Strategia ewolucyjna typu $(\mu/\mu, \lambda)$ – wsi lab 2

Autor: Patryk Zdziech

Nr.a 311028

Środowisko: Python

Użyte biblioteki: numpy, math, pyplot from matplotlib

Implementacja

Oba algorytmy przyjmują na wejściu parametry:

- `func` -funkcja podlegająca optymalizacji
- `starting_x` – początkowa wartość x dla poszukiwań
- `params` – słownik zawierający słownik resztę parametrów

  `"dimensions"` - liczba zmiennych funkcji (D)
  `"lambd"` - liczba potomnych w populacji (λ) ; wartość domyślna: D*5
  `"u"` - liczba rodziców (μ) ; wartość domyślna: λ*10
  `"maxit"` – maksymalna liczba iteracji ; wartość domyślna D*100
  `"stop_fitness"` – satysfakcjonujące nas minimum funkcji; wartość domyślna 0.1

Algorytm SA (Self-Adaptation)

Zrealizowany zgodnie z założeniami w skrypcie:

$$\xi_i^t = \tau \mathcal{N}_i(0,1)$$
$$z_i^t = \mathcal{N}_i(0, I_D)$$
$$\sigma_i^t = \sigma^t exp\{\xi_i^t\}$$
$$x_i^t = x^t + \sigma_i^t z_i^t, i \in 1 \dots \lambda$$

W szczególności:

zasięgu mutacji (σ) wyliczany średnią ważoną

I_D przyjęto jako macierz jednostkową

przyjęto τ proporcjonalny do $1/\sqrt{D}$

Algorytmie LMR (Log-Normal Mutation Rule)

Zrealizowano podobnie jednak przyjęto parametr mutacji wspólny dla wszystkich składowych x:

$$\xi_i^t = \tau\mathcal{N}_i(0,1)$$
$$z_i^t = \mathcal{N}_i(0, I_D)$$
$$\sigma^t = \sigma^{t-1}e^{\tau N(0,1)}$$
$$x_i^t = x^t + \sigma^t z_i^t, i \in 1 \dots \lambda$$

Funkcje testowe

Do badania obu algorytmów będziemy używać funkcji sferycznej o 10 wymiarach oraz funkcji:

$$q(x) = [(\| x \|^2 - D)^2]^{1/8} + D^{-1}(\frac{1}{2}\| x \|^2 + \sum_{i=1}^{D} x_i) + \frac{1}{2}, x \in [-100,100]^D$$

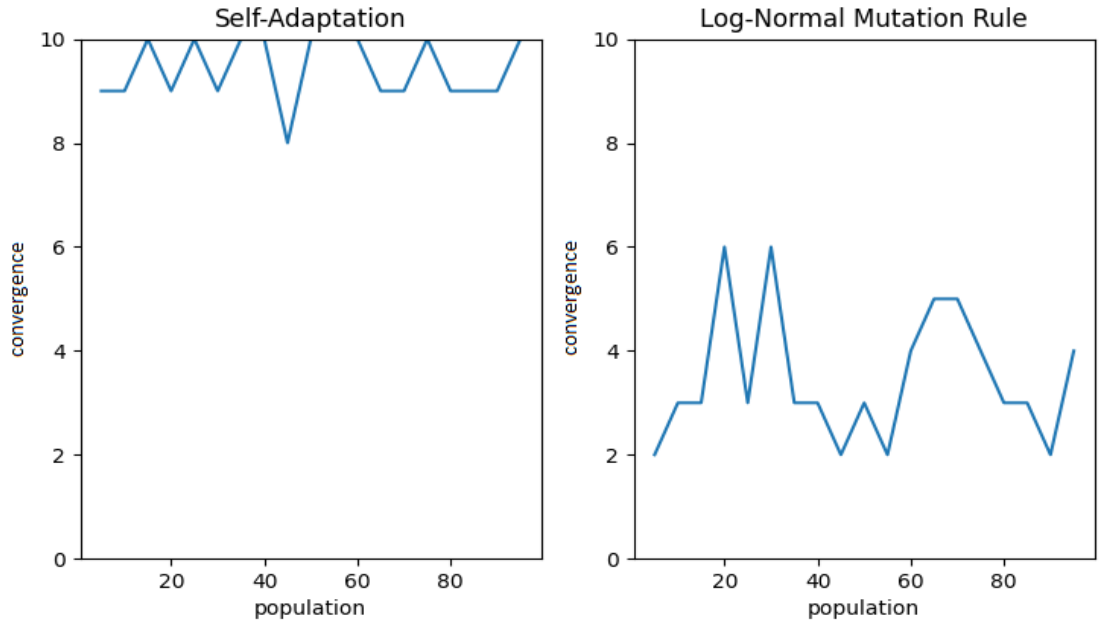
Również o 10 wymiarach (D = 10)

