Ireneusz Okniński

Nr 310228

Wprowadzenie do sztucznej inteligencji.

Laboratorium 2.

1. **Cel ćwiczenia.**
   Celem zadania było zaprojektowanie i zaimplementowanie algorytmu ewolucyjnego. Zadaniem tego algorytmu jest znajdowanie globalnego minimum funkcji celu. Interfejs algorytmu powinien być taki sam jak w zadaniu z pierwszego laboratorium tj. przyjmować funkcję celu oraz początkowy punkt/populację oraz zwracać minimum funkcji celu oraz jej wartość w tym punkcie.

2. **Struktura projektu.**
   Projekt składa się z następujących plików i podkatalogów:
   - *algorithm.py – implementacja algorytmu*
   - *experiments.py – funkcje przeprowadzające eksperymenty*
   - *plot.py – funkcje tworzące wykresy*
   - *plots – katalog zawierający wykresy*
   -*results – pliki json będące wynikami eksperymentów*
   - *cec2017 – katalog zawierający funkcje do testowania algorytmu*
   Wyniki algorytmów zapisywałem w plikach .json. Pozwoliło to na wielokrotne tworzenie wykresów na podstawie tych samych wygenerowanych jednokrotnie danych bez ponownego uruchamiania algorytmu, co bywało czasochłonne.

3. **Implementacja algorytmu**.
   Zaimplementowany przeze mnie algorytm jest algorytmem ewolucyjnym ze zmiennym odchyleniem standardowym mutacji. Po każdej iteracji algorytmu sprawdzany jest stosunek liczby udanych mutacji, czyli takich, które poprawiają jakość obecnie najlepszego punktu, do liczby wszystkich iteracji. Zastosowałem zasadę 1/5 sukcesu. Gdy udanych mutacji było mniej niż 1/5 odchylenie standardowe malało, co za tym idzie algorytm kładł większy nacisk na eksploatację. Przez to algorytm miał większą szansę na znalezienie lepszego punktu w swoim bliskim otoczeniu. W przeciwnym przypadku odchylenie standardowe rosło i algorytm eksplorować większy obszar. Dzięki temu zwiększa się prawdopodobieństwo na wyjście algorytmu z optimum lokalnego na korzyść obszaru o lepszej jakości.

   W algorytmie zastosowana jest reprodukcja turniejowa, w której losujemy z powtórzeniami tyle par osobników ile wynosi rozmiar populacji. Z każdej pary do nowej populacji przechodzi lepszy osobnik.

   Krzyżowanie jest uśredniające. Losowo wybrana z populacji para rodziców przekazuje potomkowi swoje geny. Waga dla każdego genu jest losową liczbą z przedziału <0:1>. Potomek otrzymuje sumę genu pierwszego rodzica pomnożonego przez wagę oraz genu drugiego rodzica pomnożonego przez (1 – wagę).

   Mutacje populacji korzystają z rozkładu normalnego. Do każdego osobnika z populacji dodany jest wektor liczb rzeczywistych losowanych z rozkładu Gaussa. Parametrem sterującym mutacją jest odchylenie standardowe. Gdy jest ono większe mutacje tworzą osobników bardziej różniących się od osobników początkowych.

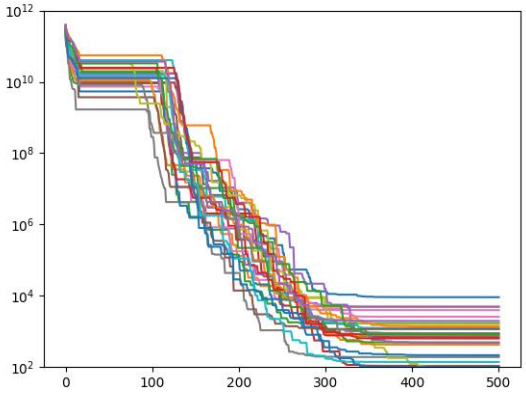
   Sukcesja stosowana w algorytmie jest sukcesją elitarną. Rozmiar elity, czyli ilość najlepszych osobników z populacji przed operacjami genetycznymi, którzy przechodzą do następnej iteracji jest parametrem sukcesji. Po wybraniu elity z początkowej populacji, nowa populacja jest uzupełniana przez najlepszych spośród osobników po krzyżowaniu i mutacjach do rozmiaru populacji.

