UNIWERSYTET WARMIŃSKO – MAZURSKI W OLSZTYNIE
WYDZIAŁ MATEMATYKI I INFORMATYKI

SYMULACJE KOMPUTEROWE

Metoda gradientu prostego vs. algorytm genetyczny

6 STYCZNIA 2024

INŻ. ADAM ZALEWSKI
155461

INFORMACJE WSTĘPNE

STRUKTURA PROGRAMU

Program składa się z 6 skryptów napisanych w Pythonie. Są to:

1. GeneticOptimizer.py
2. SymbolicGeneticOptimizer.py
3. GradientDescentOptimizer.py
4. SymbolicGradientDescentOptimizer.py
5. VisualizeOptimizer.py

Skrypty numer 1 oraz 3 zawierają kod, który rozwiązuje przykład podany w treści zadania.

Skrypty numer 2 oraz 4 zawierają kod, który rozwiązuje każdy możliwy problem (w granicach możliwości obliczeniowych i zaimplementowanych w użytych bibliotekach) który jest tożsamy z podanym w treści zadania.

Skrypt o numerze 5 służy do wizualizacji wyników działania optymalizatorów z punktów 1 oraz 3.

URUCHOMIENIE PROGRAMU

1. Zainstaluj język Python w wersji 3.11 lub nowszej. Link do strony: Python
2. Pobierz projekt z repozytorium dostępnego pod linkiem: Github
3. Aby zainstalować wymagane biblioteki należy z poziomu konsoli w folderze głównym projektu wywołać komendę
   `pip install -r requirments.txt`
4. Następnie uruchom wybrany skrypt (najlepiej w środowisku PyCharm)
5. Jeżeli chcesz dokonać zmian rób je pod sekcją `if __name__ == „__main__”`

TEORETYCZNE ROZWIĄZANIE PROBLEMU[1]

Definicja ekstremum (maksimum):

Powiedzenie, że funkcja f ma w punkcie P_0 maksimum oznacza, że P_0 jest punktem wewnętrznym dziedziny funkcji i że istnieje otoczenie punktu P_0, w którym największą wartością funkcji jest $f(P_0)$, co zapisujemy symbolicznie.

$$\bigvee_{\boldsymbol{\delta>0}} \bigwedge_{\boldsymbol{PP_0<\delta}} \boldsymbol{f(P) \leq f(P_0)}$$

Powiedzenie, że funkcja f ma w punkcie P_0 maksimum właściwe oznacza, że P_0 jest punktem wewnętrznym dziedziny funkcji i że istnieje sąsiedztwo punktu P_0 takie, że we wszystkich punktach tego sąsiedztwa funkcja f przybiera wartości mniejsze niż w punkcie P_0, co zapisujemy symbolicznie.

$$\bigvee_{\boldsymbol{\delta>0}} \bigwedge_{\boldsymbol{0<PP_0<\delta}} \boldsymbol{f(P) < f(P_0)}$$

Definicja ekstremum (minimum):

Powiedzenie, że funkcja f ma w punkcie P_0 minimum oznacza, że P_0 jest punktem wewnętrznym dziedziny funkcji i że istnieje otoczenie punktu P_0, w którym najmniejszą wartością funkcji jest $f(P_0)$, co zapisujemy symbolicznie.

$$\bigvee_{\boldsymbol{\delta>0}} \bigwedge_{\boldsymbol{PP_0<\delta}} \boldsymbol{f(P) \geq f(P_0)}$$

[1] Wszelkie definicje pochodzą z książki Roman Leitner „Zarys matematyki wyższej dla studentów część 1” – Rozdział 9 Funkcje dwóch zmiennych. Wydanie XIII – 1 dodruk (PWN) Warszawa 2020

Powiedzenie, że funkcja f ma w punkcie P_0 minimum właściwe oznacza, że P_0 jest punktem wewnętrznym dziedziny funkcji i że istnieje sąsiedztwo punktu P_0 takie, że we wszystkich punktach tego sąsiedztwa funkcja f przybiera wartości większe niż w punkcie P_0, co zapisujemy symbolicznie.

$$\bigvee_{\boldsymbol{\delta>0}} \bigwedge_{\boldsymbol{0<PP_0<\delta}} \boldsymbol{f(P) > f(P_0)}$$

Ekstremum jest lokalną własnością funkcji, charakteryzującą rozkład wartości funkcji w dowolnie małym otoczeniu danego punktu. Natomiast wartość największa i wartość najmniejsza funkcji w danym zbiorze są związane z przebiegiem funkcji w całym zbiorze.

Warunek konieczny ekstremum funkcji dwóch zmiennych

Jeśli funkcja $f(x, y)$ ma w punkcie (x_0, y_0) ekstremum i jest w tym punkcie różniczkowalna, to obie pochodne cząstkowe I rzędu w tym punkcie są równe zeru.

$$\boldsymbol{f_x(x_0, y_0) = f_y(x_0, y_0) = 0}$$

Warunek wystarczający ekstremum funkcji dwóch zmiennych

Jeśli funkcja $f(x, y)$ jest klasy C^2 w otoczeniu punktu $P = (x_0, y_0)$ i ma obie pochodne cząstkowe cząstkowe I rzędu w tym punkcie równe zeru,

$$\boldsymbol{f_x(P_0) = f_y(P_0) = 0}$$

a wyznacznik pochodnych II rzędu funkcji f jest w tym punkcie dodatni

$$\boldsymbol{W(P_0) = \begin{vmatrix} f_{xx}(P_0) & f_{xy}(P_0) \\ f_{xy}(P_0) & f_{yy}(P_0) \end{vmatrix} > 0}$$

to funkcja ma w punkcie P_0 ekstremum właściwe. Charakter tego ekstremum zależy od znaku drugich pochodnych czystych w punkcie P_0.

$$\boldsymbol{f_{xx}(P_0), \quad f_{yy}(P_0)}$$

Jeśli są one dodatnie to funkcja ma w punkcie P_0 minimum właściwe, a jeśli ujemne, to maksimum właściwe.

Przykład:

Mamy wyznaczyć ekstremum funkcji dwóch zmiennych daną wzorem:

$$f(x, y) = x^4 + y^4 - 4xy + x$$

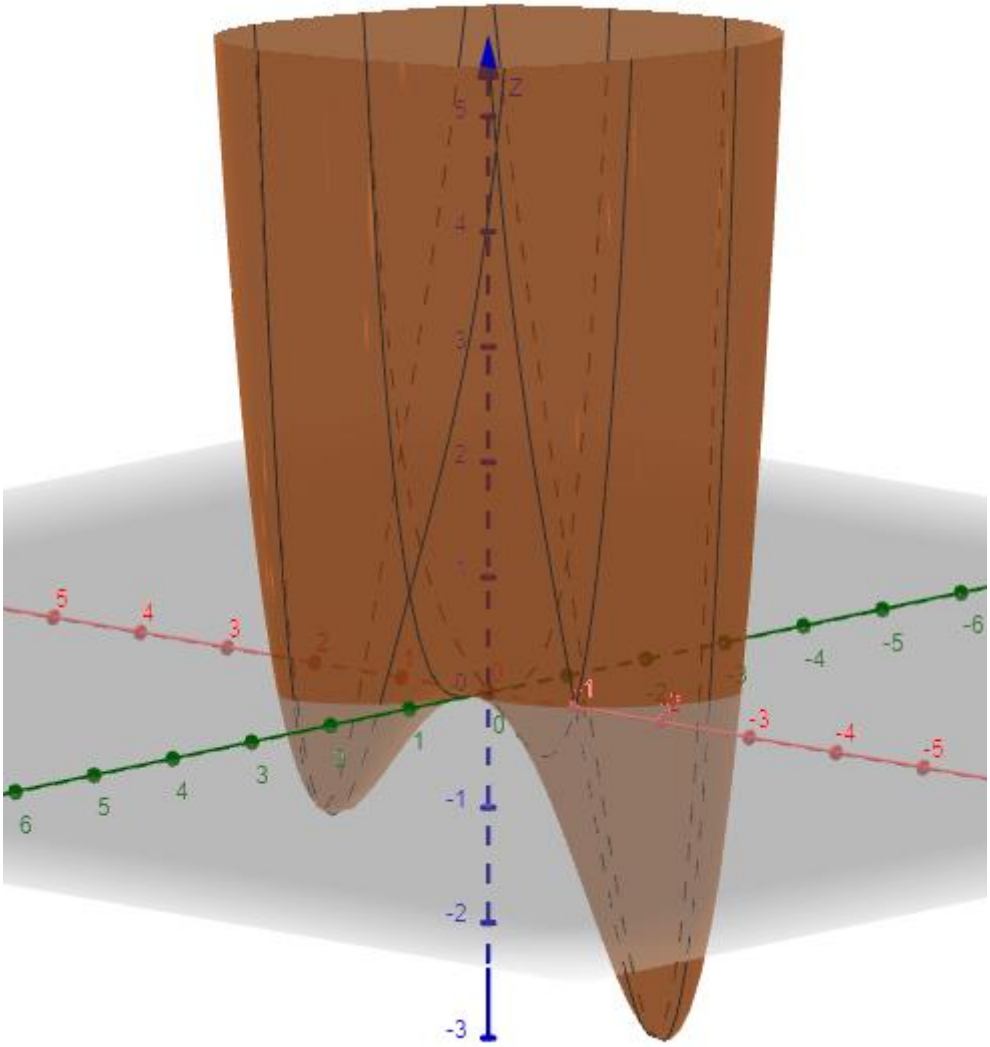
Obliczamy pochodne I rzędu i przyrównujemy do zera.

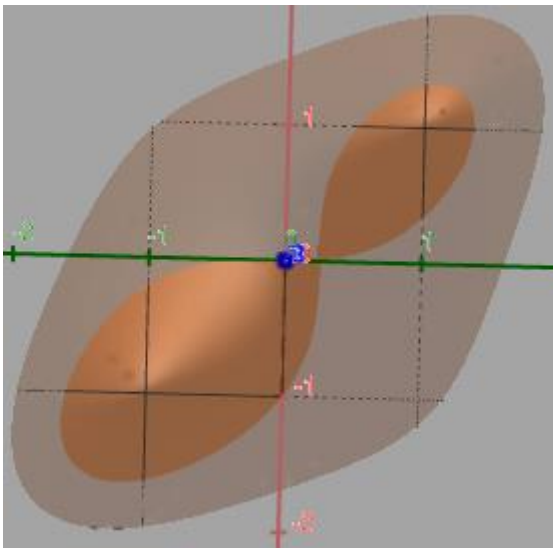
$$f_x(x, y) = 4x^3 - 4y + 1$$

$$f_{\mathrm{y}}(x, y) = 4\mathrm{y}^3 - 4\mathrm{x}$$

$$\begin{cases} 4x^3 - 4y + 1 = 0 \\ 4y^3 - 4x = 0 \end{cases} \rightarrow \begin{cases} x = \\ y = \end{cases}$$

<u>Ups…Houston mamy problem! Matematyka nie działa!</u>





Funkcja posiada dwa ekstrema – jedno lokalne, a drugie globalne. Znajdują się one blisko współrzędnych, kolejno (1, 1) oraz (-1, -1). Najprawdopodobniej algorytmy będą krążyć między jednym, a drugim punktem.

ALGORYTM GRADIENTU PROSTEGO

DOKUMENTACJA ALGORYTMU

Dokumentacja algorytmu znajduje się również w kodzie.

- Metoda __init__:

  n_min i n_max: Parametry określające zakres inicjalizacji punktu początkowego optymalizacji.

  points_x_list, points_y_list, values_list: Listy przechowujące historię wartości x, y oraz wartości funkcji w kolejnych iteracjach.

- Metoda function(x, y):

  Funkcja celu, której optymalizacja jest przeprowadzana. W tym przypadku jest to funkcja

  $$x^4 + y^4 - 4xy + x$$

- Metoda gradient_function(x, y):

  Funkcja obliczająca gradient funkcji celu względem zmiennych x i y.

- Metoda fit:

  Główna metoda przeprowadzająca optymalizację gradientową.

  objective_method: Określa, czy optymalizujemy w kierunku minimum ("min") czy maksimum ("max").

  initial_point: Początkowy punkt optymalizacji. Jeśli None, losowany jest punkt w zakresie (n_min, n_max).

  epsilon: Próg zbieżności.

  max_iterations: Maksymalna liczba iteracji.

  initial_step_size: Początkowy rozmiar kroku dla optymalizacji gradientowej.

  step_size_reduction_method: Metoda redukcji rozmiaru kroku - "factor", "sqrt_divide" lub "n_divide".

  step_size_reduction_factor: Współczynnik redukcji rozmiaru kroku dla metody "factor".

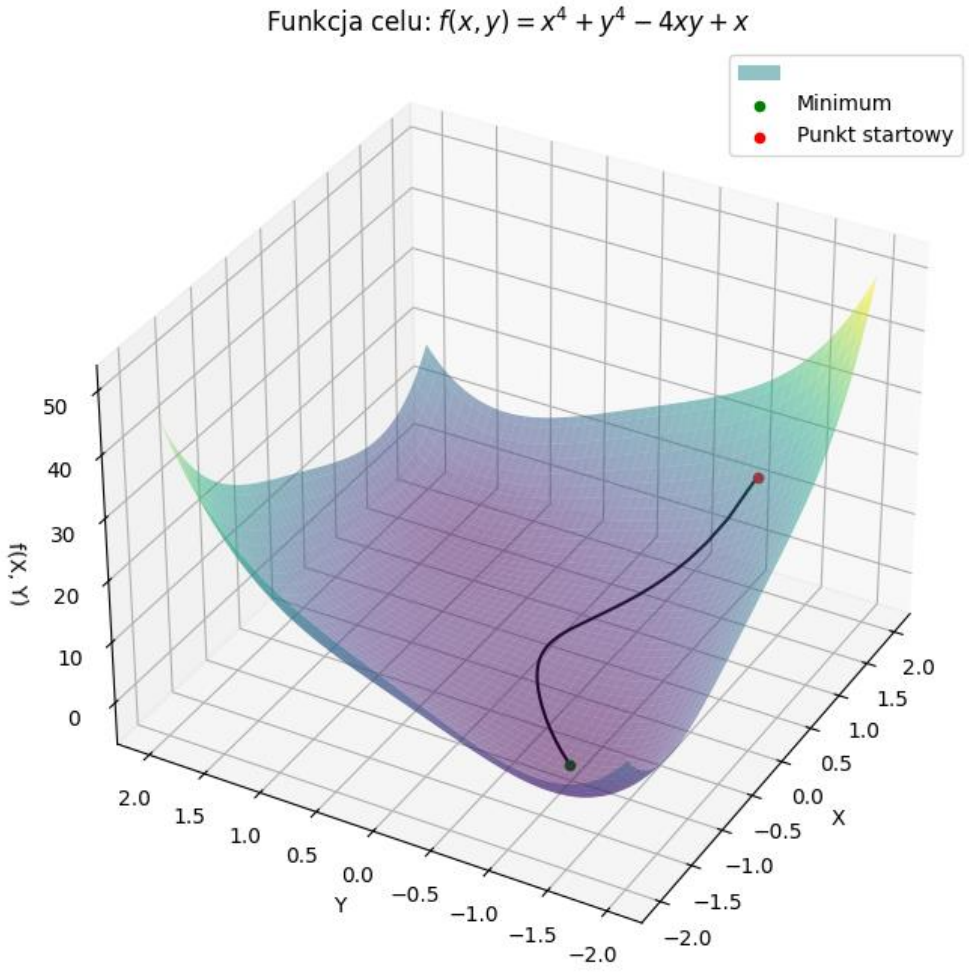
  Warunek zakończenia pętli w metodzie fit:

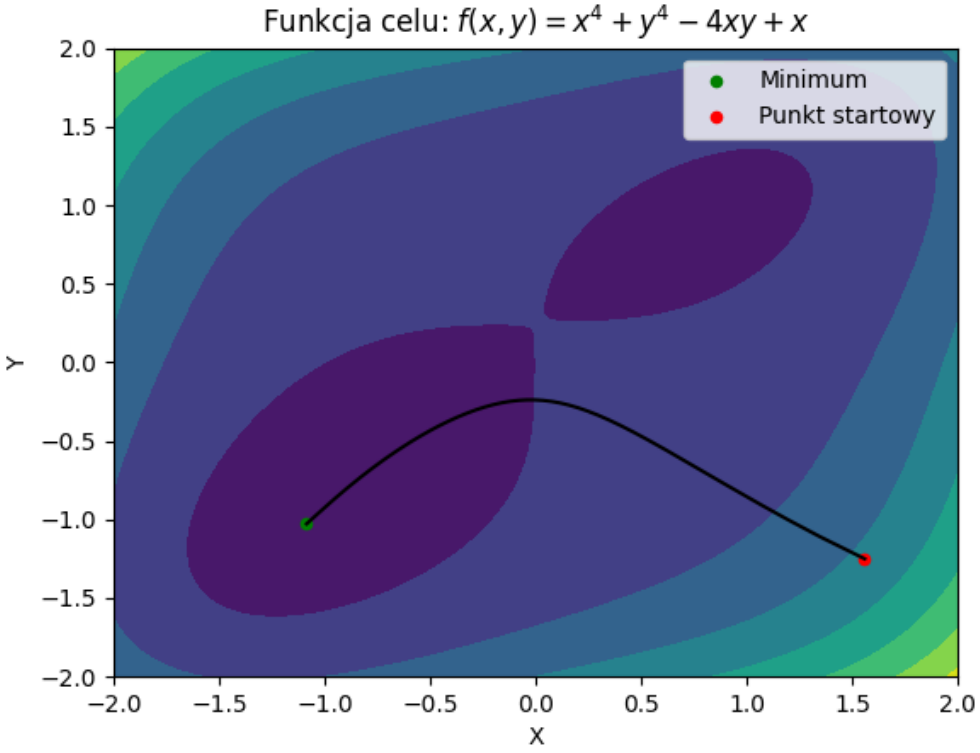
  Pętla zostanie przerwana, gdy gradient funkcji jest dostatecznie mały lub osiągnięto zadaną liczbę iteracji.

Algorytm ten ma również elastyczność w dostosowywaniu kroku optymalizacji (metoda fit), co może wpływać na szybkość zbieżności. Metoda ta oferuje także możliwość wyboru kierunku optymalizacji, czyli minimalizacji lub maksymalizacji funkcji celu.

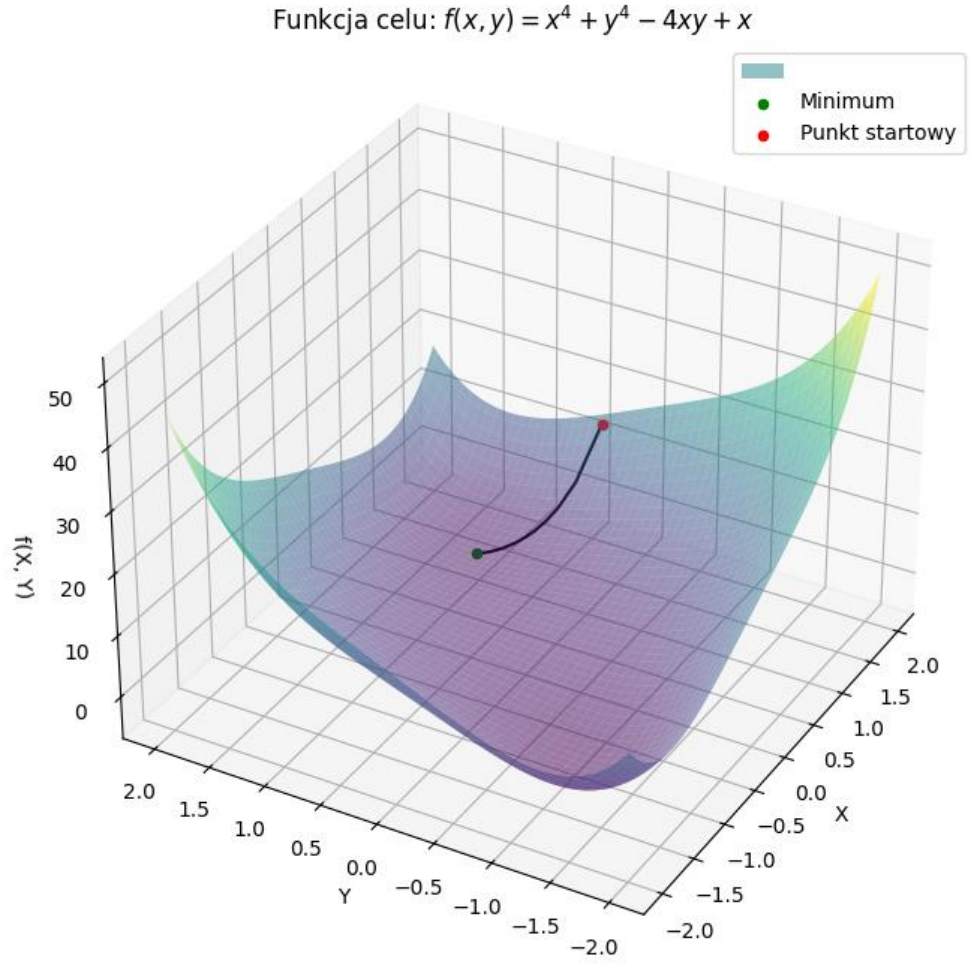
Wizualizacja wyników

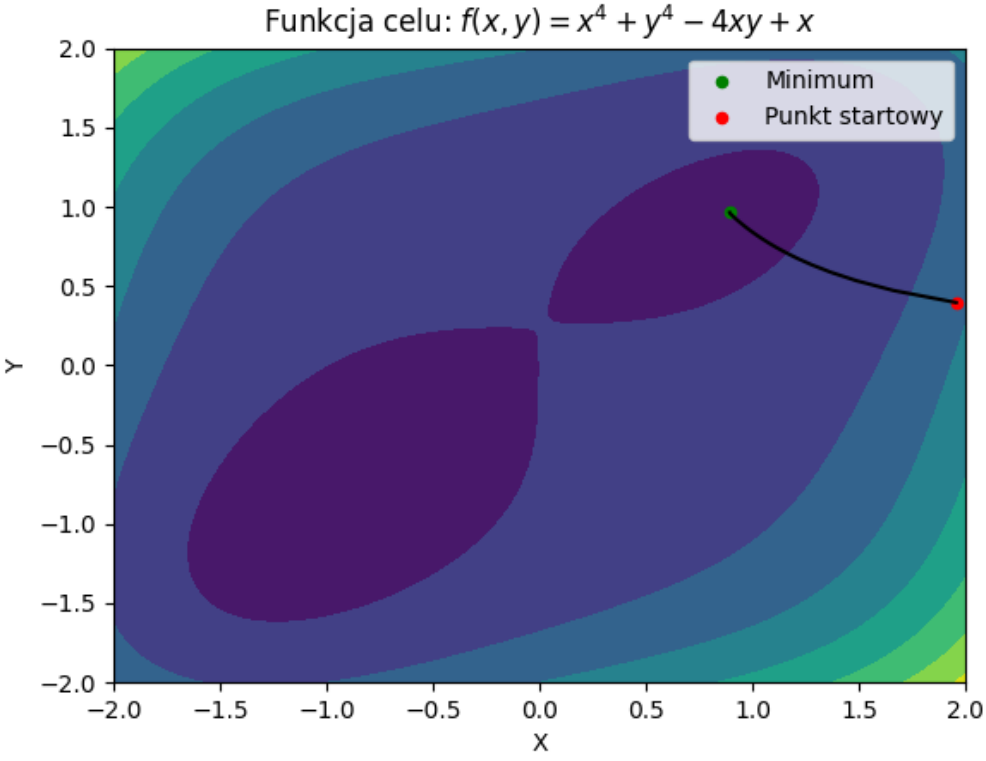
Algorytm wywołany dla randomowo wygenerowanego punktu startowego.





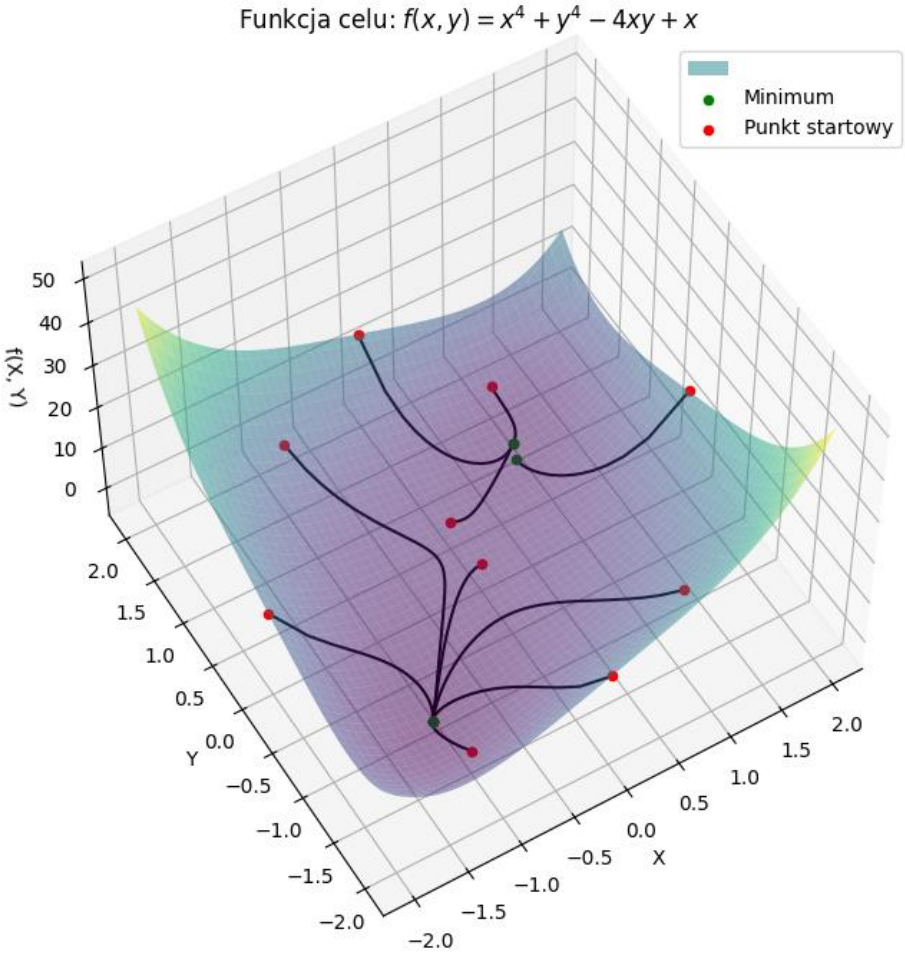
Algorytm wywołany dla randomowe wygenerowanego punktu startowego.

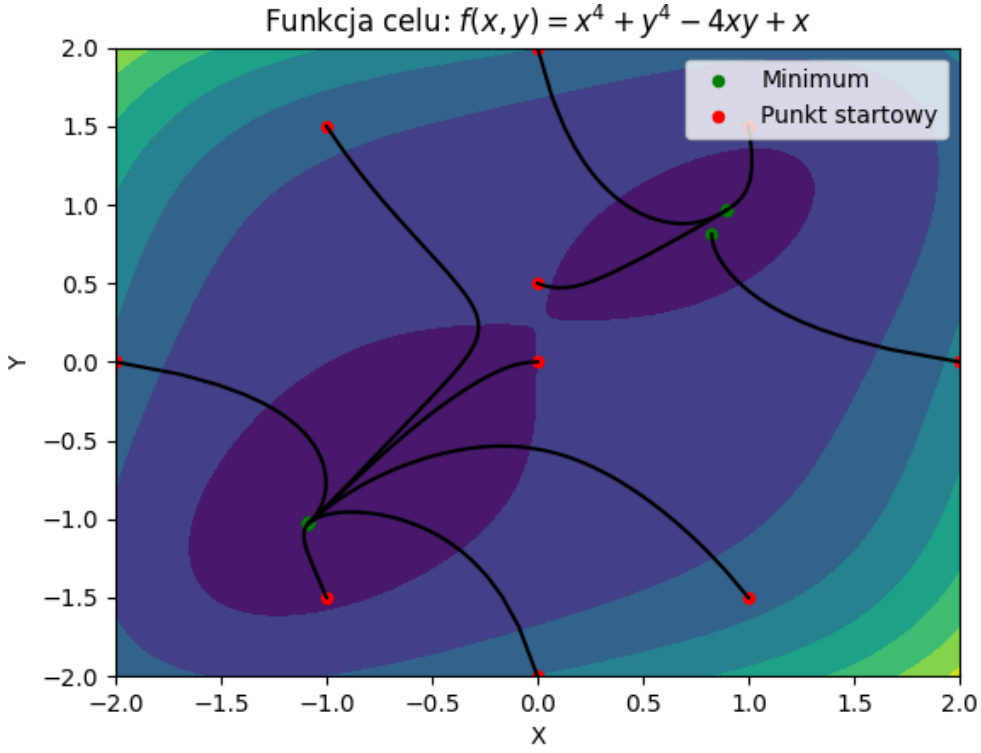




Jak można zauważyć położenie punktu startowego ma ogromne znaczenie i determinuje „podróż” algorytmu do minimum. Im bliżej punkt jest minimum globalnego/lokalnego tym większe prawdopodobieństwo, że to tam zatrzyma się nasz algorytm.

