Rozproszony algorytm genetyczny do wyszukiwania globalnej ekstremy: MPI Programowanie równoległe i rozproszone

Wykonanie:
1. Ivan Prakapets
2. Piotr Jeleniewicz

Sprawdzająca:
dr inż. Zuzanna Krawczyk

Warszawa, 2020

Spis treści

1 Opis uruchomienia i wywołania programu

Aby uruchomić program i otworzyć GUI trzeba wpisać komendę: `python3 main.py` lub poprzez IDE np. PyCharm.

Po uruchomieniu programu i otworzeniu ustawienia-GUI z kilkoma polami:

- f(x, y) input string
- chromosomes
- generations number input string liczba pokoleń
- optimizer function (min or max) funkcja optymalizatora (min lub max)

Oraz pola z ptaszkami (funkcjonalne):

- mutation - dodanie mutacji do każdej części (4 osobników) nowego pokolenia
- show statistics - pokazuje dokładne informacje o każdym pokoleniu
- save all files - zapisuje do folderu results statystyki wyników z animacją w .gif oraz ga-statystyki + do folderu generations dane csv z kolumnami x, y, f (x, y)
- show plot - pokazuje animację ewaluacji

2 Dokładny opis realizacji problemu

Opis programu

Program w przybliżeniu oblicza minimum lub maksimum funkcji o dwóch parametrach na podstawie algorytmu genetycznego. Każdy osobnik ma jedną `chromosom (x, y)` i ze względu na to, że osobnik i chromosom mają to samo znaczenie, pokolenie składa się z części, gdzie jedna część to 4 osobniki, w

konsekwencji liczba chromosomów jest wielokrotnością 4.

Zasada crossover to:

(x_better, y_best), (y_better, y_best), (x_best, y_better), (x_best, y_good)

gdzie (x_best, y_best), (x_better, y_better), (x_good, y_good) są zaznaczonymi individs.

Funkcji genetyczne i rozwiązanie algorytmiczne.

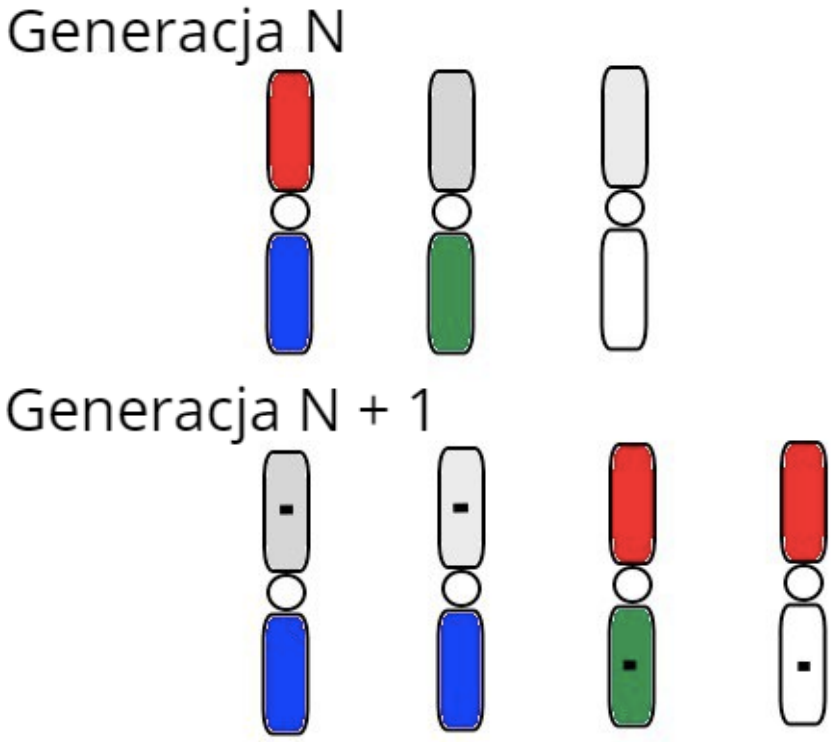
W tej chwili w programie realizowany jest genetyczne algorytm, który jest parametryzowany przez Dane wejściowe za pomocą GUI.

Pojedynczy osobnik przenosi w każdym swoim genie informacje o odpowiedniej współrzędnej X lub Y. Populacja jest określana przez wiele osobników, ale populacja jest podzielona na 4 osobniki. To rozwiązanie jest spowodowane próbą uniknięcia zbieżności do lokalnego optimum, ponieważ jest to zadanie o znalezieniu globalnego ekstremum. Taki podział, w wielu przypadkach nie pozwala zdominować jednego genotypu w całej populacji, ale wręcz przeciwnie, daje **ewolucję** większą dynamikę. Dla każdej takiej części populacji stosuje się następujący algorytm:

1. Selekcja jest podobna do metody rankingowej. Wybiera się 3 osobniki o najlepszych wskaźnikach funkcji fitness (tzn. dokonuje się sortowania osobników w kolejności rosnącej/malejącej określonej przez użytkownika funkcji),
2. Następnie stosuje się funkcję krzyżowania w taki sposób, że nowa generacja (dokładniej nowy segment populacji 4 osobników) otrzymuje 2 pary niezmutowanych genów od osobnika o lepszym odczycie funkcji fitness i po parze zmutowanych genów od pozostałych dwóch osobni-

ków.

Zasada selekcji, krzyżowania i mutacji wyraźnie wygląda tak (w generacji N chromosomów osobników są już posortowane w odpowiedniej kolejności, a mały czarny kwadrat oznacza mutację):



3 Testy programu pokazujące np. uzyskane przyspieszenie

Funkcja 1

$$f(x, y) = sin(x) + cos(y)$$

Funkcja 2

$$\frac{4\sqrt{x} - 5y}{3x^2 + 2y^2 - 2x + 1}$$

Funkcja 3

$$f(x, y) = sinx$$

Funkcja 4

$$f(x, y) = 4exp(-x^2 - y^2)$$

$$f(x, y) = (\sqrt{x} - 5y)/(x^2 + y^2 - 2x + 10)$$

4 Wnioski

1. Właściwy zestaw parametrów pozwala na dość dokładne znalezienie globalnego ekstremum funkcji z dwóch zmiennych
2. Wizualizacja może pomóc w debugowaniu i ulepszaniu algorytmu, a także pomaga zrozumieć przebieg i zasady samego algorytmu genetycznego
3. Udało nam się poprawnie zrównoleglić algorytm genetyczny i odpowiednio dobrać parametry aby osiągnąć przyśpieszenie