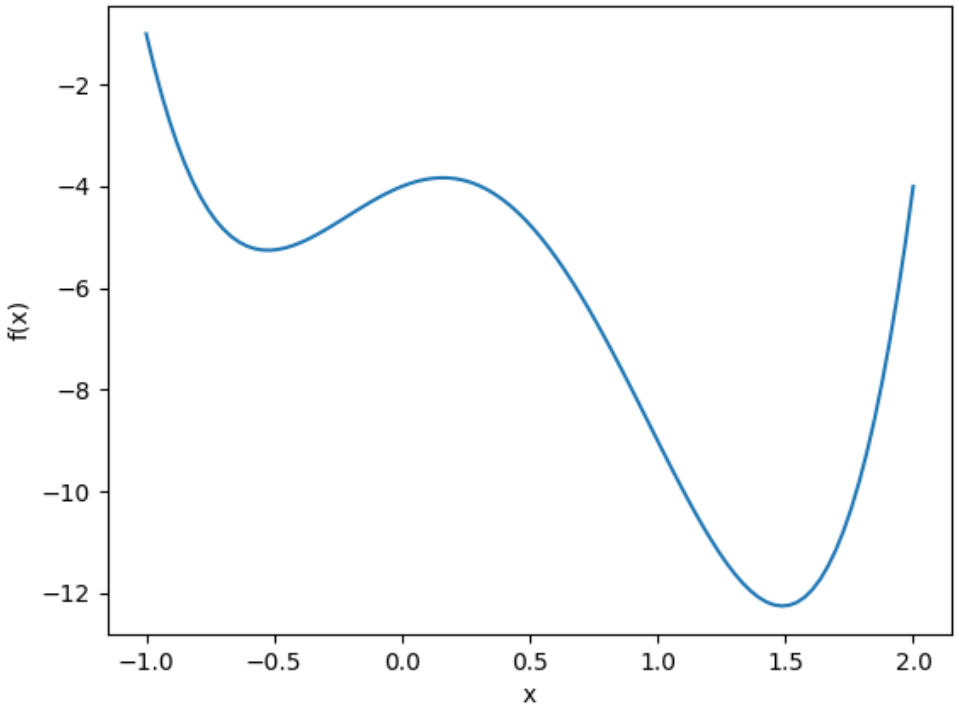
Porównanie szukania minimum f-cji metodą genetyczną i pso

Funkcja

$f(x) = 4x^2 - 6x^3 - 5x^2 + 2x - 4$

Minimum będziemy szukać na przedziale [-1,2]



Algorytm genetyczny

Kroki

1. wybór początkowej populacji łańcuchów
2. ocena jakości łańcuchów
3. selekcja łańcuchów (rodziców)
4. krzyżowanie łańcuchów
5. mutacja

Wybór początkowej populacji

Początkową populacje wybieramy losowo. Dla demonstracji działania tego algorytmu wybiorę 8 losowych punktów z przedziału na którym szukamy minimum.

[1.08940756 -0.141582 -0.31944564 0.65394431 1.15840691 0.26931938
  1.9422926   1.05448922   -1.   2.   ]

Dodatkowo dodaje ręcznie dwa punkty. Są to krańce przedziału, na którym szukam minimum. Jest to nazywane "inicjalizacją populacji z pełnym zakresem". Poprzez dodanie tych dodatkowych punktów, gwarantuje, że populacja obejmuje cały zakres przedziału, co może pomóc uniknąć sytuacji, w której minimum nie jest pomiędzy żadnymi dwoma punktami.

Rozmiar populacji jest równy 10. Oczywiście w rzeczywistości rozmiar populacji powinien być znacznie większy (np. 100), ale dla usprawnienia obliczeń rozmiar populacji ograniczyłam.

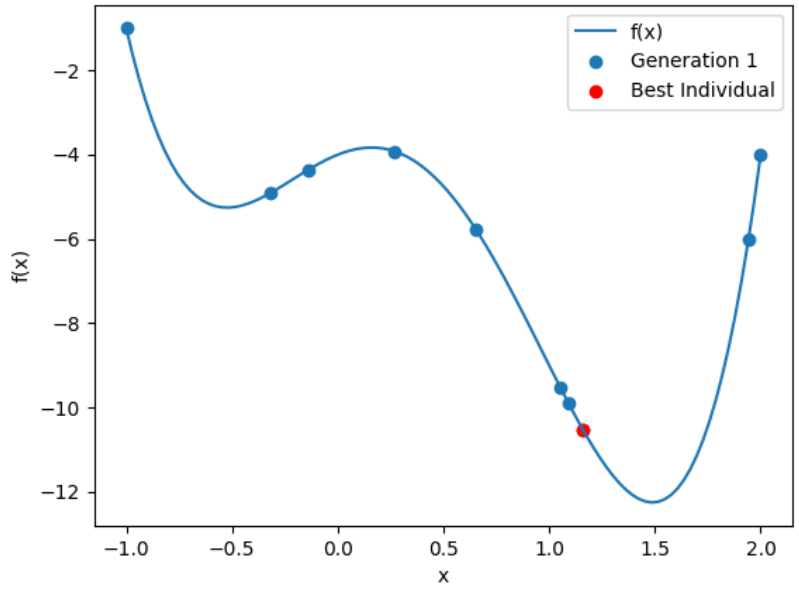
Ocena jakości łańcuchów

Każdy łańcuch podajemy na wejściu funkcji którą minimalizujemy. W rezultacie otrzymujemy takie wartości:

[-9.87867928 -4.36475556 -4.91187774 -5.77674134 -10.51670911
  -3.92018892 -6.0146756   -9.54027   -1.   -4.   ]

Najlepszym osobnikiem z tej populacji będzie 5 osobnik:
1.15840691, który przyjmuje wartość -10.51670911

Ten punkt i pozostałe zostały pokazane na wykresie:



Selekcja łańcuchów

Następnym krokiem jest ruletkowa selekcja łańcuchów. W tym celu należy każdemu łańcuchowi przypisać prawdopodobieństwo wylosowania na podstawie dopasowania.

