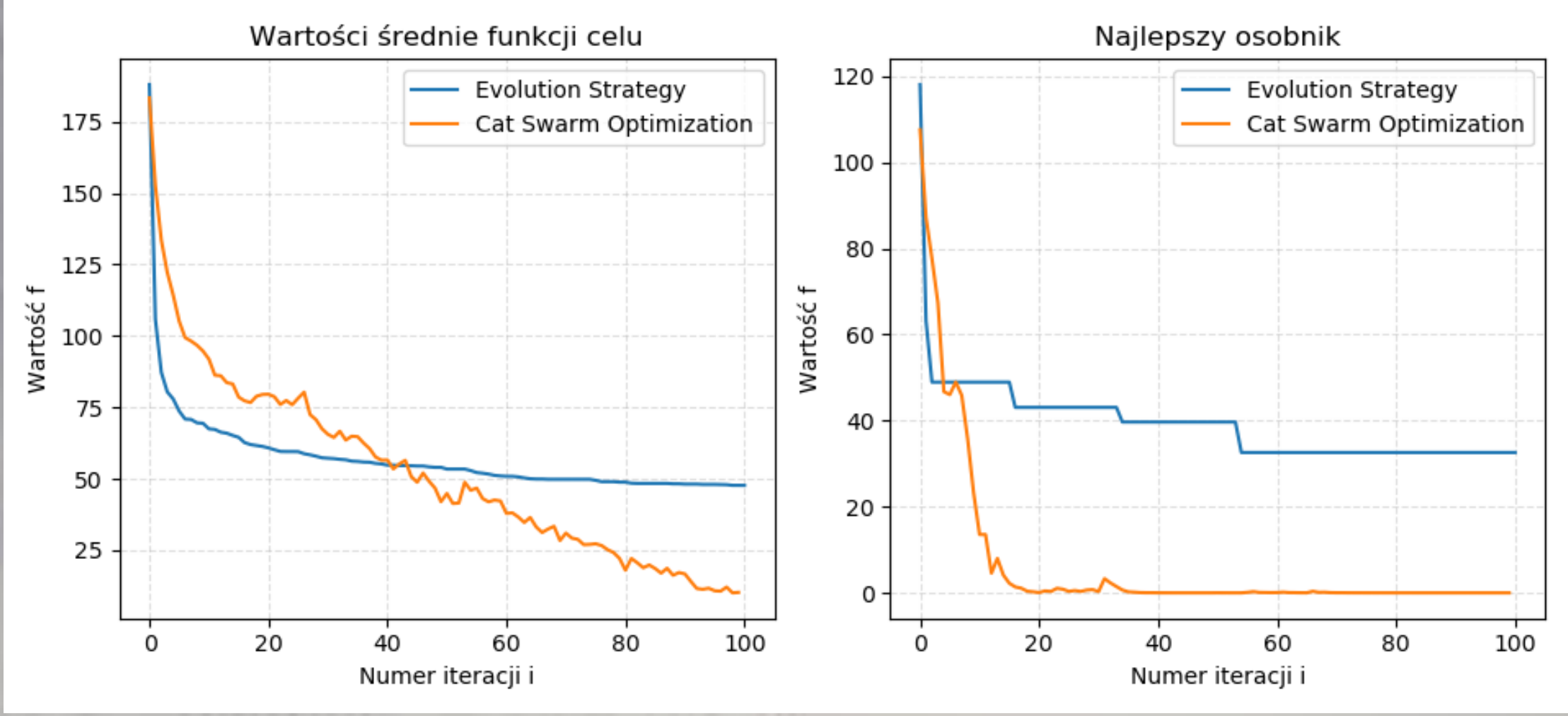
Rastrigin, 2 wymiary



Rastrigin, 4 wymiary



Rastrigin, 10 wymiarów



Demonstracja przemieszczania się cząstek

https://imgur.com/a/Rumy12i

Modyfikacje

Ahmed i in. [5] w swoim podsumowującym artykule wymienili 23 algorytmy będące modyfikacjami oryginalnego Cat Swarm Optimization. Jedną z nich jest PCSO – Parallel Cat Swarm Optimization, zaproponowane przez Tsai i in. [2]. Modyfikacja polega na podzieleniu kotów na klastry, gdzie w każdym jest przechowywany najlepsze rozwiązanie dla danego klastra oraz wymianie, w każdej iteracji, kota z najgorszą wartością funkcji celu na kota o najlepszej wartości funkcji celu, pochodzącego z innego klastra.

Zastosowania

Wang i in. w swoim artykule z 2012 roku [3], zastosował Cat Swarm Optimization w dziedzine steganografii, do optymalizacji metody simple least-significant-bit, aby otrzymać obraz (stego-image), który będzie ukrywał w sobie sekret i jednocześnie będzie odbiegał w najmniejszym stopniu od oryginału (najmniejszy błąd średniokwadratowy). Wyniki zostały porównane z wynikami otrzymanymi przy użyciu wyszukiwania wyczerpującego. Dla obrazów o wymiarze 256x256 algorytm znajdował optymalne rozwiązanie, a dla obrazów o wymiarze 512x256, gdzie wyszukiwanie wyczerpujące nie dawało już rozwiązania, Cat Swarm Oprimization znajdował rozwiązanie bliskie optymalnemu w ciągu około 2 minut.

Zastosowania

Yusiong [4] w 2013 roku zaprezentował nowy algorytm sieci neuronowej – CSONN-OBD, czyli sieć neuronową, czyli sieć wykorzystującą Cat Swarm Optimization jako algorytm uczący i Optimal Brain Damage jako metodę przycinającą i optymalizującą strukturę sieci. Testując działanie nowej metody uzyskał satysfakcjonujące wyniki, co oznacza, że zarówno Cat Swarm Optimization, jak i Optimal Brain Damage, spełniły swoje zadanie.

Zastosowania

We wspomnianym wcześniej artykule Ahmeda i in. [5], wymienione zostały następujące dziedziny, w których zastosowanie znalazło Cat Swarm Optimization:

- Elektrotechnika,
- Przetwarzanie sygnałów,
- System Management and Combinatorial Optimization,
- Bezprzewodowe sieci czujnikowe,
- Inżynieria naftowa,
- Inżynieria lądowa.

Bonus: Kotek który znalazł minimum lokalne


Dziękujemy za uwagę

Bibliografia

1. Chu, S.-C., Tsai, P.-W., and Pan, J.-S., "Cat Swarm Optimization," In Proceedings of the 9th Pacific Rim International Conference on Artificial Intelligence, LNAI 4099, 2006, pp. 854-858.
2. P. Tsai and V. Istanda, "Review on cat swarm optimization algorithms," *2013 3rd International Conference on Consumer Electronics, Communications and Networks*, Xianning, 2013, pp. 564-567.
3. Wang, Z.-H., Chang, C.-C., and Li, M.-C., "Optimizing Least-significant-bit Substitution Using Cat Swarm Optimization Strategy," Information Science, vol. 192, 2012, pp. 98-108.

Bibliografia

4. Yusiong, J. P. Y., „Optimizing Artificial Neural Networks Using Cat Swarm Optimization Algorithm,” International Journal of Intelligent Systems And Applications, vol. 01, 2013, pp. 69-80.
5. Ahmed, A. M., Rashid, T. A., and Saeed, S. A. M., „Cat Swarm Optimization Algorithm: A Survey and Performance Evaluation,” Computational Intelligence and Neuroscience, 2020, 1-20.