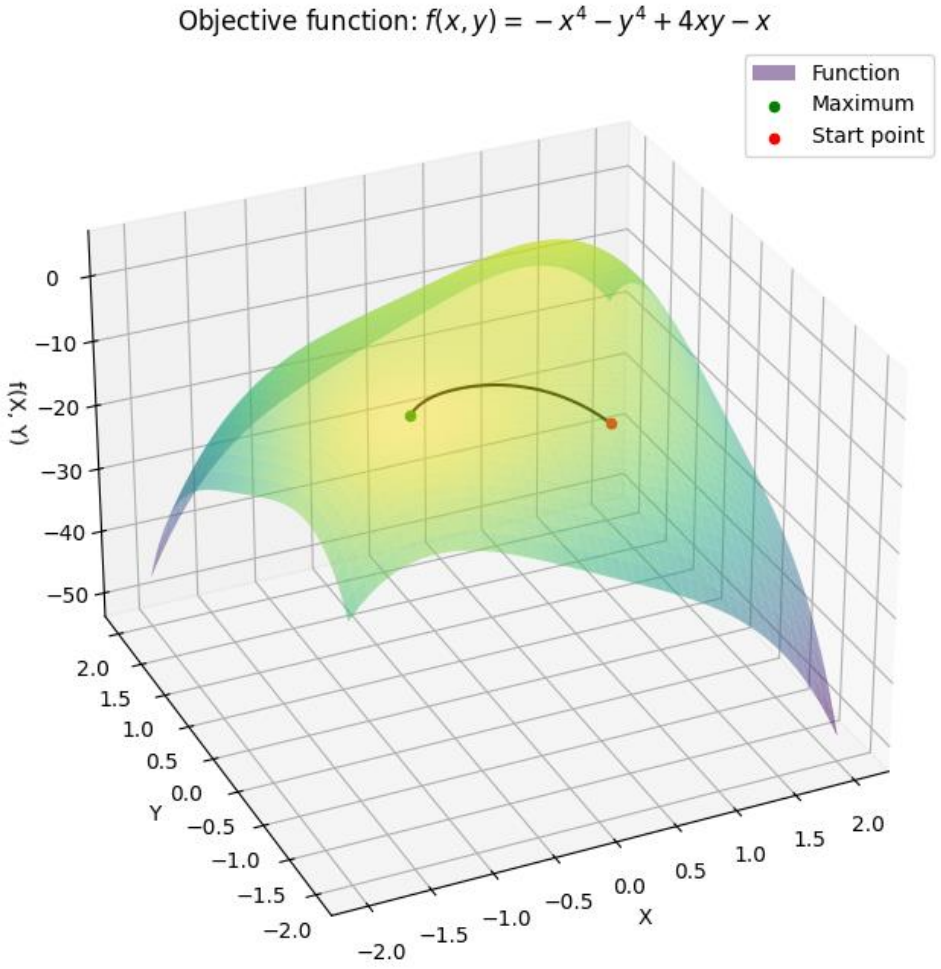
Poniżej można zaobserwować ten fakt dla większej liczby punktów, które znajdują się w różnych miejscach.

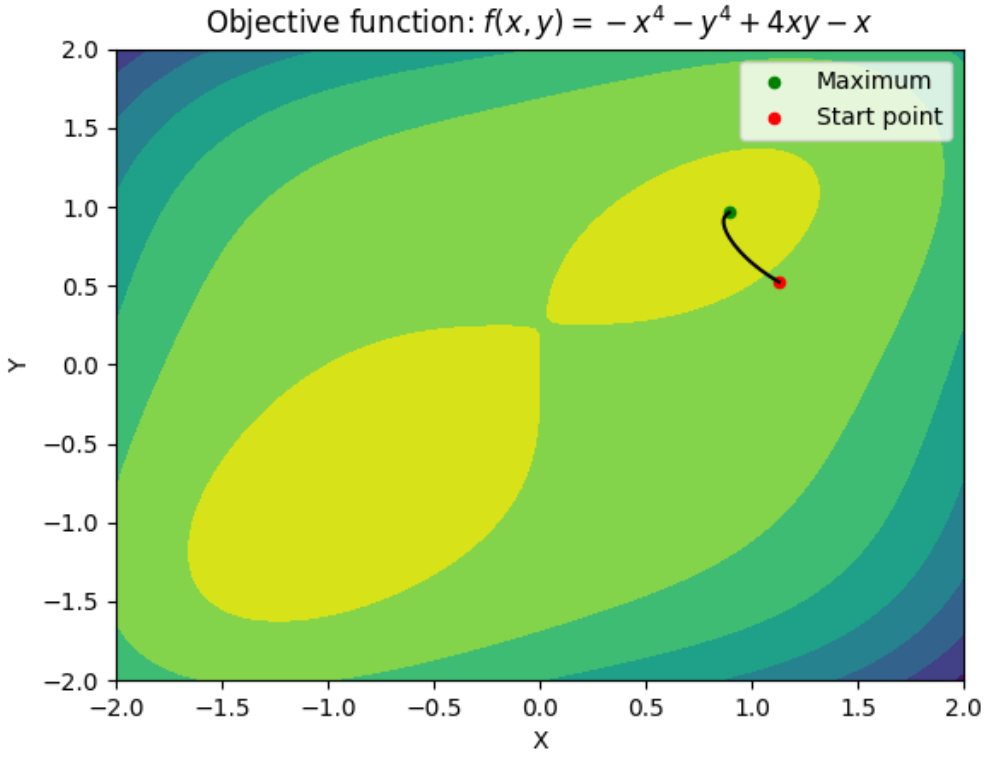




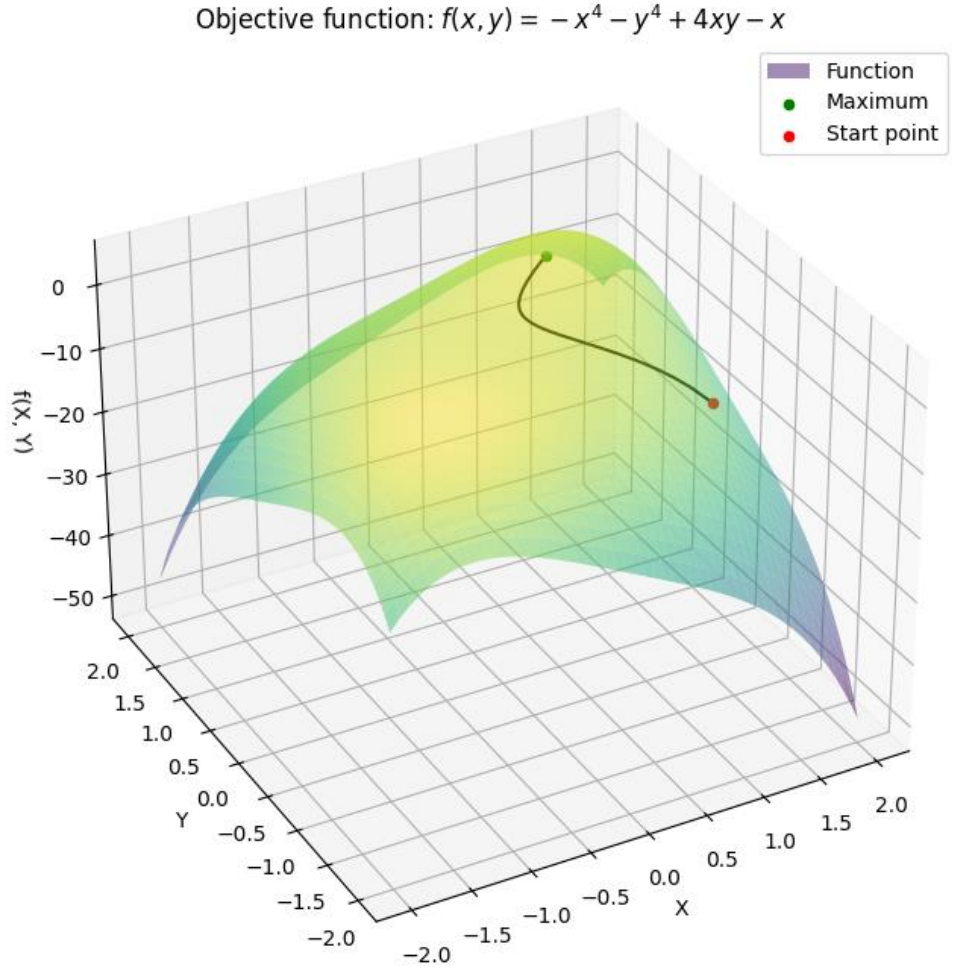
Program może również szukać maximum funkcji. Poniżej przykłady analogiczne do tych, które podałem w szukaniu minimum, a funkcja, jest funkcją przeciwną do podanej w zadaniu.

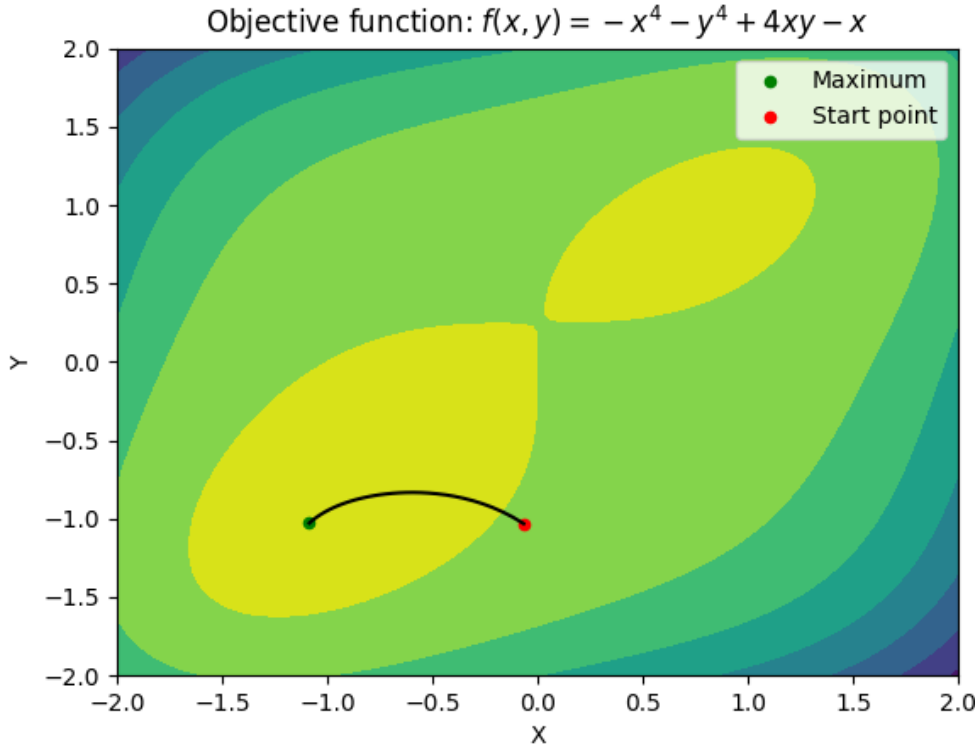
Algorytm wywołany dla randomowo wygenerowanego punktu startowego.





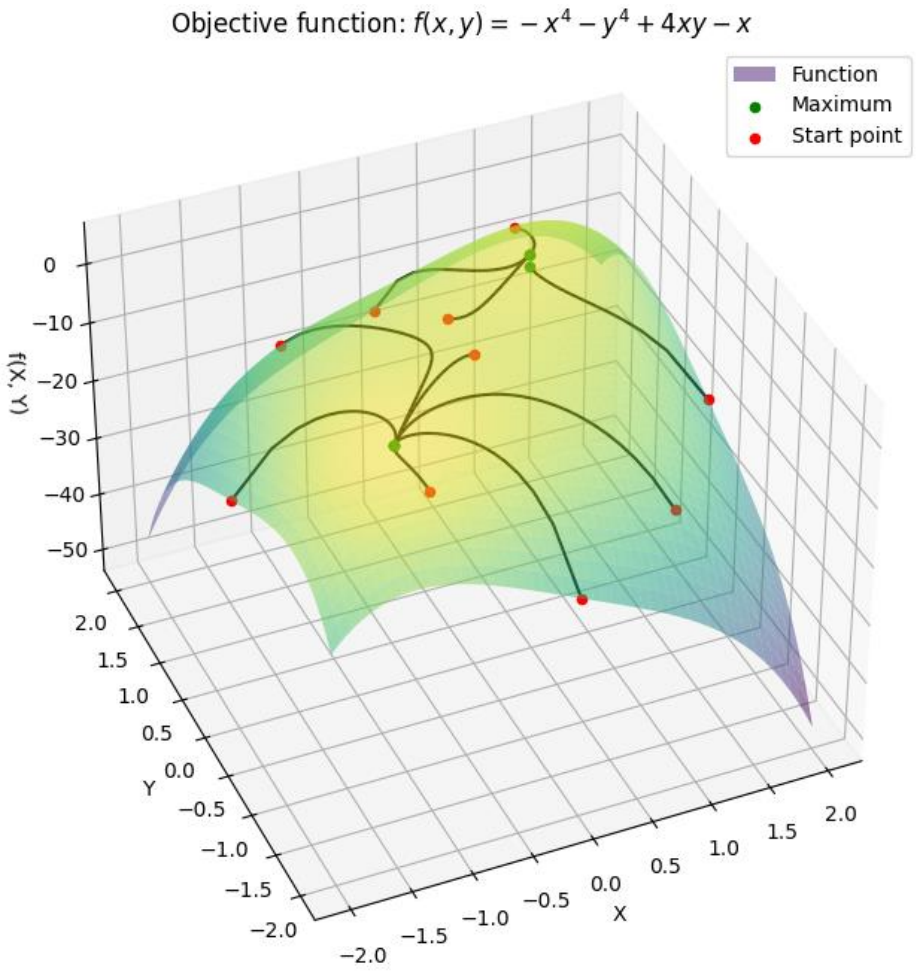
Algorytm wywołany dla randomowo wygenerowanego punktu startowego.

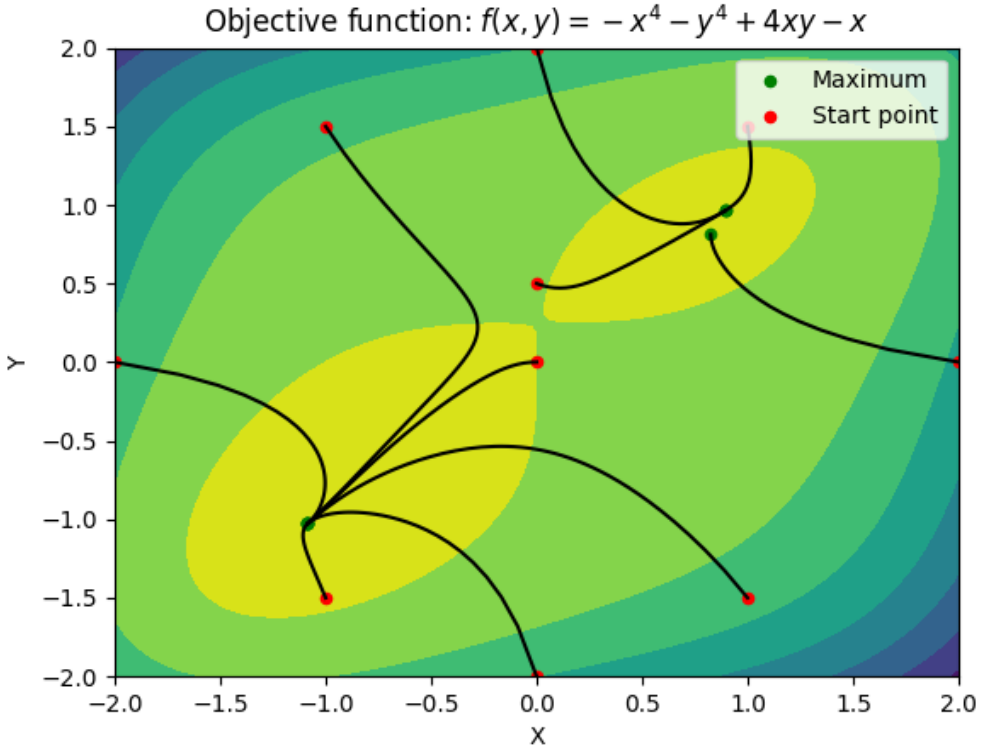




Jak można zauważyć położenie punktu startowego ma ogromne znaczenie i determinuje „podróż” algorytmu do maximum. Im bliżej punkt jest maximum globalnego/lokalnego tym większe prawdopodobieństwo, że to tam zatrzyma się nasz algorytm.