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$$

Sumą wszystkich wartości dopasowania łańcuchów które obliczamy będzie:
-59.92389753850562

i prawdopodobieństwa wyglądają następująco:
[0.1648537508897665, 0.07283831218908537, 0.08196859586118253, 0.09640129526129586, 0.1755010862840758, 0.065419458362203, 0.10037190243329196, 0.1592064332878598, 0.01668783308624785, 0.0667513323449914]

Jak widać dla naszego najlepszego łańcucha prawdopodobieństwo jest największe (około 0.18)

Teraz 6 razy losujemy dwóch rodziców których krzyżujemy poprzez policzenie średniej z nich. Dziecko które uzyskamy poprzez skrzyżowanie tych rodziców dodajemy do nowej populacji.

numer losowania: 1
rodzice: [1.089407556793585, 1.089407556793585]
dziecko: 1.089407556793585
numer losowania: 2
rodzice: [0.6539443072486737, 1.089407556793585]
dziecko: 0.8716759320211294
numer losowania: 3
rodzice: [1.05448921575459, 1.089407556793585]
dziecko: 1.0719483862740875
numer losowania: 4
rodzice: [1.1584069093566893, 0.6539443072486737]
dziecko: 0.9061756083026815
numer losowania: 5
rodzice: [1.05448921575459, 1.089407556793585]
dziecko: 1.0719483862740875
numer losowania: 6
rodzice: [0.6539443072486737, 0.6539443072486737]
dziecko: 0.6539443072486737

Z naszej początkowej populacji najbliższymi do minimum rozwiązaniami były:

1.05448922
1.15840691
1.08940756

i te rozwiązania miały wysokie prawdopodobieństwo wylosowania co ma swoje odwzorowanie w rzeczywistości bo zostały wylosowane aż 8/12 razy.

Mutacja

Następnie każde z dzieci może być poddane mutacji. Jest na to 10% szans. Mutacja w tym przypadku oznacza dodanie do dziecka losowej wartości z zakresu [-0.1; 0.1].

W tej generacji, 6 dziecko zostało poddane mutacji i jego nowa wartość to:
0.63355315831476

Aktualizacja populacji

Do początkowej populacji dodajemy dzieci, które zostały wygenerowane w poprzednim kroku. W tym momencie mam 16 łańcuchów w populacji. W wyniku krzyżowania powstało kilka duplikujących się rozwiązań. Usuwam je w celu zachowania różnorodności rozwiązań. Zwiększy to efektywność algorytmu genetycznego:

nowa populacja (poczatkowa+dzieci): [1.08940756; -0.141582; -0.31944564
0.65394431; 1.15840691; 0.26931938; 1.9422926; 1.05448922; -1.0 ; 2.0 ; 1.08940756;
0.87167593; 1.07194839; 0.90617561; 1.07194839; 0.63355316]
nowa populacja (bez powtórzeń): [-1.0; -0.31944564; -0.141582; 0.26931938
0.63355316; 0.65394431 ; 0.87167593; 0.90617561; 1.05448922l 1.07194839
1.08940756; 1.15840691; 1.9422926 ; 2.0]

Jak widać dwie wartości zostały usunięte więc nasza aktualna populacja składa się z 14 łańcuchów. To dalej za dużo - trzeba odrzucić jeszcze 4 łańcuchy. Mamy dwa warianty ograniczenia populacji:

1. **Selekcja elitarystyczna**: Wybierasz 10 najlepszych osobników na podstawie ich dopasowania i usuwasz resztę.
2. **Ruletka proporcjonalna**: Przeprowadzasz selekcję ruletkową na całej populacji, ale z prawdopodobieństwem proporcjonalnym do ich dopasowania. W ten sposób lepiej dopasowane osobniki będą miały większą szansę na przetrwanie.

W tym przypadku (losowanie 10 z 14 liczb) prawdopodobnie obie metody dadzą podobny rezultat, ale zastosuje pierwszą metodę. Moją decyzje argumentuje tym, że jeżeli uzyskaliśmy już, w którejś generacji najmniejsze rozwiązanie to stosując drugą metodę jest mała szansa, że zostanie ona na tym etapie usunięta i z jakimś prawdopodobieństwem nie

zostanie ona już nigdy osiągnięta. Oczywiście dla takiej ilości danych jest to prawie niemożliwe, ale w tej metodzie zabezpieczamy się przed taka sytuacją i możliwe jest tylko zbliżenie się do rozwiązania a nie oddalenie.

Najmniejsze wartości dopasowania wychodzą dla x-ów leżących na następujących pozycjach (licząc od 0):
[11 10 9 8 7 6 12 5 4 1]

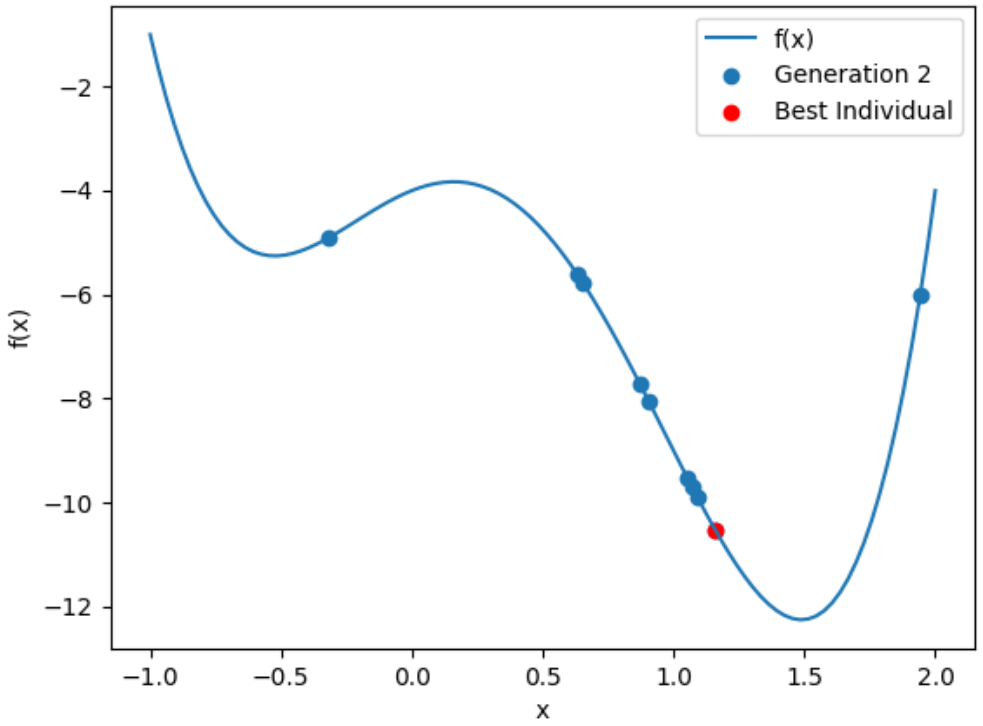
i są to:
[1.15840691 1.08940756 1.07194839 1.05448922 0.90617561 0.87167593
1.9422926 0.65394431 0.63355316 -0.31944564]

I jest to nowa populacja dla naszego algorytmu. Teraz należałoby powtórzyć te kroki wiele razy, ale ja pokaże tu jeszcze dwa takie przejścia tego algorytmu.

