

# Implementacja i ocena algorytmu NSGA-II w optymalizacji wielokryterialnej na funkcjach ZDT

Michał Żelazko

Damian Kantorowski

## Opis algorytmu

Kod programu napisany został w języku Python. Do uruchomienia wymagane są biblioteki Numpy, random, matplotlib i copy.

- **Reprezentacja i inicjalizacja**

Populacja początkowa zawiera 150 osobników składających się z wektora współrzędnych rzeczywistych z dziedziny funkcji wylosowanych z użyciem rozkładu jednostajnego oraz z wektora  $\sigma$  o takim samym rozmiarze z początkowymi wartościami zależnymi od wymiaru. Wartość  $\sigma$  jest odchyleniem standardowym rozkładu normalnego.

- **Przypisanie frontów**

Wykorzystany został algorytm Kunga – najpierw osobniki są sortowane rosnąco po funkcji oceny  $f_2$ , a gdy ta wartość jest taka sama, rosnąco po  $f_1$ . Następnie posortowana lista jest rekurencyjnie dzielona na połowy **T** i **B** aż do osiągnięcia 1 osobnika w każdej. Listy są następnie łączone poprzez zwrócenie listy niedominujących się osobników, czyli wszystkich z **T** i tych z **B** które nie są dominowane przez **T**. Sprawdzając dominację porównywana jest wyłącznie wartość oceny  $f_1$ , ponieważ posortowana lista gwarantuje lepszą wartość funkcji  $f_2$  osobników **T** względem **B**. Łączone są tak listy aż wszystkie podziały **T** i **B** zostaną rozpatrzone.

- **Obliczenie crowding distance**

Każdej instancji z tego samego frontu przypisuje się wartość wskazującą jak bardzo jego najbliżsi sąsiedzi są od siebie oddaleni. Oblicza się długość od każdego wymiaru oraz normalizuje się dzieląc przez różnicę maksymalnej i minimalnej wartości osiągniętej na froncie w danym wymiarze. Na koniec sumuje się wyniki ze wszystkich wymiarów. Punkty graniczne mają przypisaną największą odległość.

- **Selekcja**

Wykorzystano selekcję turniejową. Wybrane zostają losowo dwa osobniki, żeby następnie wybrać lepszy. Najpierw uwzględnia się hierarchie frontów Pareto, a w razie należenia do tego samego, wybierany jest ten z większą odległością crowding.

- **Krzyżowanie**

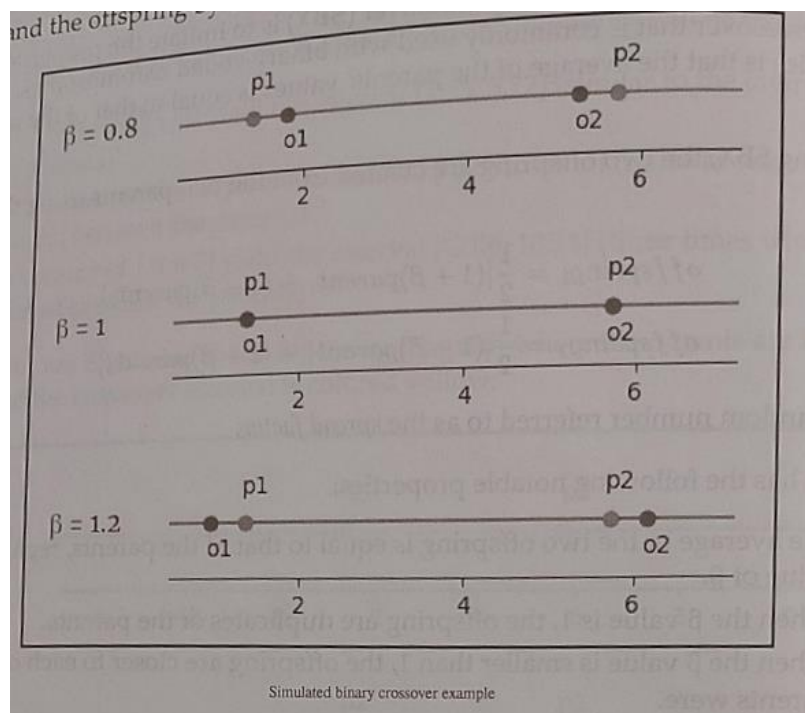
Wybrane na etapie selekcji osobniki wykorzystywane są do rekombinacji. Wykorzystany został algorytm Simulated Binary Crossover. ([Eyal Wirsansky, Hands-On Genetic Algorithms with Python](#)) Celem jest imitowanie działania krzyżowania liczb binarnych jedną linią przecięcia. Jednym z właściwości zachowanych w SBX jest średnia rodziców równa średniej potomków.

Dwóch potomków wyznacza się wzorem:

$$offspring_1 = \frac{1}{2}[(1 + \beta)parent_1 + (1 - \beta)parent_2]$$

$$offspring_2 = \frac{1}{2}[(1 - \beta)parent_1 + (1 + \beta)parent_2]$$

Potomkowie są symetryczni względem średniej rodziców. Wpływ zmiennej  $\beta$  na potomków:



Gdzie  $p1$  i  $p2$  to rodzice,  $o1$  i  $o2$  to potomkowie

$\beta$  jest wyznaczana przez wylosowanie liczby  $u$  z rozkładu jednostajnego o przedziale  $[0, 1]$ . Parametr  $\eta$  pozwala na kontrolę eksploracji/eksploatacji – większa  $\eta$  zbliża potomków do wartości pomiędzy rodzicami.

If $u \leq 0.5$ :	$\beta = (2u)^{\frac{1}{\eta+1}}$
Otherwise:	$\beta = \left[ \frac{1}{2(1-u)} \right]^{\frac{1}{\eta+1}}$

By nie dopuścić do stworzenia niepoprawnego punktu, współrzędna nie należąca do dziedziny funkcji jest do niej sprowadzana poprzez odbijanie od brzegu przedziału.

- **Mutacja**

Podczas mutacji modyfikowane najpierw wartości sigma:

$$\sigma' \leftarrow \sigma \cdot \exp(\tau \cdot N(0, 1))$$

a następnie współrzędne:

$$x_j^{t+1} = x_j^t + N(0, \sigma')$$

Gdzie  $\tau \propto \frac{1}{\sqrt{n}}$

Tak jak w trakcie krzyżowania, niepoprawne współrzędne odbijane są tak by trafiły do dziedziny.

Odbicie sprowadza wartość sigmy do przedziału  $[\sigma_{min}, \sigma_{start}]$ , gdzie  $\sigma_{min}$  i  $\sigma_{start}$  to parametry algorytmu.

- **Zastępowanie**

Operacja najpierw łączy populację rodziców i potomków. Przypisywane są fronty pareto oraz crowding distance. Następnie do nowej generacji dodawane są kolejne fronty pareto, aż maksymalna wielkość populacji zostanie osiągnięta. Jeśli ostatni wybrany front nie mieści się w całości w nowej generacji, wybierane są z niego te osobniki które mają największy crowding distance.

