WSI Laboratorium 2
Optymalizacja za pomocą Algorytmu Ewolucyjnego

Filip Misztal 310276

15 listopada 2023

1 Opis Implementowanych Algorytmów

Przedmiotem tego zadania była implementacja algorytmu ewolucyjnego do znajdowania minimum funkcji celu (pochodzących z benchmarku CEC2017). W poszukiwaniu optymalnego rozwiązania dla naszego problemu zaimplementowałem dwa algorytmy: standardowy algorytm ewolucyjny oraz algorytm realizujący strategię $\mu + \lambda$.

Pierwszy algorytm ewolucyjny realizował zarówno operację mutacji, jak i krzyżowania. Przyjmuje populację osobników w postaci wektorów cech. Krzyżowanie może się odbywać zarówno w wariancie jedno, jak i dwupunktowym. Selekcja osobników do krzyżowania jest zrobiona w stylu turniejowym (możliwość ustawienia wielkości turnieju oraz odsetku populacji, który zostanie przeznaczony do krzyżowania). Siła mutacji jest podawana na początku działania algorytmu i pozostaje niezmienna w czasie. Sposób działania:

Standardowy algorytm ewolucyjny

1. Generuję populację o wybranej przeze mnie liczebności.
2. Poddaję osobników ocenie oraz wybieram najlepiej przystosowanego osobnika.
3. Wybieram z populacji zadany odsetek osobników (selekcja turniejowa) oraz poddaję ich krzyżowaniu (jedno lub dwupunktowe).
4. Osobniki otrzymane w wyniku krzyżowania poddaję mutacji - dla każdej cechy w zależności od zadanego prawdopodobieństwa mutacji zachodzi mutacja o również podanej wcześniej stałej sile (rozkład normalny).
5. Mutanci zostają poddani ocenie oraz zostaje wyselekcjonowany ewentualny nowy najlepszy osobnik.
6. Przeprowadzam sukcesję elitarną.
7. Powtarzam kroki 3-6 aż do osiągnięcia określonej liczby iteracji algorytmu.

Drugim algorytmem, który zdecydowałem się zaimplementować, jest algorytm strategii $\mu+\lambda$. Zrezygnowałem w nim z przeprowadzania krzyżowania, natomiast mutację rozbudowałem o system zmiany jej siły w czasie. W tym celu osobnik został rozszerzony o dodatkowy chromosom zawierający wektor sił mutacji odpowiadający długością liczbie cech. Selekcja osobników do mutacji przebiega losowo, a ich ilość jest uwarunkowana parametrem λ. Sukcesja jest typu elitarnego. Sposób działania wygląda następująco:

Algorytm implementujący strategię $\mu + \lambda$

1. Generuję losową populację początkową μ osobników x_0. Osobnik posiada wektor punktów startowych oraz wektor wartości siły mutacji dla poszczególnych cech (początkowa wartość dla wszystkich cech i osobników ustalana przez użytkownika).
2. Poddaję populację ocenie - wyznaczam wartości funkcji celu dla poszczegoólnych osobników.
3. Wyznaczam najlepiej przystosowanego osobnika.

4. Wybieram z populacji losowo λ osobników (z powtórzeniami) i poddaję ich mutacji. Proces mutacji przebiega następująco:
   (a) Generuję z rozkładu normalnego parametr a oraz wektor parametrów b o liczbie elementów równej ilości cech osobnika.
   (b) Modyfikuję siłę mutacji według wzoru: $\sigma_i = \sigma_i \cdot \exp(\tau' \cdot a + \tau \cdot b_i)$, gdzie i to numer elementu z wektora wartości sił mutacji.
   (c) Mutuję osobnika na podstawie nowootrzymanego wektora sił mutacji: $M_i = P_i + \sigma \cdot N(0, 1)$, gdzie P_i to stary osobnik, a M_i to osobnik po mutacji.
5. Poddaję nowo wygenerowanych osobników ocenie oraz sprawdzam czy nie otrzymałem nowego najlepiej przystosowanego osobnika.
6. Tworzę nową populację wybierając μ najlepiej przystosowanych osobników z łącznie starej populacji i populacji mutantów.
7. Powtarzam kroki 4-6 do czasu aż nie osiągnę zadanej liczby iteracji algorytmu.