Dobieranie Parametrów λ i σ

Do początkowych badań użyjemy funkcji $q(x)$. Parametry które nie są przedmiotem badania będą przyjmowały wartość domyślną w poniższych eksperymentach.

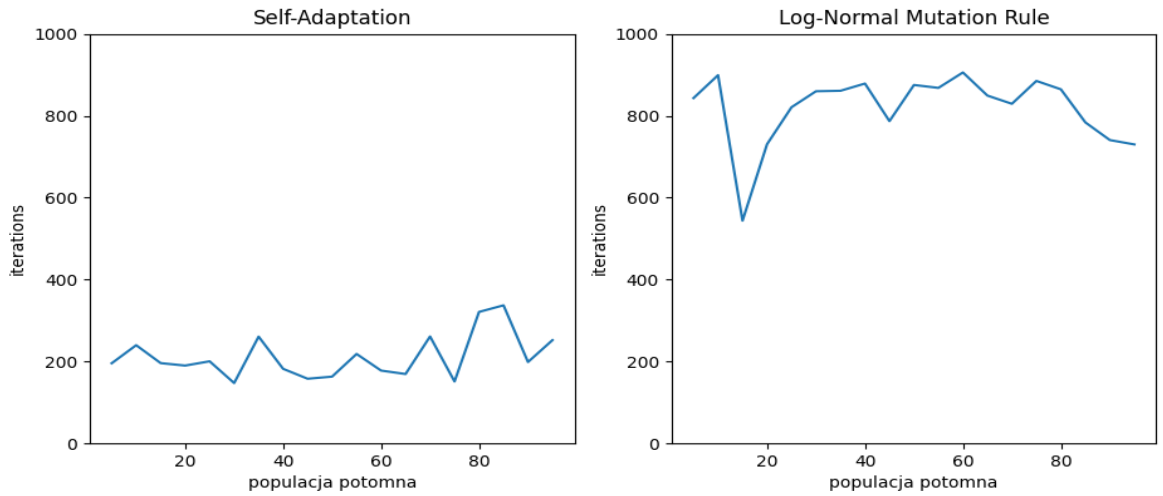
Parametr Lambda

W pierwszej próbie zbadano ile razy z 10 prób funkcja osiągnie zadaną zbieżność do minimum funkcji zależnie od populacji potomnej (parametru λ). Zadana zbieżność to odległość 0.1 od faktycznego minimum.



Widzimy że algorytm SA jest zbieżny w większości przypadków. Algorytm LMR natomiast w mniej niż połowie. Nie występuje tu widoczna korelacja między wartością parametru λ, a zbieżnością algorytmu, wydają się to raczej być zakłócenia losowe.