Poniżej można zaobserwować ten fakt dla większej liczby punktów, które znajdują się w różnych miejscach.





SYMBOLICZNY ALGORYTM GRADIENTU PROSTEGO

DOKUMENTACJA ALGORYTMU

Dokumentacja algorytmu znajduje się również w kodzie.

- Metoda __init__:

  function: Parametr przechowujący funkcję matematyczną w notacji Pythona.

  n_min i n_max: Parametry określające zakres inicjalizacji punktu początkowego optymalizacji.

  variables: Lista zmiennych, które występują w funkcji matematycznej.

  gradient_function: Lista gradientów funkcji względem zmiennych.

- Metoda fit:

  Główna metoda przeprowadzająca optymalizację gradientową.

  objective_method: Określa, czy optymalizujemy w kierunku minimum ("min") czy maksimum ("max").

  initial_point: Początkowy punkt optymalizacji. Jeśli None, losowany jest punkt w zakresie (n_min, n_max).

  epsilon: Próg zbieżności.

  max_iterations: Maksymalna liczba iteracji.

  initial_step_size: Początkowy rozmiar kroku dla optymalizacji gradientowej.

  step_size_reduction_method: Metoda redukcji rozmiaru kroku - "factor", "sqrt_divide" lub "n_divide".

  step_size_reduction_factor: Współczynnik redukcji rozmiaru kroku dla metody "factor".

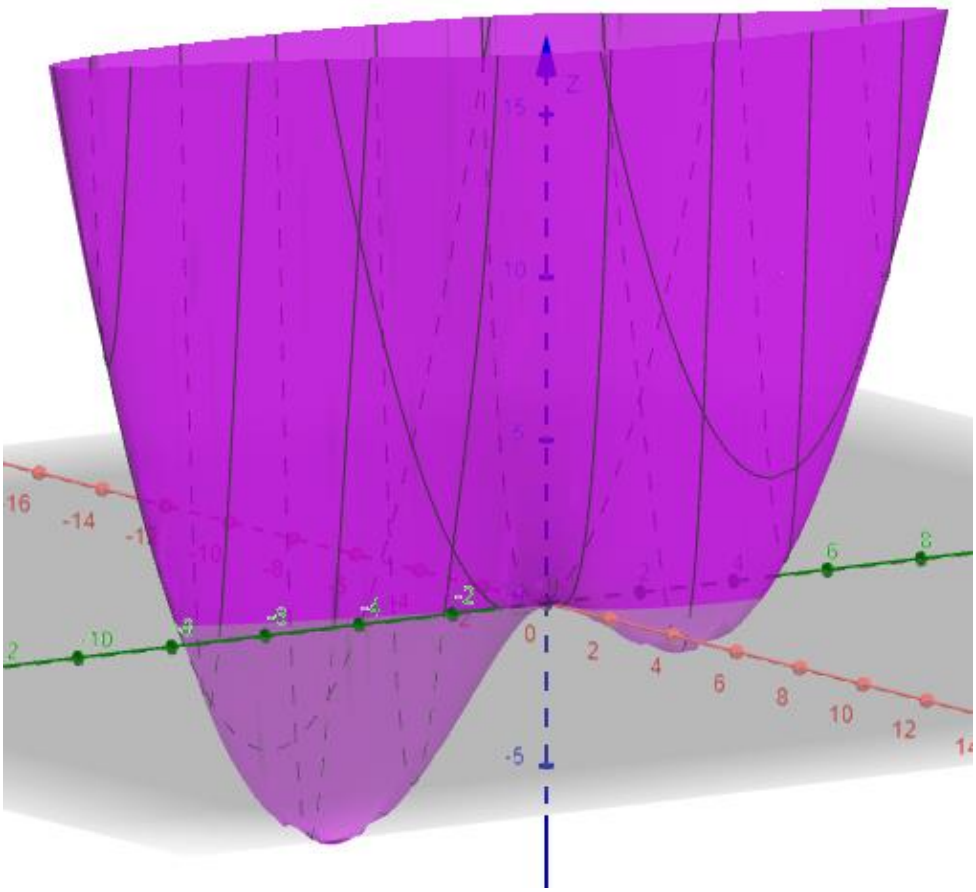
  Warunek zakończenia pętli w metodzie fit:

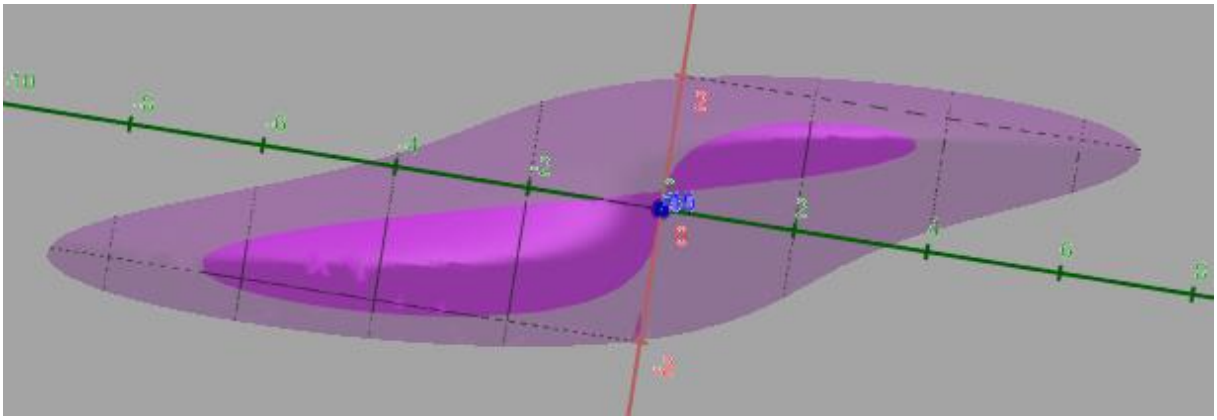
  Pętla zostanie przerwana, gdy gradient funkcji jest dostatecznie mały lub osiągnięto zadaną liczbę iteracji.

Algorytm ten może być używany do optymalizacji dowolnych funkcji matematycznych, a jego elastyczność pozwala na dostosowanie różnych parametrów, takich jak rozmiar kroku czy metoda redukcji kroku.

Mając funkcję daną wzorem:

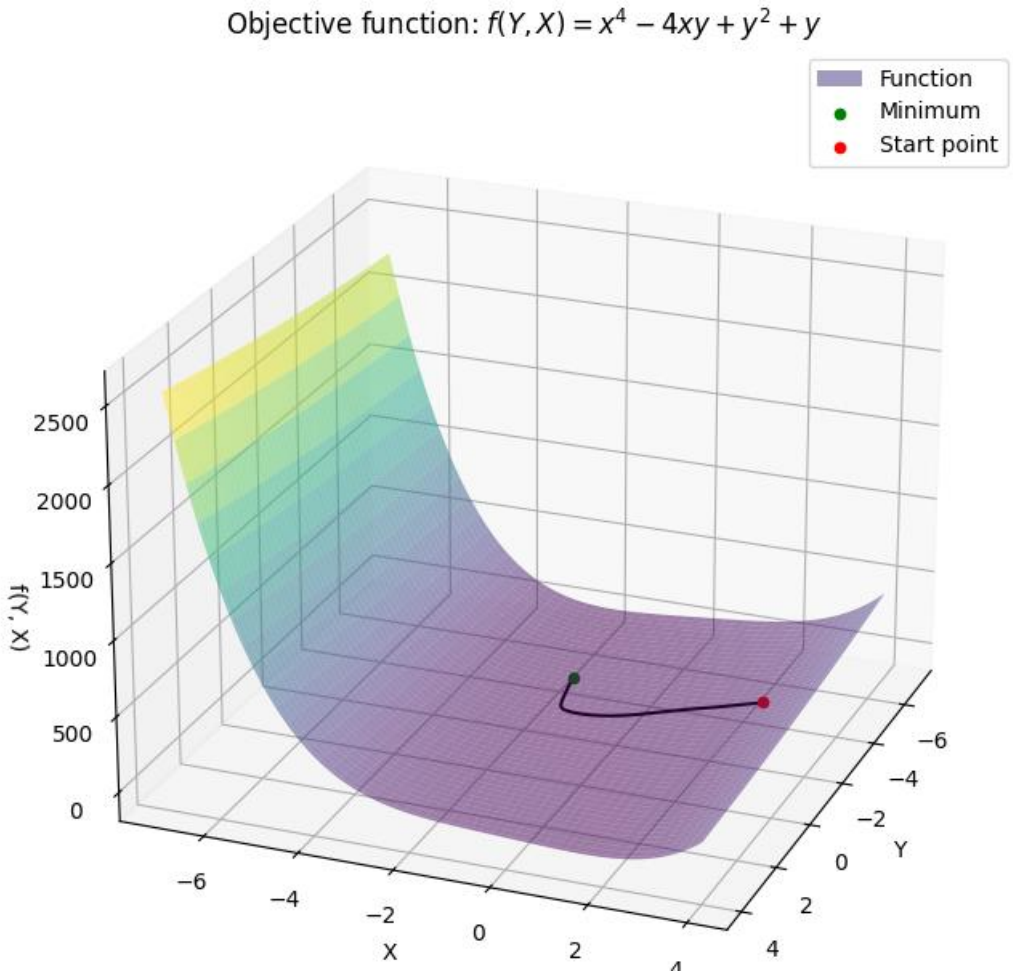
$$f(x,y) = x^4 + y^2 - 4xy + y$$

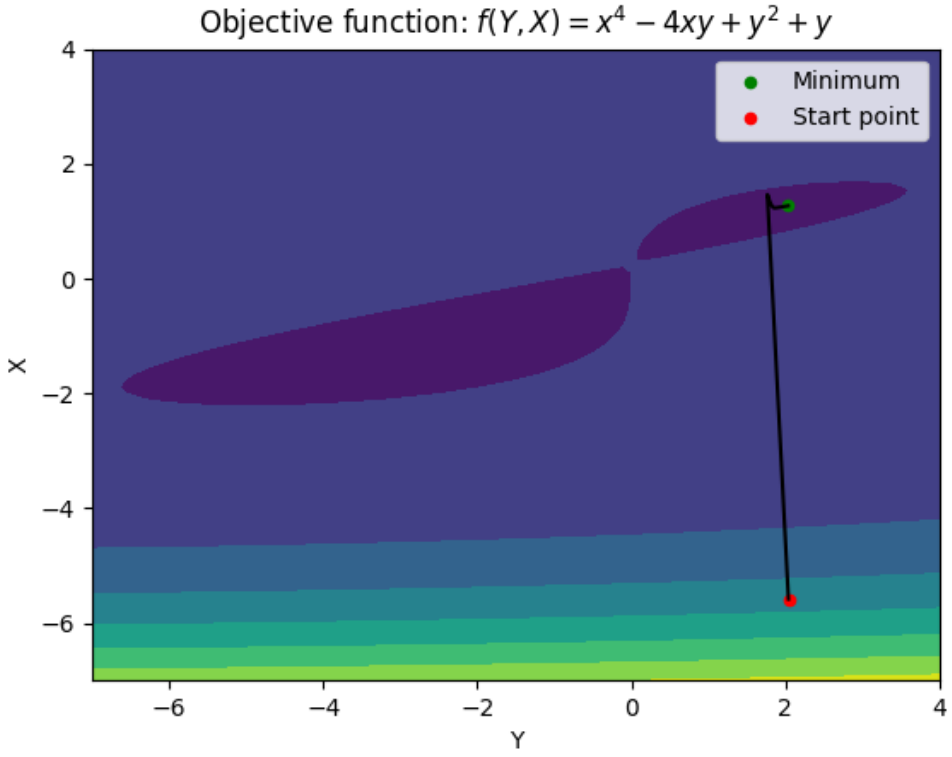




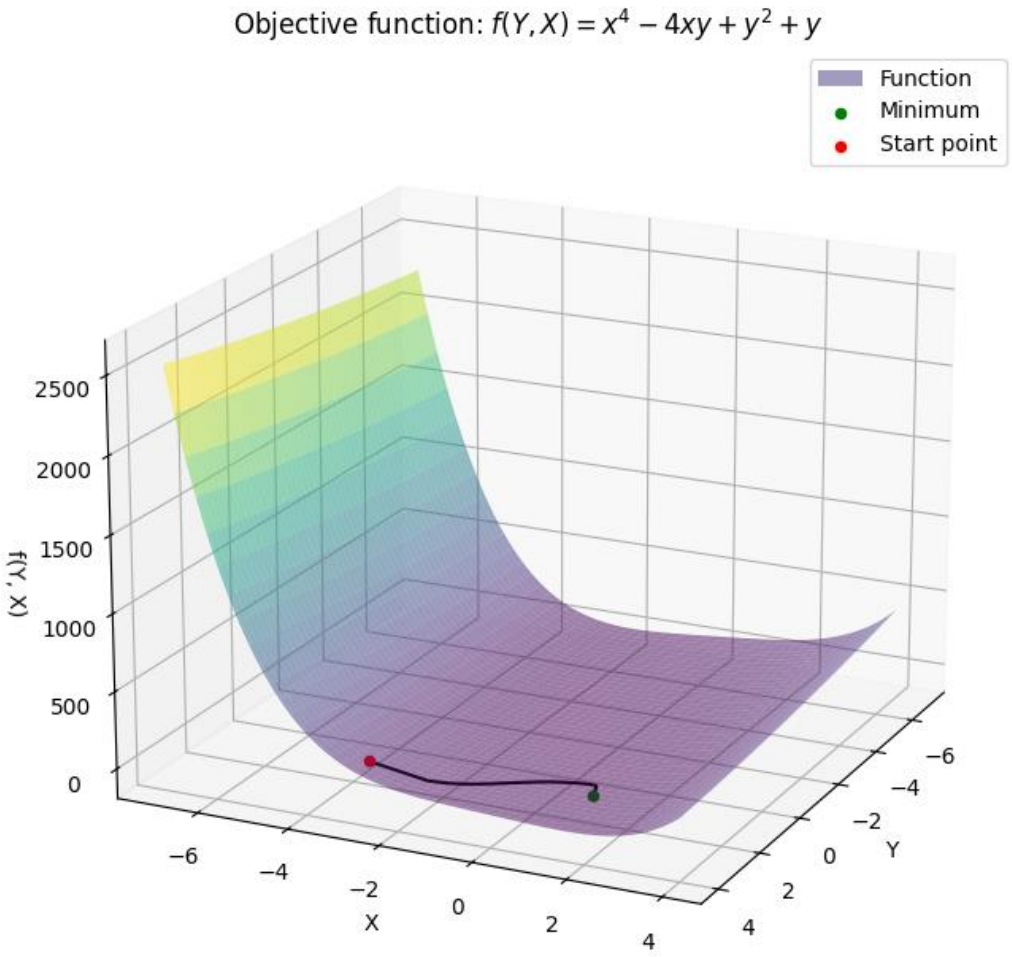
Za pomocą obliczeń symbolicznych znajdziemy minimum tej funkcji oraz maximum funkcji przeciwnej.