2 Opis planowanych eksperymentów numerycznych

W badaniach chciałem porównać działanie obu stworzonych przeze mnie implementacji. Wymagało to wcześniejszego zoptymalizowania parametrów obu tych algorytmów. Przeprowadziłem więc serie testowe, każde dla wektorów początkowych 10-wymiarowych x_0, gdzie każdy element był losowo generowany z przedziału $[-1000, 1000]$.
Testy optymalizujące dla standardowego algorytmu ewoucyjnego:

- zmiana odsetka populacji przeznaczonej do rozrodu: 0.3, 0.5, 0.7, 0.8, 1.0
- zmian siły murtacji: 0.3, 0.6, 0.8, 1.0, 2.0
- zmiana częstotliwości mutacji: 0.3, 0.6, 0.8, 1.0
- wybór techniki krzyżowania: jednopunktowe lub dwupunktowe

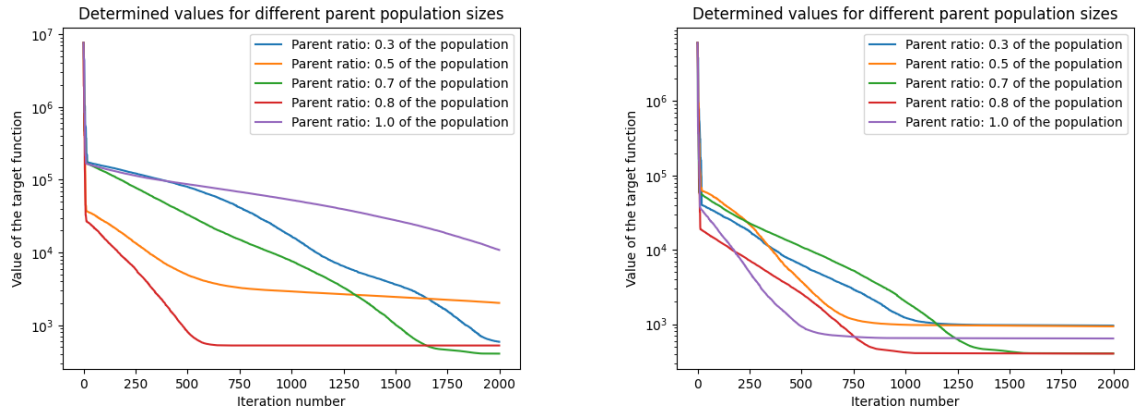
Testy optymalizujące dla algorytmu strategii $(\mu + \lambda)$:

- zmianie parametru λ (jako odsetek wartości μ): 0.2, 0.5, 0.9, 1.2, 2
- zmianie początkowej siły mutacji: 1000, 100, 10, 1, 0.1

Po znalezieniu optymalnych punktów pracy porównałem ich osiągi ze sobą, oraz z opracowanym w poprzednim ćwiczeniu algorytmem gradientu prostego.

3 Opis uzyskanych Wyników

Optymalizacja standardowego algorytmu ewolucyjnego:
Wykresy przedstawiające zbieganie standardowego algorytmu ewolucyjnego do minimum w czasie w zależności od odsetka rodziców:

