Strategie Ewolucyjne do optymalizacji funkcji testowych

Joanna Kołodziejczyk

1 Problem do rozwiązania

Celem laboratorium jest implementacja algorytmu $(\mu/\varrho, \lambda)$-ES i $(\mu/\varrho + \lambda)$-ES.

Optymalizacja ma zostać wykonana dla testowego problemu optymalizacji globalnej w przestrzeni liczb rzeczywistych. Poniżej znajduje się lista funkcji, na których należy przeprowadzić testy:

1. RASTRIGIN FUNCTION
   - `http://www.sfu.ca/~ssurjano/rastr.html`,
   - $domain = [-5.12, 5.12]$
   - Globalne minimum $f(0, 0, \ldots, 0) = 0$
   - Wymiarowość: $\{2, \ldots, d\}$
2. GRIEWANK FUNCTION
   - `http://www.sfu.ca/~ssurjano/griewank.html`,
   - $domain = [-600, 600]$
   - Globalne minimum $f(0, 0, \ldots, 0) = 0$
   - Wymiarowość: $\{2, \ldots, d\}$
3. SPHERE FUNCTION
   - `http://www.sfu.ca/~ssurjano/spheref.html`,
   - $domain = [-5.12, 5.12]$
   - Globalne minimum $f(0, 0, \ldots, 0) = 0$
   - Wymiarowość $\{2, \ldots, d\}$
4. ZAKHAROV FUNCTION
   - `http://www.sfu.ca/~ssurjano/zakharov.html`,
   - $domain = [-5, 10]$
   - Globalne minimum $f(0, 0, \ldots, 0) = 0$
   - Wymiarowość: $\{2, \ldots, d\}$

5. EASOM FUNCTION
    - `http://www.sfu.ca/~ssurjano/easom.html`,
    - $domain = [-100, 100]$
    - Globalne minimum $f(\pi, \pi) = -1$
    - Wymiarowość: 2
6. STYBLINSKI-TANG FUNCTION
    - `http://www.sfu.ca/~ssurjano/stybtang.html`,
    - $domain = [-5, 5]$
    - Globalne minimum $f(-2.903534, \ldots, -2.903534) = -39.16599d$, gdzie d jest wymiarowością problemu.
    - Wymiarowość: $\{2, \ldots, d\}$

2 Algorytm

2.1 Pseudokod algorytmu $(\mu/\varrho, \lambda)$-ES i $(\mu/\varrho + \lambda)$-ES.

1: $population \leftarrow initialize\,(\mu)$
2: $g \leftarrow 0$
3: **while** $terminal\_condition$ **not** $TRUE$ **do**
4: $\quad offspring \leftarrow prepare\,(\lambda)$
5: $\quad$ **for** $o = 1$ **to** λ **do**
6: $\quad\quad parents \leftarrow marriage\,(population, \varrho)$
7: $\quad\quad s \leftarrow s\_recombination\,(parents)$
8: $\quad\quad x \leftarrow x\_recombination\,(parents)$
9: $\quad\quad s \leftarrow s\_mutation\,(s)$
10: $\quad\quad x \leftarrow x\_mutation\,(x, s)$
11: $\quad\quad f \leftarrow fitness\_function\,(x)$
12: $\quad\quad offspring_o \leftarrow$ **struct** (x, s, f)
13: $\quad$ **end for**
14: $\quad population \leftarrow selection\,(offspring, \mu)$ lub $selection\,(offspring + \mu)$
15: $\quad g \leftarrow g + 1$
16: **end while**
17: **return** *population*

2.2 Istotne elementy algorytmu

1. **Kodowanie osobnika** i inicjalizacja:
    - x — wektor cech osobnika (współrzędne punktu w przestrzeni $\Re^n$) - zakres z dziedziny zadanej funkcji (*domain*). Liczba elementów w wektorze to wymiarowość funkcji testowej d.

