Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji

Sztuczna Inteligencja

Sprawozdanie z laboratorium

Autor

Aleksandra Walczybok

nr albumu: **272454**

kierunek: **Inżynieria systemów**

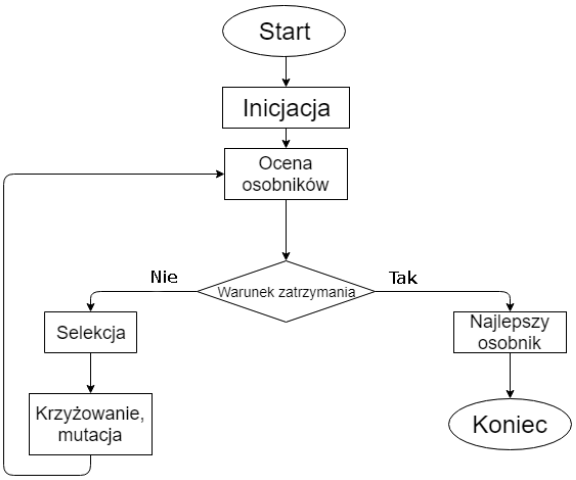
26 styczeń 2024

1 Wstęp – sformułowanie problemu

Celem projektu było opracowanie algorytmu genetycznego, który rozwiązywałby problem minimalizacji funkcji. Funkcja ma być opisana na przedziale od (-8,8).

Algorytm genetyczny jest techniką optymalizacji inspirowaną procesami ewolucji naturalnej. Należy do grupy algorytmów heurystycznych, które znajdują zastosowanie w rozwiązywaniu problemów o dużej złożoności obliczeniowej, gdzie tradycyjne metody optymalizacji mogą być zbyt kosztowne lub nieskuteczne. W procesie tym, rozwiązania potencjalne (tzw. osobniki) są reprezentowane jako chromosomy, które są poddawane operacjom takim jak selekcja, krzyżowanie czy mutacja. Celem algorytmu genetycznego jest stopniowe poprawianie jakości rozwiązań poprzez symulację procesu ewolucji, aż do osiągnięcia optymalnego lub zadowalającego wyniku.

2 Opis rozwiązania



Rysunek 1: Schemat algorytmu

Opis poszczególnych kroków rozwiązania. Użyte parametry były ustalone na domyślne/najczęściej używane. W drugiej części sprawozdania przeprowadzono doświadczenia z eksperymentowaniem z różnymi parametrami, tak aby wybrać najlepsze.

- Pierwszym krokiem była **inicjalizacja pierwszego pokolenia**. Ustalono liczbę populacji **100** oraz długość chromosomu na wartość **16**. Korzystając z metody random.randint() o zakresie 0-1 tworzono chromosomy wszystkich 100 osobników.
- Następnym istotnym krokiem była **ocena** osobników. Ocenę dokunuje się na podstawie tzw. **funkcji przystosowania**, czyli funkcji, którą chcemy minimalizować (w projekcie jest ona podawana przez użytkownika). Ważne jest to, iż ocenę oblicza się w dziedzinie liczb niebinarnych, a więc konieczne było tutaj dodanie odpowiedniej metody dekodującej wartości binarne (osobników) na wartości rzeczywiste (x), z uwzględnieniem przedziału podanego w zadaniu.

  Zakładając, że mamy chromosom binarny o długości n, który reprezentuje liczbę rzeczywistą w przedziale $[-8, 8]$, proces dekodowania można opisać za pomocą następującego wzoru:

$$\text{real\_value} = B_{\text{low}} + \left(\frac{\sum_{i=1}^{n} b_i \cdot 2^{n-i}}{2^n - 1} \right) \cdot (B_{\text{high}} - B_{\text{low}})$$

Gdzie:

- b_i to i-ty bit chromosomu (gdzie $i = 1, 2, \ldots, n$),
- 2^{n-i} to waga i-tego bitu w systemie binarnym,
- $2^n - 1$ to maksymalna wartość, jaką można przedstawić przy n-bitach,
- B_{low} i B_{high} to dolna i górna granica przedziału, w którym szukamy wartości rzeczywistej.