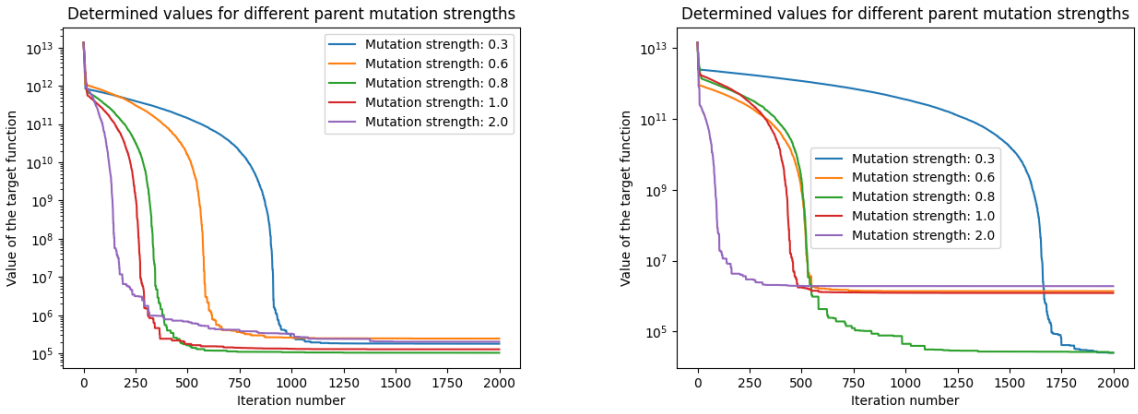


(a) (b)

Rysunek 1: Wykresy zależności od odsetka rodziców

Widać tutaj, że dopuszczenie do rozrodu zbyt szerokiego zakresu z populacji może prowadzić do zbyt dużej różnorodności genetycznej (rys a), a zbyt mała ilość rodziców spowalnia proces ewolucji zmniejszając szansę na odkrycie nowych cech. Wybrałem więc wartość ze środka, czyli 0.8.

Wykresy przedstawiające zbieganie standardowego algorytmu ewolucyjnego do minimum w czasie w zależności od siły mutacj:

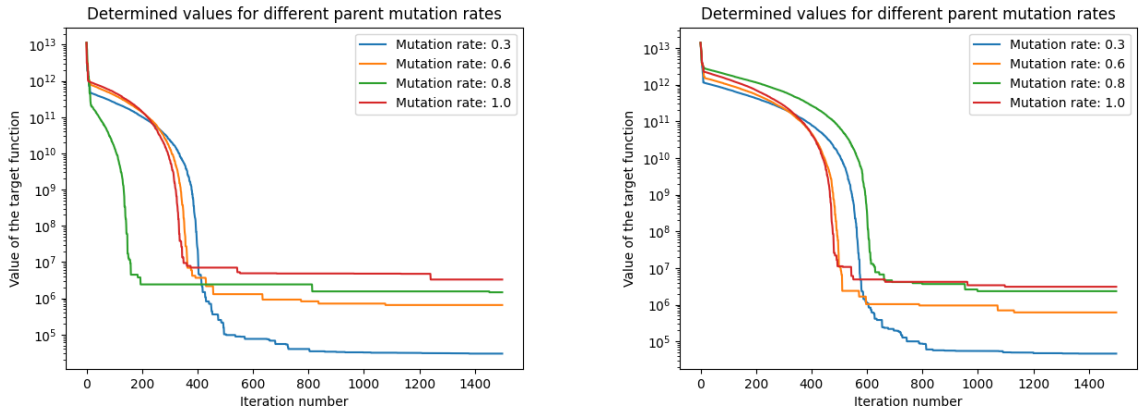


(a) (b)

Rysunek 2: Wykresy zależności od siły mutacji

Powyższe wykresy pokazują, że wysoka wartość siły mutacji zwiększa możliwości eksploracyjne i przyspiesza działanie algorytmu, jednak w końcowej fazie działania powoduje problemy ze zbieganiem do dokładnego minimum. Może też spowodować w pewnych wypadkach trudność w wyjściu z optimum lokalnego. Zbyt mała siła mutacji powoduje natomiast do małej różnorodności genetycznej, wolnej eksploracji przestrzeni oraz także problemy z minimami lokalnymi. Do dalszych badań wybrałem więc wartość ze środka, czyli 0.8.