- s — wektor odchyleń standardowych (endogenny parametr mutacji). Wartości z przedziału $\{0.1..5\}$. Liczba elementów w wektorze to wymiarowość funkcji testowej d.
- f — poziom przystosowania osobnika - zadana funkcja testowa do optymalizacji lub inaczej $fitness\_function$. f skalarem, wyliczoną wartością ze wzoru na funkcję testową.

2. **Parametry początkowe algorytmu**

   Przed uruchomieniem algorytmu należy arbitralnie podać wielkości parametrów:

   - μ —liczba osobników w populacji rodzicielskiej: zalecane wartości od 1 do 100
   - ϱ — liczba osobników w puli rodziców do reprodukcji: zalecane wartości od 1 do μ
   - λ — liczba osobników potomnych: zalecane wartości –> patrz wyjaśnienie poniżej.

   W przypadku algorytmu $(\mu/\varrho, \lambda)$-ES powinna obowiązywać zależności:

   $$\mu < \lambda$$

   Ponieważ zgodnie z algorytmem $(\mu/\varrho, \lambda)$-ES w procesie selekcji uczestniczą tylko i wyłącznie osobniki potomne, ich liczność powinna byś większa od dotychczasowej liczby wszystkich osobników populacji. W przeciwnym wypadku liczność populacji w każdej kolejnej iteracji algorytmu ulegałaby zmniejszeniu.

   W skrajnym przypadku, gdy $\mu = \lambda$ następne pokolenie będą tworzyć wszystkie osobniki potomne.

   W przeciwieństwie do algorytmu $(\mu/\varrho, \lambda)$-ES, algorytm $(\mu/\varrho+\lambda)$-ES nie wnosi żadnych teoretycznych ograniczeń na liczbę generowanych potomków.

3. Warunek zakończenia $teminal\_condition$ powinien sprawdzać zadaną maksymalną liczbę kroków algorytmu i osiągnięcie znanego dla funkcji testowej minimum globalnego przy pewnym założonym błędzie np. $fitness\_function(osobnik) - f\_globalne < 0,0001$.

4. **Wybór puli rodziców do krzyżowania**

   Procedura wybiera z puli rodzicielskiej określoną parametrem pulę do krzyżowania.

   function $parents = marriage\,(population, \varrho)$

   1: $parents \leftarrow prepare\,(\varrho)$
   2: $selected\_individuals \leftarrow randperm\,(length\,(population))$
   3: **for** $i = 1$ **to** ϱ **do**
   4: $\quad parents_i \leftarrow population_{selected\_individuals_i}$
   5: **end for**
   6: **return** $parents$

5. **Pseudokod algorytmu rekombinacji parametrów endogennych.**

   Algorytm realizuje mechanizm krzyżowania uśredniającego dla osobników kodowanych wartościami rzeczywistymi.

   function $s = s\_recombination\,(parents)$

   1: $n \leftarrow length\,(parents.s)$
   2: $s \leftarrow prepare\,(n)$
   3: **for** $i = 1$ **to** n **do**
   4: $\quad s_i \leftarrow mean\_s\,(parents, i)$
   5: **end for**
   6: **return** s

   gdzie:
   $mean\_s\,(parents, i)$ jest średnią wartością ze wszystkich pobranych z rodziców.

6. **Pseudokod algorytmu rekombinacji uśredniającej cech osobników.**

   function $y = x\_recombination\,(parents)$

   1: $n \leftarrow length\,(parents.y)$
   2: $y \leftarrow prepare\,(n)$
   3: **for** $i = 1$ **to** n **do**
   4: $\quad x_i \leftarrow mean\_x\,(parents, i)$
   5: **end for**
   6: **return** y

   gdzie:

   $mean\_x\,(parents, i)$ jest średnią wartością ze wszystkich pobranych z rodziców.

