Optymalizacja funkcji za pomocą algorytmów ewolucyjnych

Michał Żelazko

Damian Kantorowski

# Opis algorytmu

Kod programu napisany został w języku Python. Do uruchomienia wymagane są biblioteki Numpy oraz Numba.

* Reprezentacja i inicjalizacja

Populacja początkowa zawiera osobniki składające się z wektora współrzędnych rzeczywistych z dziedziny funkcji wylosowanych z użyciem rozkładu jednostajnego, którego rozmiar określa parametr n oraz z wektora sigm o takim samym rozmiarze z początkowymi wartościami 1. Wartość sigma jest odchyleniem standardowym rozkładu normalnego. Ilość osobników populacji początkowej określa parametr pop\_size.

* Selekcja

Wykorzystano selekcję turniejową. Polega ona na wyborze jednego, osiągającego najlepszy wynik spośród k osobników wylosowanych bez zwracania z populacji. Turniej powtarzany jest, aż wybrana zostanie zadana liczba osobników.

* Krzyżowanie

Wybrane na etapie selekcji osobniki wykorzystywane są do rekombinacji ważonej. Polega ona na wyznaczeniu wag dla rodziców uporządkowanych od najlepszego do najgorszego względem wyników zgodnie ze wzorem:



A następnie tworzeniu potomków składających się z wektora sigm:

Obraz zawierający tekst, Czcionka, pismo odręczne, diagram

Opis wygenerowany automatycznie

oraz współrzędnych:

Obraz zawierający tekst, Czcionka, pismo odręczne, diagram

Opis wygenerowany automatycznie

By nie dopuścić do stworzenia niepoprawnego punktu, współrzędna nie należąca do dziedziny funkcji jest do niej sprowadzana poprzez odbijanie od brzegu przedziału.

* Mutacja

Podczas mutacji modyfikowane najpierw wartości sigma:

Obraz zawierający tekst, Czcionka, zrzut ekranu, biały

Opis wygenerowany automatycznie

a następnie współrzędne:

Obraz zawierający tekst, Czcionka, zrzut ekranu, biały

Opis wygenerowany automatycznie

Gdzie Obraz zawierający tekst, Czcionka, zrzut ekranu, biały

Opis wygenerowany automatycznie oraz Obraz zawierający tekst, Czcionka, zrzut ekranu, biały

Opis wygenerowany automatycznie

Przyjęto

Tak jak w trakcie krzyżowania, niepoprawne współrzędne odbijane są tak by trafiły do dziedziny.

Odbicie sprowadza też wartość sigmy do przedziału .

* Zastępowanie

Stała liczba najlepszych osobników z poprzedniej generacji dołącza do generacji nowych rozwiązań. Na etapie krzyżowania tworzonych jest tyle osobników potomnych, by rozmiar populacji nie uległ zmianie.

# Metodologia pomiarów

Wartości parametrów:

rozmiar populacji [pop\_size] = 22

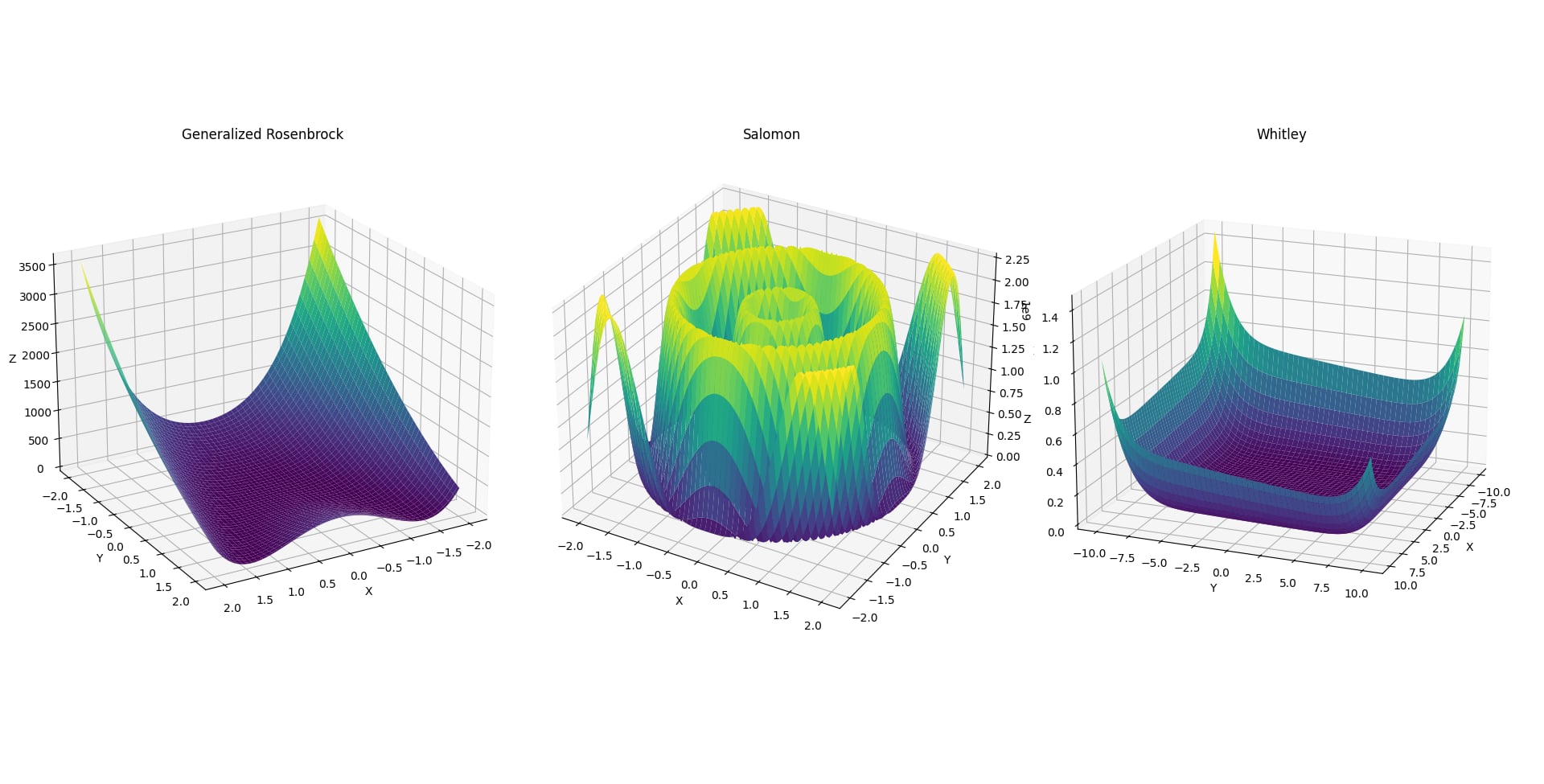
liczba pozostających najlepszych osobników [elite\_size] = 10

rozmiar turnieju [k] = 6

Parametry, jak również metody selekcji, krzyżowania i mutacji zostały wybrane w wyniku eksperymentów minimalizujących średnią ostatnich wartości wszystkich funkcji ewaluacji. Są one takie same dla wszystkich funkcji testowych oraz wszystkich wymiarów przestrzeni. Do optymalizacji wykorzystany został framework Optuna, którym wykonane zostały testy dla reprezentacji binarnej oraz rzeczywistej, selekcji: rankingowej, koła ruletki, turniejowej, krzyżowania: one\_point, two\_point, uniform, weighted, oraz mutacji: bit-flip, SBX[[1]](#footnote-1), z wektorem sigm.

W celu wygenerowania wykresów ECDF prezentujących postęp algorytmu w kolejnych krokach dla różnych progów jakości wymagane było przeprowadzenie dziewięciu niezależnych eksperymentów dla wymiarów przestrzeni n ∈ {5, 15, 30} oraz funkcji ewaluacji Generalized Rosenbrock z dziedziną [30, 30], Salomon z dziedziną [100, 100] oraz Whitley z dziedziną [-10.24, 10.24].

# Wykresy funkcji ewaluacyjnych



# Wykresy krzywych ECDF

# Obserwacje

Najlepszym osiągniętym progiem jakości dla n = 5 jest 0.15848932. Wraz ze wzrostem liczby wymiarów przestrzeni pogarszają się zdolności algorytmu. Dla n = 30 algorytm nie zbliżył się do sumy optimów trzech funkcji ewaluacji w żadnym ze 100 eksperymentów. Problemem są bardzo słabe wyniki w minimalizacji funkcji Whitley. Wartości tej funkcji w podanej dziedzinie są bardzo duże, co powoduje, że suma na podstawie której utworzone zostały wykresy nie spada poniżej 100.

1. Eyal Wirsansky "Hands-On Genetic Algorithms with Python" [↑](#footnote-ref-1)