# Metodologia pomiarów

Wartości parametrów:

parametr \ wymiar	10	30	50
eta_start	10	2	1
eta_end	18	20	5
sigma_start	1	1	1.5
sigma_min	1e-6	1-e3	1e-3

Dla każdej konfiguracji został stworzony wykres na którym widać postęp po 20, 50 , 100 i 500 iteracjach. Widoczna jest poprawa rozwiązań z zwiększając się ilością iteracji. Widoczne jest jednak zbieranie się punktów w okolicy wartości  $f1=0$ .

Aby zapobiec temu zjawisku próbowaliśmy różnych funkcji clippingu wartości, np. odbicie wartości od dziedziny pomnożony przez skalar. Zastosowane zostały również wartości parametrów, które pomagają eksploracji. (zwiększanie wart. pocz. sigmy, zwiększanie wart. min. sigmy, zmniejszanie wart początkowej i końcowej parametru  $\eta$ ). Wypróbowane zostały również inne operatory rekombinacji: BLX-alpha i uśrednianie dwóch losowych instancji. W mutacji przetestowany został także rozkład Cauchiego zamiast Normalnego przy zmianie  $\mathbf{x}$ .

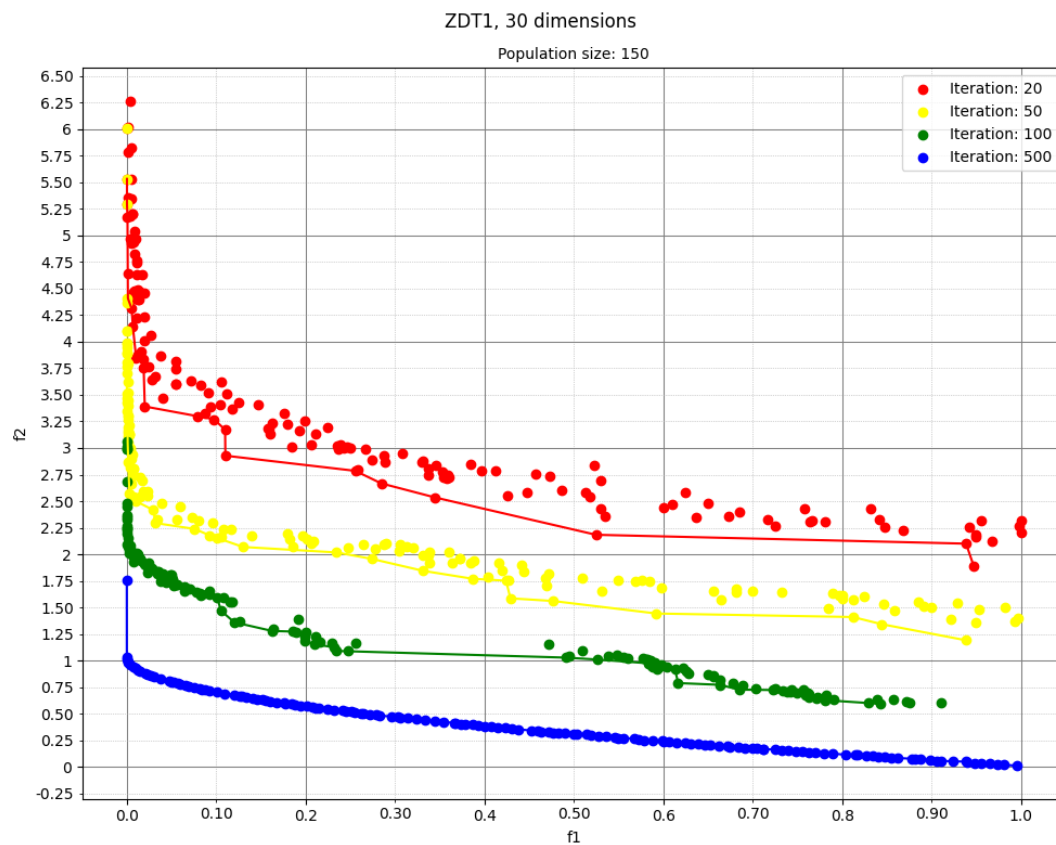
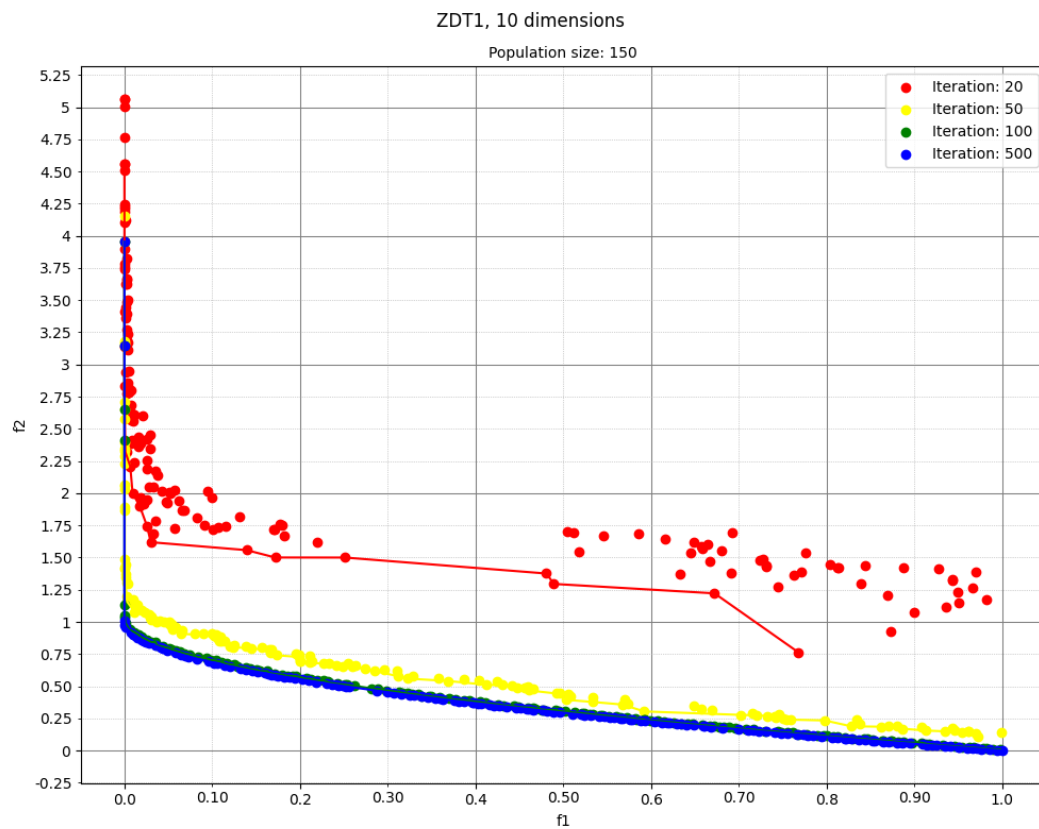
Algorytm dla ZDT1, ZDT2 i ZDT3 dla wymiarów 10 i 30 znajduje front Pareto po pełnym przebiegu. Dla wymiaru 50 jest bardzo blisko optymalnego rozwiązania.