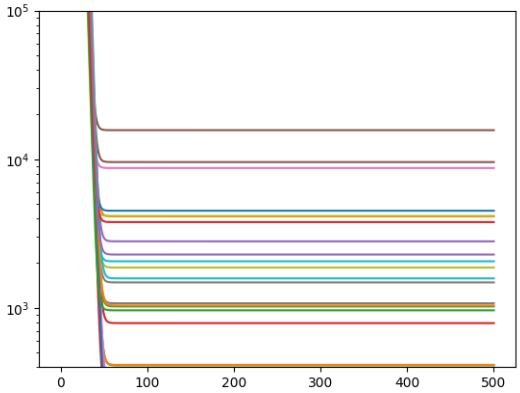
4. Testowanie.

Do testowania algorytmu użyłem funkcji f1 oraz f9 z benchmarku CEC2017. Algorytm ewolucyjny jest bardzo losowy, więc aby wyniki testowania były bardziej wiarygodne każdy z eksperymentów wywoływałem 50 razy. Za każdym razem mierzyłem jakość najlepszego wyniku oraz średnią jakość wyniku. Zbadałem wpływ rozmiaru populacji, liczby iteracji, rozmiaru elity, odchylenia standardowego mutacji na wyniki działania algorytmu.

Zbieżność algorytmów:

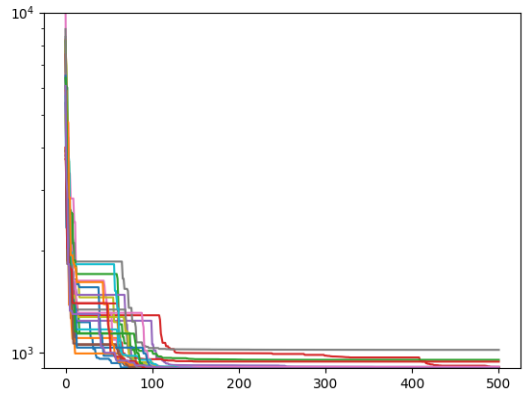
Wykresy zostały wykonane na podstawie 25 powtórzeń wywołania algorytmu.
F1:

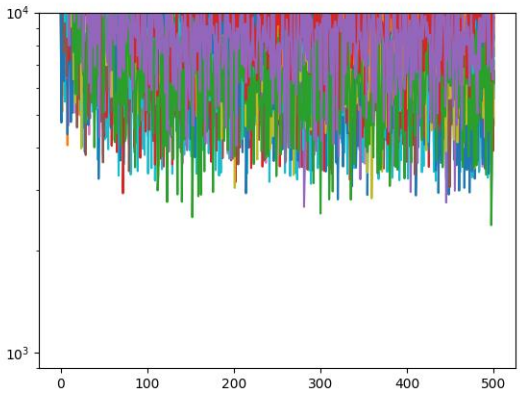




Funkcja w każdym przypadku jest zbieżna. Wartości minimum znajdowanych przez algorytm znajdują się w przedziale (100, 10000). Rozbieżność jest duża z powodu losowości algorytmu. Przy jednokrotnym powtórzeniu nie ma gwarancji znalezienia optimum globalnego funkcji celu. Z wykresu wynika również, że większość linii od ok. 400 iteracji ma stałą wartość. Spowodowane jest to znalezieniem przez algorytm optimum lokalnego.
Podobnie w przypadku algorytmu gradientu prostego, znajdowane optima są bardzo zróżnicowane. Algorytm w większości przypadków znajduje optimum lokalne. Plusem drugiego algorytmu jest to, że szybciej zbiega do małych wartości.

