

INTELIGENCJA ROJU

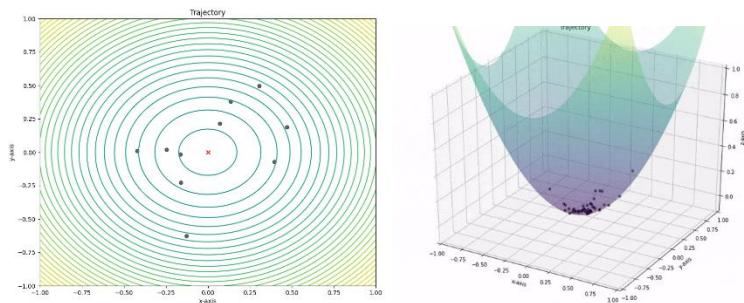
ALGORYTMY PSO I ACO

PACZKA PYSWARMS

Spróbuj uruchomić przykłady pokazane na wykładzie.

PRZYKŁAD 1 Z WYKŁADU

Użyjemy paczki **pyswarms** (<https://pyswarms.readthedocs.io/en/latest/index.html>), by uruchomić algorytm **Particle Swarm Optimizer** do wyszukiwania minimum funkcji sferycznej, która ma minimum 0 we współrzędnych (0,0) – patrz rysunki.



Ściągnij plik **pso-sphere.py** i uruchom go. Sprawdź wyniki i wykresy.

```
# Import modules
import numpy as np
from matplotlib import pyplot as plt
# Import PySwarms
import pyswarms as ps
from pyswarms.utils.functions import single_obj as fx
from pyswarms.utils.plotters import plot_cost_history

# Set-up hyperparameters
options = {'c1': 0.5, 'c2': 0.3, 'w':0.9}

# Call instance of PSO
optimizer = ps.single.GlobalBestPSO(n_particles=10, dimensions=2, options=options)

# Perform optimization
cost, pos = optimizer.optimize(fx.sphere, iters=200)

# Obtain cost history from optimizer instance
cost_history = optimizer.cost_history

# Plot!
plot_cost_history(cost_history)
plt.show()
```

PRZYKŁAD 2 Z WYKŁADU

Ściągnij plik **pso-animation.py**, który tworzy animację poruszającego się roju. Uruchom go. Sprawdź czy modyfikacja parametrów **c1**, **c2**, **w** wpływa na zachowanie roju.

Uwaga 1! Żeby zapisywać plik gif użyłem narzędzia ImageMagick do tworzenia gifów. Musiałem zainstalować na komputerze: <https://imagemagick.org/script/download.php#windows>

Uwaga 2! Niestety rysowanie i animacje w `pyswarms` nie są kompatybilne z najnowszym `matplotlibem` (>3.4). Zainstaluj starszą wersję `matplotliba`, np. wpisując w terminalu: `pip install matplotlib==3.3.4`

```
import pyswarms as ps
from pyswarms.utils.functions import single_obj as fx
from pyswarms.utils.plotters import plot_contour
from pyswarms.utils.formatters import Mesher

options = {'c1':0.5, 'c2':0.3, 'w':0.5}
optimizer = ps.single.GlobalBestPSO(n_particles=10, dimensions=2, options=options)
optimizer.optimize(fx.sphere, iters=50)
# tworzenie animacji
m = Mesher(func=fx.sphere)
animation = plot_contour(pos_history=optimizer.pos_history, mesher=m, mark=(0, 0))
animation.save('plot0.gif', writer='imagemagick', fps=10)
```

W rozwiązaniu z powyższego zadania były brane informacje ze stron:

- <https://pyswarms.readthedocs.io/en/latest/examples/tutorials/visualization.html>
- <https://www.pavlovsk.org/a-tutorial-on-optimizing-particle-swarm-in-python/>
- <https://pyswarms.readthedocs.io/en/latest/api/pyswarms.utils.plotters.html>

PROBLEM INŻYNIERYJNY: STOP METALI



Problem ten rozwiązywaliśmy algorytmem genetycznym. Spróbujmy teraz rozwiązać go algorytmem wykorzystującym inteligencję roju (PSO). Treść:

W pewnym zakładzie badawczym inżynierowie próbowali stworzyć bardzo trwały stop sześciu metali. Ilości wszystkich 6 metali w stopie oznaczone zostały symbolami x, y, z, u, v, w i są to liczby z przedziału $[0, 1]$. Okazało się, że wytrzymałość stopu określona jest przez funkcję:

$$\begin{aligned} \text{endurance}(x, y, z, v, u, w) \\ = e^{-2 \cdot (y - \sin(x))^2} + \sin(z \cdot u) + \cos(v \cdot w) \end{aligned}$$

```
def endurance(x, y, z, v, u, w):
    return math.exp(-2 * (y-math.sin(x))**2)+math.sin(z*u)+math.cos(v*w)
```

Obliczenie maksymalnej wytrzymałości (endurance) było dla inżynierów problematyczne. Poproszono Ciebie, eksperta od sztucznej inteligencji, o rozwiązanie problemu.

ZADANIE 1: PSO DLA STOPU METALI

Użyjemy paczki `pyswarms`, by uruchomić algorytm `Particle Swarm Optimizer` do wyszukania maksimum funkcji Endurance, z problemu inżynieryjnego.

- Rzucimy okiem na tutorial o podstawowej optymalizacji:

https://pyswarms.readthedocs.io/en/latest/examples/tutorials/basic_optimization.html#Optimizing-a-function

i wykorzystajmy ją na początek do minimalizacji funkcji Sphere.

```
import pyswarms as ps
from pyswarms.utils.functions import single_obj as fx

options = {'c1': 0.5, 'c2': 0.3, 'w':0.9}

optimizer = ps.single.GlobalBestPSO(n_particles=10, dimensions=2,
options=options)

optimizer.optimize(fx.sphere, iters=1000)
```

- b) Spróbujmy teraz dodać ograniczenia dla dziedziny (obszaru), w którym szukamy minimum. Trzeba ustalić ograniczenie górne i dolne dla wszystkich zmiennych, weźmy minimum 1, maksimum 2 – dla wszystkich zmiennych.

```
x_max = [2, 2]
x_min = [1, 1]
my_bounds = (x_min, x_max)
```

I oczywiście ograniczenia trzeba przekazać jako parametr do optimizera poprzez argument:

```
bounds=my_bounds
```

Przy takich ograniczeniach mój wynik był następujący:

```
best cost: 2.037488183282666, best pos: [1.00140036 1.01719492]
```

Czy masz podobny? 😊

- c) Zmieńmy teraz ten kod tak, by rozwiązywał problem inżynierijny. Po pierwsze trzeba zmienić limity: min 0, max 1 dla wszystkich sześciu zmiennych. Oczywiście dimensions trzeba ustawić na 6.

Uwaga, zamiast wypisywać długie wektory limitów ręcznie, można użyć numpy: np.zeros(6), np.ones(6).

- d) Teraz najciekawsza część, czyli zmiana funkcji fx.sphere na endurance. Pojawia się tu mały problem: funkcja endurance dostaje jeden punkt z 6 współrzędnymi i zwraca jedną wartość liczbową. Natomiast optimizer potrzebuje funkcji która dostaje cały rój punktów z sześcioma zmiennymi i zwraca tablicę wartości funkcji dla całego roju.

Musimy więc napisać funkcję do optymalizacji, która wywołuje funkcję endurance dla wszystkich częstek. Warto zajrzeć do tutoriala, gdzie to zrobiono na innym przykładzie:

https://pyswarms.readthedocs.io/en/latest/examples/usecases/train_neural_network.html#Constructing-a-custom-objective-function

W tutorialu chcemy optymalizować funkcję forward_prop, ale musimy dodatkowo napisać funkcję f dla całego roju, która przebiega pętlą po całym roju. Wykorzystaj ten pomysł i:

- Popraw funkcję endurance, by pobierała tablicę sześciu argumentów, a nie sześć oddzielnych argumentów
- Dopolacz funkcję f, która przebiegnie po całym roju uruchamiając dla każdej częstki endurance.
- Wrzuć f do optymalizatora.