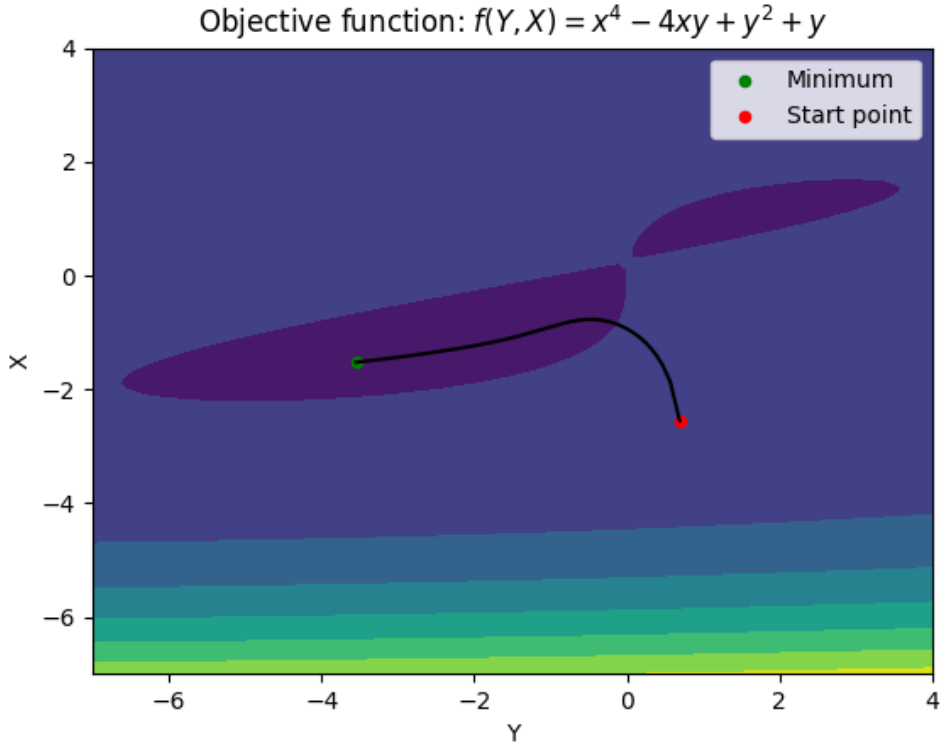
Algorytm wywołany dla randomowo wygenerowanego punktu startowego.





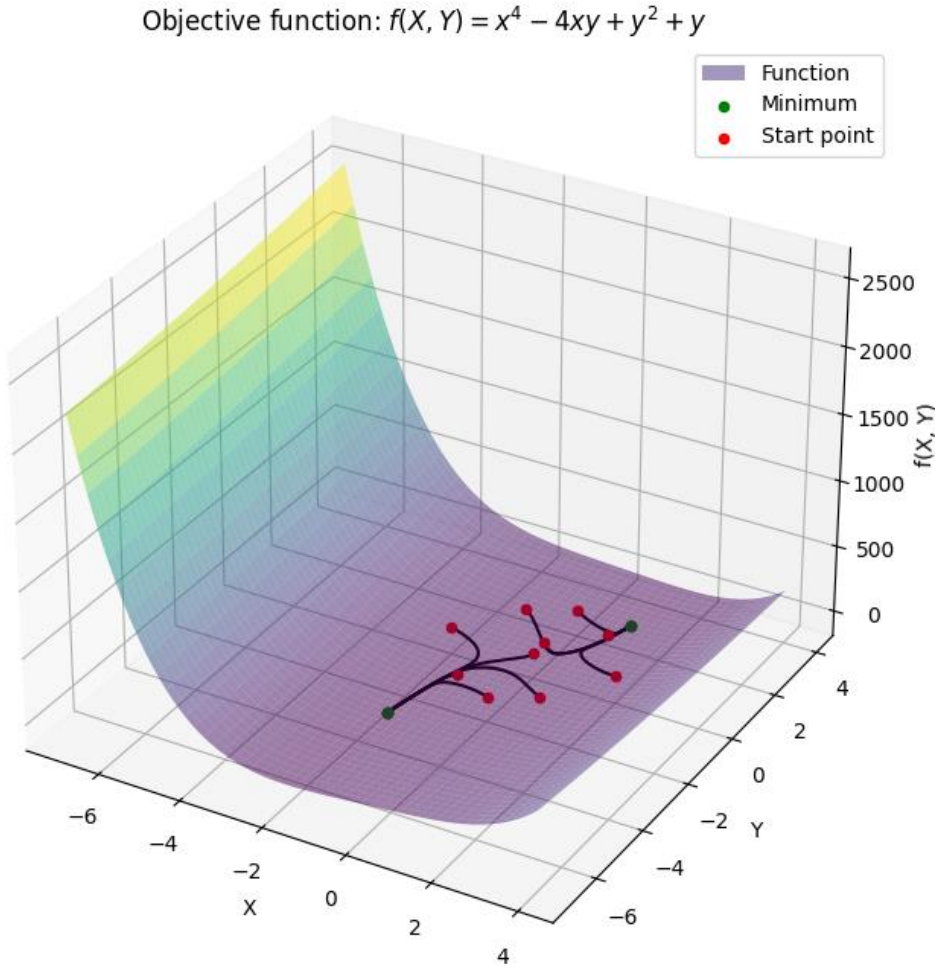
Algorytm wywołany dla randomowo wygenerowanego punktu startowego.

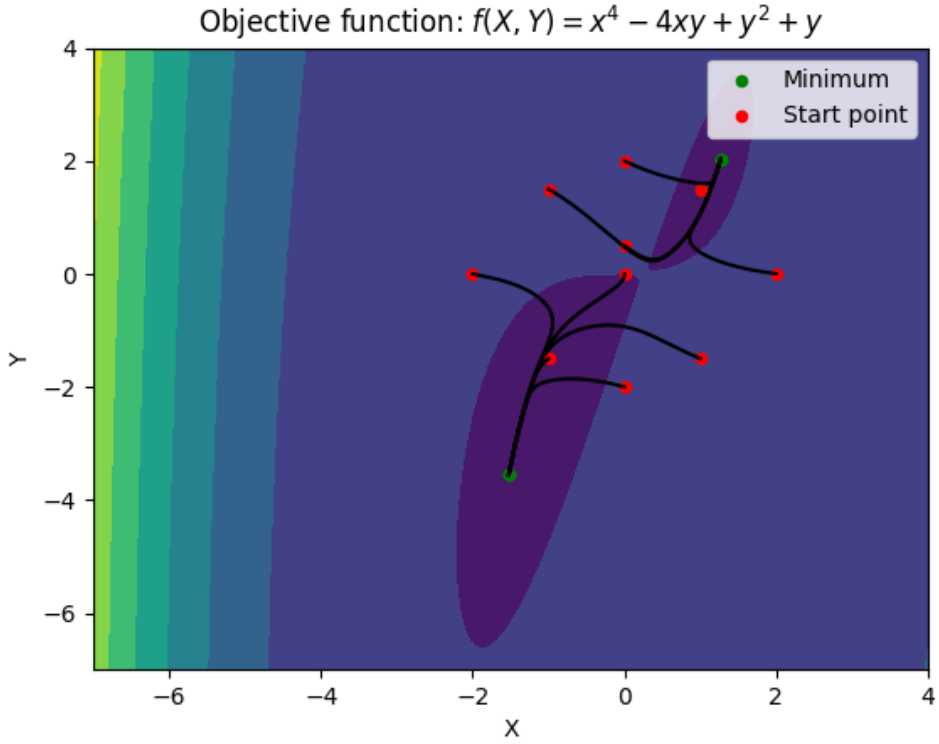




Jak można zauważyć położenie punktu startowego ma ogromne znaczenie i determinuje „podróż” algorytmu do minimum. Im bliżej punkt jest minimum globalnego/lokalnego tym większe prawdopodobieństwo, że to tam zatrzyma się nasz algorytm.

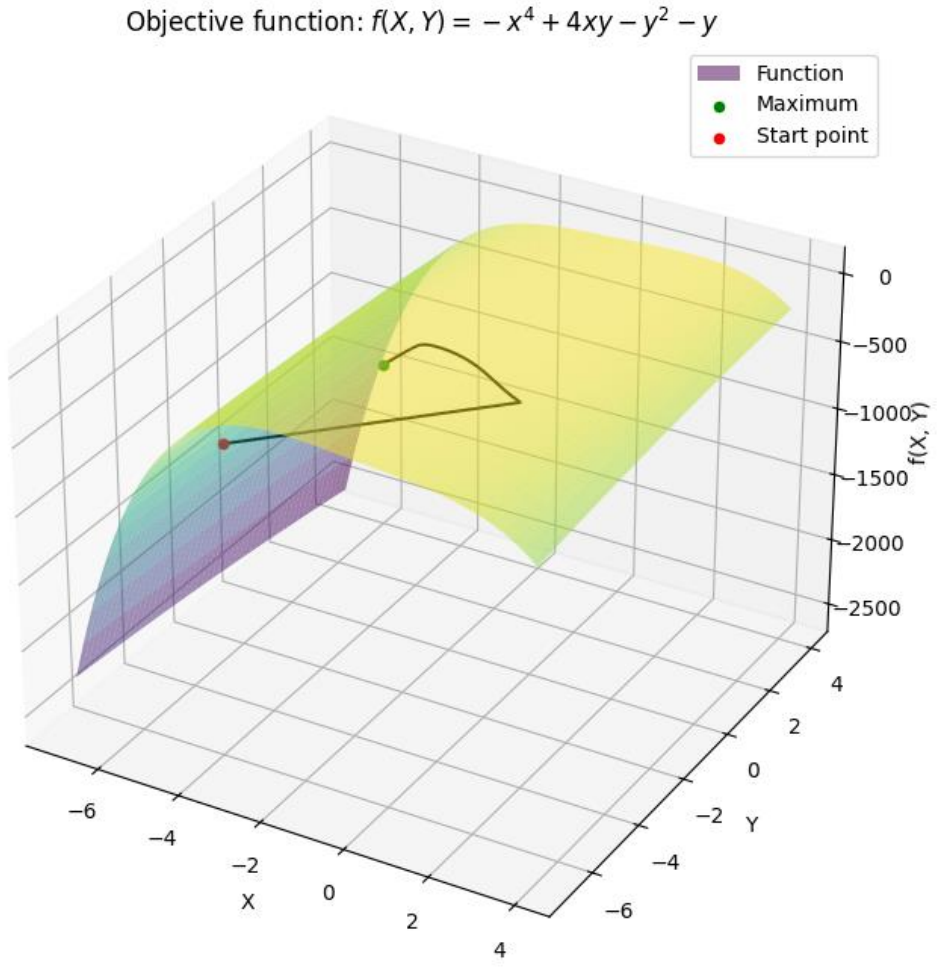
Poniżej można zaobserwować ten fakt dla większej liczby punktów, które znajdują się w różnych miejscach.

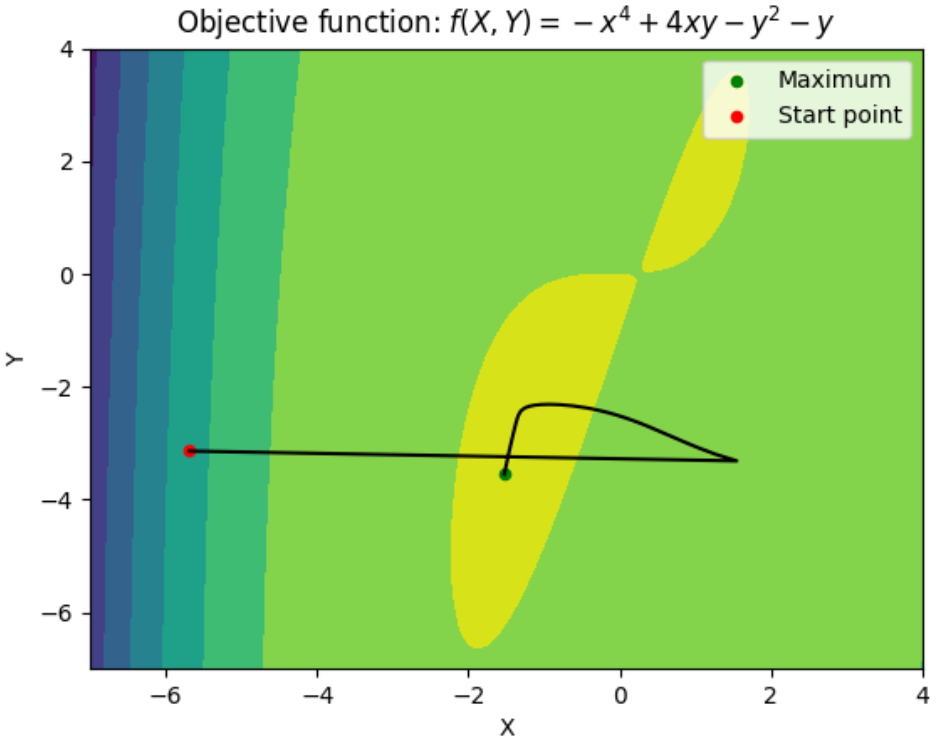




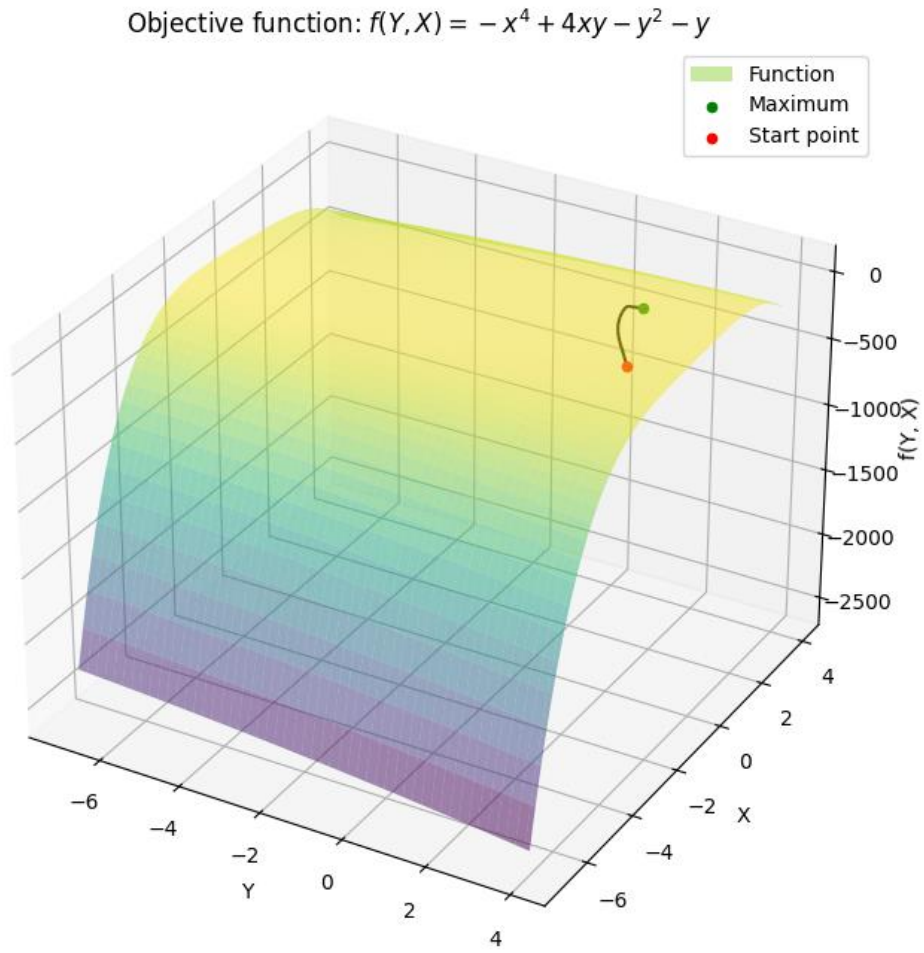
Program może również szukać maximum funkcji. Poniżej przykłady analogiczne do tych, które podałem w szukaniu minimum, a funkcja, jest funkcją przeciwną do podanej w zadaniu.