Funkcje ZDT4 i ZDT6 są dużą trudnością dla naszego algorytmu. Dla ZDT4 odległość funkcji celu do frontu Pareto po 500 iteracjach jest średnio: 1, 13 i 18 dla kolejnych wymiarów. Funkcja ZDT6 nieco bliżej frontu: 0.5, 2 , 3.

## Obserwacje

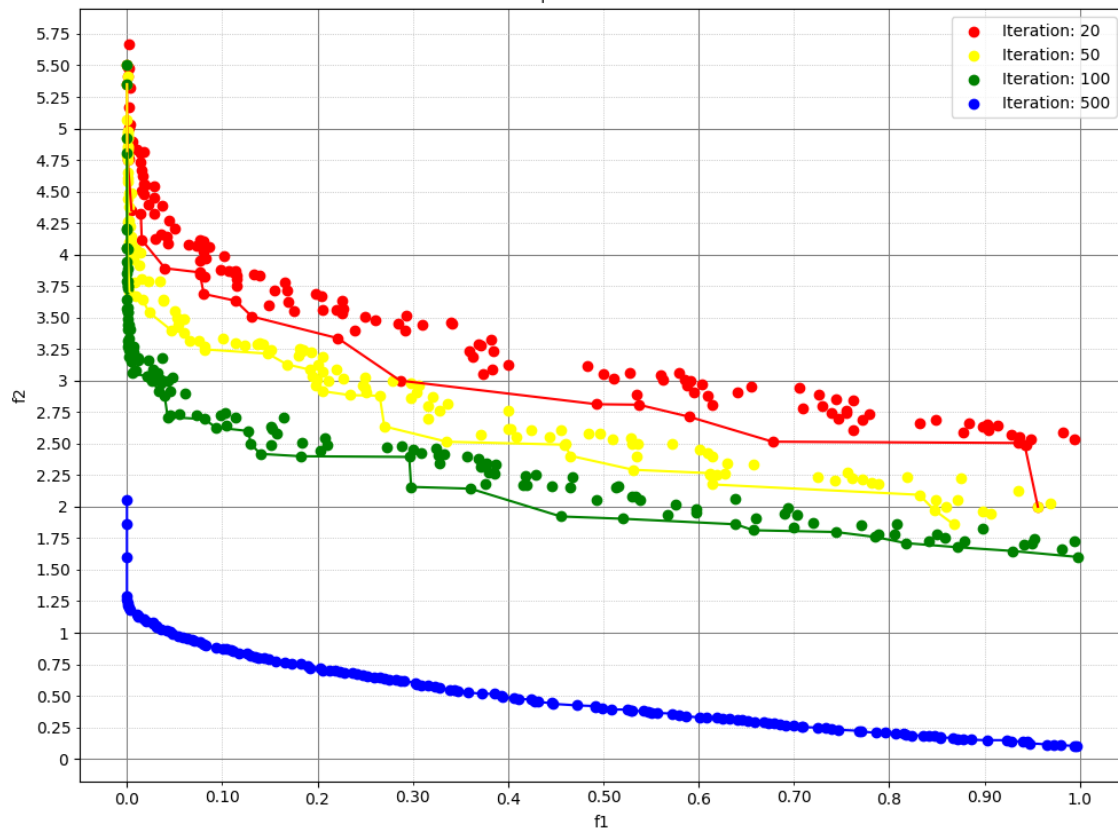
Wydaje się, że gdyby nie zbieranie się punktów w pobliżu jednego miejsca, algorytm mógłby sobie poradzić znacznie lepiej. Możliwą modyfikacją jest zastąpienie rozkładu normalnego przy mutacji  $\mathbf{x}$  przez  $\alpha$ -stable distribution. Mimo wszystko widać, że algorytm potrafi znajdować fronty Pareto lub systematycznie zbliżać się do nich.