F9:

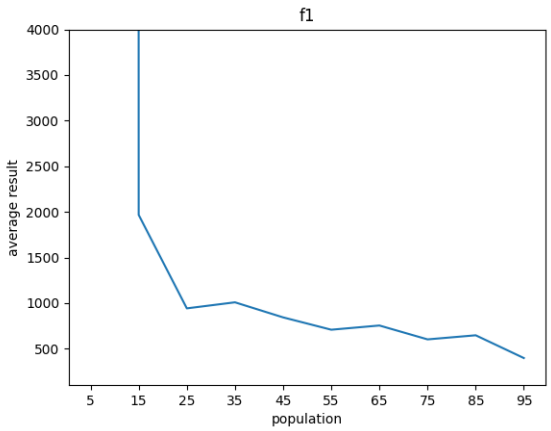


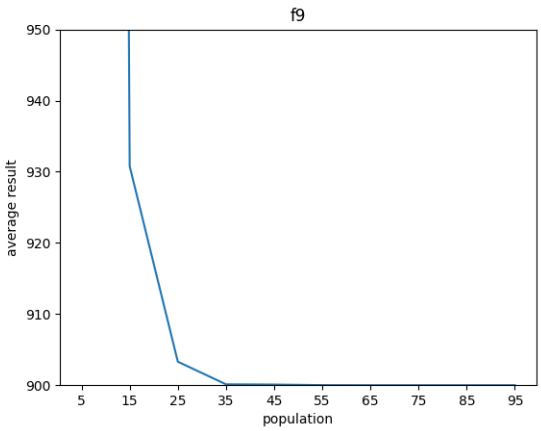


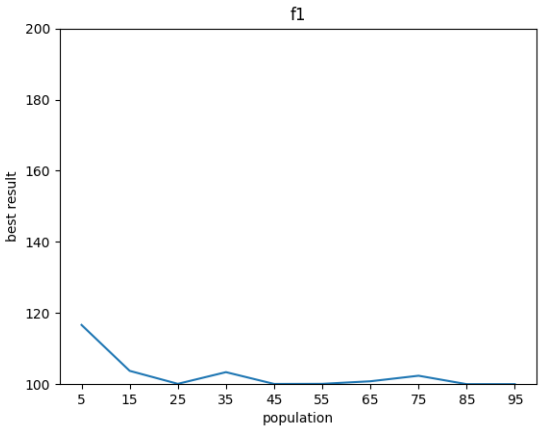
Dla funkcji f9 minimum było znajdywane znacznie częściej niż dla f1. Powodem mogła być budowa funkcji, która bardziej sprzyja algorytmowi genetycznemu. Z wykresu wynika, że wystarczające do znalezienia optimum byłoby ok. 200 iteracji, co jest lepszym wynikiem niż w przypadku f1.

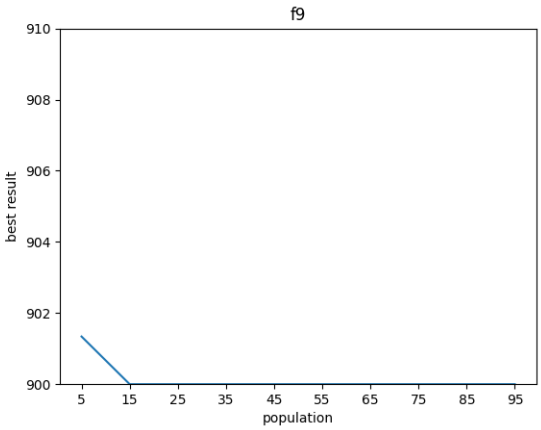
Algorytm gradientu prostego nie sprawdził się dla funkcji f9. Nie znalazłem parametrów dla których algorytm byłby zbieżny w 500 krokach.