- **Selekcja** w algorytmie genetycznym to proces wybierania osobników z populacji, którzy mają największe szanse na przekazanie swojego materiału genetycznego do kolejnego pokolenia. Celem selekcji jest promowanie najlepszych rozwiązań, co prowadzi do poprawy jakości populacji w kolejnych generacjach.

  W projekcie zaimplementowano 3 możliwe metody selekcji osobników, które w dalszym etapie były analizowane pod względem lepszego i szybszego działania.

  - **Metoda ruletkowa** - polega na tym, że prawdopodobieństwo wyboru osobnika jest proporcjonalne do jego wartości przystosowania (*fitness*). Osobniki o wyższym przystosowaniu mają większą szansę na wybór. Proces ten można porównać do losowania sektora na kole ruletki.

    Problemem w projekcie, było to, że należało minimalizować funckję, a więc na początku w tej metodzie dostosowano funkcje przystosowania w poniższy sposób:

    $$\text{adjusted\_fitness} = \frac{1}{1 + \text{fitness} - \min(\text{fitness})}$$

    Należy odjąć wartość minimalną w mianowniku, po to, aby uniknąć wartości ujemnych.
    Prawdopodbieństwo wylosowanie osobnika i:

    $$p(i) = \frac{f(i)}{\sum_{j=1}^{n} f(j)}$$

    Gdzie:

    * $f(i)$ – wartość funkcji przystosowania dla osobnika i,
    * $p(i)$ – prawdopodobieństwo wyboru osobnika i,
    * n – liczba osobników w populacji.

  - **Metoda turniejowa** - losowo wybierana jest podgrupa (turniej) k osobników z populacji. Następnie najlepszy osobnik z tej grupy zostaje wybrany na podstawie wartości funkcji przystosowania (*fitness*). W projekcie rozmiar grupy wynosił k = 2.

    $$i = \arg\max_{j \in T} f(j)$$

    Gdzie:

    * T – zbiór losowo wybranych k osobników (turniej),

  - $f(j)$ – wartość funkcji przystosowania dla osobnika j,
  - i – indeks najlepszego osobnika w turnieju.
- **Metoda rankingowa** - w tej metodzie osobniki są sortowane według wartości funkcji przystosowania (u nas na podstawie funkcji przekształconej jak w przykładzie z metodą ruletkową), a prawdopodobieństwo wyboru zależy od ich pozycji w rankingu, a nie od rzeczywistej wartości funkcji przystosowania.

$$p(i) = \frac{q^{r(i)}}{\sum_{j=1}^{n} q^{r(j)}}$$

Gdzie:

  - $r(i)$ – pozycja osobnika i w rankingu (1 dla najlepszego),
  - q – parametr określający, jak szybko maleje prawdopodobieństwo wyboru w zależności od pozycji w rankingu (przykładowo $q > 1$),
  - n – liczba osobników w populacji.

- **Krzyżowanie** ma na celu generowanie nowych osobników (*potomków*) na podstawie istniejących osobników (*rodziców*). Proces ten polega na wymianie informacji genetycznej pomiędzy rodzicami, co umożliwia eksplorację przestrzeni rozwiązań. W projekcie badano dwie metody krzyżowania:

  - **Krzyżowanie jednopunktowe** W krzyżowaniu jednopunktowym wybierany jest losowy punkt przecięcia k w chromosomie rodzica. Następnie fragmenty chromosomów od tego punktu są zamieniane między rodzicami, co prowadzi do powstania dwóch nowych potomków.
  - **Krzyżowanie jednolite** Krzyżowanie z maską polega na użyciu specjalnej maski binarnej M, która określa, które geny są przekazywane od którego rodzica. Maska jest budowana losowo. Wartość M_i w masce M decyduje, czy gen na i-tej pozycji pochodzi od rodzica 1, czy od rodzica 2.

  Krzyżowanie zachodzi dla losowo wybranych (ze zbioru po selekcji) dwóch osobników, jeśli wylosowana wartość dla tej pary będzie mniejsza niż prawdopodobieństwo krzyżowania. W projekcie ustalone na początku było jako **0.7**.

- **Mutacja** wprowadza losowe zmiany w chromosomie osobnika. Jej celem jest zwiększenie różnorodności w populacji oraz umożliwienie eksploracji nowych obszarów przestrzeni rozwiązań, co zapobiega przedwczesnej konwergencji algorytmu genetycznego. Dla każdego genu w chromosomie losowana jest wartość z przedziału [0,1],jeśli jest ona mniejsza niż prawdopodobieństwo mutacji, to mutacja zachodzi. p_{mut} ustalone zostało na **0.01**.

