WSI Labolatorium 2

Jonatan Kasperczak 341208

March 2025

1 Wprowadzenie

Celem projektu jest porównanie efektywności dwóch metod optymalizacji:

- algorytmu **ewolucyjnego**, opartego na losowej modyfikacji i selekcji populacji kandydatów,
- klasycznego **algorytmu gradientu prostego**, korzystającego z pochodnej funkcji celu.

Obie metody przetestowano na dwóch funkcjach testowych z zestawu **CEC2017**:

1. **F3** – Rotated High Conditioned Elliptic Function
2. **F19** – Composition Function złożona z funkcji Rosenbrocka, Ackley i Weierstrass

Funkcje te są znane z trudności optymalizacji, a ich minimum globalne wynosi $f(x^*) = 0$. W obu przypadkach zastosowano przestrzeń $\mathbb{R}^{10}$ i punkt początkowy losowany z rozkładu jednostajnego. Dla algorytmu ewolucyjnego przeprowadzono uśrednione testy, natomiast metoda gradientowa była oceniana osobno. Celem było zaobserwowanie zbieżności, stabilności oraz jakości uzyskanych wyników.

2 Algorytm ewolucyjny

Algorytm ewolucyjny działa na populacji P wektorów rzeczywistych $x \in \mathbb{R}^n$, które są kandydatami do optymalnego rozwiązania. W każdej iteracji obliczana jest wartość funkcji celu dla każdego osobnika, a następnie generowana jest nowa populacja na podstawie losowo wybranych rodziców oraz mutacji Gaussowskiej.

Zastosowano elitaryzm: najlepszy osobnik z dotychczasowych iteracji zostaje przeniesiony bez zmian do nowej populacji. Mutacja osłabia się w trakcie działania algorytmu (*mutation strength* maleje liniowo). Obliczenia są przerywane, jeśli przez 300 kolejnych iteracji nie uda się poprawić najlepszego wyniku.

2.1 Parametry algorytmu

- **pop_size** – liczba osobników w populacji (np. 500),
- **max_iter** – maksymalna liczba iteracji (np. 3000),
- **mutation_prob** – prawdopodobieństwo mutacji (np. 0.8),
- **mutation_strength** – siła mutacji początkowej (np. 6.0),
- **tol** – tolerancja (nieaktywna w tej wersji),
- **max_improve_c** – liczba iteracji bez poprawy po której następuje przerwanie działania (ustalona na 300).

Algorithm 1 Algorytm ewolucyjny z elitaryzmem i zanikającą mutacją

Require: Funkcja celu f; liczba osobników N; maksymalna liczba iteracji maxIter; parametry mutacji.

$P \leftarrow$ populacja losowa z rozkładu $[-100, 100]^n$
best $\leftarrow \arg\min_{x \in P} f(x)$
bestFit $\leftarrow f(\text{best})$
counter $\leftarrow 0$
for $i = 1$ to maxIter **do**
  Oblicz wartości $f(x)$ dla wszystkich $x \in P$
  $x_{\text{best}} \leftarrow \arg\min_{x \in P} f(x)$
  if $f(x_{\text{best}}) <$ bestFit **then**
    best $\leftarrow x_{\text{best}}$, bestFit $\leftarrow f(x_{\text{best}})$
    counter $\leftarrow 0$
  else
    counter $\leftarrow$ counter $+ 1$
  end if
  if counter $= 300$ **then**
    break
  end if
  Wylicz siłę mutacji: $\sigma = \text{mutationStrength} \cdot (1 - \frac{i}{\text{maxIter}})$
  $P_{\text{new}} \leftarrow [\,]$
  for $j = 1$ to $N - 1$ **do**
    Wybierz losowego osobnika $x \in P$
    if losowa liczba $<$ mutation_prob **then**
      Dodaj mutację $x \leftarrow x + \mathcal{N}(0, \sigma^2)$
    end if
    Dodaj x do P_{new}
  end for
  Dodaj najlepszy osobnik do P_{new}
  $P \leftarrow P_{\text{new}}$
end for
return najlepszy osobnik i jego wartość

3 Testowanie algorytmu ewolucyjnego

W celu zbadania zbieżności algorytmu ewolucyjnego przeprowadzono eksperymenty na funkcjach F3 oraz F19 z zestawu CEC2017. Każda funkcja była optymalizowana 10-krotnie dla wymiarowości $n = 10$, a wyniki uśredniane.