7. **Pseudokod algorytmu mutacji parametrów endogennych**

   function $out\_s = s\_mutation\,(s)$

   1: $n \leftarrow length\,(s)$
   2: $out\_s \leftarrow prepare\,(n)$
   3: $c \leftarrow 1$
   4: $\tau_0 \leftarrow e^{\frac{c}{\sqrt{2n}} \cdot \aleph(0,1)}$
   5: $\tau \leftarrow \frac{c}{\sqrt{2\sqrt{n}}}$
   6: **for** $i = 1$ **to** n **do**
   7: $\quad out\_s_i \leftarrow \tau_0 \cdot s_i \cdot e^{\tau \cdot \aleph(0,1)}$
   8: **end for**
   9: **return** out_s

   gdzie:

   - $\aleph\,(0, 1)$ to liczba losowa z rozkładu normalnego o wartości oczekiwanej 0 i odchyleniu standardowym 1

- zmiennej c przypisujemy wartość 1. Jest to akceptowalna wartość dla algorytmu (10,100)-ES.

8. **Pseudokod algorytmu mutacji eliptycznej cech osobników.**

   function $out\_x = x\_mutation\,(x, s)$

   1: $n \leftarrow length\,(y)$
   2: $out\_x \leftarrow prepare\,(n)$
   3: **for** $i = 1$ **to** n **do**
   4: $\quad out\_x_i \leftarrow x_i + s_i \cdot \aleph\,(0, 1)$
   5: **end for**
   6: **return** out_x

   gdzie
   $\aleph\,(0, 1)$ to liczba losowa z rozkładu normalnego o wartości oczekiwanej 0 i odchyleniu standardowym 1

9. Selekcja

   Selekcja wybiera μ osobników z początku posortowanej rosnąco listy potomków (wariant selekcji *comma*) lub z posortowanej rosnąco połączonej listy populacji poprzedniej i nowej (wariant selekcji *plus*). Dzięki temu wybieramy osobniki o najlepszym (najmniejszym) przystosowaniu. Pozostałe zostają zapomniane. Ten typ selekcji nazywa się *truncation selection*.

   Selekcja realizowana w algorytmie $(\mu/\varrho, \lambda)$-ES:

   function $new\_population = selection\,(offspring, \mu)$

   1: $new\_population \leftarrow sort\,(offspring, ascd)$
   2: $new\_population \leftarrow new\_population\,(1 : \mu)$
   3: **return** $new\_population$

   Selekcja realizowana w algorytmie $(\mu/\varrho + \lambda)$-ES:

   function $new\_population = selection\,(offspring, population, \mu)$

   1: $new\_population \leftarrow sort\,([offspring, population]\,, ascd)$
   2: $new\_population \leftarrow new\_population\,(1 : \mu)$
   3: **return** $new\_population$

3 Wymagania dla implementacji

Język implementacji algorytmu jest dowolny.
Wymagane elementy programu:

- Do optymalizacji wybrać jedną z podanych na początku funkcji testowych. Wczytać wszystkie parametry z nią związane. Wymiarowość funkcji d jest parametrem programu.

- Zaimplementować algorytm zgodnie z wytycznymi podanymi w punkcie 2 i 2.1.

- Program po wykonaniu optymalizacji powinien wyświetlać:
  - położenie znalezionego przez algorytm optimum i wartość przystosowania/wartość funkcji w tym punkcie
  - numer generacji, w której znaleziono rozwiązanie
  - wykres zmienności przystosowania najlepszego osobnika w generacjach
  - wykres zmienności średniego przystosowania populacji w generacjach
  - Nieobowiązkowo:
    * odległość od znanego globalnego optimum (miara Euklidesa)
    * rozproszenie populacji (średni dystans pomiędzy osobnikami w populacji) w generacjach

Każde własne (rozsądne) propozycje modyfikacji lub wykorzystane z literatury zmiany w algorytmie lub reprezentacji lub prezentacji wyników ponad te, wymagane w instrukcji, będą dodatkowo punktowane.

3.1 Przekazanie programu

Kod z rozwiązaniem proszę wysłać na mail: jkolodziejczyk@zut.edu.pl