Prawo dżungli
czyli
ewolucja w działaniu

Celem projektu jest implementacja prostej wersji algorytmu minimalizacji bez ograniczeń funkcji 2 zmiennych opartego na analogii do działania ewolucji, czyli tzw. algorytmu ewolucyjnego.

Możecie przy tym wykorzystać gotową procedurę obliczającą wartość funkcji celu w obszarze $[-1,1] \times [-1,1]$ oraz skrypt rysujący mapę poziomicową tejże funkcji.

Ponieważ w zapisie algorytmu będziemy często wykorzystywać liczby losowe, przyjmijmy konwencję, że r oznacza liczbę losową o rozkładzie równomiernym w przedziale $[0,1]$, inną przy każdym swoim pojawieniu się, tzn. faktycznie wywołanie funkcji `rand(1,1)`.

Ewolucja gatunku najogólniej mówiąc polega na następujących procesach:

1. Pojawienie się populacji startowej, np na izolowanej wyspie.
2. Dobór rodziców w pary.
3. Rozmnażanie polegające na mieszaniu genów rodziców.
4. Mutacje polegające na przypadkowej zmianie genotypu, zachodzące z pewnym prawdopodobieństwem $0 < p < 1$.
5. Selekcja polegająca na eliminacji najgorzej przystosowanych osobników z puli potencjalnych rodziców.

W naszym przypadku osobniki to punkty na płaszczyźnie, a ich genotyp to wartości liczbowe współrzędnych: $\boldsymbol{x}_k = [x_{1k}, x_{2k}]$. Miarą ich przystosowania jest wartość funkcji celu, im mniejsza tym lepiej, gdyż przeprowadzamy minimalizację.

Parametrami wejściowymi naszego programu będą: N – liczebność populacji, p – prawdopodobieństwo mutacji oraz M – ilość generacji.

Rozmnażanie dla pary rodziców $\boldsymbol{x}_f$, $\boldsymbol{x}_m$ będzie polegało na utworzeniu leżącego gdzieś na odcinku pomiędzy nimi potomka (tzw. mieszanie arytmetyczne):

$$t := r; \qquad \boldsymbol{x}_c := (1-t)\boldsymbol{x}_f + t\boldsymbol{x}_m;$$

Mutacja, zachodząca z prawdopodobieństwem p będzie polegała po prostu na zaburzeniu genotypu potomka o mała wartość, np

$$\text{jeżeli} \quad r < p \quad \text{to} \quad \boldsymbol{x}_c := \boldsymbol{x}_c + 0.2[0.5 - r, 0.5 - r];$$

Dobór par rodziców pozostawiamy Waszej inwencji. Dla przykładu taką regułą może być zasada, że osobnik najlepiej przystosowany ma potomstwo z każdym z pozostałych (władca haremu – patrz jelenie, słonie morskie itp.) produkując $N-1$ potomków, z których każdy następnie może mutować. W rezultacie powstaje populacja zawierająca $N+(N-1)$ osobników (rodzice + potomstwo).

W ramach selekcji będziemy eliminowali osobników gorzej przystosowanych, zostawiając tylko N najlepszych, tak że w każdej generacji liczebność populacji rodziców pozostaje taka sama.

Formalnie algorytm ten można zapisać następująco:

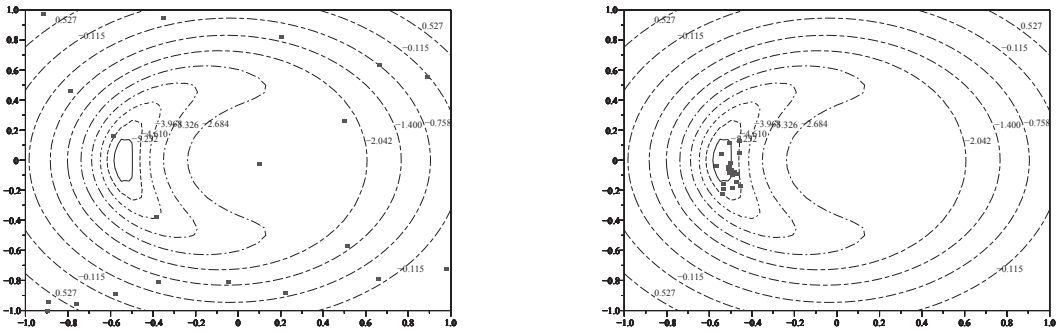
- Wczytaj dane.
- Wygeneruj losowo początkową populację N punktów z obszaru $[-1,1]\times[-1,1]$.
- Powtórz M razy
  - wyświetl mapę i położenie populacji rodziców;
  - dobierz pary rodziców;
  - wygeneruj potomstwo;
  - każdego potomka poddaj z prawdopodobieństwem p mutacji;
  - wybierz N najlepiej przystosowanych osobników i uczyń z nich nową populację rodziców;
- Wyświetl mapę i położenie najlepszego punktu oraz podaj jego współrzędne wraz z wartością funkcji celu.

Uwagi realizacyjne.
– Dla szeregowania wartości elementów wektora w kolejności rosnącej służy funkcja wbudowana `[fsorted,rows]=sort(f)` (patrz pomoc do programu).
– Ponieważ kolejne generacje będą prawdopodobnie migać na ekranie zbyt szybko, jako komendy opóźniającej można użyć np

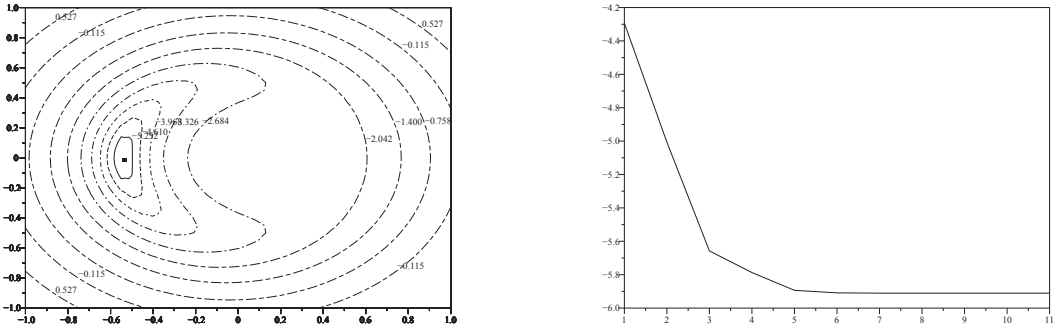
```
A=(rand(300,300))^(-1);
```

gdzie wymiar macierzy zależy od szybkości komputera.

Przykładowe wyniki dla $N=20$, $M=10$, $p=0.2$ oraz funkcji `rogalikowa dolina`.



Rysunek 1: Pierwsza (na lewo) i trzecia generacja na tle mapy poziomicowej.



Rysunek 2: Dziesiąta generacja (po lewej) i przebieg najlepszej wartości funkcji celu w kolejnych generacjach.