Wykresy przedstawiające zbieganie standardowego algorytmu ewolucyjnego do minimum w czasie w zależności od szansy mutacji:

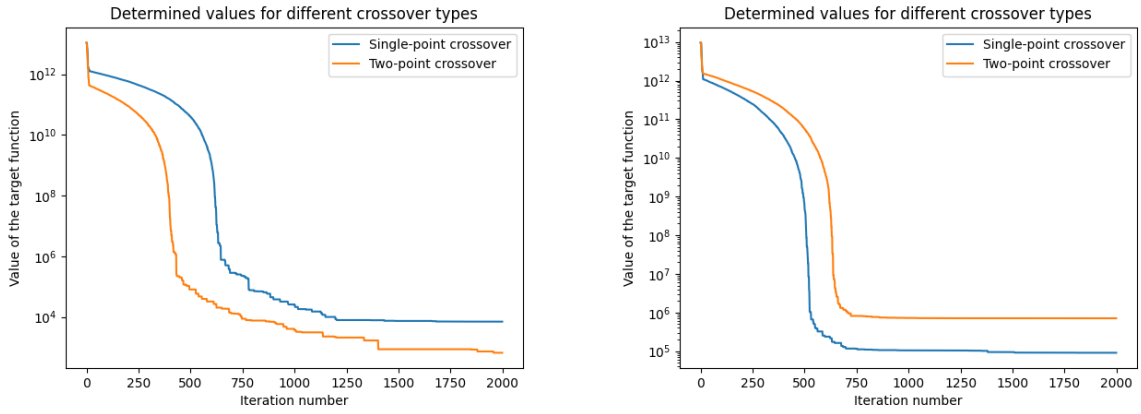


(a) (b)

Rysunek 3: Wykresy zależności od szansy mutacji

Można na podstawie powyższych wykresów zauważyć, że większa szansa mutacji z reguły przyspiesza zbieganie algorytmu, jednak zwiększa znacząco szansę na utknięcie w minimum lokalnym. Dlatego też pozostałem przy wartości 0.3.

Wykresy przedstawiające zbieganie standardowego algorytmu ewolucyjnego do minimum w czasie w zależności od rodzaju krzyżowania:

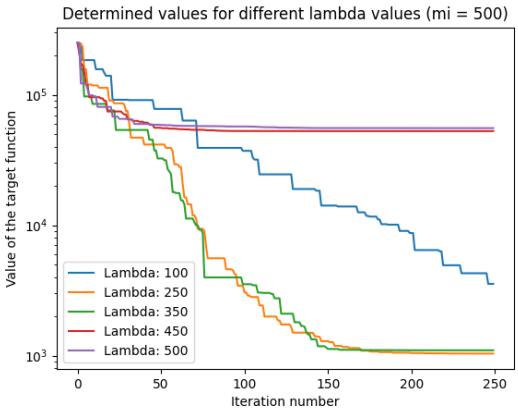


(a) (b)

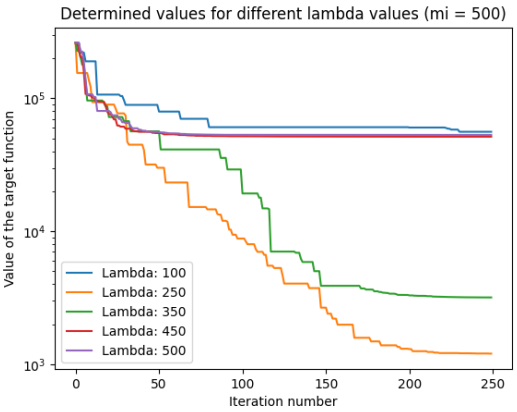
Rysunek 4: Wykresy zależności od rodzaju krzyżowania

W tym punkcie badań nie doszedłem do jednoznacznych wniosków, która metoda jest lepsza. Żadna nie oferowała w naszym problemie znaczącej przewagi. Zdecydowałem się jednak na pozostanie przy krzyżowaniu jednopunktowym ze względu na mniejszy koszt obliczeniowy.

