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:
  $$w_j = \log(\mu+1) - log(j);\ \forall_j \in \{1, \ldots, \mu\}$$
  A następnie tworzeniu potomków składających się z wektora sigm:
  $$\sigma_i' = \left(\frac{1}{\sum_{j=1}^{\mu} w_j} \sum_{j=1}^{\mu} w_j \cdot \sigma_{i,j} \right) \exp(\tau' N(0,1) + \tau N_i(0,1))$$
  oraz współrzędnych:
  $$x_i' = \left(\frac{1}{\sum_{j=1}^{\mu} w_j} \sum_{j=1}^{\mu} w_j \cdot x_{i,j} \right) + \sigma_i' \cdot N_i(0,1)$$
  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:
  $\sigma' \leftarrow \sigma \cdot \exp(\tau \cdot N(0,1))$
  a następnie współrzędne:
  $x_i^{t+1} = x_i^t + N(0,\sigma')$
  Gdzie $\tau \propto \frac{1}{\sqrt{n}}$ oraz $\sigma' < \varepsilon_0 \Rightarrow \sigma' = \varepsilon_0$
  Przyjęto $\varepsilon_0 = 1e-6$
  Tak jak w trakcie krzyżowania, niepoprawne współrzędne odbijane są tak by trafiły do dziedziny.
  Odbicie sprowadza też wartość sigmy do przedziału $[\varepsilon_0, 1]$.
- 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

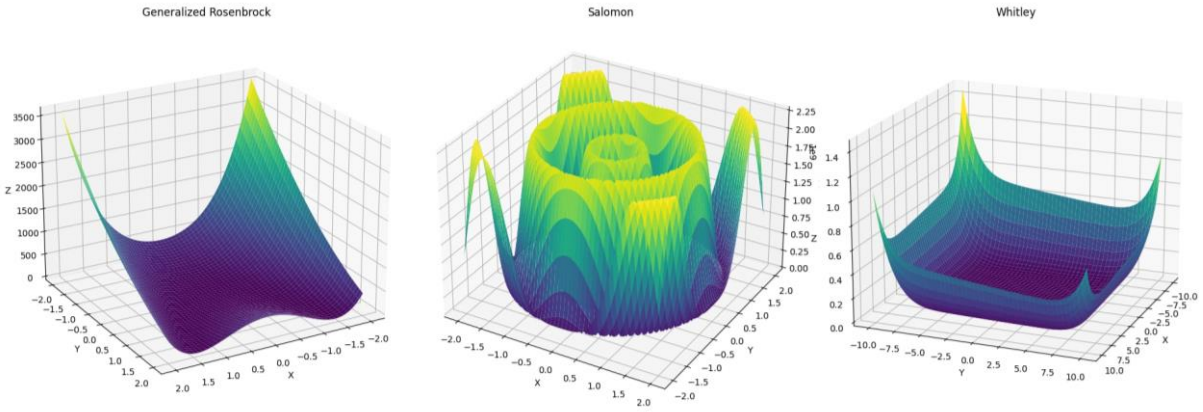
rozmiar turnieju [k] = 6

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, ze zmiennym prawdopodobieństwem zajścia krzyżowania oraz mutacji: bit-flip, SBX[1], z wektorem sigm.

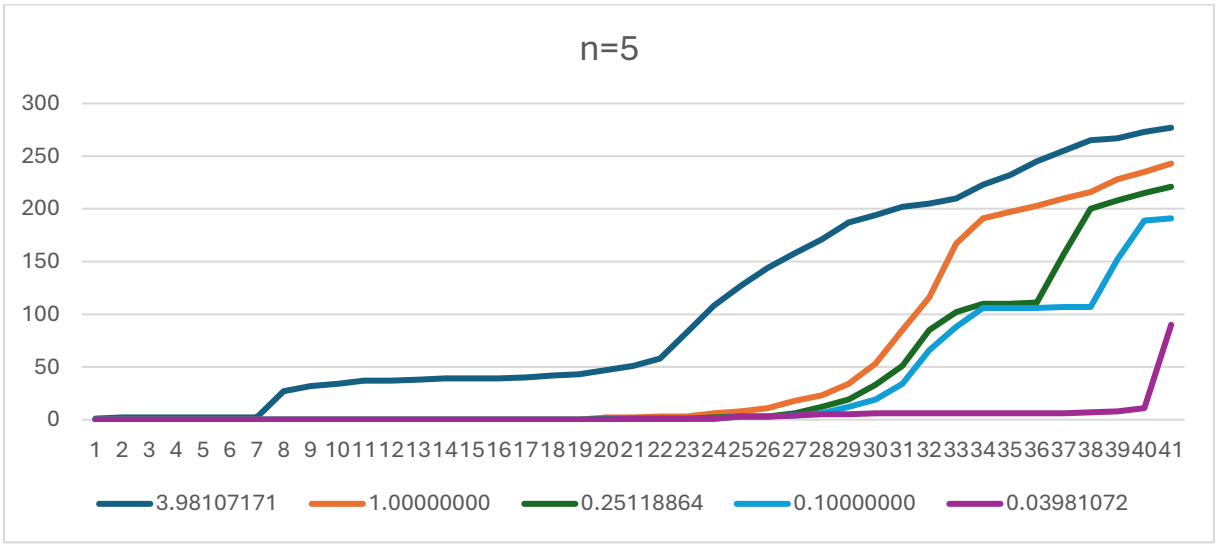
W celu wygenerowania wykresów ECDF prezentujących postęp algorytmu w kolejnych krokach dla różnych progów jakości wymagane było przeprowadzenie dziewięciu niezależnych eksperymentów dla wymiarów przestrzeni n ∈ {5, 15, 30} oraz funkcji ewaluacji Generalized Rosenbrock z dziedziną [30, 30], Salomon z dziedziną [100, 100] oraz Whitley z dziedziną [-10.24, 10.24]. Czas działania pojedynczego eksperymentu ograniczona jest liczbą wywołań funkcji, która wynosi 10000 * n.