Ocena jakości:

[-10.51670911 -9.87867928 -9.71047565 -9.54027 -8.06090247
-7.72033881 -6.0146756 -5.77674134 -5.62119716 -4.91187774]

Wykres



Selekcja łańcuchów

Suma: -77.75186714208438
Wagi: [**0.1352598914848575**, 0.12705391703461372, 0.12489057823399283, 0.12270149060701882, 0.10367471248096706, 0.09929457760910732, 0.07735731395576913, 0.07429713976206716, 0.07229661953416495, 0.06317375929744136]
Największa pogrubiona

Krzyżowanie

numer losowania: 1
rodzice: [1.089407556793585, 1.1584069093566893]
dziecko: 1.1239072330751372
numer losowania: 2
rodzice: [1.05448921575459, 1.1584069093566893]
dziecko: 1.1064480625556397
numer losowania: 3
rodzice: [0.9061756083026815, 1.1584069093566893]
dziecko: 1.0322912588296855
numer losowania: 4
rodzice: [1.0719483862740875, 1.05448921575459]
dziecko: 1.0632188010143389
numer losowania: 5
rodzice: [0.63355315831476, 1.1584069093566893]
dziecko: 0.8959800338357247
numer losowania: 6
rodzice: [1.089407556793585, 1.9422925951538463]
dziecko: 1.5158500759737157

Mutacja

W tej iteracji nie nastąpiła mutacja

Aktualizacja populacji

nowa populacja (bez powtórzeń): [-0.31944564 0.63355316 0.65394431 0.87167593 0.89598003 0.90617561 1.03229126 1.05448922 1.0632188 1.07194839 1.08940756 1.10644806 1.12390723 1.15840691 1.51585008 1.9422926]

Wartości dopasowania: [-4.91187774 -5.62119716 -5.77674134 -7.72033881 -7.959767 -8.06090247 -9.3215288 -9.54027 -9.62560089 -9.71047565 -9.87867928 -10.04057402 -10.20375299 -10.51670911 -12.23647609 -6.0146756]

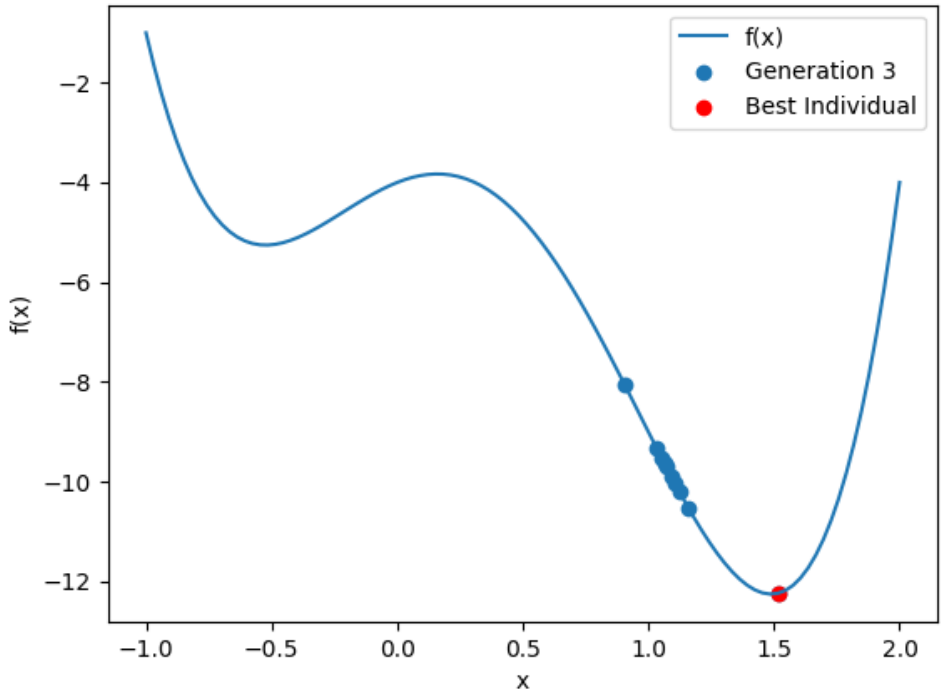
indeksy osobników z najniższą wartością funkcji dopasowania**:** [14 13 12 11 10 9 8 7 6 5]

nowa populacja: [1.51585008 1.15840691 1.12390723 1.10644806 1.08940756
1.07194839 1.0632188 1.05448922 1.03229126 0.90617561]

Ocena jakości

Wartości dopasowania: [-12.23647609 -10.51670911 -10.20375299 -10.04057402
-9.87867928 -9.71047565 -9.62560089 -9.54027 -9.3215288 -8.06090247]

Wykres



Selekcja łańcuchów

Suma: -99.13496929100641
Wagi: [**0.12343248980935845**, 0.10608475684813143, 0.10292788772829929,
0.10128185938454146, 0.09964878536612305, 0.09795207196405967,
0.09709591839464395, 0.09623516367678081, 0.094028664872789,
0.0813124019552729]

Krzyżowanie

numer losowania: 1
rodzice: [1.5158500759737157, 1.0322912588296855]
dziecko: 1.2740706674017006
numer losowania: 2
rodzice: [1.0719483862740875, 1.1239072330751372]

dziecko: 1.0979278096746123
numer losowania: 3
rodzice: [1.0322912588296855, 1.05448921575459]
dziecko: 1.0433902372921378
numer losowania: 4
rodzice: [1.1064480625556397, 1.05448921575459]
dziecko: 1.0804686391551148
numer losowania: 5
rodzice: [1.1584069093566893, 1.0719483862740875]
dziecko: 1.1151776478153885
numer losowania: 6
rodzice: [1.089407556793585, 1.05448921575459]
dziecko: 1.0719483862740875

Znowu brak mutacji.