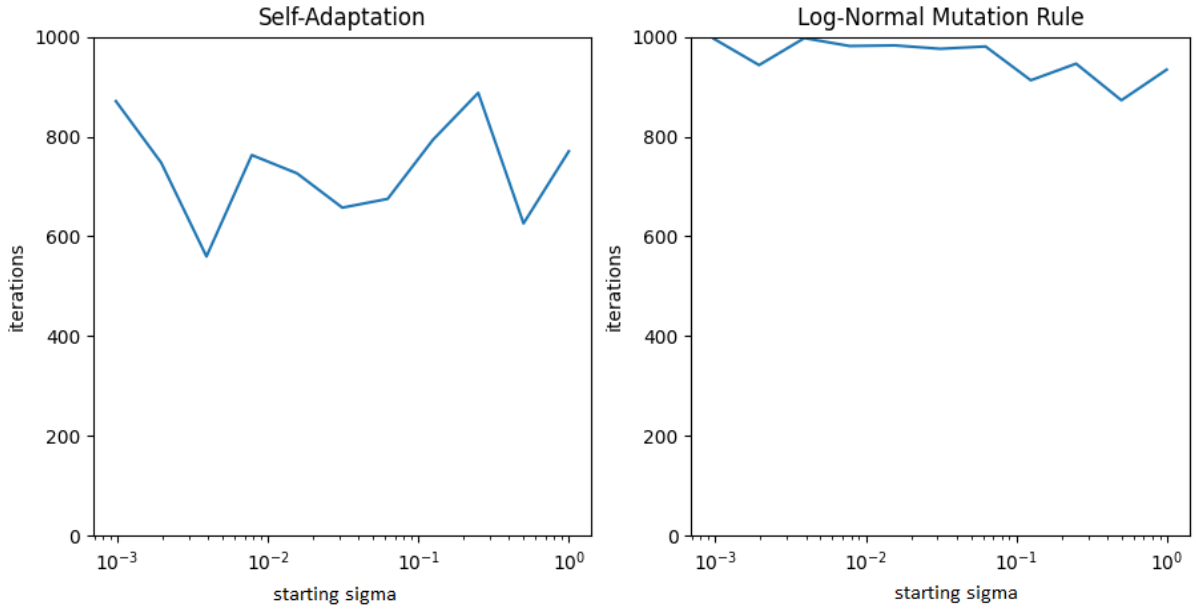
Ponieważ próba nie przyniosła efektów przeprowadzimy kolejny eksperyment. Sprawdzimy szybkość zbiegania algorytmów. Założymy tutaj że czym mniej iteracji potrzebował algorytm tym szybciej zbiega on do wartości minimalnej. Wartości uśrednione z 20 prób:



Nadal nie jest widoczna bezpośrednia korelacja. Wybierzemy zatem stosunkowo niską wartość by uzyskać szybsze działanie algorytmu. Dla obu algorytmów wartość 20 wydaje się osiągać akceptowalne wyniki więc będziemy jej używać w kolejnych eksperymentach.

Parametr Sigma

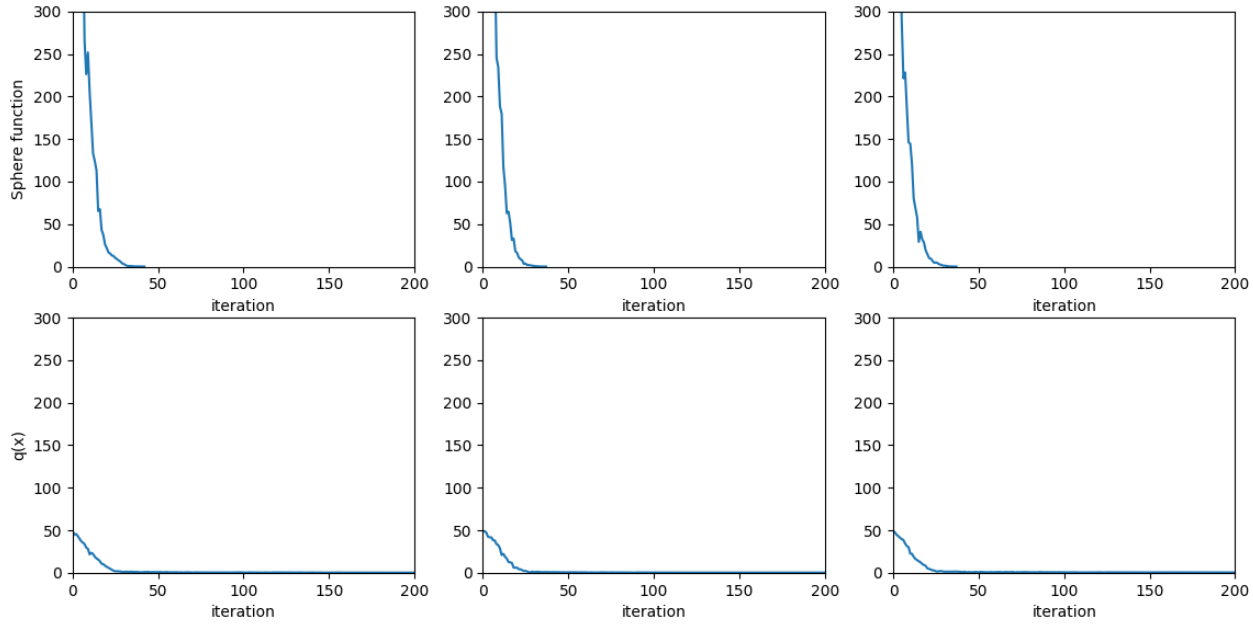
Użyjemy metody zwracającej uwagę na ilość iteracji algorytmu, by ustalić jak szybko algorytm zbiega do minimum. Ponieważ przewiduję że sigma będzie z zakresu od 0.001 do 1 użyję skali logarytmicznej przy dobieraniu parametrów, gdyż bardziej będzie nas interesował rząd wielkości σ. Wartości uśrednione z 20 prób:



Ponownie nie widać bezpośredniej korelacji, jednak algorytm LMR zdaje radzić sobie nieco lepiej dla wartości 0.5. Ponieważ Algorytm SA również posiada minimum lokalne w tej wartości będę jej używać w kolejnych eksperymentach. Wygląda natomiast na to że algorytm SA radzi sobie porównywalnie dobrze również dla pozostałych wartości σ.

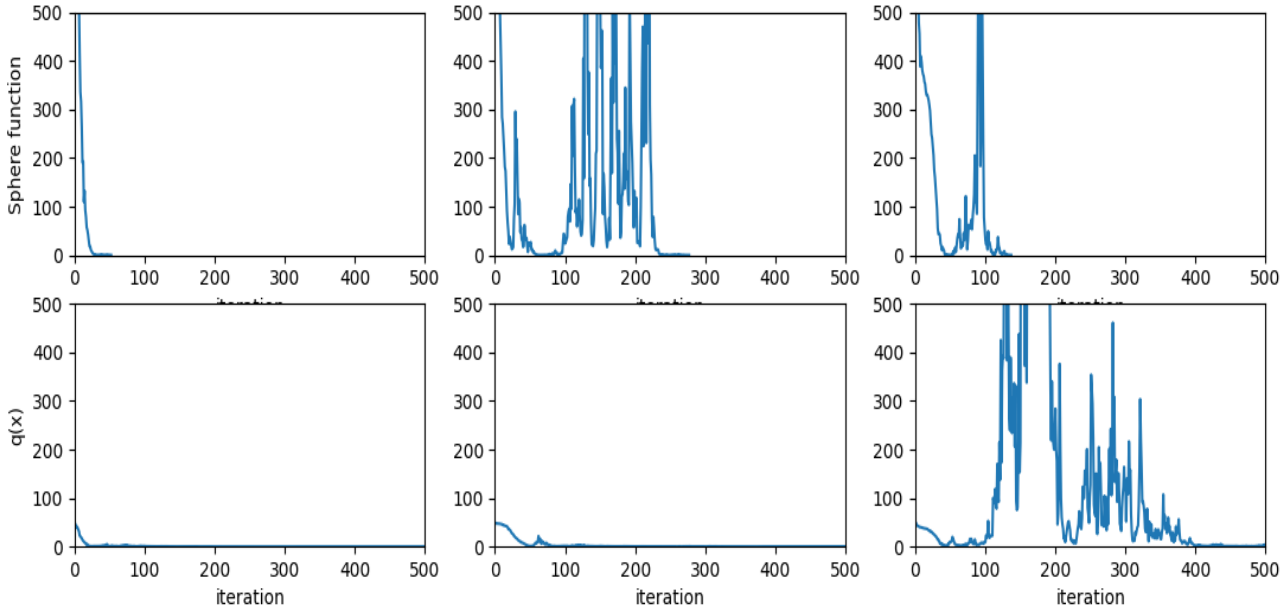
Badanie zbieżności wprost dla ustalonych parametrów.

Po dobraniu parametrów λ i σ wygeneruje wykresy działania algorytmu dla konkretnych przykładów. Będą one przedstawiać wartość funkcji dla najlepszego potomka w danej iteracji. W ten sposób zobaczymy jak algorytm przybliża się i oddala od wartości minimalnej. Poniżej trzy wykresy dla funkcji sferycznej i trzy wykresy funkcji $q(x)$ dla algorytmu SA:



Widzimy dużą stabilność w dążeniu algorytmu SA do minimum, bez znaczących oscylacji. Zwykle do 50 tej iteracji osiąga wartość bardzo bliską minimum.

Wygenerujmy teraz wykresy dla algorytmu LMR:



Widzimy że algorytm już w pierwszych 100 iteracjach zbliża się bardzo blisko do wartości minimalnej. Wpada on niekiedy w oscylacje ponieważ nasza pożądana dokładność jest dość duża, ale przy szerszym kryterium stopu radziłby sobie niewiele gorzej od algorytmu SA.