Funkcja	**Średnia liczba iteracji**
F3	1117.4
F19	587.6

Table 1: Średnia liczba iteracji do zatrzymania dla 10 uruchomień

Uruchomienie	F3	F19
Run 1	538.60	4636.10
Run 2	374.10	2738.60
Run 3	408.92	2283.40
Run 4	423.75	2908.80
Run 5	513.71	15420.00
Run 6	413.88	3643.80
Run 7	806.56	2736.30
Run 8	342.33	1958.80
Run 9	385.23	2366.60
Run 10	674.08	6150.70
Średnia	**488.32**	**4304.27**

Table 2: Końcowe wartości $f(x)$ po działaniu algorytmu ewolucyjnego (10 uruchomień)

3.1 Średnia zbieżność dla F3 i F19

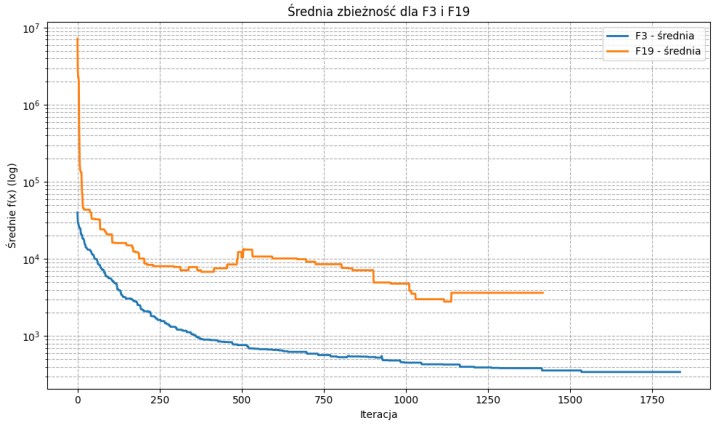

Figure 1: Średnia wartość $f(x)$ w skali logarytmicznej (10 uruchomień algorytmu ewolucyjnego)

Na rysunku 1 przedstawiono uśrednioną wartość $f(x)$ w kolejnych iteracjach. Widać wyraźnie, że dla funkcji F3 zbieżność jest szybsza i stabilniejsza – już po kilkuset iteracjach następuje spłaszczenie krzywej. Dla F19 wartości są początkowo wyższe i opadają wolniej, a moment stabilizacji jest mniej wyraźny. Oznacza to większą trudność w eksploracji przestrzeni i potrzebę większej liczby prób losowych.

3.2 Zbieżność funkcji F3 – wszystkie uruchomienia

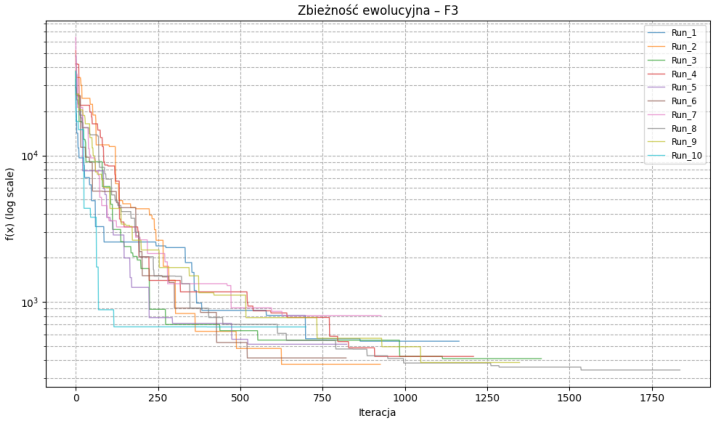

Figure 2: Zbieżność funkcji F3 (10 uruchomień, skala logarytmiczna)

Na wykresie 2 widzimy, że przebiegi 10 uruchomień są bardzo zbliżone – każda krzywa opada szybko w kierunku minimum, co świadczy o niskiej wariancji rozwiązania. Algorytm ewolucyjny w tym przypadku działa stabilnie i powtarzalnie.