# Wykresy funkcji ewaluacyjnych



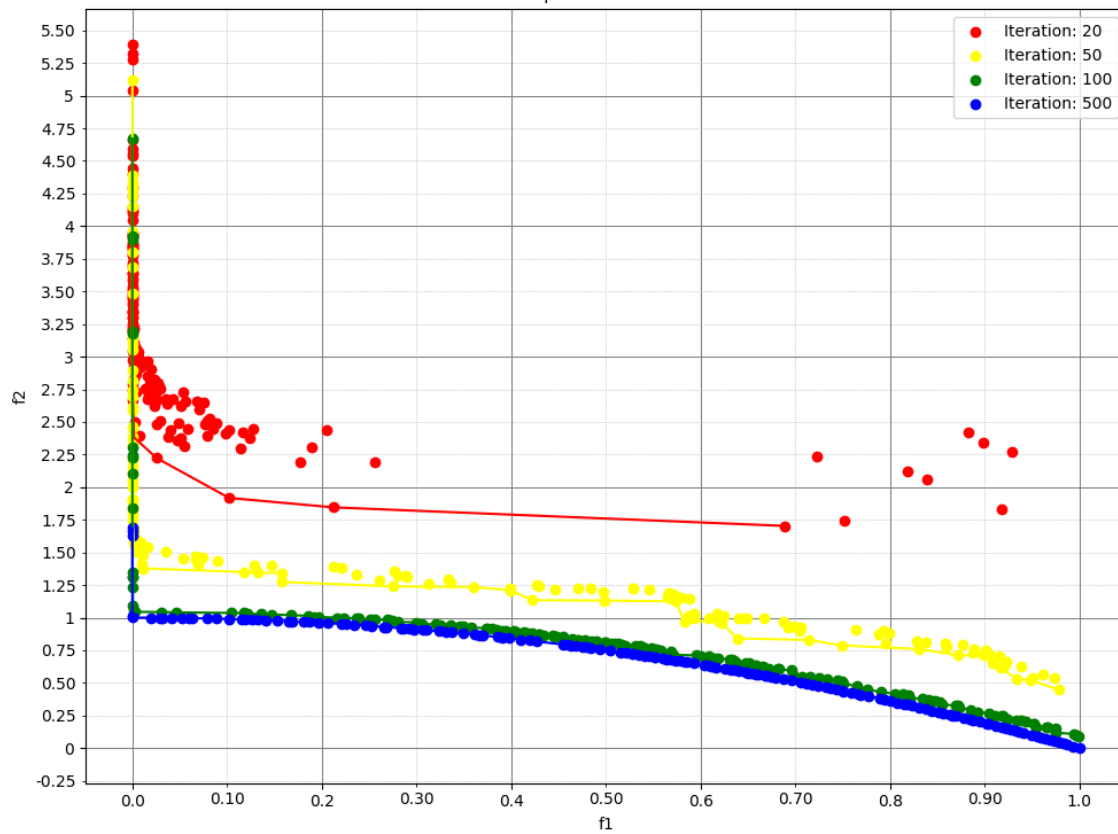
ZDT1, 50 dimensions

Population size: 150



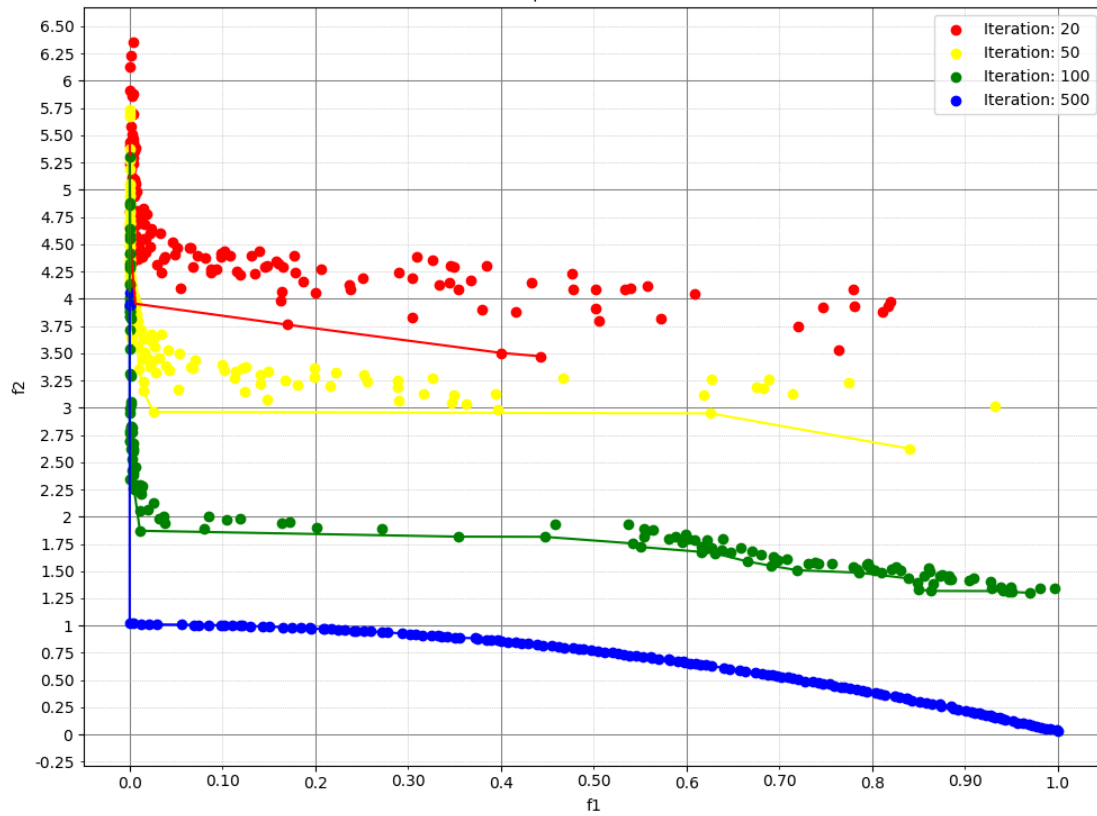
ZDT2, 10 dimensions

