|  |  |
| --- | --- |
| **PSZT Projekt 1 Przeszukiwanie** | |
| **Tytuł projektu** | RB.S8 A może ślub? (bogata rodzina) |
| **Wykonujący projekt** | |
| Mateusz Bajdak | Nr albumu: 277125 |
| Tomasz Indeka | Nr albumu: 293457 |

# Interpretacja treści zadania

Należy zmodyfikować klasyczny algorytm ewolucyjny losowo łącząc w pary osobniki w populacji. Dodatkowo – wartość funkcji dopasowania (q) dla osobników w parze (i oraz j):  
 .

# Wkład autorów

Mateusz Bajdak:

* Parser opcji wywołania programu
* Architektura aplikacji
* Interpretacja i opis wyników
* Implementacja funkcji testowych

Tomasz Indeka:

* Implementacja algorytmu ewolucyjnego:
  + selekcja
  + replikacja
  + mutacja
  + krzyżowanie
* Prezentacja statystyk

# Wykorzystane narzędzia i biblioteki

Projekt został napisany w programie PyCharm Community Edition, w języku programowania   
Python (v3.7) z wykorzystaniem biblioteki zewnętrznej **matplotlib** oraz bibliotek standardowych,  
tj. **math**, **random**, **functools, statistics**.

# Instrukcja wykonania programu

* Na komputerze, na którym wykonywany będzie program powinien być zainstalowany Python w wersji 3.7 lub wyższej,
* Będąc w katalogu głównym aplikacji (tam gdzie znajduje się plik *main\_app.py*) należy wykonać następujące polecenie: *python main\_app.py.*

Algorytm można skonfigurować korzystając z dostępnych opcji (pełna nazwa (skrót)):

* iterations (i) – liczba iteracji (w kontekście zadania – budżet) głównej pętli algorytmu
* function (f) – rodzaj funkcji do przetestowania, do wyboru:
  + griewank (<http://benchmarkfcns.xyz/benchmarkfcns/griewankfcn.html>)
  + cigar (<https://al-roomi.org/benchmarks/unconstrained/n-dimensions/164-bent-cigar-function>)
  + branin-1 (<http://infinity77.net/global_optimization/test_functions_nd_B.html>)
  + bird (<http://benchmarkfcns.xyz/benchmarkfcns/birdfcn.html>)
  + alpine-1 (<http://benchmarkfcns.xyz/benchmarkfcns/alpinen1fcn.html>)
  + ackley (<http://benchmarkfcns.xyz/benchmarkfcns/ackleyfcn.html>)
* dimensions (d) – liczba wymiarów funkcji celu
* crossover\_p (C) – prawdopodobieństwo krzyżowania
* cardinality (n) – liczność populacji
* attempts (a) – liczba wykonań „pełnego obiegu” algorytmu
* mut\_sigma (S) – wartość odchylenia standardowego rozkładu normalnego przy mutowaniu
* x\_min (m) – ograniczenie od dołu wartości osobników populacji początkowej
* x\_max (M) – ograniczenie od góry wartości osobników populacji początkowej

Daną opcje można zastosować poprzez dodanie po nazwie programu: --<pełna nazwa>=wartość lub  
-<skrót>=wartość lub -<skrót>wartość. Przykładowa komenda uruchomienia programu:

*python main\_app.py –iterations=1000 -d2 -C0.5 -n=200 -a25 -m-100 -M100 -fcigar*

# Cele i tezy przeprowadzonych badań

Celem przeprowadzonych badań jest porównanie i wskazanie ewentualnych różnic w wynikach zwróconych przez wymienione w rozdziale 1 algorytmy.

Charakterystyka zmodyfikowanego algorytmu może spowodować, że słabsze osobniki, które w przypadku klasycznym zostałyby „uśmiercone”, przeżyją dzięki silniejszemu partnerowi – mają szansę pokonać słabszy obszar funkcji celu.