- **Warunek stopu** - dodano również warunek stopu, gdzie jeśli przez 10 generacji nie widać poprawy w najlepszym rozwiązaniu, to przerywamy algorytm, ponieważ istnieje ryzyko stagnacji.

3 Przykład działania

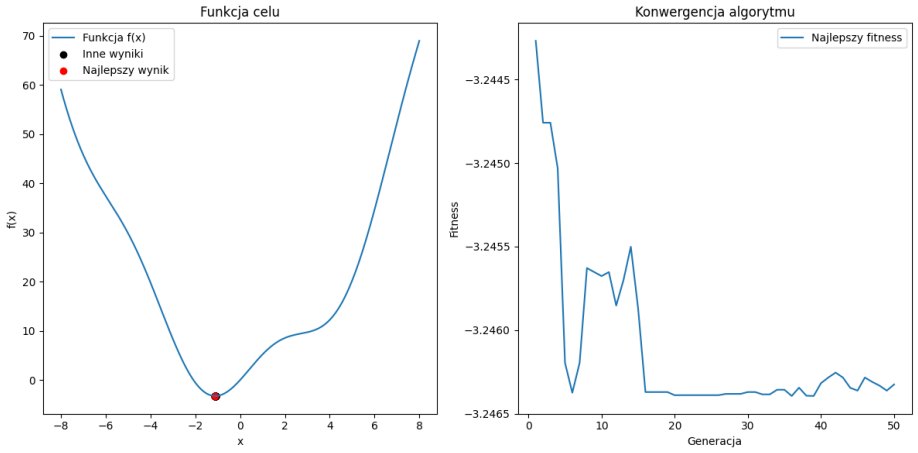
Algorytm był testowany na przykładowej funkcji jednoargumentowej:

$$f(x) = x^2 + 5\sin(x)$$

Ustalone parametry początkowe:
POPULATION SIZE = 100
CHROMOSOME LENGTH = 16
MUTATION RATE = 0.01
CROSSOVER RATE = 0.7
GENERATIONS = 50
STALL GENERATIONS=10
BOUND LOW, BOUND HIGH = -8, 8

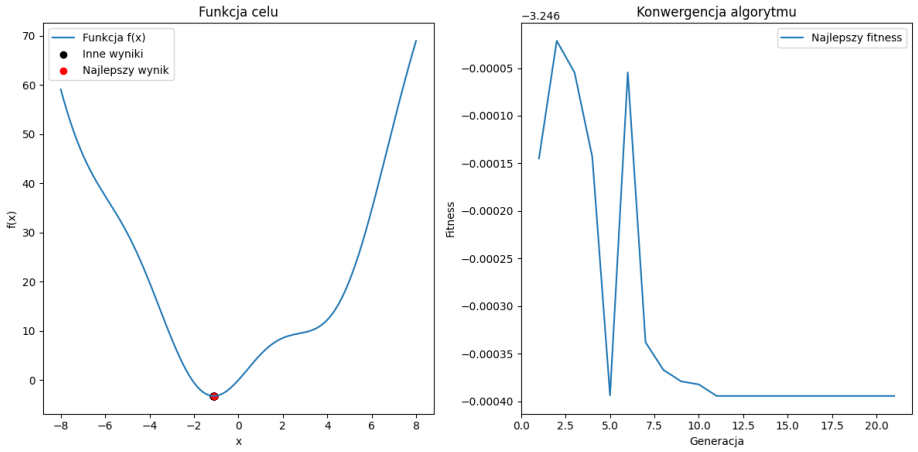
W każdej generacji zapisywano najlepsze rozwiązanie oraz najlepszą wartość funkcji przystosowania. Dokonywano selekcji, krzyżowania i mutacji, tworząc nową populację do następnej generacji.