Population size: 150



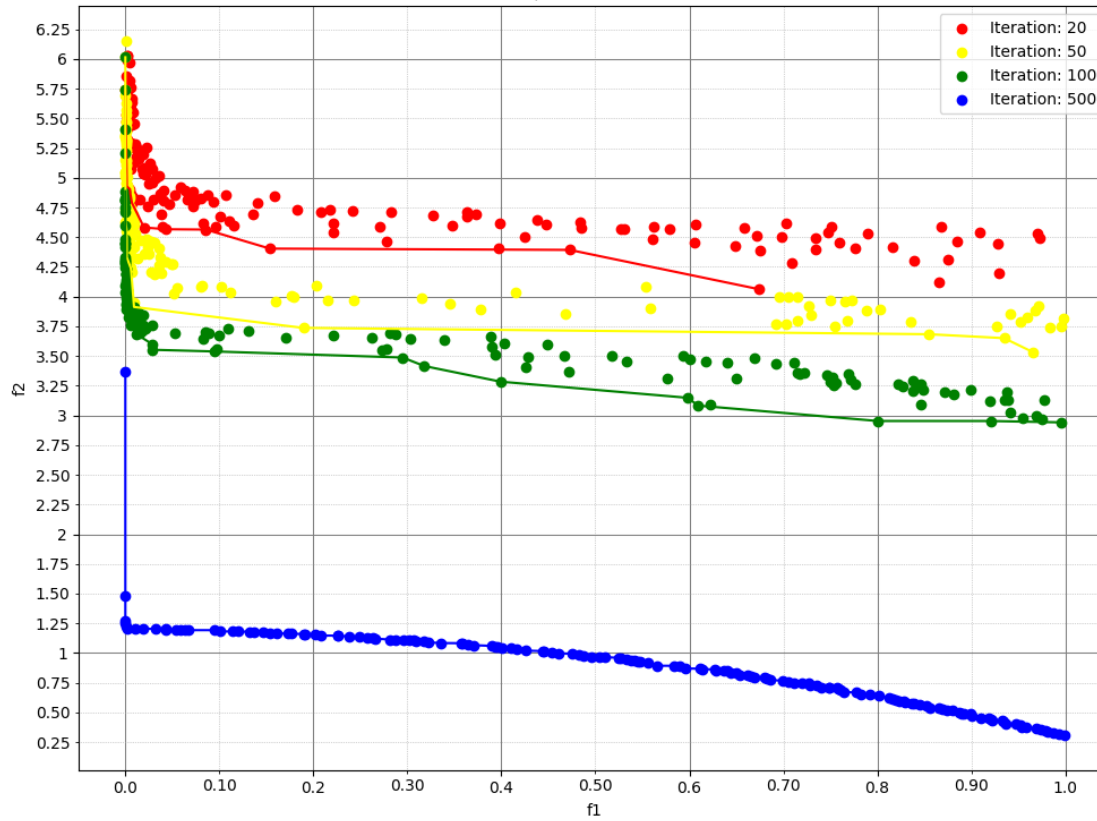
ZDT2, 30 dimensions

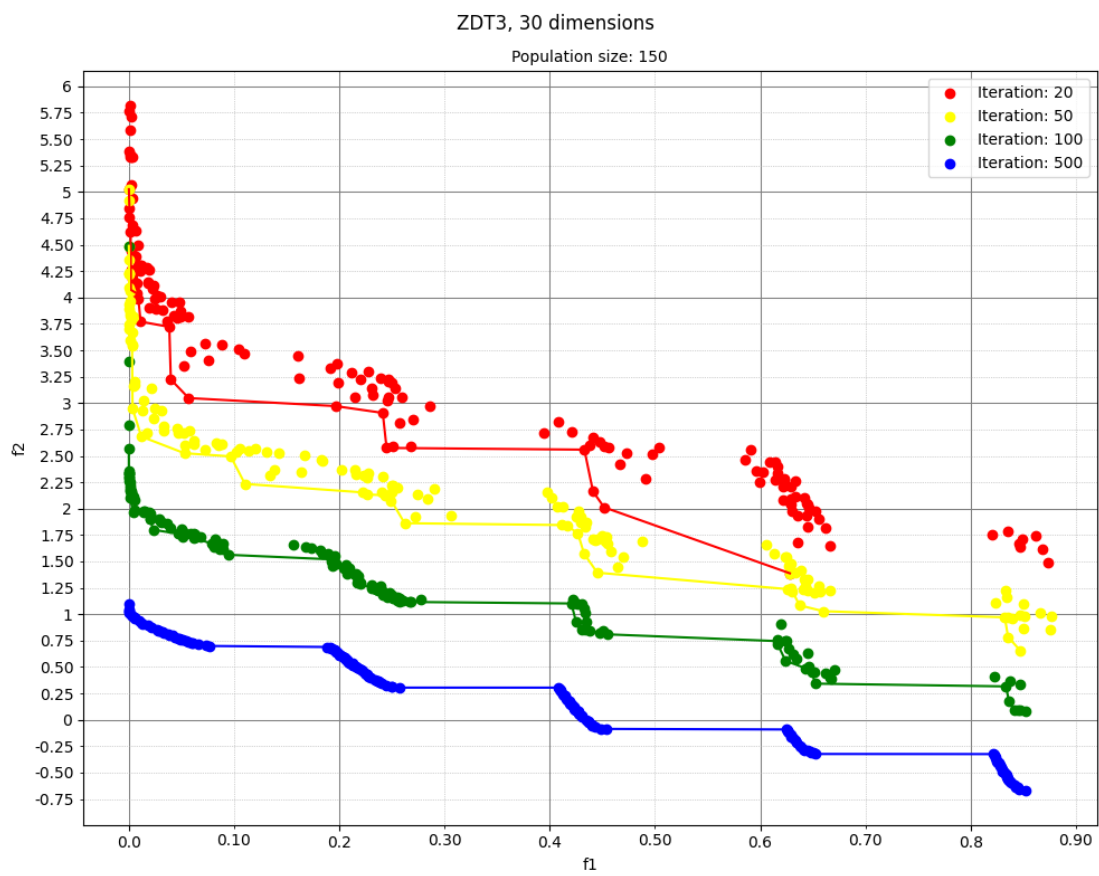
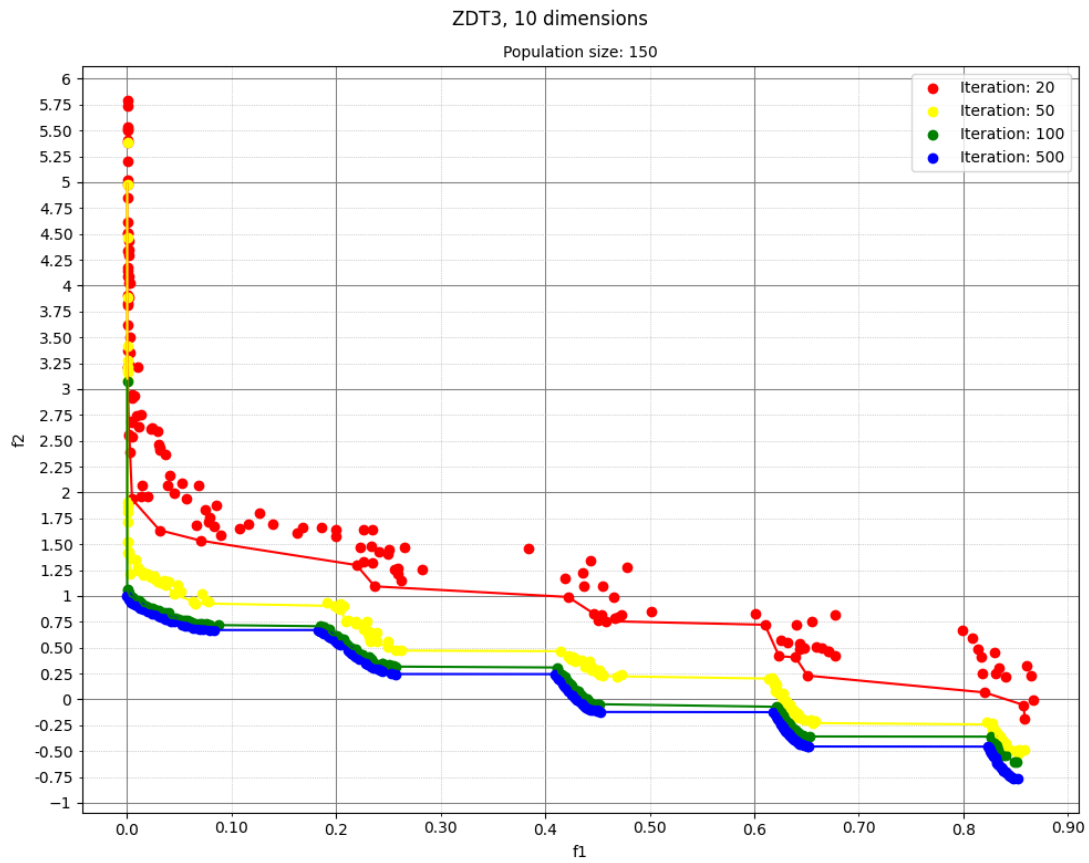
Population size: 150