Aktualizacja populacji

nowa populacja (bez powtórzeń): [0.90617561 1.03229126 1.04339024 1.05448922 1.0632188 1.07194839
1.08046864 1.08940756 1.09792781 1.10644806 1.11517765 1.12390723
1.15840691 1.27407067 1.51585008]
Wartości dopasowania: [-8.06090247 -9.3215288 -9.43118857 -9.54027 -9.62560089 -9.71047565 -9.79283297 -9.87867928 -9.95992924 -10.04057402
-10.12252731 -10.20375299 -10.51670911 -11.43715561 -12.23647609]
indeksy osobników z najniższą wartością funkcji dopasowania: [14 13 12 11 10 9 8 7 6 5]
nowa populacja (bez osobników z najwyższą wartością funkcji dopasowania): [1.51585008 1.27407067 1.15840691 1.12390723 1.11517765 1.10644806
1.09792781 1.08940756 1.08046864 1.07194839]

Wynik

W rzeczywistości należałoby przeprowadzić tu jeszcze dużo iteracji i początkowych łańcuchów powinno być więcej, ale po takiej liczbie iteracji otrzymujemy następujący wynik:

Najlepszy osobnik: 1.5158500759737157, wartosc: -12.236476086763211

Wynik ten uzyskaliśmy już w drugiej generacji, co potwierdza właściwość mojej argumentacji wyboru metody ograniczenia populacji.

Prawdziwe minimum tej funkcji leży w punkcie:
(1.4884738753768, -12.2528678203824)

Co oznacza, że nasz algorytm nie pomylił się bardzo bo tylko o około: **0.02738.** Co dla tak małej ilości punktów i generacji jest bardzo dobrym wynikiem.

Warto zauważyć, że w przypadku większej ilości generacji jest możliwe ze poprzez mutacje jeszcze bardziej zbliżylibyśmy się do prawdziwego rozwiązania.

Inną ważną rzeczą jest to, że mogłoby to się potoczyć dużo inaczej i w trzech generacjach nawet nie zbliżylibyśmy się do rozwiązania, ze względu na fakt, że początkowa populacja i rodzice do krzyżowania są losowani.

Algorytm PSO

Wstęp

Jednostka wybiera algorytm swojego postępowania jako wypadkową swojego dotychczas najlepszego postępowania, postępowania jednostki w populacji odnoszącej największe sukcesy oraz pewnej swojej dynamiki zmian.
W każdym kroku cząstka przemieszcza się w kierunku określonym przez:
1. wektor wskazujący najlepsze dotychczasowe położenie cząstki
2. wektor wskazujący najlepsze dotychczasowe położenie cząstek sąsiednich
3. chwilowy wektor swojej prędkości.

Minimalizować będę tą samą funkcję.
$f(x) = 4x^2 - 6x^3 - 5x^2 + 2x - 4$

Kroki

1. Inicjalizacja roju cząstek: Tworzona jest populacja cząstek, gdzie każda cząstka reprezentuje potencjalne rozwiązanie. Cząstki są losowo inicjalizowane w przestrzeni poszukiwań.
2. Inicjalizacja prędkości i pamięci cząstek: Dla każdej cząstki inicjalizowana jest prędkość oraz pamięć, która przechowuje informacje o najlepszym dotychczas znalezionym rozwiązaniu przez daną cząstkę.
3. Obliczanie funkcji celu: Dla każdej cząstki obliczane jest wartość funkcji celu na podstawie jej aktualnej pozycji.
4. Aktualizacja pamięci cząstek: Jeśli aktualna wartość funkcji celu jest lepsza od najlepszej dotychczas znalezionej wartości przez daną cząstkę, to jej pamięć jest aktualizowana.
5. Aktualizacja prędkości i pozycji: Prędkość i pozycja każdej cząstki są aktualizowane na podstawie jej aktualnej prędkości, najlepszej pamięci cząstki oraz najlepszej pamięci spośród wszystkich cząstek w roju.
6. Powtarzanie kroków 3-5: Kroki 3-5 są powtarzane przez określoną liczbę iteracji lub do spełnienia warunku zakończenia, np. osiągnięcia zadowalającego minimum funkcji.
7. Zwrócenie najlepszego rozwiązania: Po zakończeniu iteracji algorytm zwraca najlepsze znalezione rozwiązanie (najlepszą pozycję) spośród wszystkich cząstek.

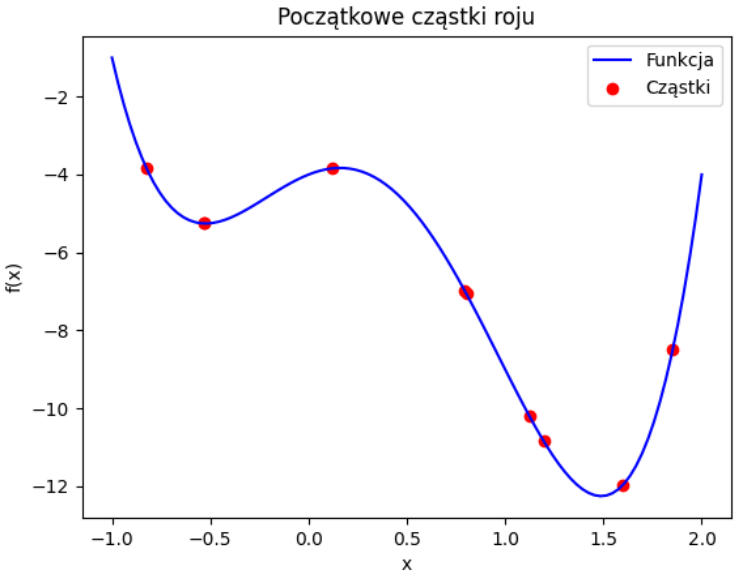
Parametry

Liczba cząstek = 10
Liczba iteracji = 3
Współczynnik wagowy = 0.5
Współczynnik przyśpieszenia cząstki (dla najlepszej pamięci) = 1
Współczynnik przyśpieszenia cząstki (dla najlepszej pamięci społecznej) = 2

Inicjalizacja roju cząstek

Początkowy rój cząstek inicjalizowany jest losowo. Jest to 10 punktów x z zakresu poszukiwań czyli (-1, 2)

Początkowe pozycje cząstek:
[0.12362036]
[1.85214292]
[1.19598183]
[0.79597545]
[-0.53194408]
[-0.53201644]
[-0.82574916]
[1.59852844]
[0.80334504]
[1.12421773]



Inicjalizacja prędkości i pamięci cząstek

Na początku prędkości dla cząstek są równe 0. Najlepsze pozycje dla cząstek to jedyne jakie na razie mamy czyli początkowe.