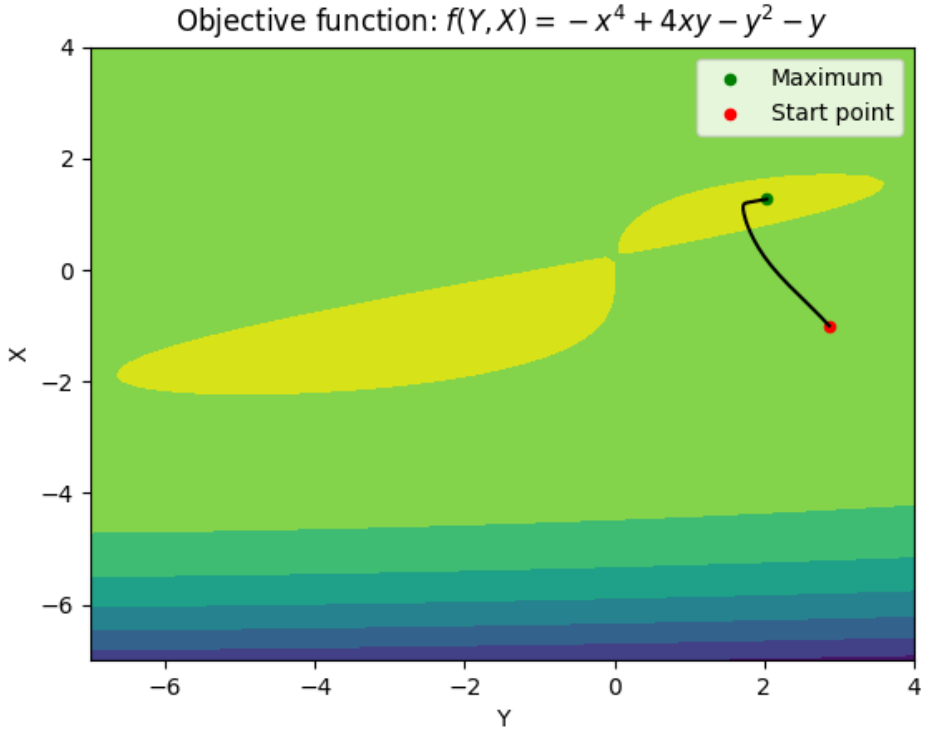
Algorytm wywołany dla randomowo wygenerowanego punktu startowego.





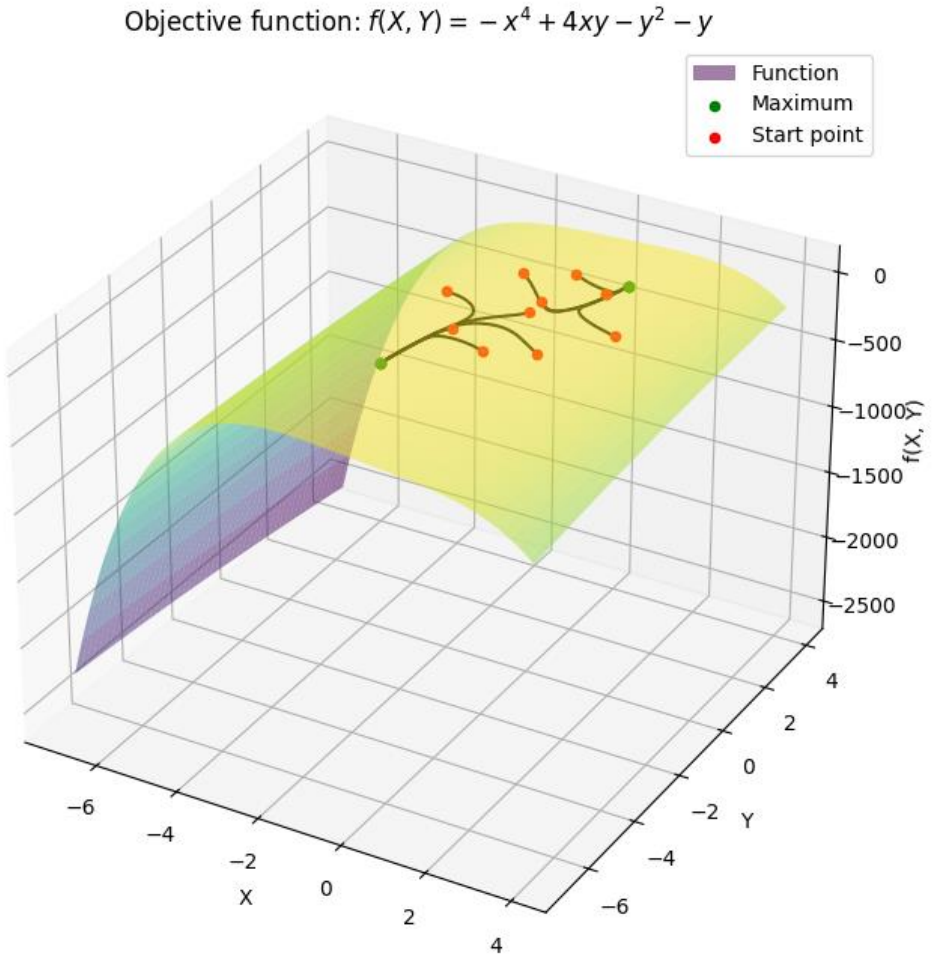
Algorytm wywołany dla randomowo wygenerowanego punktu startowego.

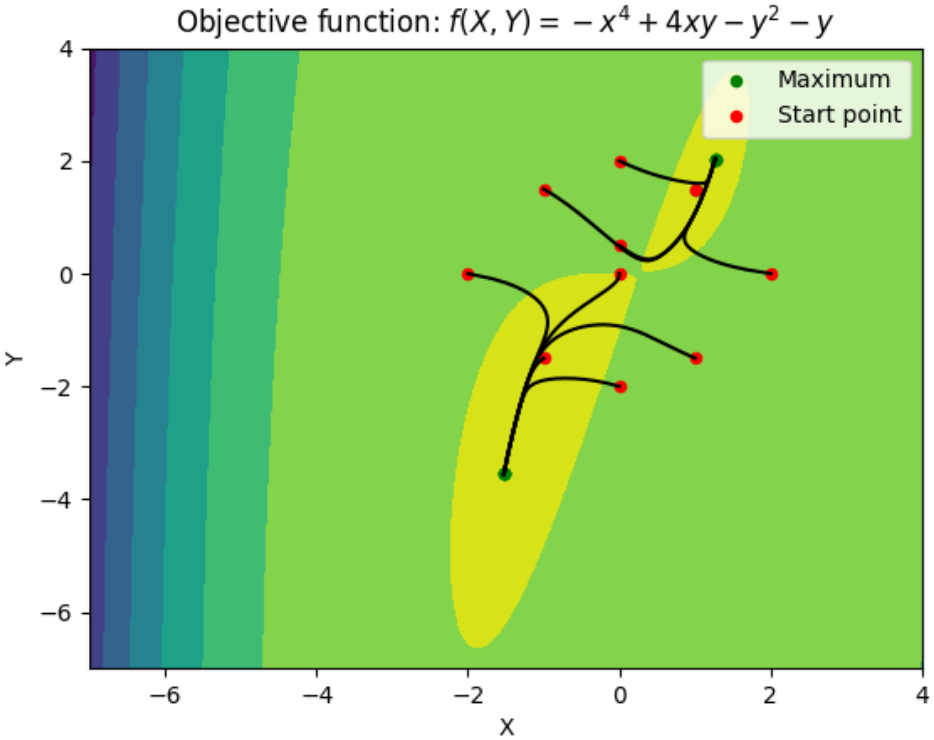




Jak można zauważyć położenie punktu startowego ma ogromne znaczenie i determinuje „podróż” algorytmu do maximum. Im bliżej punkt jest maximum globalnego/lokalnego tym większe prawdopodobieństwo, że to tam zatrzyma się nasz algorytm.

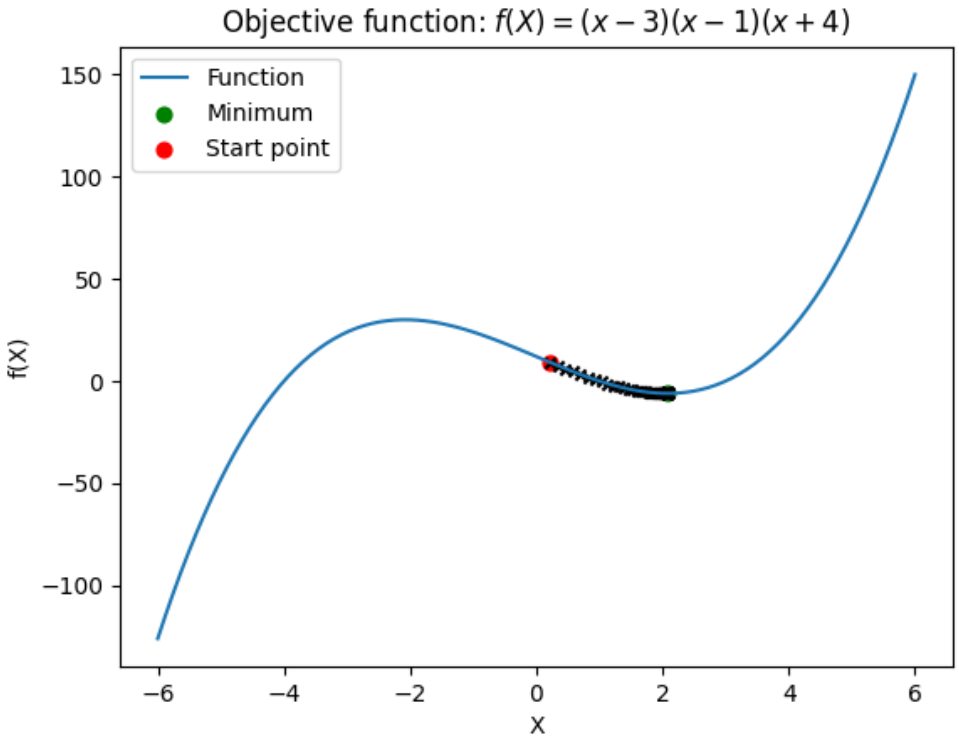
Poniżej można zaobserwować ten fakt dla większej liczby punktów, które znajdują się w różnych miejscach.



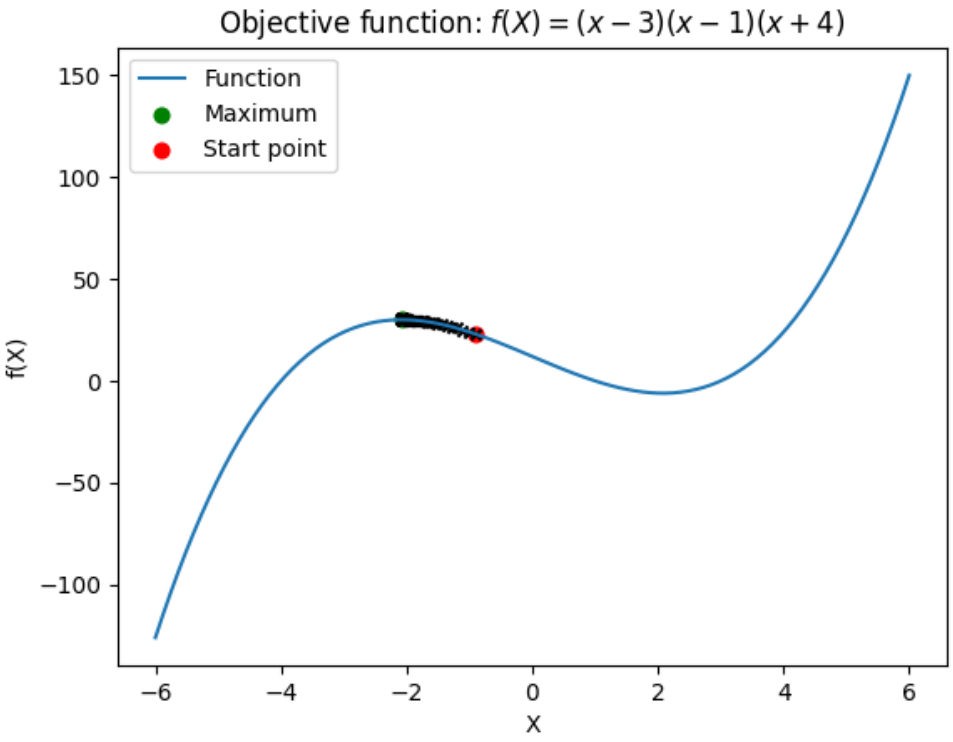


Algorytm może również znajdywać ekstrema funkcji jednej zmiennej i wizualizować je.

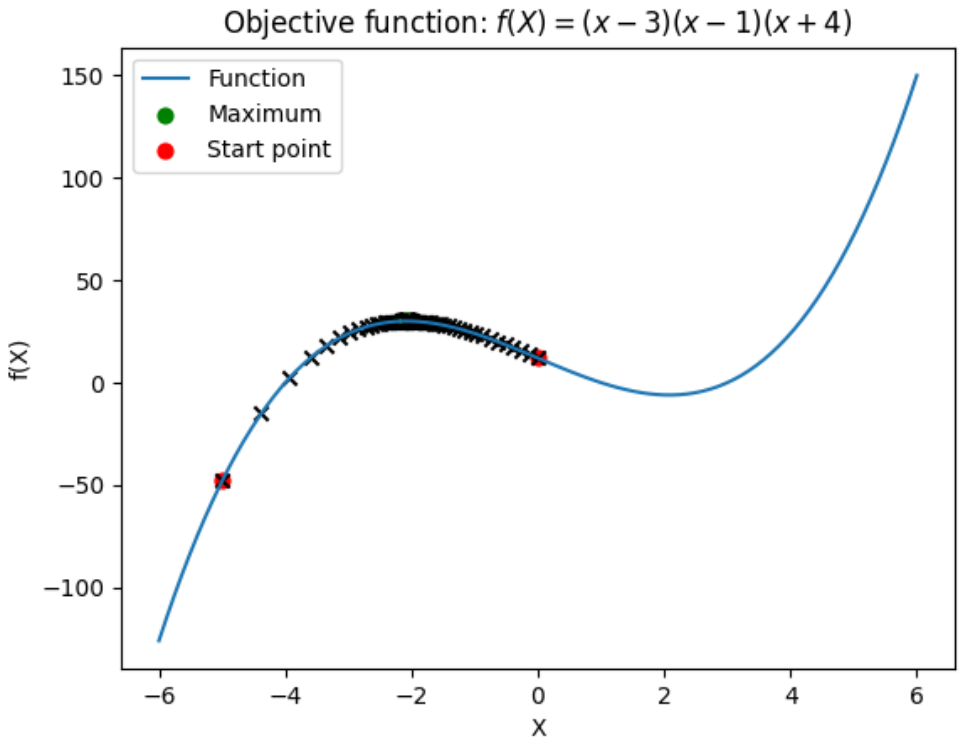
Algorytm wywołany dla randomowo wygenerowanego punktu startowego.