Wyniki dla selekcji ruletkowej i krzyżowania jednopunktowego:



Rysunek 2: Wykres działania algorytmu 1

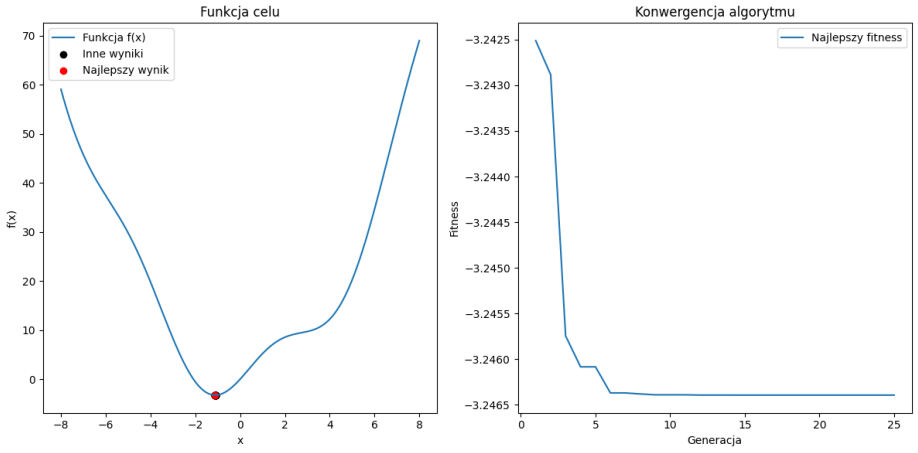
Wyniki dla selekcji turniejowej i krzyżowania jednopunktowego:



Rysunek 3: Wykres działania algorytmu 2

Tutaj "zastosował się" warunek stopu, ponieważ od 11 do 21 generacji nie było poprawy w wartości fitness.

Wyniki dla selekcji rankingowej i krzyżowania jednopunktowego:



Rysunek 4: Wykres działania algorytmu 3

Tutaj również warunek stopu zadziałał w pokoleniu 24.

4 Eksperymenty

Testowano działanie algorytmu, wykorzystując różne wartości parametrów.

$$\text{population\_sizes} = \{10, 20, 50, 100\}$$

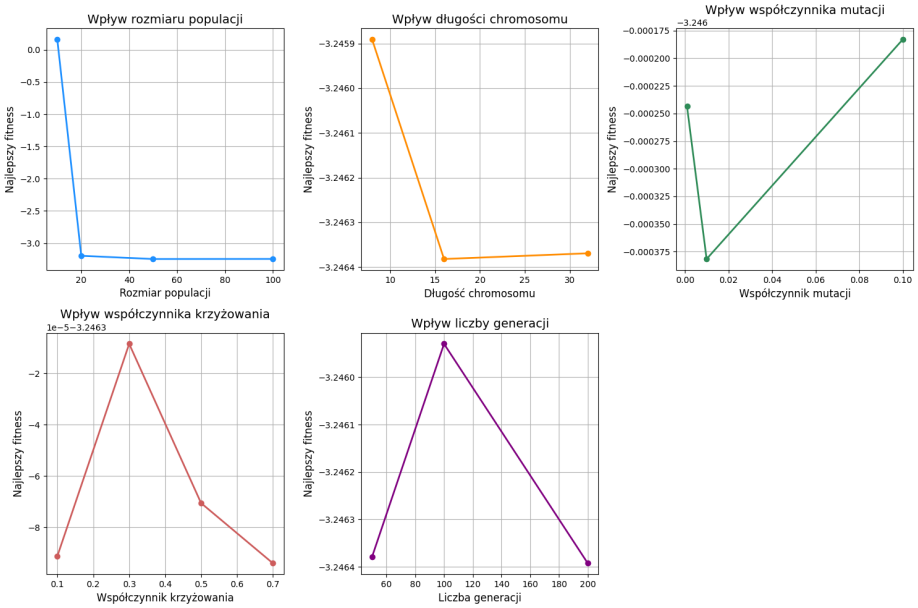
$$\text{chromosome\_lengths} = \{8, 16, 32\}$$