Obliczenie funkcji celu

[-3.83957008]
[-8.49830558]
[-10.840234]
[-6.9961249]
[-5.25530728]
[-5.25529403]
[-3.82278505]
[-11.96956905]
[-7.06484744]
[-10.20662809]

Jako, że już wiemy z poprzedniego algorytmu, jaki powinien być wynik działania algorytmu widać, że jedno rozwiązanie jest dość blisko prawidłowego. Zostało ono pogrubione w obu listach.

Aktualizacja pamięci cząstek

Położenie cząstki w kroku iteracyjnym zmienia się według formuły:

$$w_i(t+1) = w_i(t) + v_i(t+1)$$

gdzie:

- $w_i(t)$ oznacza pozycję cząstki i w kroku t
- $v_i(t+1)$ oznacza prędkość cząstki obliczaną według wzoru:

$$v_i(t+1) = v_i(t) + c_1 r_1(t) [\, y_i(t) - w_i(t) \,] + c_2 r_2(t) [\, \hat{y}(t) - w_i(t) \,]$$

gdzie:

- c1 i c2 - wyżej opisane współczynniki (1 i 2)
- r1, r2 - losowe wartości z przedziału [0, 1]
- $y_i(t)$ - najlepsza dotychczasowa pozycja cząstki spośród wszystkich osiąganych
- $\hat{y}(t)$ - najlepsza spośród wszystkich najlepszych pozycji osobistych wszystkich cząstek

A więc dla każdej cząstki najpierw wyznaczamy dwie wartości losowe **r1 i r2:**

r1: [[0.02058449]
 [0.96990985]
 [0.83244264]
 [0.21233911]
 [0.18182497]
 [0.18340451]
 [0.30424224]
 [0.52475643]
 [0.43194502]
 [0.29122914]]

r2: [[0.61185289]
 [0.13949386]
 [0.29214465]
 [0.36636184]
 [0.45606998]
 [0.78517596]
 [0.19967378]
 [0.51423444]
 [0.59241457]
 [0.04645041]]

oraz wyznaczamy zgodnie z wzorem **wektor prędkości i aktualizujemy pozycje:**

[1.80485356]
 [-0.07075533]
 [0.23520368]
 [0.46543345]
 [1.79064895]
 [3.08291794]
 [0.90130136]
 [-0.172107]
 [0.74388368]
 [0.02851756]

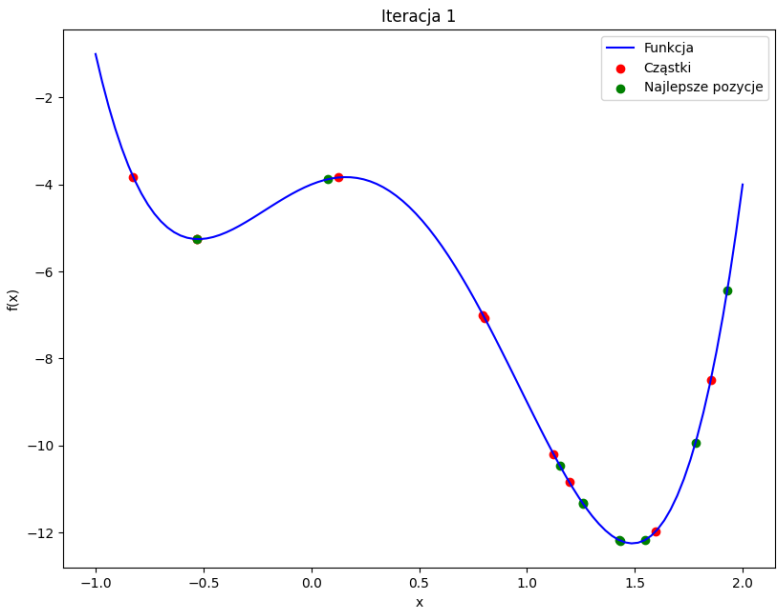
[1.92847391]
 [1.78138759]
 [1.4311855]
 [1.2614089]
 [1.25870488]
 [2.]
 [0.0755522]
 [1.42642144]
 [1.54722871]
 [1.15273529]

Należy tu zauważyć, że wartości mogą zostać zmniejszone (lub zwiększone) jeśli wychodzą poza zakres poszukiwań i tak stało się z 6 cząstką.

Wszystkie cząstki są dodatnie co sugeruje, że algorytm słusznie decyduje na zwiększenie wartości ujemnych i zbliża się do naszego rozwiązania znajdującego się bliżej prawego krańca zakresu.

Następnie obliczamy wartość funkcji celu dla nowej pozycji cząstki i sprawdzamy czy jest ona lepszym rozwiązaniem niż to, które obecnie uważamy za najlepsze. W pierwszej iteracji sprawdzamy po prostu czy nowa pozycja danej cząstki jest bliżej rozwiązania niż początkowa.