Wpływ wielkości populacji:



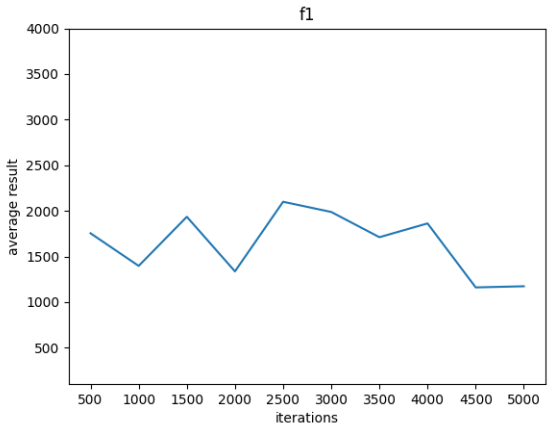


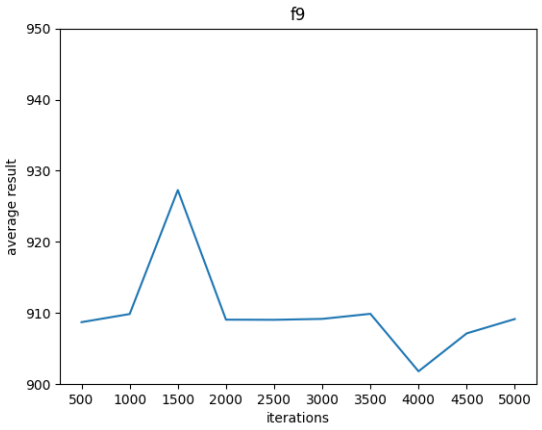


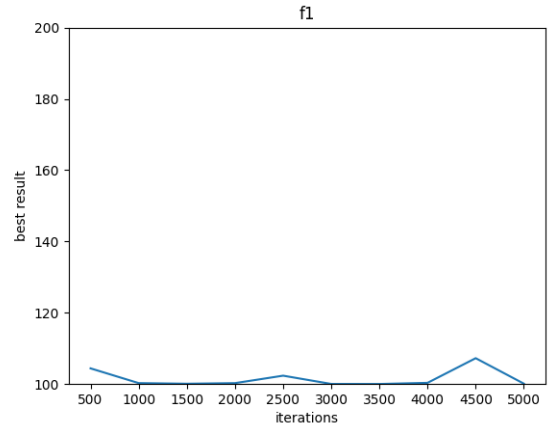


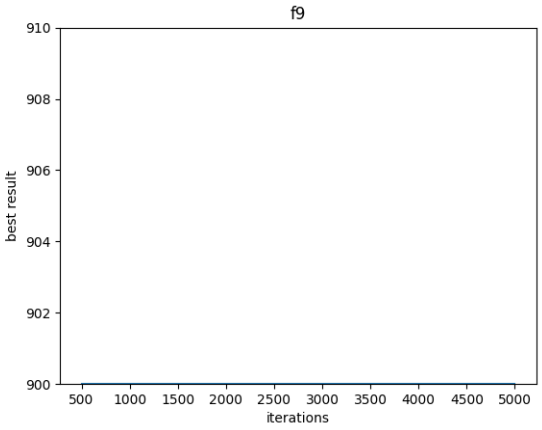
Elita stanowiła zawsze połowę populacji. Zwiększenie wielkości populacji zwiększa skuteczność algorytmu, odbywa się to kosztem czasu wykonania, który liniowo zależy od iloczynu rozmiaru populacji i liczby iteracji.

Wpływ liczby iteracji:



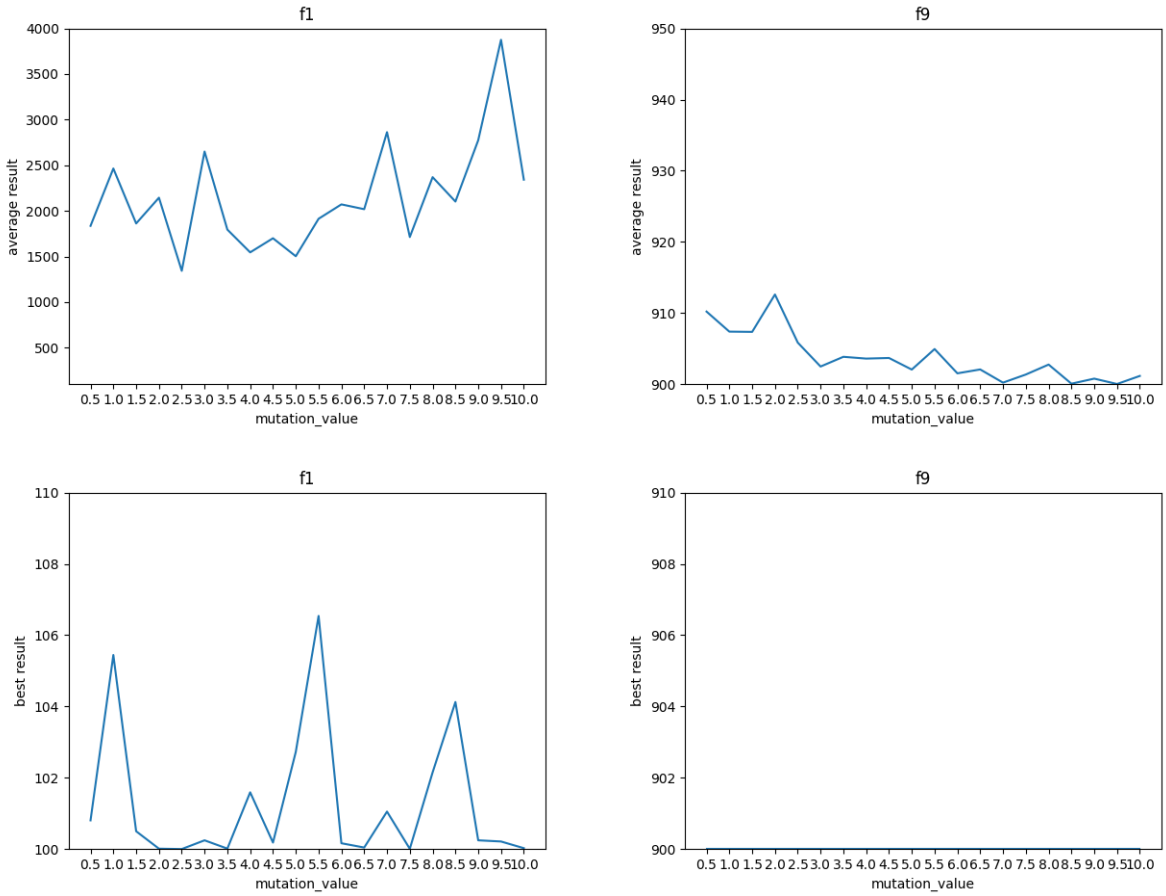






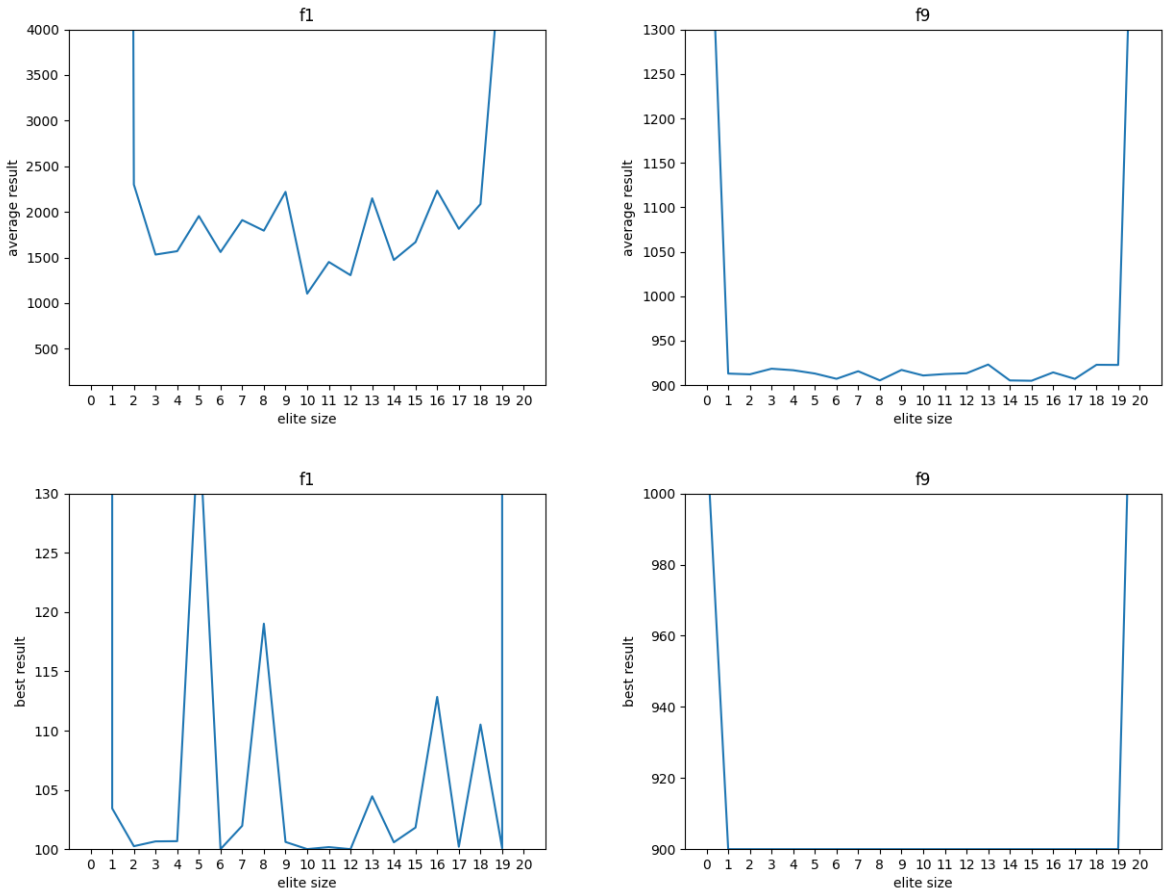
Liczba iteracji miała niewielki wpływ na poprawę jakości algorytmu. Zmienna wartość mutacji powodowała, że algorytm wykonywał to samo (eksplorację i eksploatację) więcej razy.