$$\text{mutation\_rates} = \{0.001, 0.01, 0.1\}$$

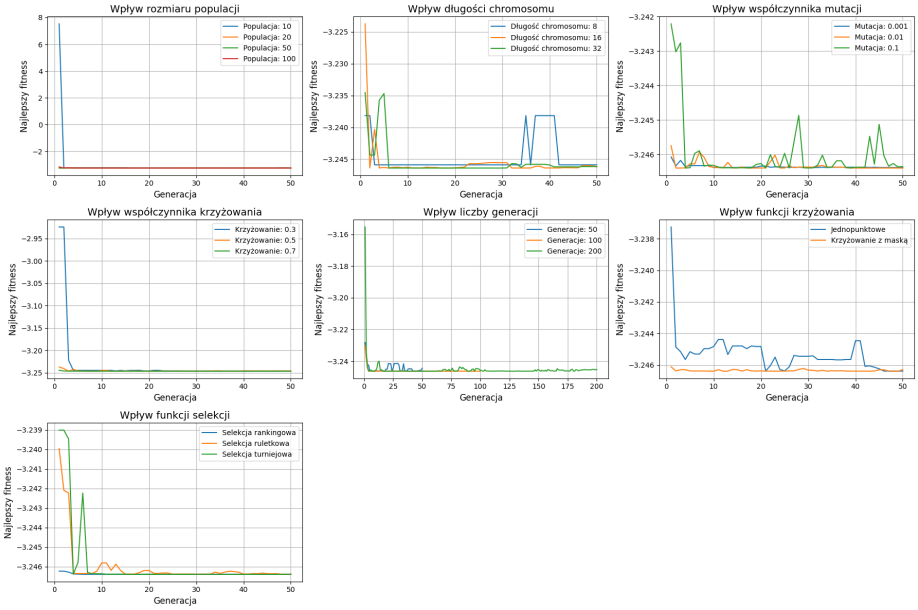
$$\text{crossover\_rates} = \{0.3, 0.5, 0.7\}$$

$$\text{selection\_funcs} = \{\text{ranking\_selection}, \text{roulette\_wheel\_selection}, \text{tournament\_selection}\}$$

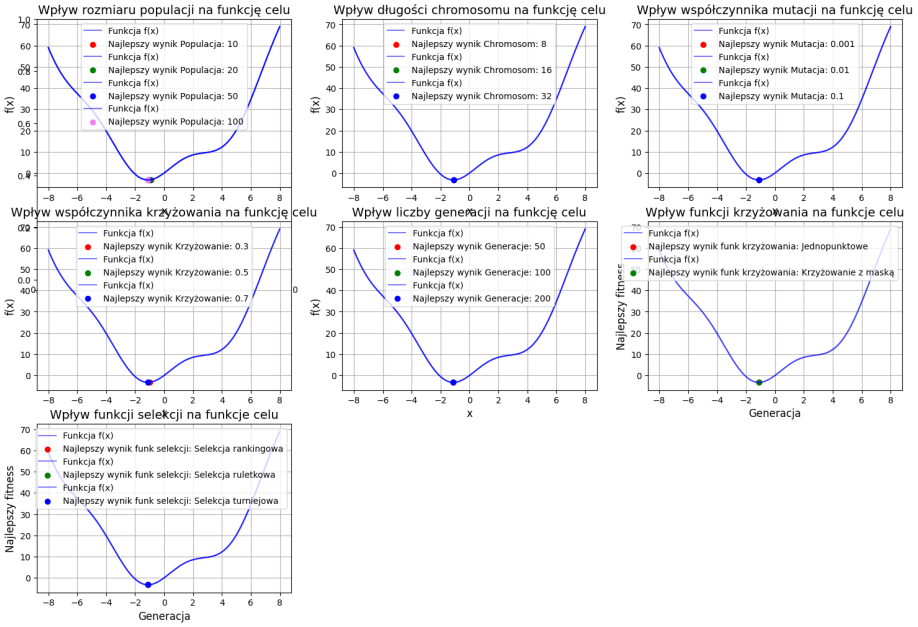
$$\text{crossover\_funcs} = \{\text{crossover}, \text{crossover\_with\_mask}\}$$