Cząstka: 1, Pozycja: [1.92847391], Wartość: [-6.4460217]
Zaktualizowano najlepszą pozycję cząstki
Cząstka: 2, Pozycja: [1.78138759], Wartość: [-9.94125179]
Zaktualizowano najlepszą pozycję cząstki
Cząstka: 3, Pozycja: [1.4311855], Wartość: [-12.18600358]
Zaktualizowano najlepszą pozycję cząstki
Cząstka: 4, Pozycja: [1.2614089], Wartość: [-11.34844316]
Zaktualizowano najlepszą pozycję cząstki
Cząstka: 5, Pozycja: [1.25870488], Wartość: [-11.32905645]
Zaktualizowano najlepszą pozycję cząstki
Cząstka: 6, Pozycja: [2.], Wartość: [-4.]
Nie zaktualizowano najlepszej pozycji
Cząstka: 7, Pozycja: [0.0755522], Wartość: [-3.87989352]
Zaktualizowano najlepszą pozycję cząstki
Cząstka: 8, Pozycja: [1.42642144], Wartość: [-12.17473846]
Zaktualizowano najlepszą pozycję cząstki
Cząstka: 9, Pozycja: [1.54722871], Wartość: [-12.17539738]
Zaktualizowano najlepszą pozycję cząstki
Cząstka: 10, Pozycja: [1.15273529], Wartość: [-10.46621786]
Zaktualizowano najlepszą pozycję cząstki



Jak widać prawie każdą cząstkę należało podmienić. Dla pierwszej iteracji jest to dość normalne ponieważ obecne najlepsze rozwiązanie jest losowe. Wyjątek stanowi cząstka 6, która tak jak wcześniej napisałam wyszła poza zakres.

Wykres pokazujący początkowe położenie cząstek oraz aktualnie najlepsze pozycje cząstek.

Pierwszą iteracje można uznać za zakończoną

Najlepsze pozycje i wartości po pierwszej iteracji

Najlepsze pozycje: [[1.92847391]
 [1.78138759]
 [1.4311855]
 [1.2614089]
 [1.25870488]
 [-0.53201644]
 [0.0755522]
 [1.42642144]
 [1.54722871]
 [1.15273529]]

Najlepsze wartości: [[-6.4460217]
 [-9.94125179]
 [-12.18600358]
 [-11.34844316]
 [-11.32905645]
 [-5.25529403]
 [-3.87989352]
 [-12.17473846]
 [-12.17539738]
 [-10.46621786]]

8 i 9 pozycja są bardzo blisko minimum co widać również na wykresie.

Druga iteracja

Wylosowane wektory r1 i r2:

r1: [0.60754485]
 [0.17052412]
 [0.06505159]
 [0.94888554]
 [0.96563203]
 [0.80839735]
 [0.30461377]
 [0.09767211]
 [0.68423303]
 [0.44015249]

r2: [0.12203823]
 [0.49517691]
 [0.03438852]
 [0.9093204]
 [0.25877998]
 [0.66252228]
 [0.31171108]
 [0.52006802]
 [0.54671028]
 [0.18485446]

Nowy wektor prędkości i nowe pozycje cząstek:

Prędkości: [0.78105038]
 [-0.38220164]
 [0.11760184]
 [0.54147938]
 [0.98459354]
 [-1.25912096]
 [1.29578251]
 [-0.08109822]
 [0.24505781]
 [0.1172043]

Pozycje: [2.   ]
 [1.39918595]
 [1.54878734]
 [1.80288828]
 [2.   ]
 [0.74087904]
 [1.37133471]
 [1.34532321]
 [1.79228652]
 [1.2699396]

Obliczamy wartość funkcji dla cząstek i podmieniamy w razie potrzeby:
Cząstka: 1, Pozycja: [2.], Wartość: [-4.]
Cząstka: 2, Pozycja: [1.39918595], Wartość: [-12.09484062]
Zaktualizowano najlepszą pozycję cząstki

Cząstka: 3, Pozycja: [1.54878734], Wartość: [-12.17112894]
Cząstka: 4, Pozycja: [1.80288828], Wartość: [-9.54640855]
Cząstka: 5, Pozycja: [2.], Wartość: [-4.]
Cząstka: 6, Pozycja: [0.74087904], Wartość: [-6.49759693]
Zaktualizowano najlepszą pozycję cząstki
Cząstka: 7, Pozycja: [1.37133471], Wartość: [-11.98737217]
Zaktualizowano najlepszą pozycję cząstki
Cząstka: 8, Pozycja: [1.34532321], Wartość: [-11.86531211]
Cząstka: 9, Pozycja: [1.79228652], Wartość: [-9.74570353]
Cząstka: 10, Pozycja: [1.2699396], Wartość: [-11.40859202]
Zaktualizowano najlepszą pozycję cząstki

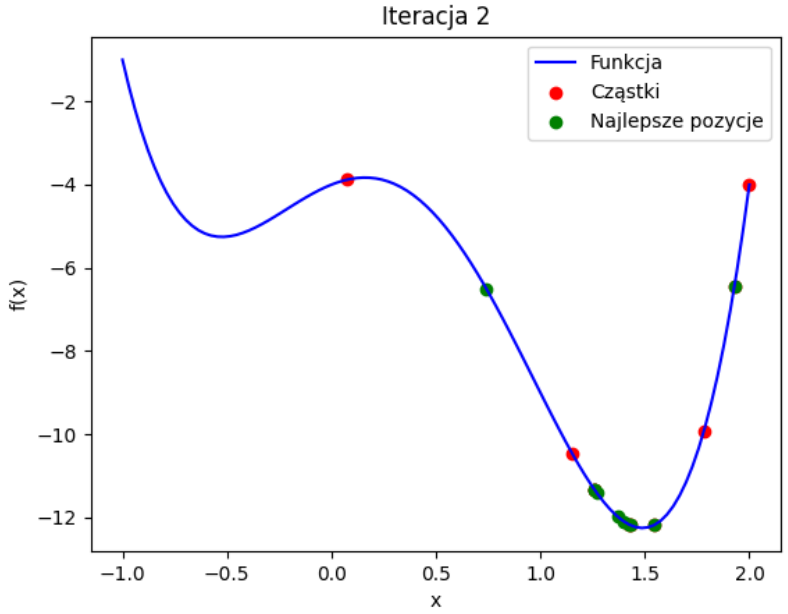
Jak widać w drugiej iteracji tylko 4 cząstki zmieniły najmniejszą wartość

Zaktualizowane wektory najlepszych rozwiązań dla cząstek i przypisanych im wartości wyglądają następująco:

Najlepsze pozycje: [[1.92847391]
[1.39918595]
[1.4311855]
[1.2614089]
[1.25870488]
[0.74087904]
[1.37133471]
[1.42642144]
[1.54722871]
[1.2699396]]

Najlepsze wartości: [[-6.4460217]
[-12.09484062]
[-12.18600358]
[-11.34844316]
[-11.32905645]
[-6.49759693]
[-11.98737217]
[-12.17473846]
[-12.17539738]
[-11.40859202]]

Najlepsze znalezione rozwiązanie zostało pogrubione, ale widać, że coraz więcej cząstek zbliża się do takiej wartości.