Wpływ odchylenia standardowego mutacji



Podobnie jak ilość iteracji, początkowy rozmiar kroku nie ma dużego wpływu na jakość algorytmu. Można jednak zauważyć, że początkowo powinien być większy. Dzięki temu algorytm początkowo bardziej skupi się na eksploracji obszaru poszukiwań i znajdzie więcej potencjalnie dobrych miejsc.

Wpływ rozmiaru elity:



Pomiary zostały wykonane dla populacji o rozmiarze 20. Dla funkcji f9 algorytm osiągał zadowalające efekty dla rozmiaru elity od 2 do 19, a dla f1 od 4 do 18. Można z tego wyciągnąć wniosek, że zbyt duże i zbyt małe rozmiary elit nie są dobre dla algorytmu. Zbyt mała elita skutkuje małym wpływem mutacji na populację, a przez zbyt dużą wielkość elity do następnej iteracji przechodzi zbyt mało osobników z obecnie najlepszej populacji.

5. Wnioski końcowe.

Algorytm genetyczny jest algorytmem losowym. Jakość rozwiązania, które dzięki niemu uzyskamy zależy od tego w jakim kierunku przebiegnie mutacja populacji. Średnie wyniki, są zazwyczaj znacznie słabsze od najlepszych rezultatów. Dlatego też, aby mieć większe szanse znalezienia optimum globalnego należy wywołać algorytm wiele razy. Na przykład: wywołanie tego samego algorytmu 10 razy powinno dać lepszą jakość najlepszego rozwiązania niż jednokrotne wywołanie algorytmu dla 10 razy większej populacji lub liczby iteracji.

W mojej implementacji zastosowałem zasadę 1/5 sukcesów, dzięki której odchylenie standardowe mutacji dostosowuje się do działania algorytmu. Pozwala to na osiągnięcie lepszych rezultatów kosztem mniejszej kontroli nad algorytmem. Parametry takie jak rozmiar elity lub początkowe odchylenie standardowe mutacji, które pozwalałyby sterować algorytmem w klasycznej wersji, tutaj stają się mniej znaczące. W szczególności drugi z nich, który jak zaobserwowałem przyjmuje w czasie działania algorytmu wartości z zakresu (0+ε, 10000). Być może wprowadzenie dodatkowych warunków zmiany odchylenia standardowego pozwoliłoby na usprawnienie algorytmu.