Mateusz Kołacz, 336360

**WSI – CWICZENIA 2**

## Zadanie

Zaimplementować klasyczny algorytm ewolucyjny z selekcją turniejową i sukcesją generacyjną, bez krzyżowania. Dostępny budżet to 10000 ewaluacji funkcji celu. Optymalizujemy funkcje numer 2 i 13 z CEC 2017 w 10 wymiarach. Ograniczenia kostkowe przestrzeni to -100, 100.

## Uwaga

* Ponieważ algorytmy ewolucyjne wykorzystują losowość, nie wolno wyciągać wniosków na podstawie wyników pojedynczego uruchomienia. Należy porównywać średnie z co najmniej 25 uruchomień. W celu uzyskania pełnego obrazu, w tabelach z wynikami oprócz średniej zamieszcza się odchylenie standardowe, oraz najlepszy i najgorszy ze znalezionych wyników.
* Nie należy podawać zbyt wielu miejsc po przecinku.
* Liczbę iteracji wyliczmy w kodzie: tmax=budżet/mu, gdzie mu to aktualnie przyjęta liczba osobników w populacji.
* Oczywiście optymalne wartości parametrów są zależne od siebie i analizując parametry oddzielnie możemy ich nie znaleźć. Na tym ćwiczeniu nie szukamy jednak optymalnych parametrów, szukamy rozsądnych, wyrabiamy podstawowe intuicje. Na początek przyjmujemy rozsądne wartości parametrów, np. sigma=3. Na podstawie eksperymentów znajdujemy najlepszy dla danego zadania rozmiar populacji (np. sprawdzamy kolejne potęgi 2). Mając rozmiar populacji badamy wpływ siły mutacji, tj. wykonujemy eksperymenty i tworzymy kolejną tabelę.

## Pytania (odpowiedzi proszę umieścić w dokumencie tekstowym)

Zbadać wpływ:

* liczby osobników w populacji,
* siły mutacji,

na jakość uzyskanych rozwiązań.  
Po dobraniu parametrów należy sprawdzić czy wyniki poprawią się po użyciu pięciokrotnie większego budżetu.

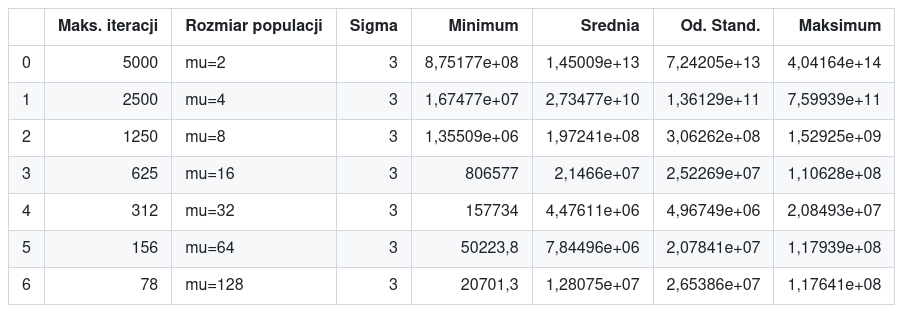
Jakie wnioski można wyciągnąć na podstawie wyników tego ćwiczenia?

**Badanie wpływu liczby osobników w populacji**

Na potrzeby ćwiczenia zaimplementowano algorytm ewolucyjny bez krzyżowania, z reprodukcją turniejową oraz sukcesją generacyjną. Działanie algorytmu testowano w pierwszym etapie dla dwóch funkcji z CEC2017 (odpowiednio f2 i f13) dla budżetu o wysokości 10000 z stałym parametrem SIGMA=3 oraz dla różnych rozmiarów populacji. Postanowiono wziąć pod obserwację kolejne potęgi liczby 2. Z uwagi na losowość, wnioski wyciągano na podstawie zagregowanych statystyk dla 30 uruchomień każdej kolejnej wartości parametru „mu”.

Wyniki przedstawiono w tabelach poniżej:

**Eksperyment 1.1 : badanie wpływu rozmiaru populacji dla funkcji f2**

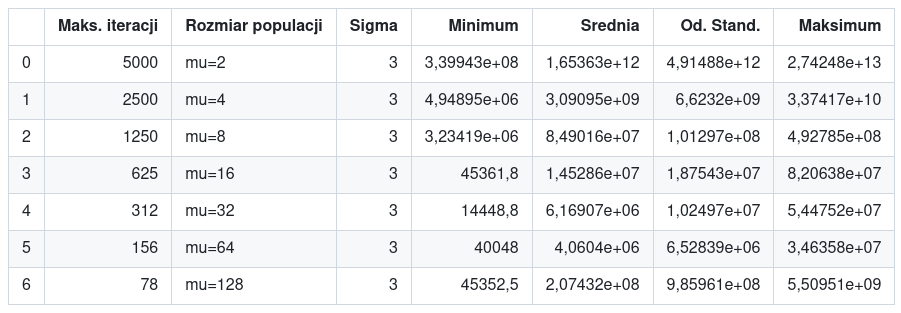


Jak można zauważyć, najlepiej dla funkcji f2 sprawdził się rozmiar populacji mu=32. Świadczy o tym najniższa średnia (4,47611 \* 10^6) a także najniższa wartość maksymalna (2,0849 \* 10^7).

Wartość minimalna 157734 pozostawia jednakże wiele do życzenia, dla parametrów mu równych 64 i 128 można było uzyskać wyniki w tej statystyce niższe nawet 3- i 7-krotnie, aczkolwiek średnia dla tych parametrów były większe i tym samym dalej rozwiązania optymalnego.