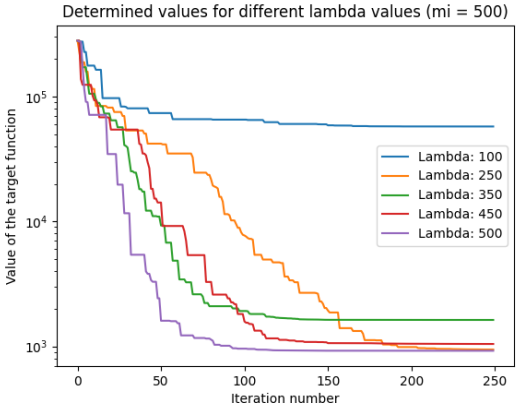
Optymalizacja algorytmu strategii $(\mu + \lambda)$:
Wykresy przedstawiające zbieganie algorytmu strategii do minimum w czasie w zależności od parameru λ:



(a)



(b)

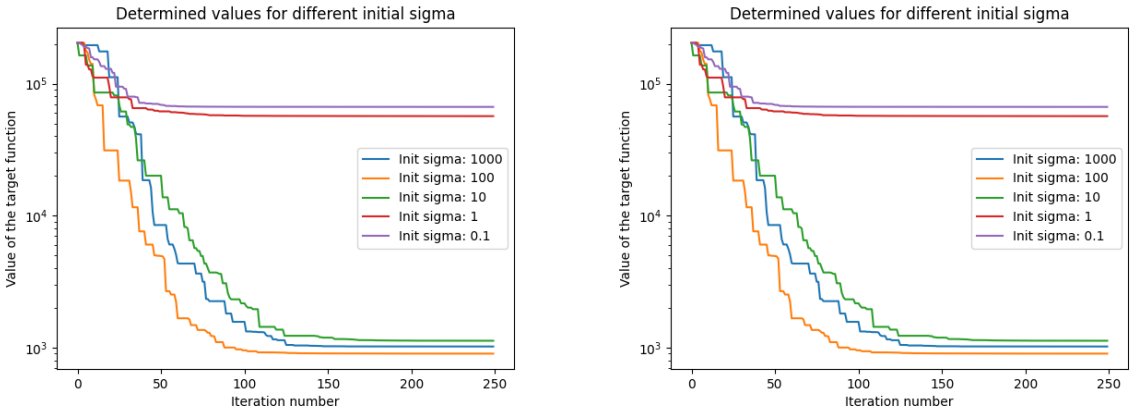


(c)

Rysunek 5: Wykresy zależności od parametru λ

Na powyższych wykresach można zauważyć, że algorytm ma większą tendencję do zatrzymywania się na minimum lokalnym gdy λ jest skrajnie niska albo wysoka w porównaniu do μ. Zdecydowałem się dlatego w dalej korzystać z wartości lambda w wysokości 80% wielkości μ.