Mój wynik był taki:

```
best cost: 1.1535708386757408, best pos: [0.03641204 0.95752916  
0.50499987 0.01549592 0.33011641 0.83217486]
```

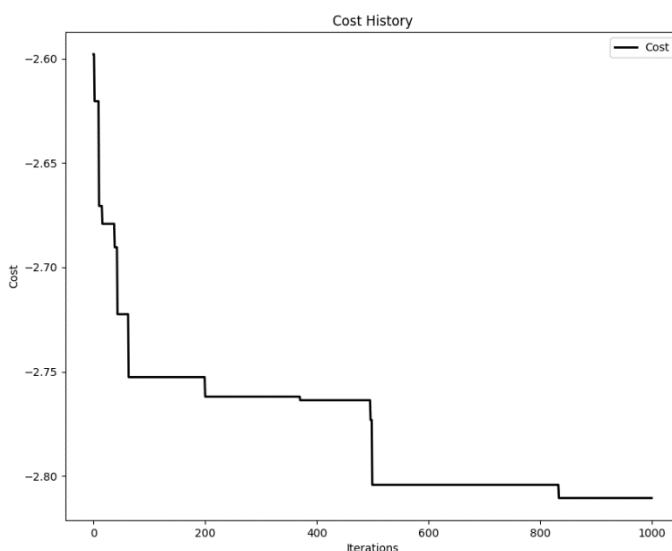
- e) Powyższy wynik jest oczywiście zły, bo funkcja szuka minimum zamiast maximum. Czy można to zmienić w prosty sposób w optimizerze? Ja niestety nie znalazłem takiej opcji 😞 Trzeba więc zrobić sztuczkę, taką jak przy algorytmie genetycznym i dopisać minus do funkcji endurance. Teraz wynik jest już dobry (podobny do tego z algorytmu genetycznego), ale oczywiście z minusem:
- ```
best cost: -2.833945350484618, best pos: [0.43027521 0.39007732
0.99986895 0.99134853 0.09893475 0.51286469]
```

- f) Wyświetlmy jeszcze wykres kosztu zgodnie ze wskazówkami z:  
<https://pyswarms.readthedocs.io/en/latest/api/pyswarms.utils.plotters.html>

Nie zapomnij zainportować potrzebnych rzeczy:

```
from pyswarms.utils import plot_cost_history
import matplotlib.pyplot as plt
```

Mój wynik:



## ■ ZADANIE 2: ACO DLA KOMIWOJAŻERA

Na wykładzie była prezentowana paczka `aco`, do rozwiązywania problemu komiwojażera za pomocą kolonii mrówek.

- Ściagnij i uruchom `aco-tsp.py`.
- Wygeneruj wersję TSP z większą liczbą wierzchołków i uruchom dla niej ponownie algorytm ACO.  
**COORDS = (**  
**(20, 52),**  
**(43, 50),**  
**(20, 84),**  
**(70, 65),**  
**(29, 90),**  
**(87, 83),**  
**(73, 23)**  
**)**

Tutaj było 7 punktów, zrób test dla 15 losowo wybranych punktów z przedziału od 0 do 100.

- c) Sprawdź czy modyfikacja parametrów poprawi skuteczność algorytmu. Wykonaj parę eksperymentów z różnymi zestawami danych (różne atrybuty w obiekcie AntColony).

```
AntColony(COORDS, ant_count=300, alpha=0.5, beta=1.2,
 pheromone_evaporation_rate=0.40, pheromone_constant=1000.0,
 iterations=300)
```

Twoje wnioski zapisz w postaci komentarza w notebooku lub komentarza w programie.

- d) Przeanalizuj jeszcze jeden przypadek: grid 5x5 np. ze współrzędnymi:

COORDS = (

(0, 0),

(10, 0),

(20, 0),

(30, 0),

(40, 0),

(0, 10),

(10, 10),

(20, 10),

(30, 10),

(40, 10),

(0, 20),

(10, 20),

(20, 20),

(30, 20),

(40, 20),

(0, 30),

(10, 30),

(20, 30),

(30, 30),

(40, 30),

(0, 40),

(10, 40),

(20, 40),

(30, 40),

(40, 40),

)

Jak Ci się wydaje, czy zalezienie najkrótszej drogi jest łatwe w takim przypadku? Jak to zrobić „po ludzku” i jaką długość będzie miała taka droga? Policz ręcznie/skryptem.

Sprawdź jak z gridem poradzi sobie algorytm mrówkowy.