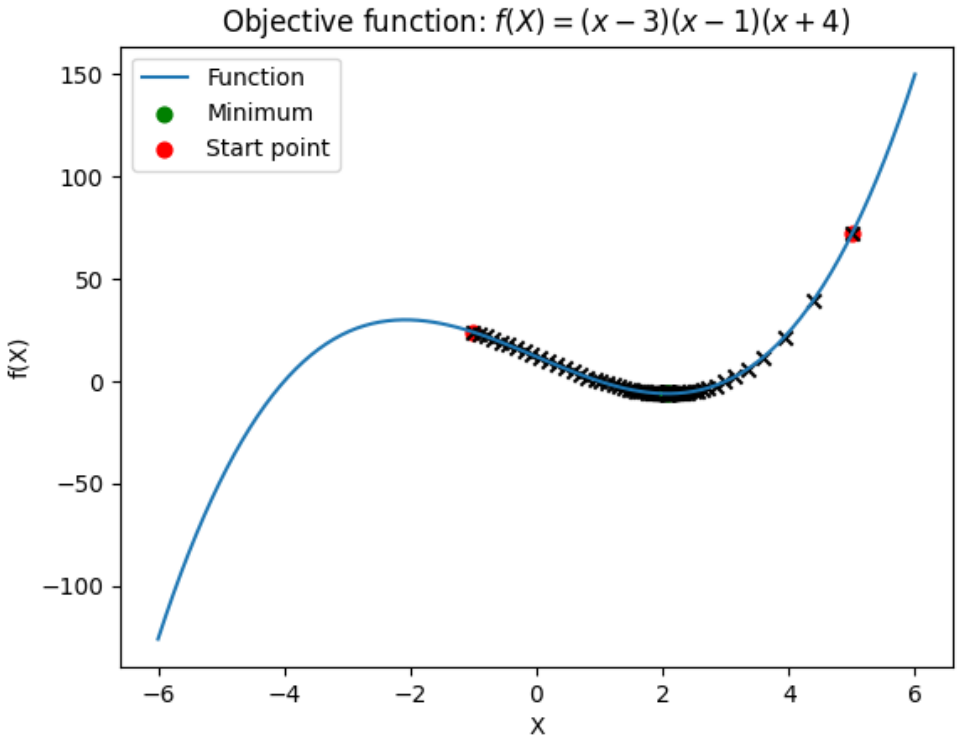
Algorytm wywołany dla randomowo wygenerowanego punktu startowego.



Algorytm wywołany dla punktów x=-5 oraz x=0.



Algorytm wywołany dla punktów x=-1 oraz x=5.



ALGORYTM GENETYCZNY

Algorytm genetyczny (AG) to heurystyczna metoda optymalizacji inspirowana procesem ewolucji biologicznej. Jego głównym celem jest rozwiązanie problemów optymalizacyjnych poprzez symulację procesów genetycznych, takich jak selekcja naturalna, krzyżowanie i mutacja. Poniżej opisuję główne kroki algorytmu genetycznego:

- Inicjalizacja populacji: Algorytm zaczyna od stworzenia losowej populacji osobników, gdzie każdy osobnik reprezentuje potencjalne rozwiązanie problemu.
- Funkcja oceny (fitnes): Dla każdego osobnika w populacji obliczana jest wartość funkcji oceny (fitnes), która określa, jak dobrze dany osobnik radzi sobie w rozwiązaniu problemu optymalizacyjnego.
- Selekcja: Osobniki są wybierane do reprodukcji na podstawie ich fitnesu. Możliwe są różne metody selekcji, takie jak ruletka, turniej czy ranking.
- Krzyżowanie (crossover): Pary osobników są wybierane do krzyżowania, czyli wymiany części swoich informacji genetycznych. Krzyżowanie symuluje genetyczne rekombinacje i pomaga stworzyć potomstwo, które dziedziczy cechy obu rodziców.
- Mutacja: Pewne losowo wybrane elementy potomstwa podlegają mutacji, czyli losowej zmianie ich cech. Mutacja wprowadza zróżnicowanie genetyczne do populacji, co może pomóc uniknąć zbytniego skupienia się wokół lokalnych optimum.
- Zastąpienie starej populacji: Nowo utworzone potomstwo zastępuje starą populację. Proces ten może być powtarzany przez kilka generacji.
- Kryterium zatrzymania: Algorytm kontynuuje ewolucję populacji przez określoną liczbę generacji lub do momentu spełnienia określonego warunku zatrzymania, takiego jak osiągnięcie wystarczająco dobrego rozwiązania.

Algorytm genetyczny jest używany w różnych dziedzinach do rozwiązywania problemów optymalizacyjnych, takich jak projektowanie układów, planowanie tras, optymalizacja funkcji matematycznych itp. Jego siła tkwi w zdolności do przeszukiwania przestrzeni rozwiązań, znajdowania globalnych optimum oraz radzenia sobie z złożonymi i wielowymiarowymi problemami.

Jednym z przykładów algorytmu genetycznego jest algorytm Mu+Lambda.

WYNIKI PRÓB

The best individual:

x = 0.7387674208730459

y = 0.8948781285434961

Fitness value: -0.966495308727263

The best individual:

x = -1.0078379856422544

y = -0.7656135400757194

Fitness value: -2.7189847975248536

SYMBOLICZNY ALGORYTM GENETYCZNY

WYNIKI PRÓB

Crossbreeding
The best individual:
x = -0.915215986505447
y = -1.0040661098009673
Fitness value: -2.87299372917602

Mutating
The best individual:
x = 0.6195856062695384
y = 0.4880199683830142
Fitness value: -0.385804428541673

Crossbreeding
The best individual:
y = -0.43239297261754484
x = 1.5059990576402824
Fitness value: 9.28966253566221

Mutating
The best individual:
y = -0.6906013339757919
x = -0.9825449520722032
Fitness value: -2.53728325181307

PORÓWNANIE ALGORYTMÓW

CZAS WYKONANIA ALGORYTMÓW A LICZBA ITERACJI

Algorytmy zostały wywołane na funkcji z treści zadania.
Znaczenie skrótów w tabelce:

GDO – Gradient Descent Optimizer

SGDO – Symboli Gradient Descent Optimizer

GO – Genetic Optimizer

SGO – Symbolic Genetic Optimizer

	GDO	SGDO	GO		SGO	
			Mutacja	Krzyżowanie	Mutacja	Krzyżowanie
10 iteracji	0.0001 s	0.02 s	0.00003 s	0.03 s	0.009 s	0.42 s
50 iteracji	0.0003 s	0.1 s	0.00007 s	0.13 s	0.01 s	2.04 s
100 iteracji	0.0005 s	0.24 s	0.0005 s	0.29 s	0.03 s	4.06 s
500 iteracji	0.0007 s	0.42 s	0.003 s	1.27 s	0.12 s	20.16 s
1000 iteracji	0.001 s	0.35 s	0.007 s	2.54 s	0.26 s	40.12 s

Porównanie GDO a SGDO:

- Czasy Obliczeń:

  SGDO ma wyraźnie większe czasy obliczeń w porównaniu do GDO dla wszystkich ilości iteracji i wynika to z faktu obliczeń symbolicznych.

- Wpływ Iteracji:

  Czasy wykonania zarówno dla GDO, jak i SGDO, rosną wraz z liczbą iteracji, co jest zgodne z oczekiwaniami.

- Ocena Wydajności:

  Wybór między GDO a SGDO zależy od priorytetów, takich jak precyzja obliczeń (SGDO może być bardziej precyzyjne, ale kosztowne obliczeniowo).

Porównanie GO a SGO:

- Czasy Mutacji i Krzyżowania:

  Dla obu algorytmów (GO i SGO), zarówno czasy mutacji, jak i krzyżowania, zwykle rosną wraz z postępem iteracji.

- Wpływ Obliczeń Symbolicznych:

  Czasy krzyżowania dla SGO są znacznie wyższe niż dla GO, co jest związane z kosztami obliczeń symbolicznych.

  Algorytmy symboliczne, takie jak SGO, mogą być bardziej kosztowne obliczeniowo, ale oferują potencjalną precyzję i elastyczność w rozwiązywaniu problemów optymalizacyjnych.

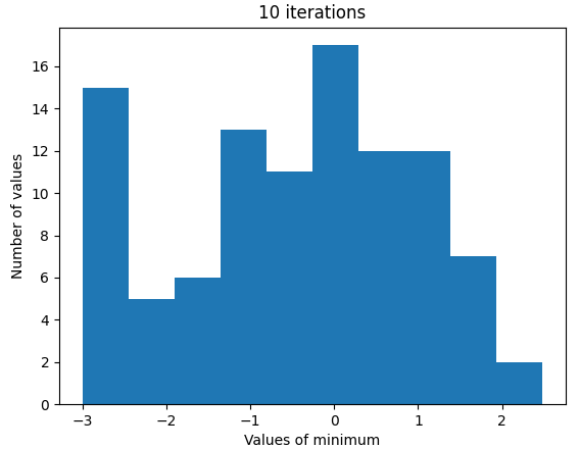
- Ogólna Ocena:

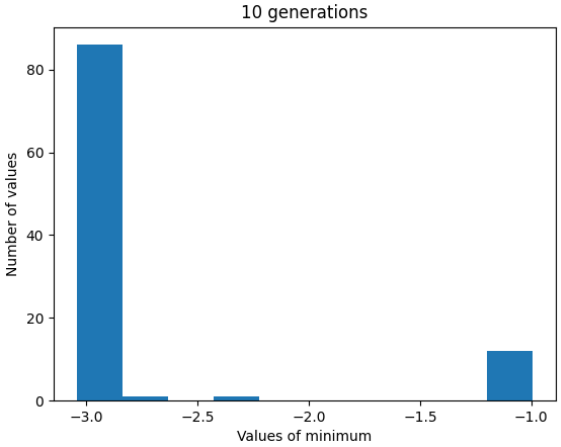
  Wybór między GO a SGO zależy od konkretnego przypadku użycia, priorytetów dotyczących precyzji obliczeń i dostępności zasobów obliczeniowych.

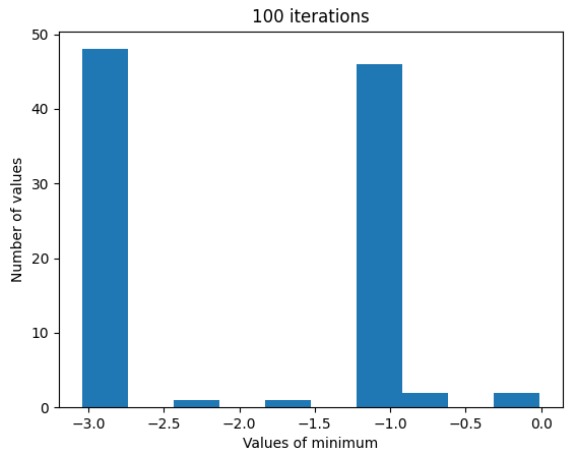
Z ogólnej oceny wynika, że GDO oraz GO są dobrymi algorytmami optymalizacyjnymi.

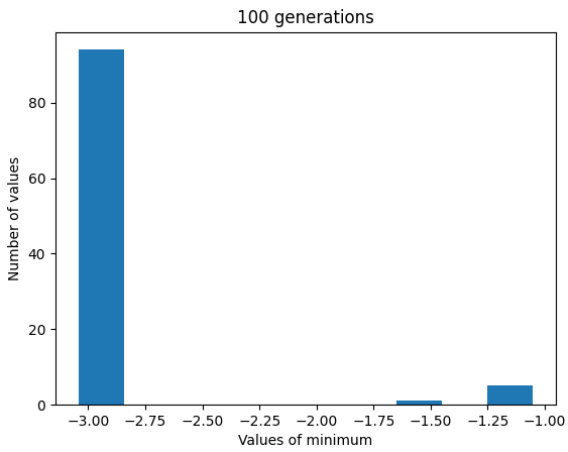
ANALIZA WARTOŚCI MINIMÓW I ICH ILOŚĆ

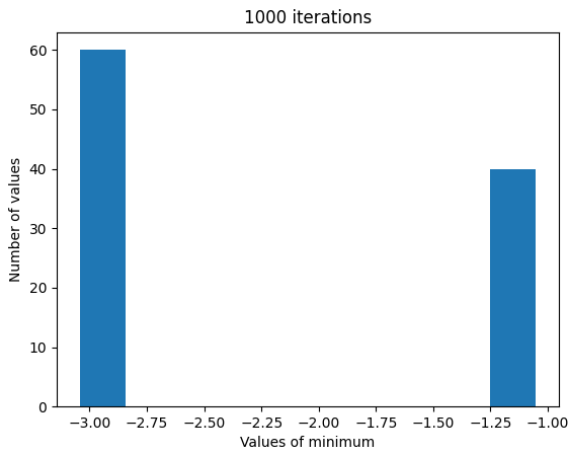
Gradient Descent Optimizer	Genetic Optimizer Krzyżowanie