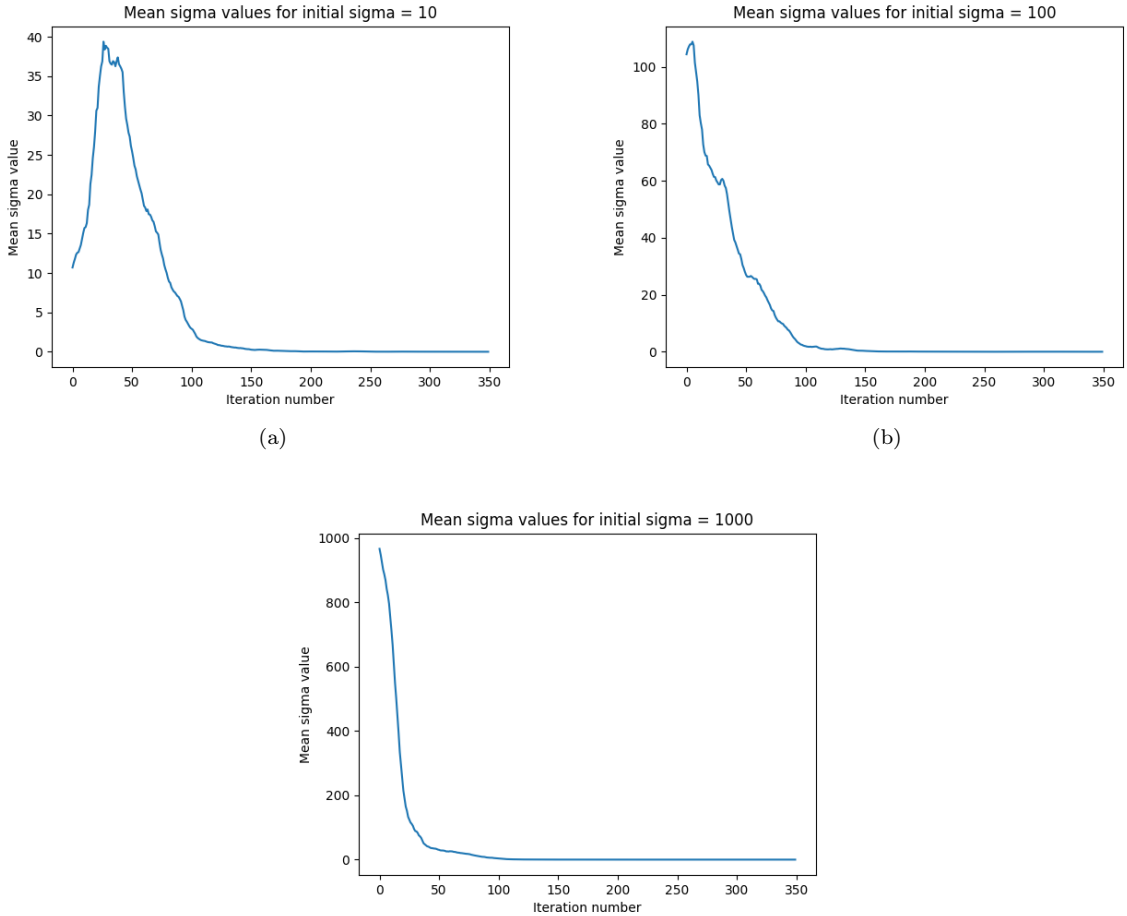
Wykresy przedstawiające zbieganie algorytmu strategii do minimum w czasie w zależności od początkowej siły mutacji:



(a) (b)

Rysunek 6: Wykresy zależności od początkowej wartości siły mutacji

Powyższe obrazki pokazują, że niskie wartości początkowe σ skutkowały poważnym ograniczeniem możliwości eksploracyjnych i przedwczesnym zatrzymywywaniem postępu algorytmu. Dlatego też zrezygnowałem od razu z używania sigmy początkowej o wartości rzędu $< 10^0$. Dalej analizując zależność algorytmu od tego parametru, utworzyłem wykresy zmiany średniej wartości σ w populacji w miarę postępowania ewolucji:

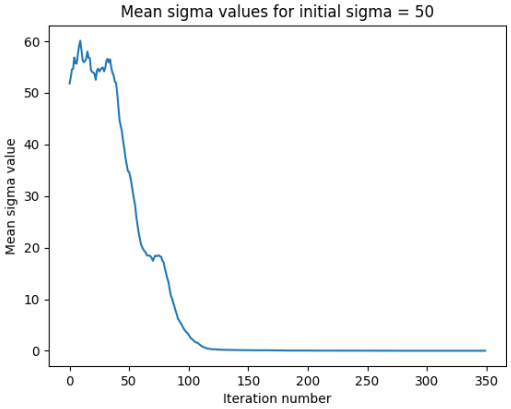


(a) (b)

(c)