Rysunek 5: Eksperyment 1



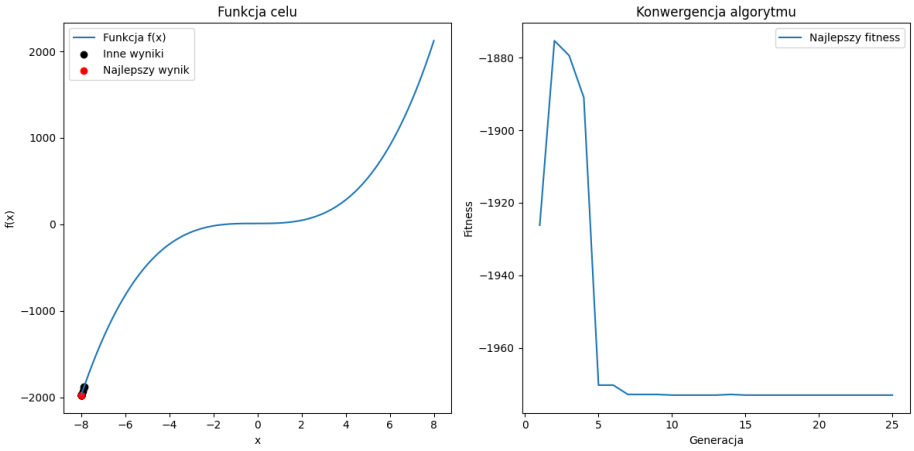
Rysunek 6: Eksperyment 2



Rysunek 7: Eksperyment 3

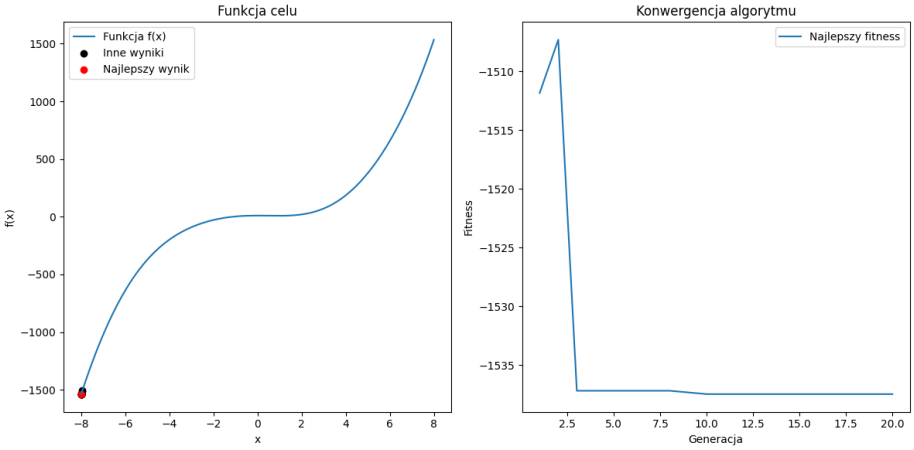
Przykłady działania dla innych funkcji:

$$f(x) = 4x^3 + x^2 + 11$$



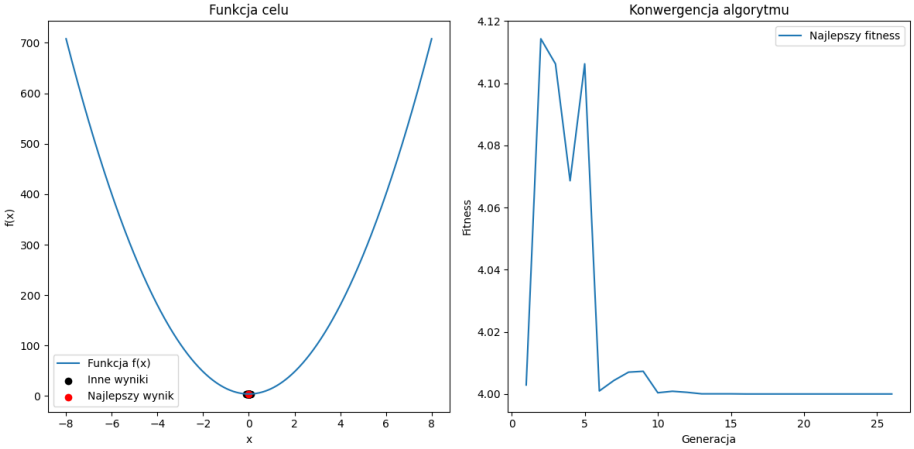
Rysunek 8: Przykład 1

$$f(x) = 3x^3 + 10\cos(x)$$



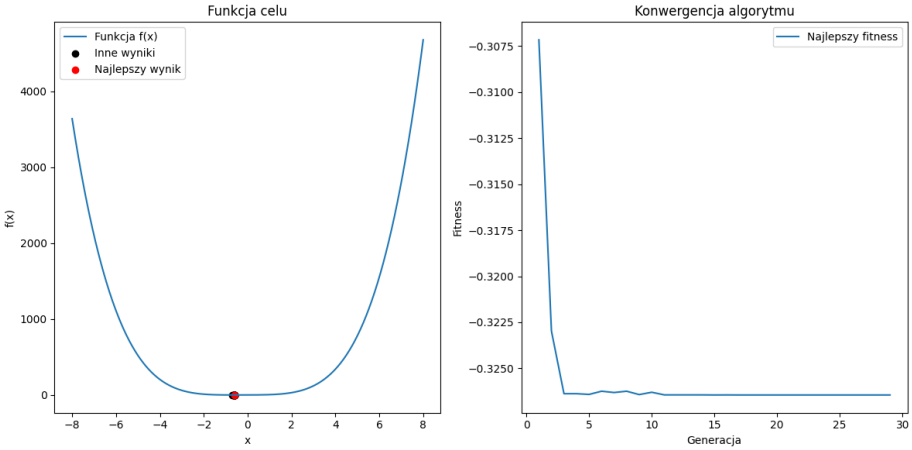
Rysunek 9: Przykład 2

$$f(x) = 11x^2 + 4$$



Rysunek 10: Przykład 3

$$f(x) = x^4 + x^3 + x^2 + x$$



Rysunek 11: Przykład 4

5 Wnioski z przykładu działania oraz eksperymentów

- **Populacja** - Zwiększenie rozmiaru populacji poprawia wyniki algorytmu. Z eksperymentu przedstawionego na rysunku 5 wynika, że populacja składająca się z 10 osobników była zdecydowanie gorsza niż populacja licząca 20 osobników. Co ciekawe, różnice między populacjami 20 a 100 osobników były minimalne.

- **Długość chromosomu** - Najlepsze wyniki uzyskano dla długości chromosomu równej 16. Długość 8 okazała się zdecydowanie za mała, podczas gdy długość 32 działała podobnie jak 16. Więcej szczegółów znajduje się na wykresach 6.
- **Współczynnik mutacji** - Najlepsze rozwiązania uzyskano dla wartości współczynnika mutacji $p_m = 0.01$. Wartość $p_m = 0.1$ okazała się za wysoka, powodując zbyt duże zmiany w genotypach osobników, co prowadziło do gorszych wyników.
- **Współczynnik krzyżowania** - Najlepsze wyniki uzyskano dla wartości $p_k = 0.1$, $p_k = 0.5$ i $p_k = 0.7$. Najgorsze rezultaty pojawiły się przy $p_k = 0.3$.
- **Liczba generacji** - W eksperymencie przedstawionym na rysunku 5 okazało się, że dla populacji liczącej 100 osobników wyniki były najgorsze. Jednak zastosowany warunek stopu pozwalał na skuteczne monitorowanie zmian wartości fitness, co umożliwiało dostosowanie liczby generacji do zachodzących zmian.
- **Funkcja krzyżowania** - Z analizy wykresu 6 wynika, że krzyżowanie z maską działa lepiej niż krzyżowanie jednopunktowe. Po około 42 generacjach obie metody osiągają zbliżone wyniki.
- **Funkcja selekcji** - Dla małej liczby generacji selekcja ruletkowa i turniejowa nie dawały najlepszych rezultatów (6). Po 15 generacjach wszystkie metody selekcji zaczęły działać podobnie. Podczas testowania algorytmu (w sekcji przykład działania) można zuważyć, że zarówno selekcja turniejowa i rankingowa szybciej doszły do minimum niż selekcja ruletkowa.

Jak widać na wykresach 7, mimo różnych ustawionych parametrów, algorytm genetyczny radził sobie bardzo dobrze i za każdym razem zbliżał się do minimum funkcji.

Warto również zauważyć, że ocena parametrów w algorytmach genetycznych jest trudna, ponieważ wyniki są silnie zależne od losowości. Z tego względu wyniki uzyskane w kilku próbach mogą nie wystarczyć do rzetelnej oceny skuteczności wszystkich parametrów.

A Dodatek

Kody źródłowe umieszczone zostały w repozytorium GitHub:
`https://github.com/aleksandra0014/genetic_algorithm`.