Trochę odmienną sytuację ukazują wyniki dla funkcji f13.

**Eksperyment 1.2 : badanie wpływu rozmiaru populacji dla funkcji f13**



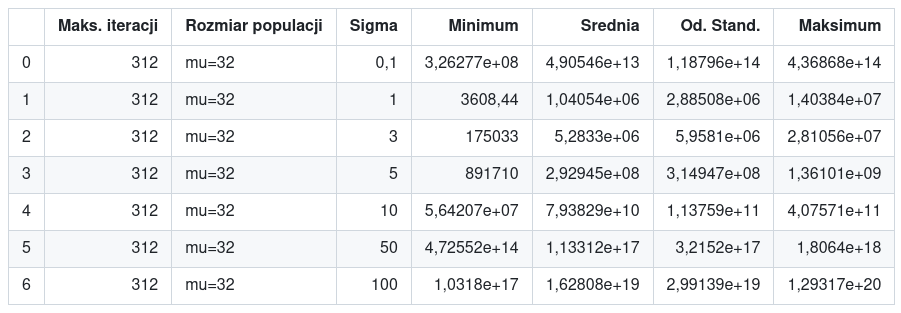
W przypadku funkcji f13, najlepiej sprawdził się rozmiar populacji o wysokości mu=64.

Średnia dla tego parametru była najmniejsza (4,0604 \* 10^6), a co więcej biorąc pod uwagę wartość Minimum i Maksimum, były to również wartości najbliżej optimum (minimum globalnemu) spośród ich odpowiedników dla pozostałych rozmiarów populacji.

W związku z powyższym można wysnuć wniosek, że zbyt duże populacje, ale też i zbyt małe, pogarszają jakość otrzymanego rozwiązania. Dla wybadanych rozmiarów populacji przeprowadzono następnie badanie wpływu siły mutacji poprzez manipulowanie wartością parametru Sigma.

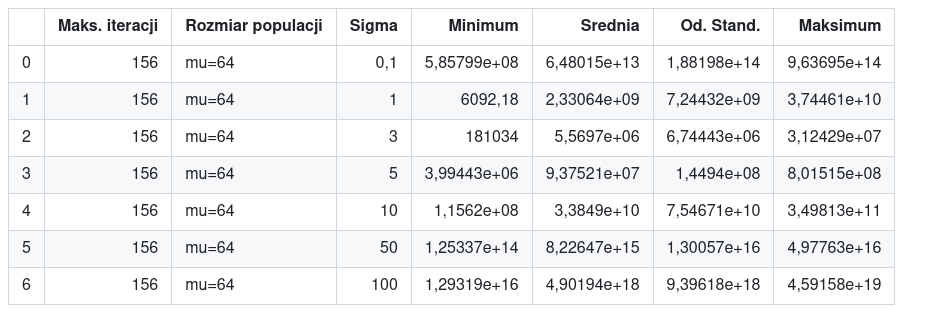
**Badanie wpływu siły mutacji**

**Eksperyment 2.1 : badanie wpływu siły mutacji dla funkcji f2**



W przypadku funkcji f2, algorytm ewolucyjny z stałą populacją mu=32 najlepsze wyniki uzyskał dla parametru sigma = 1. Średnia, minimum oraz maksimum były stosunkowo najbliższe minimum globalnemu.

**Eksperyment 2.2 : badanie wpływu siły mutacji dla funkcji f13**

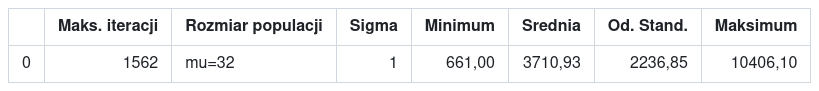


Jednakże zmiana parametru Sigma dla funkcji f13 nie przyniosła poprawy rezultatów. Zmiana wartości na wyższą lub niższą dawała wartości funkcji celu średnio wyższe niż dla sigma = 3.

**Sprawdzanie 5-krotnie większego budżetu**

Postanowiono zatem przetestować jak na otrzymane w ramach testów kombinacje rozmiaru populacji i parametru Sigma wpłynie wykorzystanie pięciokrotnie wyższego budżetu do przeprowadzenia oceny. Wyniki były bardzo zadowalające, uzyskano rezultaty o zdecydowanie lepszej jakości. Przedstawiają to zestawienia w tabelach poniżej.

**Eksperyment 3.1 : Funkcja f2 z budżetem 50 000, mu=32 i sigma=1**



Zdecydowana poprawa wyników na skutek zwiększenia budżetu (maksymalnej liczby iteracji) dla obydwu funkcji. Im dłużej pozwolono na ewoluowanie rozwiązania tym wyniki były bliżej minimum (optimum).

**Eksperyment 3.2 : Funkcja f13 z budżetem 50 000, mu=64 i sigma=3**



**Wnioski końcowe:**

Na podstawie powyższego ćwiczenia można wysnuć następujące wnioski:

* algorytm ewolucyjny stanowi bardzo prosty w implementacji i skuteczny w stosowaniu algorytm,
* stosując go należy wykazać się umiejętnością doboru rozmiaru populacji, nie za dużej i nie za małej.
* wysokość parametru sigma, który zwiększa wpływ mutacji powinna być rozsądnie mała i raczej unikana. Zalecane jest bardziej konfigurowanie rozmiaru populacji i dostępnego budżetu,
* używanie krzyżowania nie jest wymagane.