

Politechnika Krakowska

Wydział Informatyki i Telekomunikacji

Studia Stacjonarne

Sprawozdanie z projektu 1:

**Implementacja klasycznego algorytmu genetycznego**

Wykonali:

Dawid Wilgucki

Magdalena Skwarczek

Mateusz Stec

Mikołaj Wnuk

**1. Wprowadzenie**

Projekt polega na implementacji klasycznego algorytmu genetycznego w języku Python. Celem jest rozwiązanie problemu optymalizacji funkcji wielu zmiennych poprzez maksymalizację oraz minimalizację wartości funkcji testowych.

**2. Technologie wykorzystane w projekcie**

* **Język programowania:** Python
* **Biblioteki:** numpy, matplotlib, tkinter, sqlite3
* **Środowisko programistyczne:** VS Code
* **System kontroli wersji:** Git (GitHub)

**3. Wymagania środowiska**

Aby uruchomić aplikację, należy mieć zainstalowane:

* Python w wersji 3.8+
* Poniższe biblioteki:

pip install numpy matplotlib tkinter sqlite3

* System operacyjny: Windows/Linux/MacOS

**4. Opis implementacji algorytmu genetycznego**

Algorytm składa się z następujących etapów:

1. Inicjalizacja populacji chromosomów w reprezentacji binarnej.
2. Ocena populacji na podstawie funkcji celu.
3. Selekcja najlepszych osobników metodą turniejową lub ruletki.
4. Krzyżowanie: jednopunktowe, dwupunktowe, jednorodne, ziarniste.
5. Mutacja: bit flip, brzegowa, dwupunktowa.
6. Inwersja z określonym prawdopodobieństwem.
7. Strategia elitarna zapewniająca przejście najlepszych osobników do następnej populacji.

**5. Wybrane funkcje, które będą testowane wraz z rysunkiem funkcji oraz wartościami optimum wraz z argumentami dla których to optimum jest osiągnięte. Jeżeli w sprawozdaniu testują Państwo funkcje 10, 20, 30 zmiennych to proszę zamieścić informacje o optimach i argumentach dla których ono zachodzi dla 10, 20 i 30 zmiennych, tak by można było łatwo porównać wyniki które Państwo osiągnęli ewolucyjną optymalizacją z rzeczywistami wartościami optimum**

Do testów wybrano dwie funkcje:

1. **Łatwa funkcja testowa:** Rastrigin
   * Wzór funkcji:
   * Rzeczywiste optimum:
2. **Trudna funkcja testowa:** Hyperellipsoid
   * Wzór funkcji:
   * Rzeczywiste optimum:

Testy przeprowadzono dla 10, 20 i 30 zmiennych.

**6. Wyniki eksperymentów**

**6.1 Wykresy zależności wartości funkcji celu od kolejnej iteracji**

**A graph with numbers and a line

AI-generated content may be incorrect.**

**6.2 Średniej wartości funkcji celu oraz odchylenia standardowego w kolejnej iteracji**

**A graph with red lines

AI-generated content may be incorrect.**

**6.3 Porównanie osiągniętych wyników przy różnych konfiguracjach algorytmu + porównanie czasu obliczeń. Każda konfiguracja którą uruchomicie powinna być powtórzona przynajmniej 10 razy – tak by wyniki zostały odpowiednio uśrednione i wolne od efektu losowości. W sprawozdaniu proszę zamieścić średnie wyników z 10 uruchomień, najlepszy wynik oraz najgorszy wynik. Wykresy z podpunktów d oraz e proszę zamieścić tylko z najlepszego uruchomienia.**

**A close-up of a graph

AI-generated content may be incorrect.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Konfiguracja** | **Średnia wartość** | **Najlepsza wartość** | **Najgorsza wartość** | **Średni czas [s]** | **Odchylenie czasu [s]** |
| Konfiguracja bazowa | 8.00 | 7.00 | 9.00 | 0.59 | 0.11 |
| Wysoka mutacja | 8.10 | 7.00 | 9.00 | 0.44 | 0.02 |
| Krzyżowanie dwupunktowe | 8.40 | 7.00 | 10.00 | 0.46 | 0.05 |
| Koło ruletki + niska mutacja | 8.00 | 7.00 | 9.00 | 0.34 | 0.04 |

1. Najlepsze wyniki osiągnęły konfiguracje:
   * Koło ruletki + niska mutacja (średnia: 8.00, najszybsza: 0.34s).
   * Konfiguracja bazowa (średnia: 8.00, ale wolniejsza: 0.59s).
2. Wysoka mutacja miała niższe odchylenie czasu (0.02s), co wskazuje na stabilność.
3. Krzyżowanie dwupunktowe dało najgorszy maksymalny wynik (10.00), ale najlepszą zbieżność (patrz wykres).

**6.4 W podsumowaniu zamieść tabelkę zawierającą nazwę optymalizowanej funkcji, liczbę zmiennych, rzeczywista wartość optimum, wartość optimum którą udało się uzyskać z wykorzystaniem algorytmu genetycznego oraz błąd – czyli różnicę rzeczywistej wartości z osiągniętą.**

**7. Podsumowanie**

Algorytm genetyczny pozwolił znaleźć optymalne lub bliskie optymalnym wartości funkcji testowych. Wyniki wykazały, że większa liczba generacji oraz większa populacja zwiększają dokładność wyników, ale kosztem czasu obliczeń.

**8. Załączniki**

1. **Kod źródłowy** – zawarty w archiwum .zip.
2. **Nagranie wideo** – demonstracja działania aplikacji.
3. **Pliki wynikowe** – zapisane w bazie SQLite (result.db) oraz plikach CSV.