ZDT2, 50 dimensions

Population size: 150

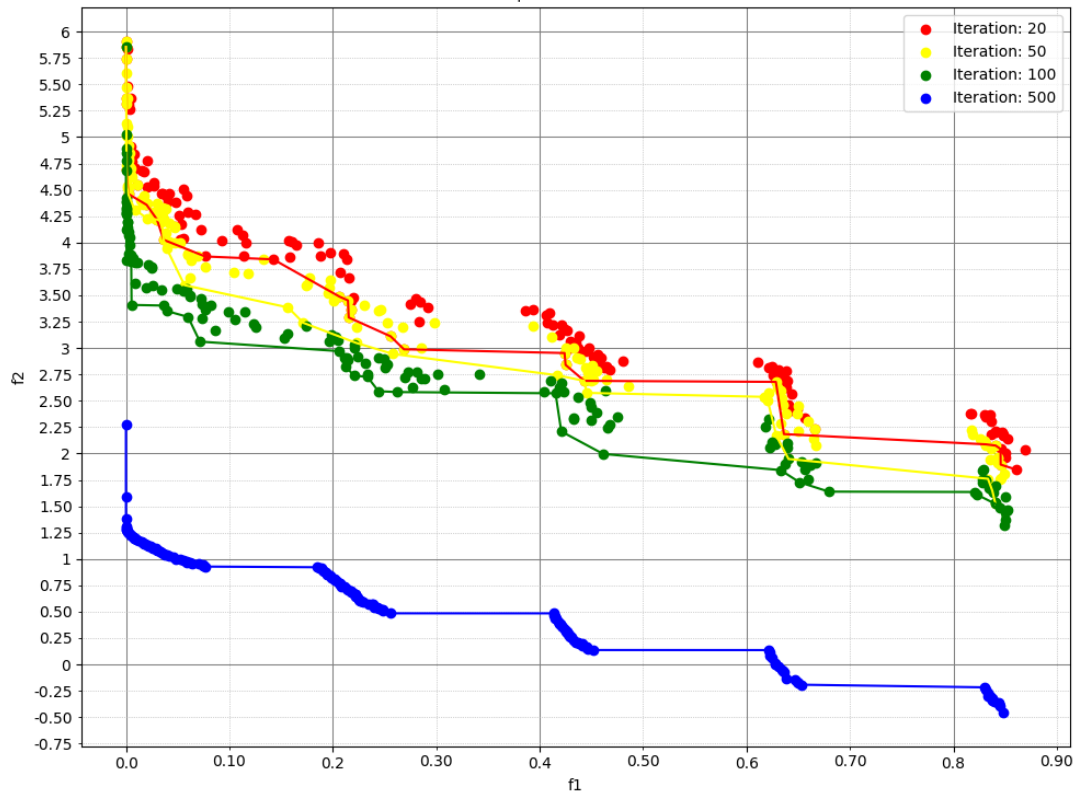


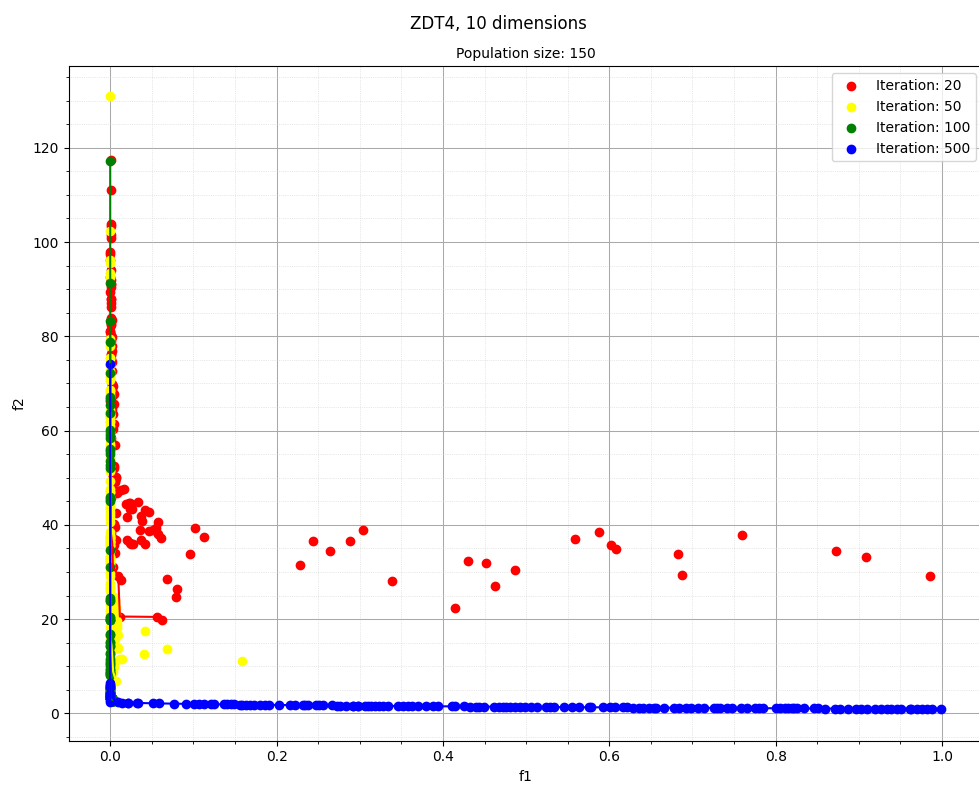
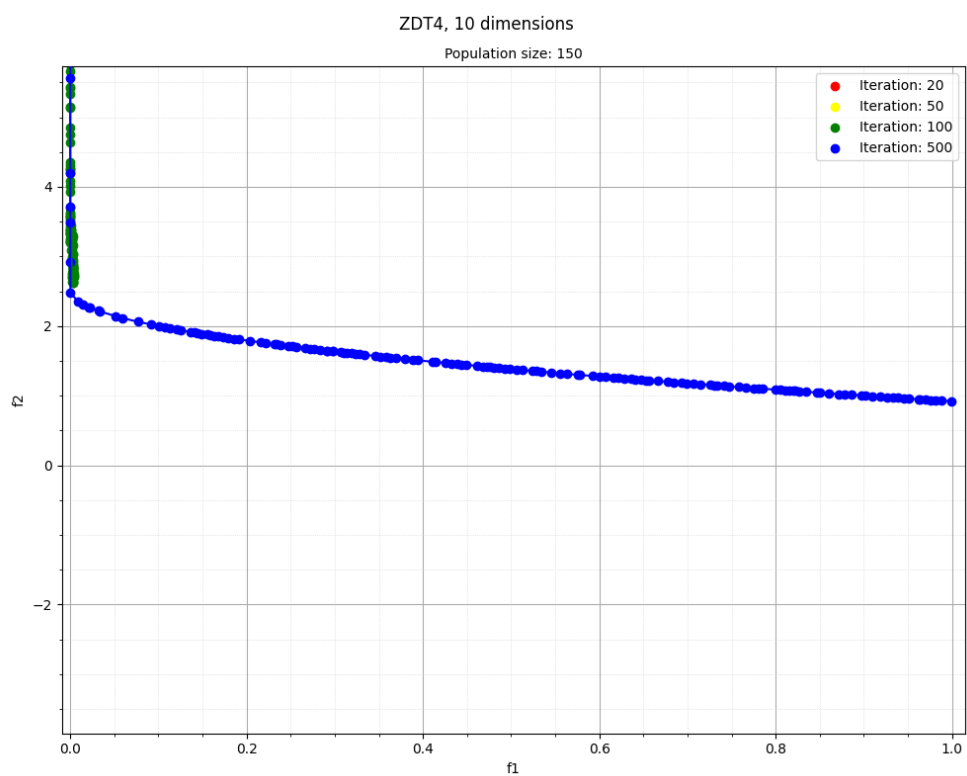