Widać to również na wykresie.

Iteracja 3

r1: [[0.96958463]
[0.77513282]
[0.93949894]
[0.89482735]
[0.59789998]
[0.92187424]
[0.0884925]
[0.19598286]
[0.04522729]
[0.32533033]]

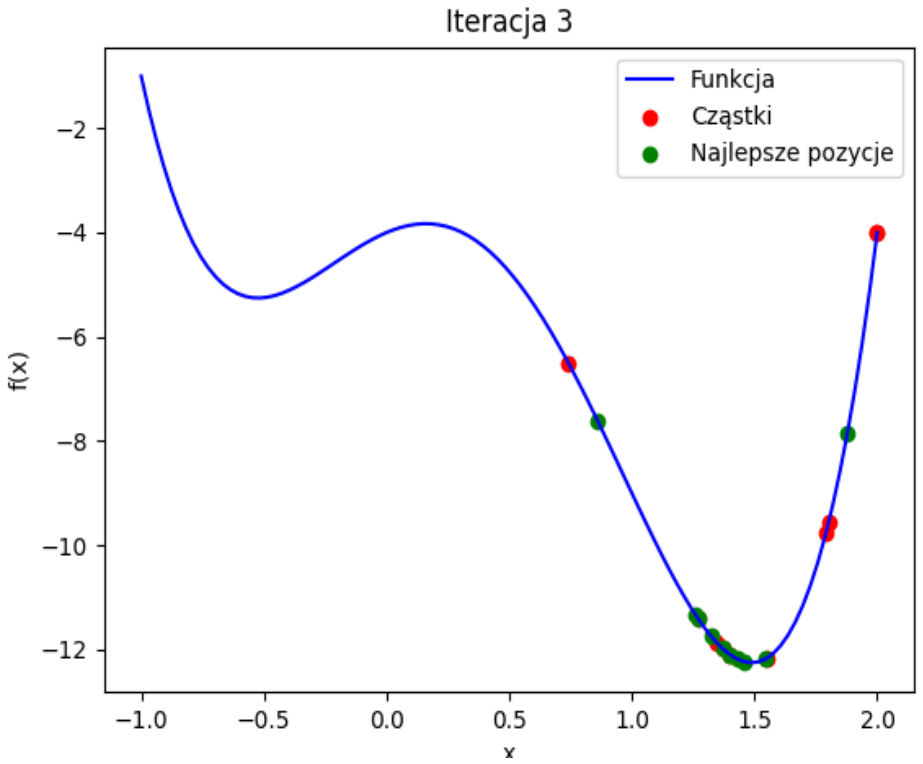
r2: [[0.38867729]
[0.27134903]
[0.82873751]
[0.35675333]
[0.28093451]
[0.54269608]
[0.14092422]
[0.80219698]
[0.07455064]
[0.98688694]]

Prędkości: [[-0.12099596]
[-0.17373473]
[-0.24660799]
[-0.47900327]
[-0.27052281]
[0.11969275]
[0.66476011]
[0.11310169]
[0.06166641]
[0.43062952]]

Pozycje: [[1.87900404]
[1.22545123]
[1.30217935]
[1.32388501]
[1.72947719]
[0.86057179]
[2.   ]
[1.4584249]
[1.85395293]
[1.70056912]]

Cząstka: 1, Pozycja: [1.87900404], Wartość: [-7.83784387]
Zaktualizowano najlepszą pozycję cząstki
Cząstka: 2, Pozycja: [1.22545123], Wartość: [-11.07875703]
Cząstka: 3, Pozycja: [1.30217935], Wartość: [-11.62120224]
Cząstka: 4, Pozycja: [1.32388501], Wartość: [-11.75017105]
Zaktualizowano najlepszą pozycję cząstki
Cząstka: 5, Pozycja: [1.72947719], Wartość: [-10.7481381]
Cząstka: 6, Pozycja: [0.86057179], Wartość: [-7.6118711]
Zaktualizowano najlepszą pozycję cząstki
Cząstka: 7, Pozycja: [2.], Wartość: [-4.]
Cząstka: 8, Pozycja: [1.4584249], Wartość: [-12.23404236]
Zaktualizowano najlepszą pozycję cząstki
Cząstka: 9, Pozycja: [1.85395293], Wartość: [-8.45582668]
Cząstka: 10, Pozycja: [1.70056912], Wartość: [-11.1129941]

Znowu tylko 4 cząstki zmieniły najlepsze pozycje. Warto zauważyć, że 8 cząstka, która była bardzo blisko rozwiązania zmieniła swoją wartość. niewiele bo z 1.42642144 na 1.4584249 ale przybliżyła się jeszcze bardziej do rozwiązania (dla przypomnienia: 1.48847387)



Najlepsze pozycje:
[[1.87900404]
[1.39918595]
[1.4311855]
[1.32388501]
[1.25870488]
[0.86057179]
[1.37133471]
[**1.4584249**]
[1.54722871]
[1.2699396]]
Najlepsze wartości: [[
-7.83784387]
[-12.09484062]
[-12.18600358]
[-11.75017105]
[-11.32905645]
[-7.6118711]
[-11.98737217]
[**-12.23404236**]
[-12.17539738]
[-11.40859202]]

Wynik

Cząstka 8, która zbliżyła się w tej iteracji do rozwiązania stała się najlepszą cząstką po tej iteracji. Tak jak poprzednio zwiększenie liczby iteracji prawdopodobnie dałoby nam lepszy wynik.

Prawdziwe minimum tej funkcji leży w punkcie:
(1.4884738753768, -12.2528678203824)

Minimum wyznaczone w trzech iteracjach:
(1.4584249, -12.23404236)

Błąd między rzeczywistym rozwiązaniem to około: **0.03005**
A więc niewiele więcej niż w przypadku algorytmu genetycznego. Oczywiście warto zaznaczyć, że jest to losowe i równie dobrze dla innego wylosowania początkowych cząstek mogłoby być lepiej lub gorzej.

W tym przypadku liczba cząstek może nie być dużym utrudnieniem ponieważ czasami 10 cząstek dla tak prostego problemu (prosta funkcja, mały przedział poszukiwań) może w zupełności wystarczyć.