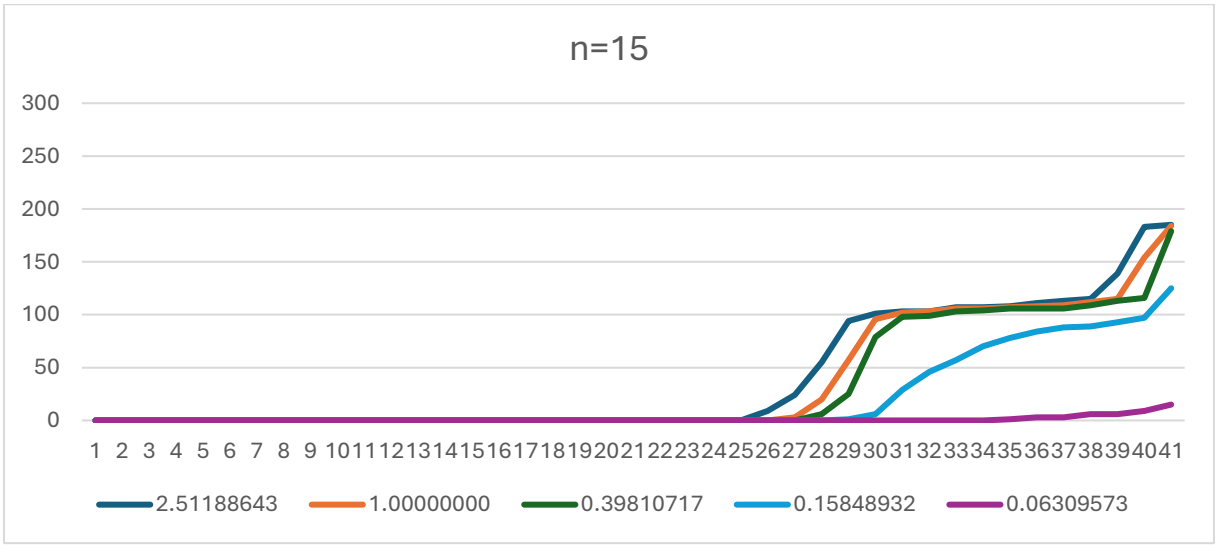
[1] Eyal Wirsansky "Hands-On Genetic Algorithms with Python"

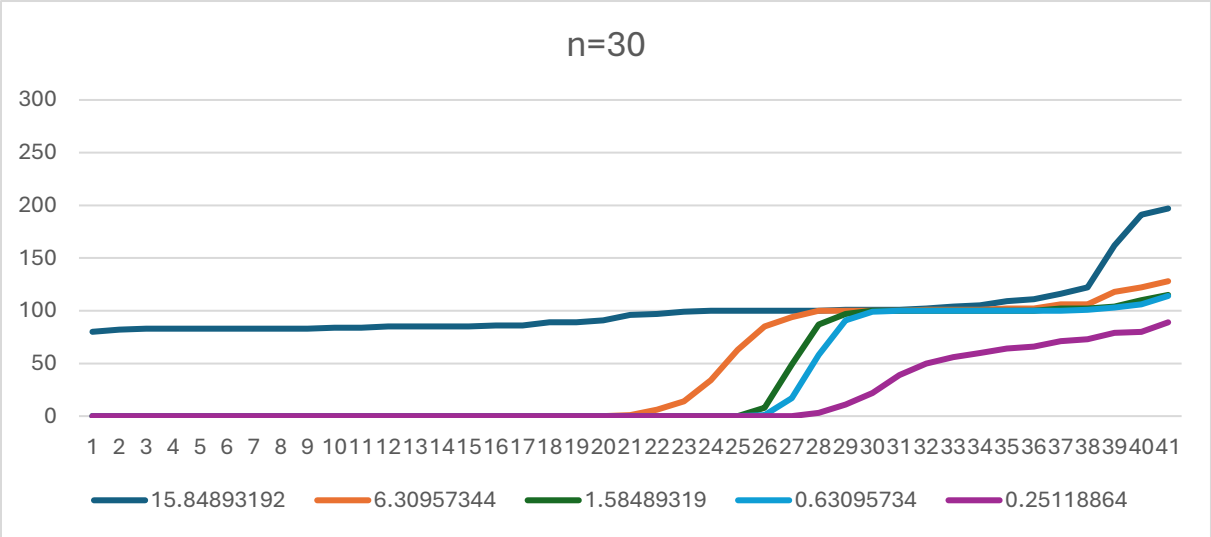
Wykresy funkcji ewaluacyjnych



Wykresy krzywych ECDF







Obserwacje

Na wykresach zaobserwować można polepszanie się rozwiązań w kolejnych progach ewaluacji dla różnych dokładności opisanych w legendzie. Pięć z trzystu przebiegów algorytmu dla n=5 osiąga maksymalną założoną dokładność względem prawdziwego minimum wynoszącą $1e-8$. Wyniki algorytmu stają się dużo mniej dokładne dla większych wymiarów przestrzeni. Przy n=30 tylko jeden eksperyment osiągnął dokładność lepszą niż 0.01584893.