3.3 Zbieżność funkcji F19 – wszystkie uruchomienia

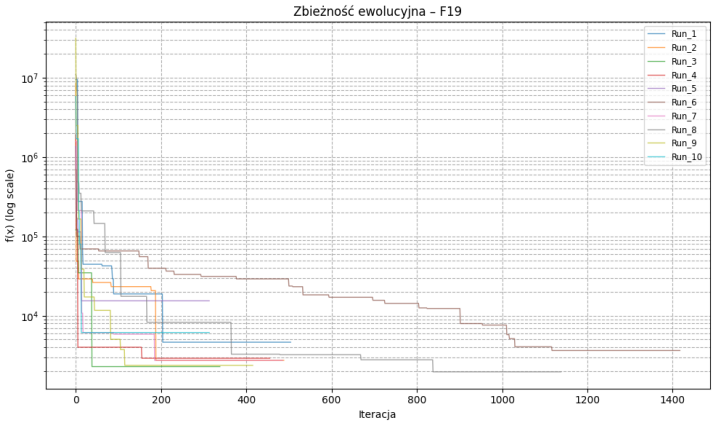

Figure 3: Zbieżność funkcji F19 (10 uruchomień, skala logarytmiczna)

Rysunek 3 pokazuje znacznie większe rozproszenie wartości końcowych. Część uruchomień zakończyła się z bardzo niskim $f(x)$, inne „utknęły" wyżej – co potwierdza obecność licznych minimów lokalnych i trudność funkcji F19. Jest to typowe dla funkcji kompozycyjnych, w których różne składniki (np. Weierstrass) generują wielomodalne krajobrazy.

4 Porównanie z metodą gradientu prostego

W celu pełnego zrozumienia skuteczności zaprojektowanego algorytmu ewolucyjnego, przeprowadzono także analogiczne testy z wykorzystaniem metody gradientu prostego.

4.1 Funkcja F3 – Gradient prosty

Funkcja F3 to *Rotated High Conditioned Elliptic Function*, czyli eliptyczna funkcja o bardzo dużej kondycji (kondycja macierzy hessjan to rzędu 10^6). Gradient tej funkcji w różnych kierunkach ma dramatycznie różne skale, co sprawia, że stały krok α nie pozwala na skuteczną aktualizację współrzędnych jednocześnie.

Wszystkie uruchomienia gradientu zakończyły się bez sukcesu, z bardzo dużymi końcowymi wartościami funkcji celu (rzędu 10^{10}), pomimo pełnego wykonania 20000 iteracji. Wektor rozwiązania nie zbiegał do minimum, lecz często „wahał się" w rejonach dalekich od optimum globalnego.

4.2 Funkcja F19 – Gradient prosty

Funkcja F19 jest jedną z najbardziej złożonych funkcji benchmarku CEC2017. Stanowi kompozycję funkcji Rosenbrocka, Ackley i Weierstrassa, czyli łączy trudności wynikające z:

- licznych minimów lokalnych,
- bardzo płaskich fragmentów przestrzeni,
- silnie niestabilnych i oscylujących gradientów.

Dla funkcji F19 jedynie przy ekstremalnie małych wartościach kroku uczenia ($\alpha = 10^{-5}$) udawało się zbiec do względnie sensownych wyników (np. $f(x) \approx 500$), ale nawet w takich przypadkach nie był spełniony warunek stopu i algorytm wykonywał pełne 20000 iteracji.