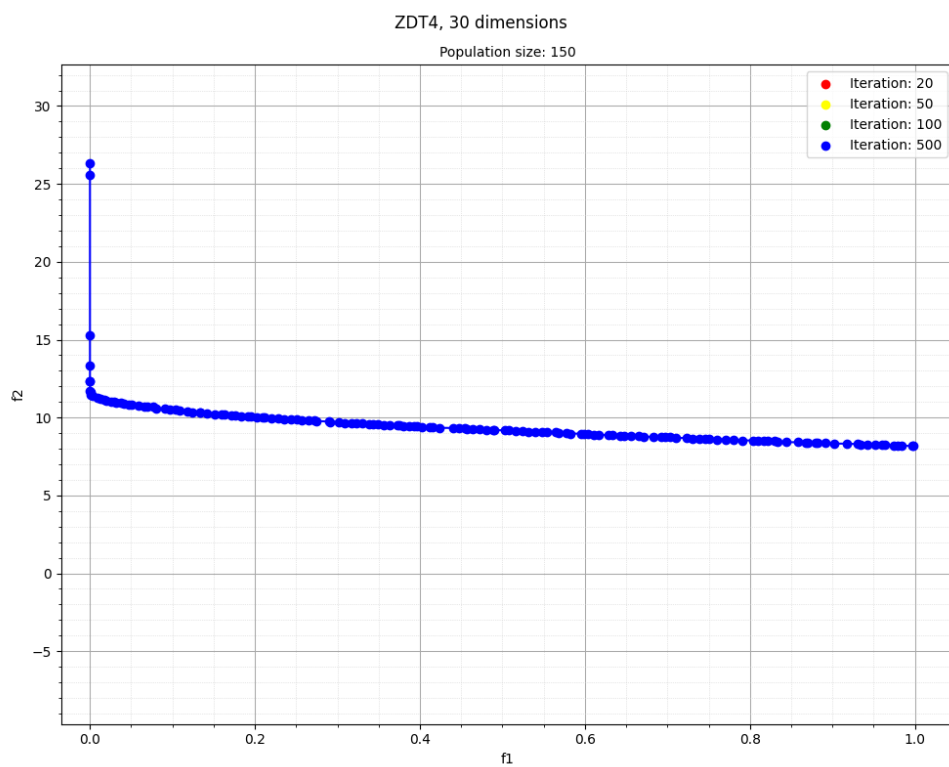
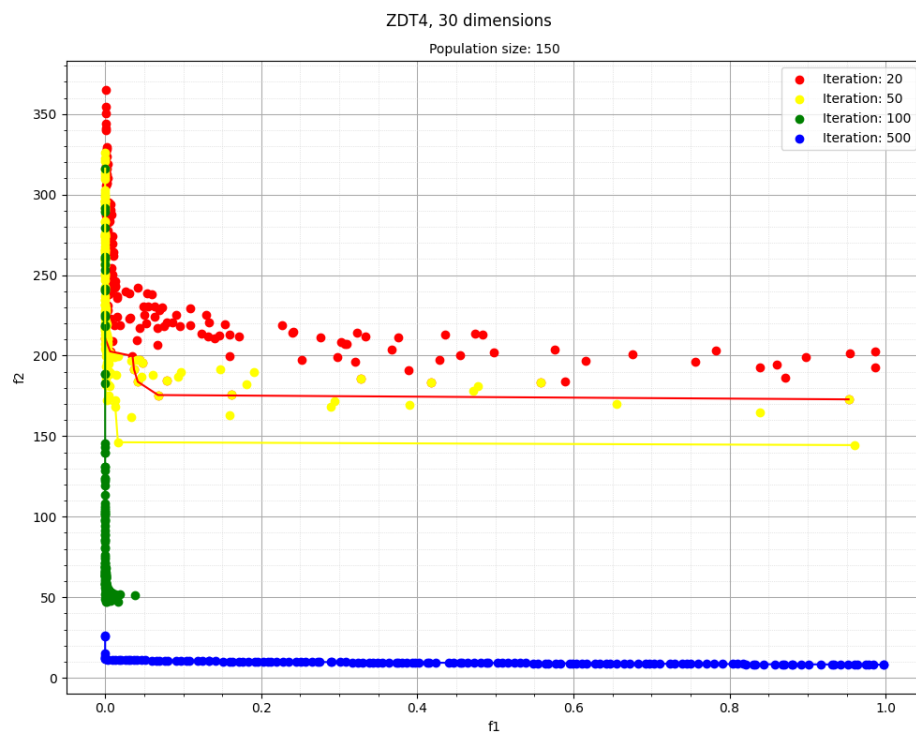


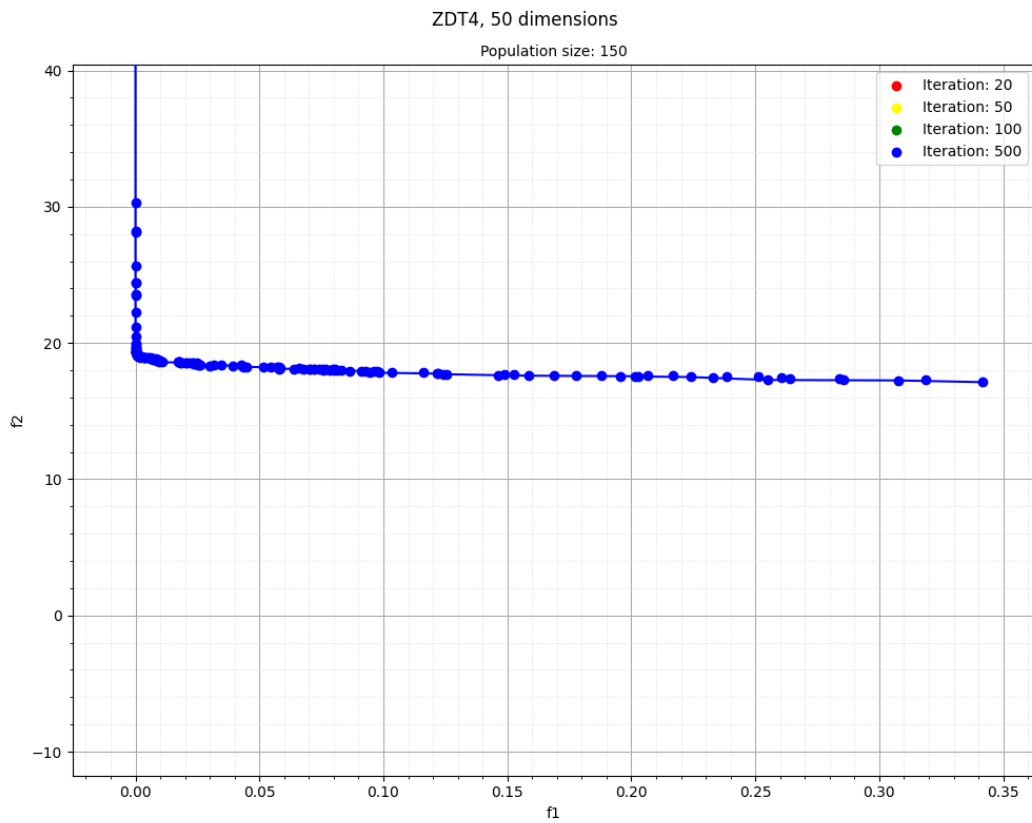
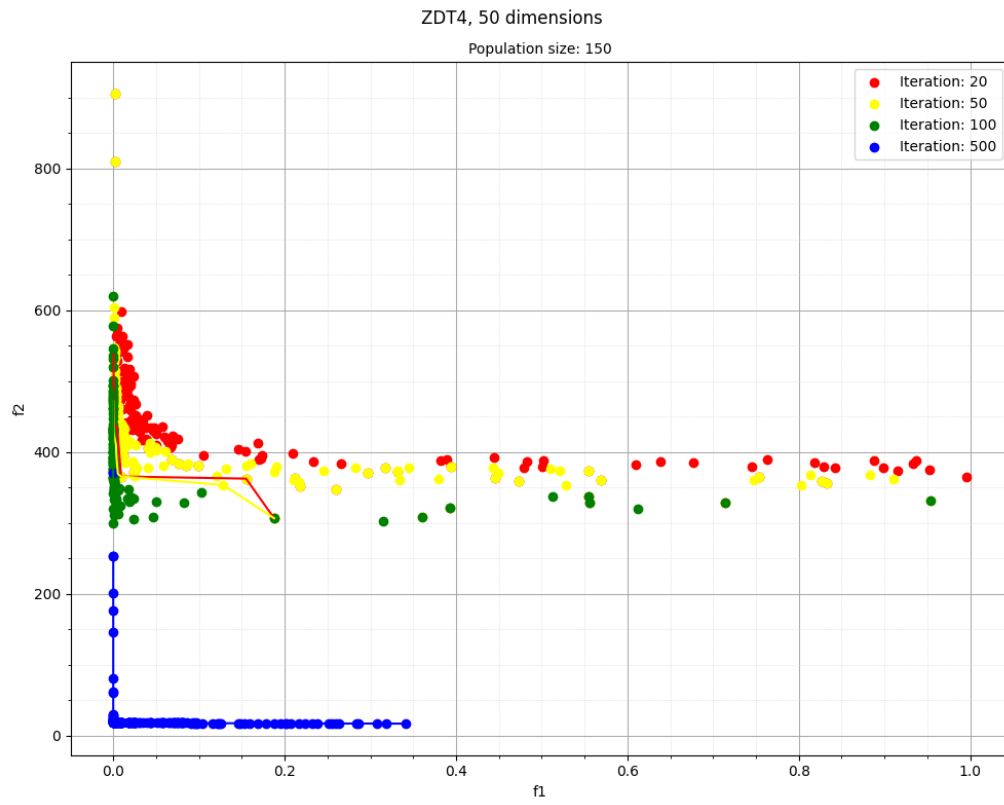
ZDT3, 50 dimensions