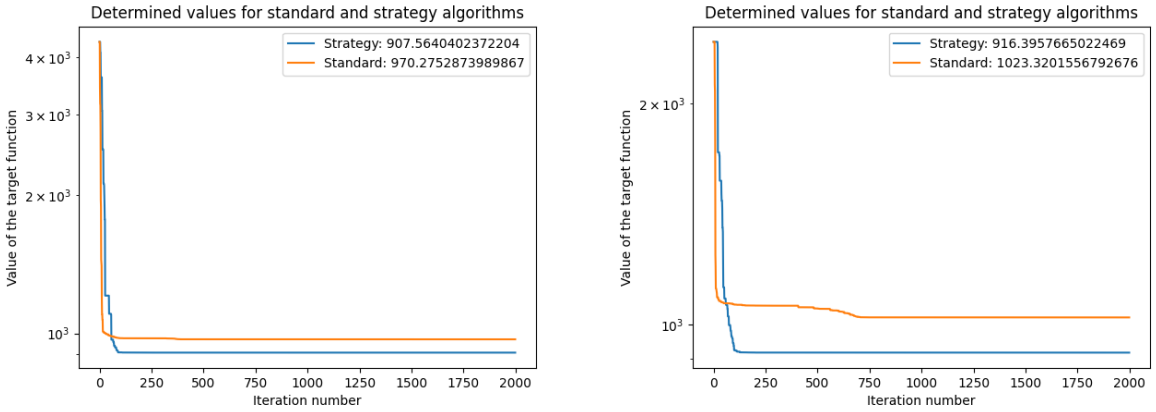
Rysunek 7: Wykresy zmian średniej wartości siły mutacji populacji w czasie

Możemy zaobserwować, że w przypadku σ początkowej równej 1000 i 100 wraz z rozpoczęciem pracy jej wartości zaczynają gwałtownie spadać, natomiast dla $\sigma = 10$ wartość średnia przez pierwsze 50 iteracji rosła, aż do okolic 40. Zdecydowałem więc, że obtymalną wartością początkową będzie 50:



Rysunek 8: Wykresy zmian średniej wartości siły mutacji populacji w czasie dla $\sigma = 50$

Porównanie algorytmów ewolucyjnych (funkcja f9):

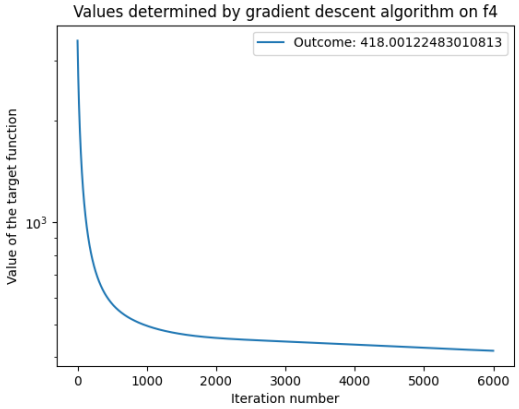


(a) (b)

Rysunek 9: Porównanie algorytmu standardowego z algorytmem strategii $(\mu + \lambda)$

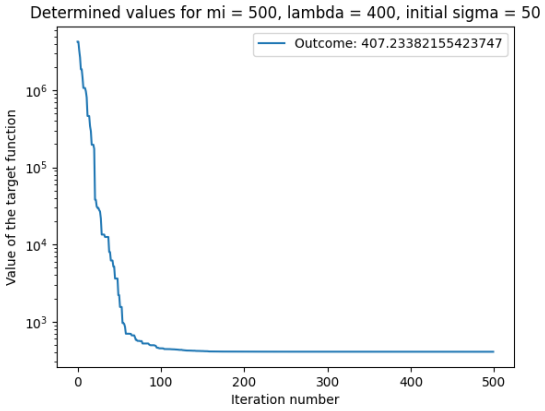
Średnio lepiej wypadał algorytm ze strategią $\mu + \lambda$. Mogło być to spowodowane tym, że w naszym problemie mutacja dawała była bardziej efektywna niż proces krzyżowania. Mutacja z siłą poddawaną ewolucji oferowała bardziej dopasowaną intensywność eksploracji i możliwość omijania minimów lokalnych. Ponadto algorytm ze strategią $\mu + \lambda$ okazał się być szybszy, bowiem przy zadanym w tym badaniu problemie znalezienie rozwiązania zajęło mu ok. 2 sekundy, podczas gdy standardowy algorytm potrzebował 3 sekund.