5 Wykresy dla metody gradientu prostego

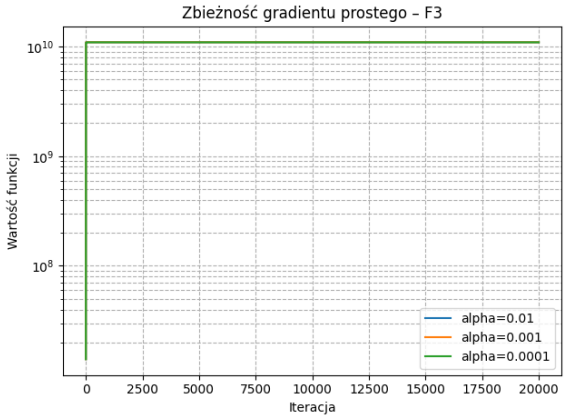

Figure 4: Gradient – przebieg wartości $f(x)$ dla F3

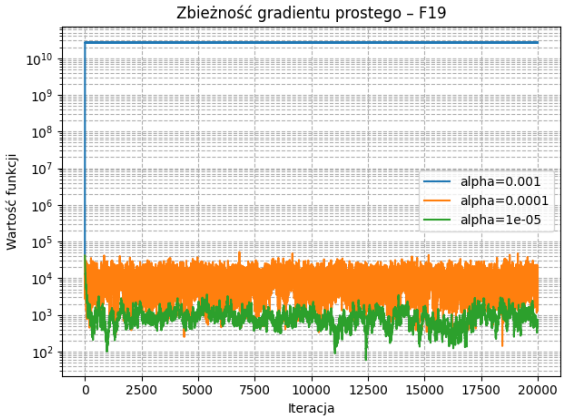

Figure 5: Gradient – przebieg wartości $f(x)$ dla F19

5.1 Wnioski wynikające z porównania metod

- Metoda gradientu prostego z jednym, stałym krokiem α okazuje się nieefektywna na złożonych funkcjach testowych z benchmarku CEC2017.
- Dla funkcji F3 różnice w skali gradientu pomiędzy współrzędnymi powodują, że krok dobrany odpowiednio dla jednej współrzędnej jest zbyt duży lub zbyt mały dla innych.
- Dla funkcji F19 obecność oscylujących składników i wielu minimów lokalnych prowadzi do bardzo niestabilnej zbieżności.
- Algorytm ewolucyjny, mimo swojej prostoty i braku wykorzystania pochodnej, poradził sobie zdecydowanie lepiej — co jest szczególnie widoczne w przypadku F19, gdzie wartości $f(x)$ po 1000 iteracjach były nawet 10-krotnie niższe niż w metodzie gradientowej po 20000 iteracjach.
- Dla trudnych funkcji nieliniowych o niegładkich lub nieciągłych gradientach algorytmy populacyjne okazują się bardziej odporne i skuteczniejsze.

6 Podsumowanie

Na podstawie przeprowadzonych eksperymentów można sformułować następujące wnioski:

- **Algorytm ewolucyjny** poradził sobie bardzo dobrze z optymalizacją funkcji F3 i F19 – szczególnie przy zastosowaniu elitaryzmu oraz osłabiającej się mutacji.
- Dla F3 wyniki były stabilne i powtarzalne, z niską wariancją i szybką zbieżnością. Średnia liczba iteracji do zatrzymania to około 1100.
- Dla F19 zbieżność była trudniejsza – z powodu lokalnych minimów i bardziej złożonego krajobrazu funkcji. Algorytm jednak osiągał dobre wyniki w mniej niż 600 iteracjach średnio.
- **Metoda gradientu prostego** nie była w stanie skutecznie zoptymalizować żadnej z funkcji przy zastosowaniu stałego kroku. Dla F3 – rozbieżność przez różną skalę gradientu, dla F19 – niestabilność i utknięcie w minimach lokalnych.
- Optymalizacja funkcji o złej kondycji lub złożonej strukturze wymaga metod nieliniowych i adaptacyjnych. Algorytmy populacyjne są tu znacznie bardziej odporne i efektywne, nawet bez informacji o pochodnej.

Projekt potwierdził zasadność stosowania algorytmów ewolucyjnych w zadaniach, gdzie klasyczne metody gradientowe zawodzą. W zastosowaniach praktycznych, gdy funkcje są trudne do analitycznego opisu lub mają nieregularne krajobrazy optymalizacji, metody populacyjne stanowią bardziej uniwersalne i odporne podejście.