Podsumowanie

PSO jest często bardziej skuteczny w osiąganiu szybkiej zbieżności do optymalnego rozwiązania. Dzięki dynamicznemu dostosowaniu prędkości cząstek, PSO może skoncentrować wysiłki na najlepszych obszarach przestrzeni poszukiwań, co przyspiesza proces optymalizacji.

Algorytm genetyczny z kolei często lepiej radzi sobie ze znalezieniem globalnego minimum niż algorytm PSO. Poprzez zastosowanie operatorów krzyżowania i mutacji, AG może eksplorować większą przestrzeń poszukiwań, co zwiększa szansę na znalezienie globalnego optimum.

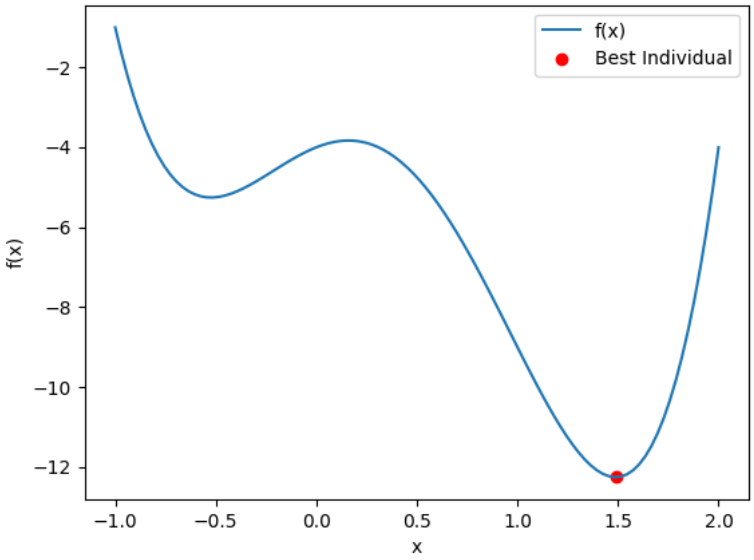
Algorytm genetyczny cechuje także dobra skalowalność: Algorytmy genetyczne są dobrze skalowalne dla problemów o większych rozmiarach przestrzeni poszukiwań i większych populacji. Można z łatwością zwiększać liczbę osobników i iteracji, co pozwala na lepsze eksplorowanie przestrzeni rozwiązań.

Na koniec warto porównać algorytmy w pełnej okazałości.

Algorytm genetyczny

Dla tego algorytmu zastosujemy nowe parametry:

Rozmiar populacji	100
Liczba dzieci	60
Liczba generacji	100



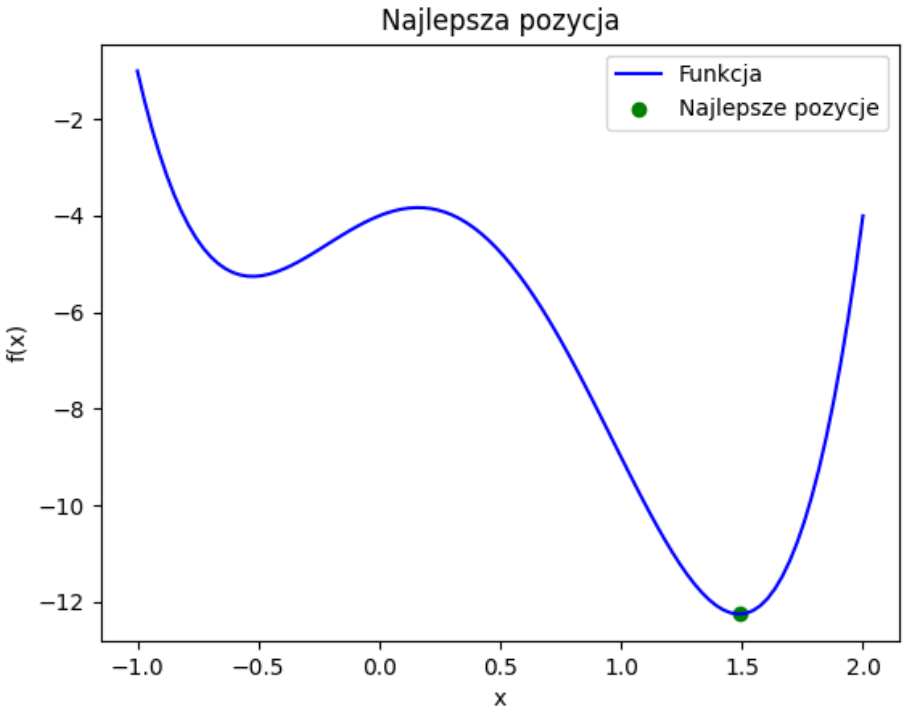
Najlepszy osobnik: 1.4884738785220617, wartosc: -12.252867820382367
Prawdziwe minimum: (1.4884738753768, -12.2528678203824)
Jak widać wartość osiągnięta przez ten algorytm jest bardzo zbliżona do rzeczywistej. Różni się dopiero na 8 miejscu po przecinku. Błąd jest więc pomijalny.

PSO

Parametry:

Liczba cząstek	50
Liczba iteracji	100

Najlepsza pozycja: [1.48847388]. Wartość: -12.252867820382367



Podobnie znalezione minimum jest bardzo dokładne. Warto tu jednak zauważyć ze taka liczba cząstek i iteracji może być zbędna. Podobny efekt uzyskamy przy mniejszej liczbie cząstek lub iteracji (lub obu).

Przykładowe wyniki dla różnych parametrów:
Liczba cząstek: 10 **Liczba iteracji**: 100
Wynik: Najlepsza pozycja: [1.48847387]. Wartość: -12.252867820382367

Liczba cząstek: 50 **Liczba iteracji**: 10
Wynik: Najlepsza pozycja: [1.48861431]. Wartość: -12.252867398642314

Liczba cząstek: 10 **Liczba iteracji**: 10
Wynik: Najlepsza pozycja: [1.48825454]. Wartość: -12.252866791981512

Jak widać dla tego problemu tak małe parametry jak 10 i 10 mogą w zupełności wystarczyć, ale trzeba pamiętać, że dla bardziej złożonych problemów parametry trzeba odpowiednio dostosować, najlepiej metodą prób i błędów.