Grzegorz Topór 337294

WSI LAB1 - ALGORYTMY EWOLUCYJNE I GENETYCZNE

1. Opis zaimplementowanego algorytmu

Solver do znalezienia minimum globalnego podanej funkcji używa strategii ewolucyjnej (1+1) oraz posiada dwa warunki stopu: maksymalna liczba wywołań funkcji zależna od wymiarowości wektora zmiennych (n x 100) oraz maksymalna liczba wywołań funkcji bez znalezienia nowej, lepszej wartości funkcji. Solver zwraca dictionary gdzie: dict[t]= [f(x_t), best-so-far].
jak działa algorytm w dużym uproszczeniu:

1. stwórz nowego potomka:
   x' ← x + moc adaptacji * N(0, I) (rozklad gaussa)

2. Oblicz wartość funkcji:
   q ← q(x')

3. Jeśli q' < najlepsza_wartość:
   najlepsza_wartość ← q

4. co okres adaptacji mocy::
   jeśli (liczba_popraw / K) ≥ 1/5:
      σ ← σ * 1.2 # zwiększ krok
   w przeciwnym razie:
      σ ← σ * 0.82 # zmniejsz krok
5. liczba_iteracji ← liczba_iteracji + 1

2. Planowane eksperymenty numeryczne

Eksperymenty skupiają się na sprawdzaniu zależności mocy siły adaptacji w stosunku do możliwości eksploracji. Program tworzy 12 wykresów, z czego 6 pokazuje średnie wartości best so far w zależności od iteracji z określonej liczby uruchomień (R=10). Pozostałe 6 wykresów pokazuje zestawienie pojedynczych uruchomień funkcji oraz zachowanie best so far value i f(x_t). Eksperymenty zostały przeprowadzone na trzech funkcjach (kwadratowa, ackley'a i rosenbrocka) dla wymiarowości n = 10, 30. Moc mutacji = 0.5, 1, 5.(oraz 2 dla funkcji ackley'a). Liczba wywołań funkcji została ograniczona do 200. Moc adopotowana co 10 iteracji.

3. Opis uzyskanych wyników i wnioski

Funkcja kwadratowa:

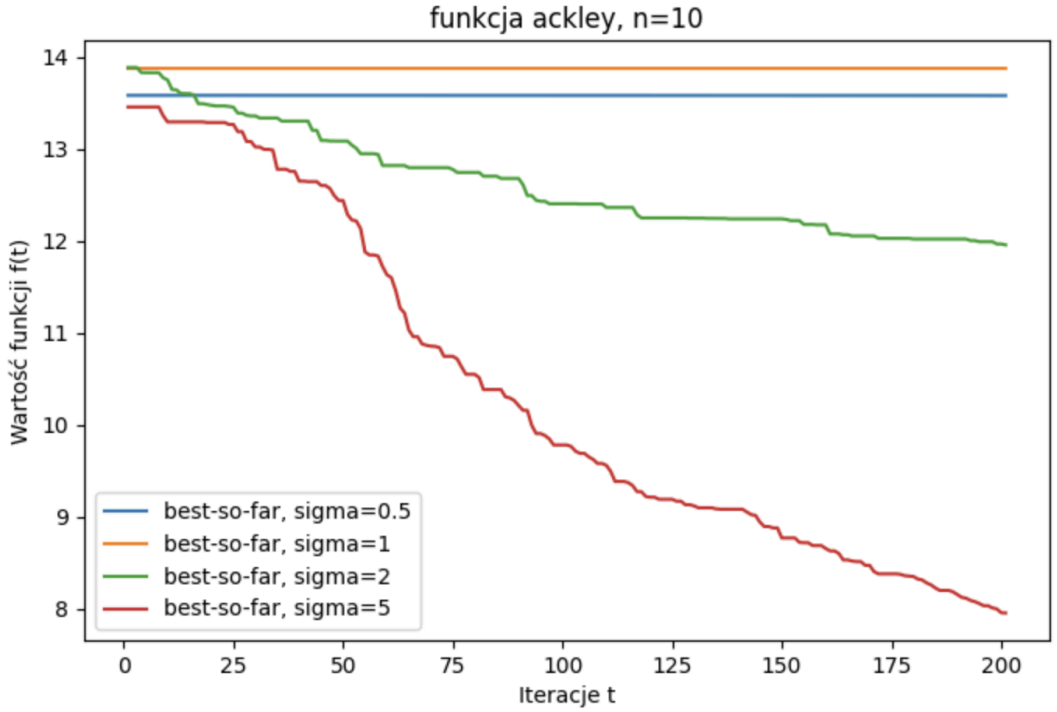
funkcja kwadratowa jest prosta dla algorytmu (1+1), nie utyka w minimum lokalnym i znajduje globalne (nawet dla n=30)

Funkcja rosenbrocka:

funkcja rosenbrocka jest dość prosta dla algorytmu (1+1), nie utyka w minimum lokalnym i znajduje globalne (nawet dla n=30)

Funkcja ackleya:

funkcja ackleya jest najtrudniejsza i najciekawiej widać problem strategii (1+1) i jego wady wynikające z braku populacji. Dla wymiarowości n=10:



widać że moc 0.5 i 1 jest zbyt mała i funkcja utyka w minimach lokalnych, natomiast 2 i 5 wychodzą z nich i znajdują lepsze minima.

Dla wymiarowości n=30, algorytm całkowicie utyka w minimach lokalnych i nie wychodzi z nich, niezależnie od sigmy. Aby znaleźć lepsze minima należałoby użyć algorytmu z populacją, która gwarantuje lepszą eksploracje.