Zakładamy, że zmodyfikowany algorytm będzie znajdował optimum wolniej, ale finalnie osiągnie wynik co najmniej tak samo dobry jak standardowy algorytm ewolucyjny.

# Wyniki eksperymentów

Przedstawione najlepsze rezultaty z 25 prób dla 6 funkcji testowych (w wariantach dwuwymiarowych), przy budżecie 500/1000, liczności populacji 200, prawdopodobieństwu krzyżowania 50% oraz odchyleniu standardowemu dla rozkładu normalnego w mutacji 5.

## Funkcja Ackleya (minimum: 0) - .

Minimum dla algorytmu tradycyjnego: **0.00**

Minimum dla algorytmu zmodyfikowanego: **0.00**

|  |  |
| --- | --- |
| Obraz zawierający tekst, mapa  Opis wygenerowany automatycznie |  |
|  | |

## Funkcja Alpine01 (minimum: 0) - .

Minimum dla algorytmu tradycyjnego: **0.00**

Minimum dla algorytmu zmodyfikowanego: **0.00**

|  |  |
| --- | --- |
|  |  |
|  | |

## Funkcja Bird (minimum: -106.76) - .

Minimum dla algorytmu tradycyjnego: -**106.76**

Minimum dla algorytmu zmodyfikowanego: -**106.76**

|  |  |
| --- | --- |
|  |  |
|  | |

## Funkcja Branin01 (minimum: 0.39) - .

Minimum dla algorytmu tradycyjnego: **0.39**

Minimum dla algorytmu zmodyfikowanego: **0.39**

|  |  |
| --- | --- |
|  |  |
|  | |

## Funkcja Cigar (minimum: 0) - .

Minimum dla algorytmu tradycyjnego: **0.00**

Minimum dla algorytmu zmodyfikowanego: **0.00**

|  |  |
| --- | --- |
|  |  |
|  | |

## Funkcja Griewank (minimum: 0) - .

Minimum dla algorytmu tradycyjnego: **0.00**

Minimum dla algorytmu zmodyfikowanego: **0.00**

|  |  |
| --- | --- |
|  |  |
|  | |

# Omówienie wyników

Wyniki obu algorytmów zbliżone. Widoczna różnica w szybkości dochodzenia do wartości  
minimalnej – algorytm klasyczny w większości przypadków szybciej osiągał mniejsze wartości funkcji celu. Widoczny również wolniejszy spadek wariancji wartości funkcji celu w kolejnych iteracjach.

# Wnioski

Dla wybranych funkcji celu nie udało się uzyskać sytuacji, w której widoczna byłaby wyższość algorytmu zmodyfikowanego nad klasycznym – algorytmy sprawiły się bardzo podobnie z widoczną lekką przewagą po stronie wersji klasycznej. Oba pomyślnie znajdują minimum zadanej funkcji, choć algorytm standardowy zdaje się to robić szybciej. Jest to spowodowane przez utrzymywanie przez testowany algorytm informacji o niektórych punktach z niekorzystnym położeniem w przestrzeni, ale posiadających partnera o dobrym dopasowaniu. Standardowy algorytm nie przechowuje informacji o źle dopasowanych osobnikach przez co dzięki mutacji w okolicy optimum funkcji jest w stanie szybciej znaleźć rozwiązanie.

Przechowywanie informacji o źle dopasowanych osobnikach może prowadzić do znajdowania minimów znajdujących się za dużymi słabymi obszarami funkcji celu. Z naszych obserwacji wynika jednak, że informacja o źle dopasowanych osobnikach była często tracona w wyniku krzyżowania i znajdowania lepszych rozwiązań, przez co para ze słabym osobnikiem była gorzej dopasowana i odrzucana. Niemniej jednak na wykresach można zauważyć, że w przypadku testowanego algorytmu wariancja otrzymanych funkcji celu jest większa co może świadczyć o przeszukiwaniu większej przestrzeni.