Dalej chciałem porównać działanie algorytmu ewolucyjnego (strategii $\mu + \lambda$) z algorytmem gradientu prostego. Badania przeprowadziłem na funkcji f4 z benchmarku, jako że tylko tą funkcję był w stanie obsłużyć alg. gradientu prostego.



Rysunek 10: Zbieganie do minimum przez algorytm gradientu prostego

Widać, że algorytm bardzo płynnie i stabilnie minimalizował funkcję celu, jednak wiązało się to z dosyć sporym czasem działania algorytmu. Zajęło to bowiem aż ok. 6000 iteracji, czyli ok.26 sekund. Teraz dla tej samej funkcji szukam minimum za pomocą algorytmu ewolucyjnego:



Rysunek 11: Zbieganie do minimum przez algorytm ewolucyjny ze strategią $(\mu + \lambda)$

Łatwo zauważyć, że algorytm ewolucyjny osiągną porównywalną dokładność w dużo krótszym czasie, bowiem już po zaledwie ok. 200 iteracjach, co trwało mniej niż 2 sekundy.

4 Wnioski z przeprowadzonych badań

Na podstawie przeprowadzonych eksperymentów można wyciągnąć następujące wnioski:

1. W standardowym algorytmie ewolucyjnym:
   - Nie zauważyłem istotnej różnicy między korzyściami, jakie dawały procesy krzyżowania jedno i dwupunktowego w naszym problemie.
   - Dopuszczanie do krzyżowania zbyt wielkiej części populacji może zwiększyć szansę na utknięcie w minimum lokalnym, natomiast zbyt małej powoduje spowolnienie zbiegania algorytmu do minimum.
   - Jako, że mutacja miała być tutaj tylko dodatkiem do procesu krzyżowania, nie należało przeprowadzać jej zbyt intensywnie, tj. ze zbyt dużą siłą i częstotliwością. W przeciwnym razie zwiększa się prawdopodobieństwo utknięcia w minimum lokalnym na wskutek zmniejszenia stabilności populacji i zmniejszenie zbieżności. Zbyt małe wartośći tych parametrów również mogą skutkować trudnościami w przeskakiwaniu minimów lokalnych i wydłużyć czas działania algorytmu.

2. W algorytmie ewolucyjnym ze strategią $\mu + \lambda$:
    - Algorytm ma większą tendencję do zatrzymywania się na minimum lokalnym gdy λ jest skrajnie niska albo wysoka w porównaniu do μ.
    - Małe wartości początkowe σ zmniejszają możliwości eksploracyjne algorytmu i powodują zbieganie do minimum lokalnego. Ze względu na to, że siła mutacji również podlega ewolucji, a także stopniowej redukcji, dobre efekty przyniosło stosowanie stosunkowo wysokich wartości.
3. Wybór optymalnej strategii ewolucyjnej zależy od specyfiki problemu. W przeprowadzonych badaniach porównawczych algorytmów ewolucyjnych, strategia $\mu + \lambda$ osiągała lepsze wyniki niż standardowy algorytm ewolucyjny (dokładniej lokalizowała minimum globalne oraz działała szybciej). Może to wynikać z bardziej elastycznego podejścia do mutacji.
4. W przypadku obu algorytmów kluczowe było dostosowanie parametrów do konkretnego problemu.
5. Algorytm gradientu prostego okazał się zdecydowanie wolniejszy niż algorytm ewolucyjny. Nie był on również w stanie poradzić sobie z wieloma funkcjami z benchmarku, ponieważ mogły być one nieciągłe lub nieróżniczkowalne.