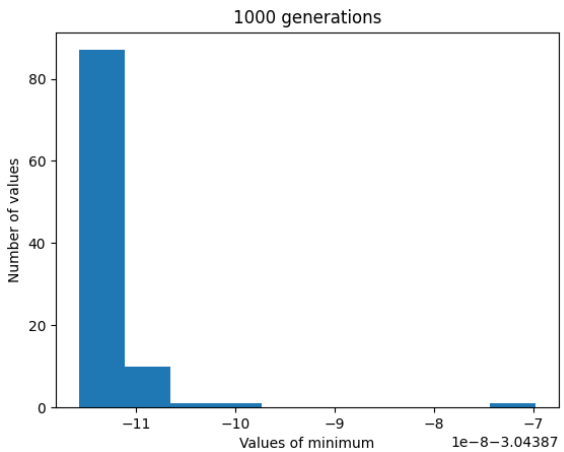




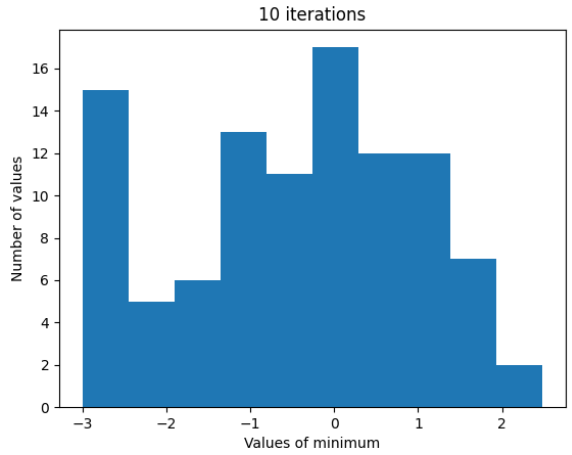
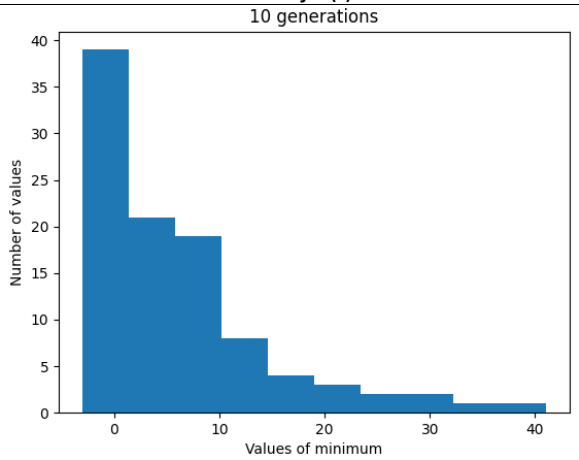
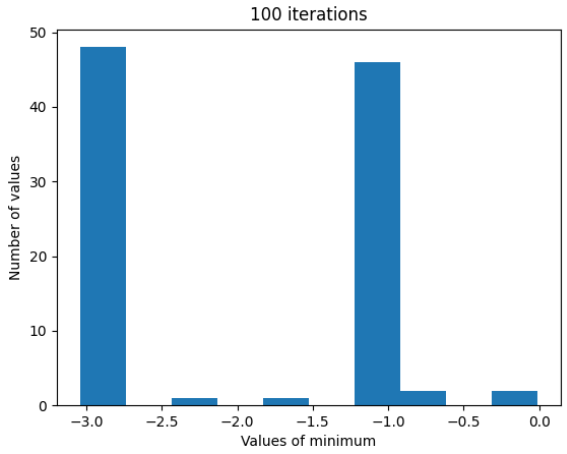
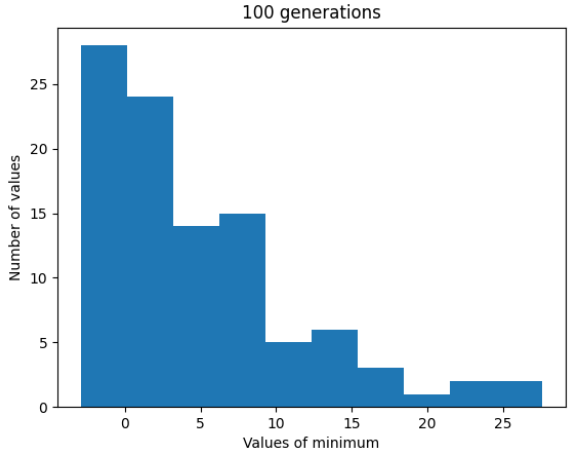
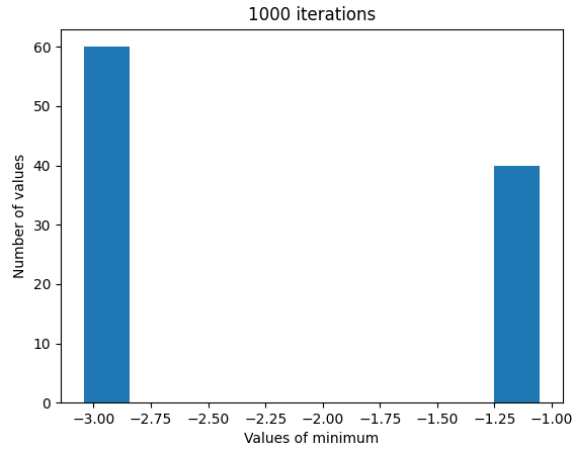
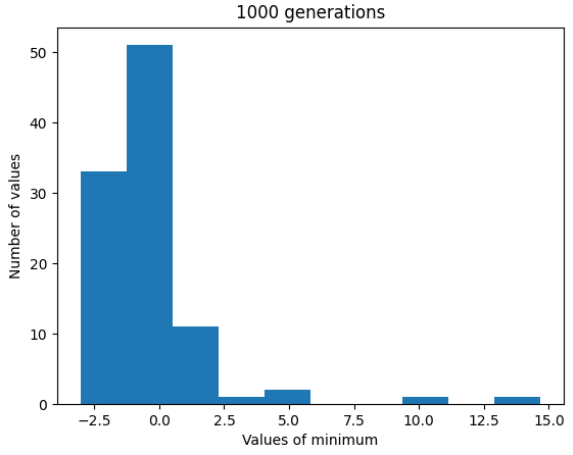




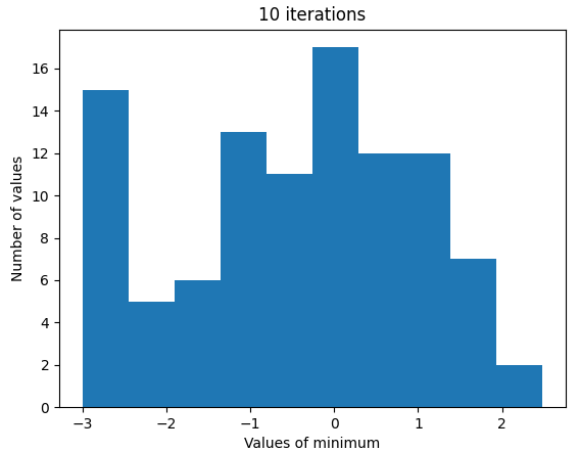
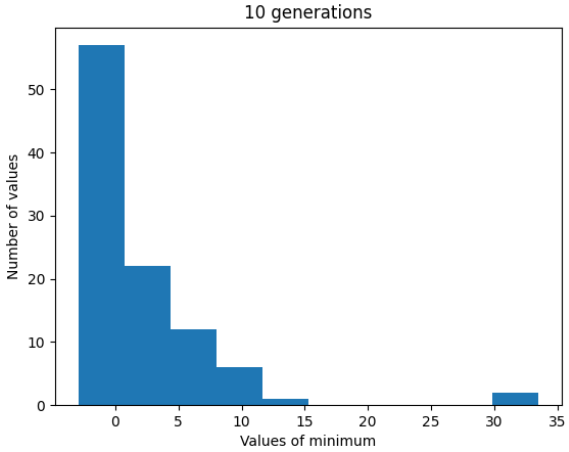
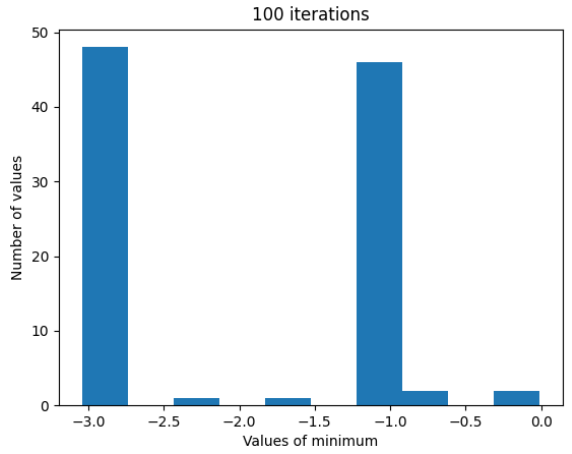
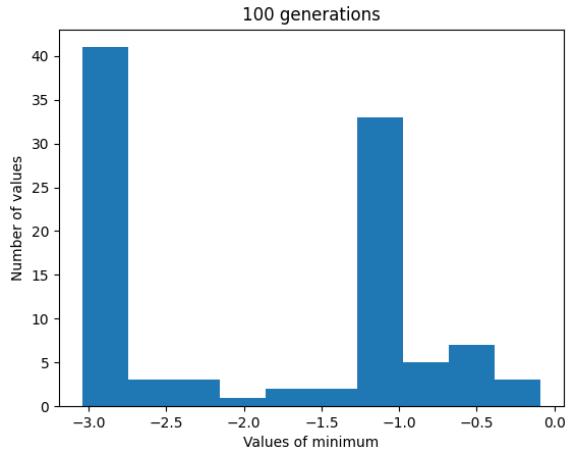
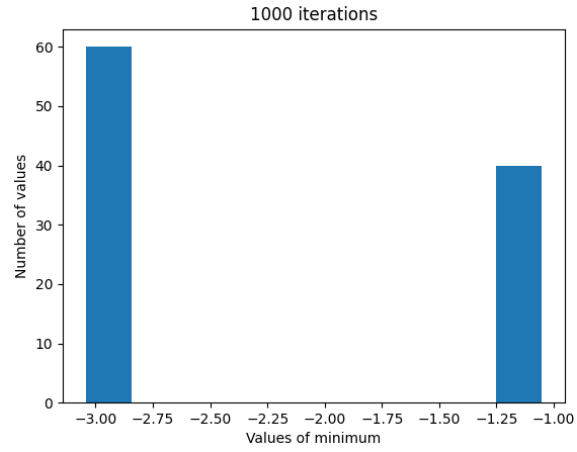
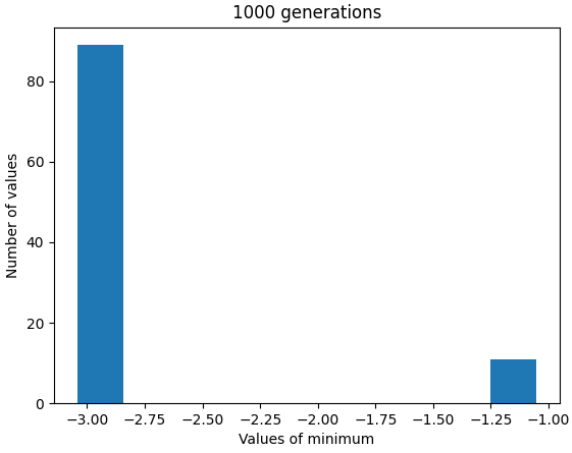




Analizując histogramy możemy dojść do wniosku, że metoda genetycznego krzyżowania jest lepsza od metody gradientowej. Szczególnie dla 100 iteracji/generacji widać, że algorytm genetyczny osiąga częściej minimum globalne funkcji.

Gradient Descent Optimizer	Genetic Optimizer Mutacja (I)
	
	
	

Analizując histogramy możemy dojść do wniosku, że metoda genetycznej mutacji jest gorsza od metody gradientowej. Szczególnie dla 1000 iteracji/generacji widać, że algorytm genetyczny osiąga częściej minimum lokalne funkcji.

Gradient Descent Optimizer	Genetic Optimizer Mutacja (II)
	
	
	

Analizując histogramy możemy dojść do wniosku, że metoda genetycznej mutacji jest lepsza od metody gradientowej. Szczególnie dla 1000 iteracji/generacji widać, że algorytm genetyczny osiąga częściej minimum globalne funkcji. W tym przypadku mutacja osobnika zachodzi zawsze, a nie w warunkach przedstawionych na wykładzie.

Wrażliwość na warunek początkowy

Wrażliwość algorytmów optymalizacyjnych, takich jak algorytm gradientu prostego i algorytm genetyczny, na warunki początkowe może znacząco wpływać na ich skuteczność. Poniżej przedstawiam krótkie omówienie wrażliwości obu tych algorytmów na warunki początkowe:

Algorytm gradientu prostego:

- Wrażliwość na warunki początkowe jest jednym z istotnych aspektów algorytmu gradientu prostego.
- Jeśli algorytm startuje z punktu początkowego bliskiego do minimum lokalnego, to może szybko zbiec do optymalnego rozwiązania.
- Jednakże, jeśli punkt początkowy jest daleko od minimum lokalnego lub jeśli istnieje wiele minimów lokalnych, algorytm może utknąć w jednym z lokalnych minimów, co prowadzi do suboptymalnego rozwiązania.

Algorytm genetyczny:

- Wrażliwość algorytmów genetycznych na warunki początkowe jest zazwyczaj niższa niż w przypadku algorytmu gradientu prostego.
- Algorytmy genetyczne operują na populacji rozwiązań, a nie na pojedynczym punkcie początkowym, co sprawia, że są bardziej elastyczne i zdolne do eksploracji różnych obszarów przestrzeni rozwiązań.
- Jednak efektywność algorytmów genetycznych w dużej mierze zależy od ustawień parametrów takich jak rozmiar populacji, prawdopodobieństwo krzyżowania, i prawdopodobieństwo mutacji.

W praktyce, dobór odpowiednich warunków początkowych dla obu algorytmów jest istotny. Dla algorytmu gradientu prostego, może to oznaczać eksplorację obszaru wokół potencjalnych minimów lokalnych, a dla algorytmu genetycznego, może to oznaczać dostosowanie parametrów tak, aby uwzględnić charakterystykę przestrzeni rozwiązań.

W obu przypadkach, oprócz warunków początkowych, ważne jest także monitorowanie postępu algorytmu i dostosowywanie parametrów w trakcie działania, aby zapewnić skuteczną optymalizację.

Podsumowanie

Problem optymalizacji występuje, gdy dążymy do znalezienia najlepszego rozwiązania spośród dostępnych możliwości, przy jednoczesnym uwzględnieniu pewnych ograniczeń lub kryteriów. Optymalizacja jest powszechnie spotykana w wielu dziedzinach, takich jak inżynieria, ekonomia, nauki przyrodnicze, informatyka czy sztuczna inteligencja.

Algorytm gradientu prostego i algorytm genetyczny są dwoma różnymi podejściami do rozwiązywania problemów optymalizacji, i każdy z nich ma swoje zastosowanie w zależności od charakterystyki problemu.

Algorytm gradientu prostego:

- Algorytm gradientu prostego jest powszechnie stosowany w zadaniach optymalizacji, zwłaszcza w kontekście funkcji różniczkowalnych.
- Działa poprzez iteracyjne kroki w kierunku przeciwnym do gradientu funkcji celu, aby znaleźć minimum lokalne.
- Jest skuteczny w problemach, gdzie funkcja celu jest gładka i różniczkowalna, a także gdy dostępna jest informacja o pochodnych funkcji.

Algorytm genetyczny:

- Algorytm genetyczny jest inspirowany procesami ewolucyjnymi i działa na zasadzie selekcji naturalnej, krzyżowania i mutacji.
- Jest bardziej uniwersalny i może być stosowany w problemach optymalizacji, które mogą mieć skomplikowaną, nieliniową strukturę, a także gdy funkcja celu może być niemiarowa lub trudna do opisania matematycznie.
- Algorytmy genetyczne są bardziej odporniejsze na lokalne minima niż algorytmy gradientu prostego, ponieważ potrafią eksplorować przestrzeń rozwiązań w sposób bardziej globalny.

Podsumowując, wybór pomiędzy algorytmem gradientu prostego a algorytmem genetycznym zależy od charakterystyki konkretnego problemu optymalizacji. Algorytm gradientu prostego jest skuteczny w zadaniach, gdzie mamy do czynienia z funkcją gładką i różniczkowalną, podczas gdy algorytm genetyczny może być lepszym wyborem, gdy struktura problemu jest bardziej złożona, a przestrzeń poszukiwań jest trudniejsza do zbadania. W praktyce często stosuje się też kombinacje różnych algorytmów optymalizacyjnych w celu osiągnięcia lepszych wyników.