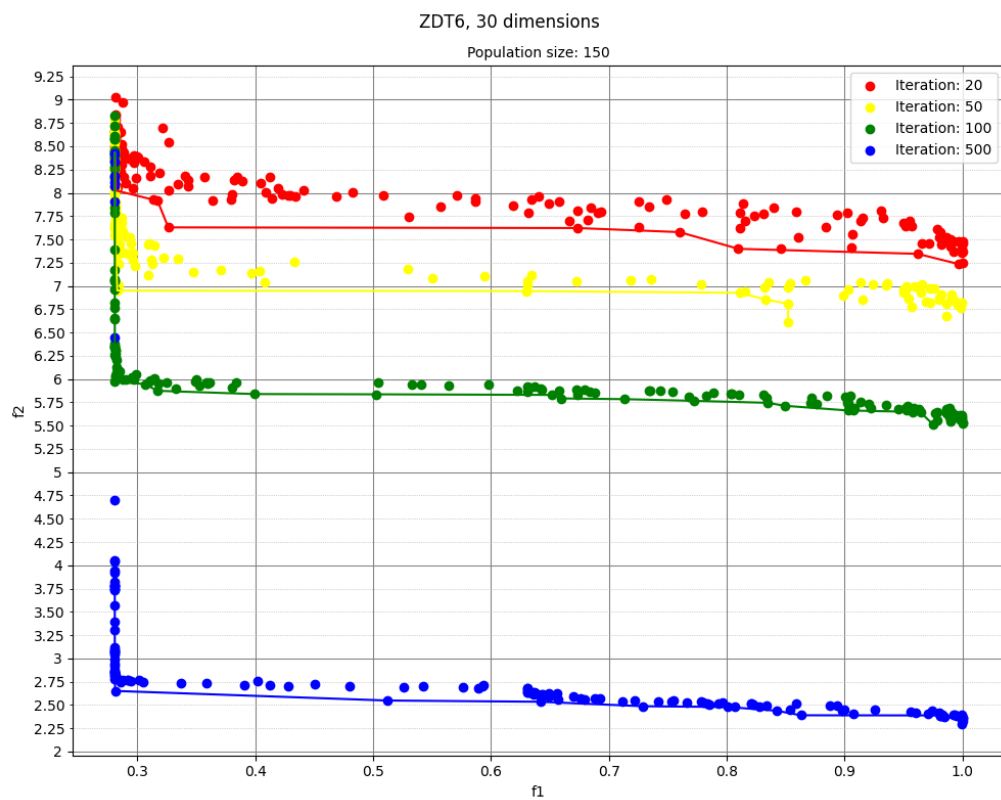
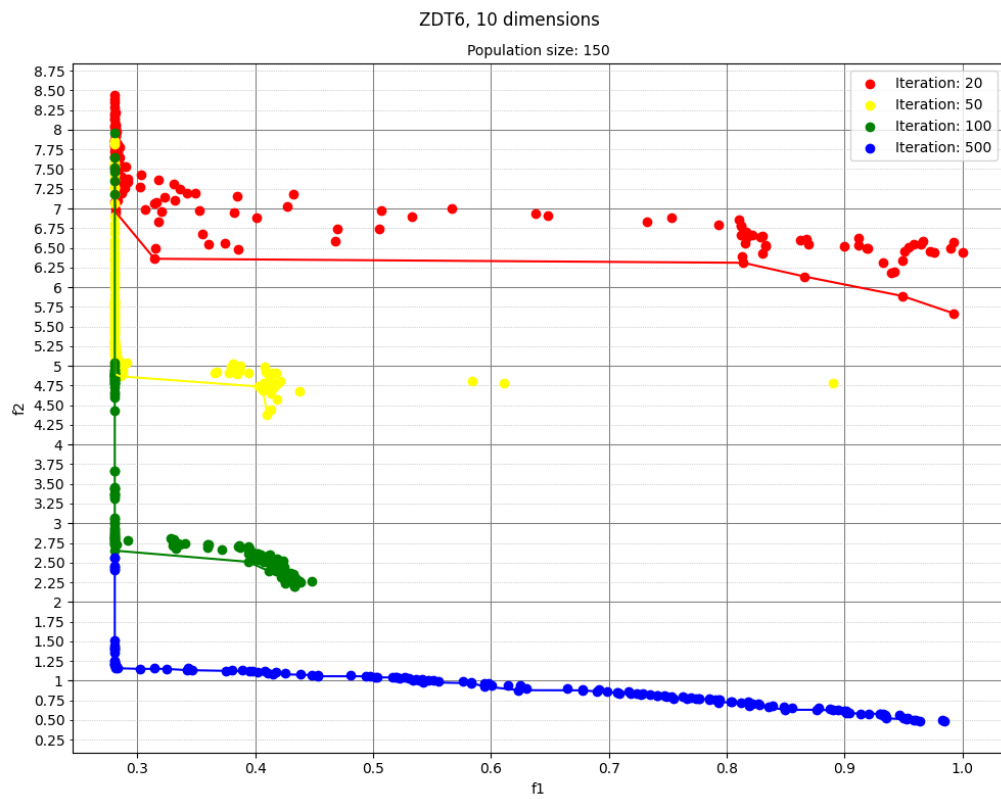
Population size: 150











ZDT6, 50 dimensions

Population size: 150

