Optymalizacja wielokryterialna

Aplikacja do porównań metod OWD

Tomisław Tarnawski
Jakub Wąsik

1. Wstęp

Aplikacja została stworzona w języku Python 3, z wykorzystaniem bibliotek Tkinter i Matplotlib, aby zapewnić użytkownikowi intuicyjny interfejs graficzny do analizy i optymalizacji wielokryterialnej. Tkinter posłużył do zbudowania interaktywnej aplikacji, umożliwiającej łatwe wprowadzanie danych, zarządzanie punktami oraz wybór i konfigurację algorytmów optymalizacyjnych. Dzięki integracji z Matplotlib, program pozwala na wizualizację wyników w postaci wykresów 2D i 3D, co ułatwia analizę i interpretację danych przez użytkownika.

Celem aplikacji jest umożliwienie przeprowadzenia optymalizacji wielokryterialnej na zbiorze punktów w przestrzeni wielowymiarowej, z wykorzystaniem różnych algorytmów wyszukiwania punktów niezdominowanych. Program wspiera generowanie losowych punktów, zarządzanie danymi oraz przeprowadzanie benchmarkingu, co pozwala na porównanie efektywności poszczególnych algorytmów.

2. Opis interfejsu aplikacji

Główne sekcje aplikacji

Aplikacja składa się z kilku sekcji, które pozwalają na różnorodne operacje na danych punktach. Na początku użytkownik może ręcznie wprowadzić zestaw punktów w formacie krotek, lub wygenerować losowy zbiór punktów o zadanej liczbie kryteriów i rozkładzie. Wybór rozkładu (Gaussian, Exponential, Poisson) oraz parametry, takie jak średnia i odchylenie standardowe, pozwalają na dostosowanie danych do konkretnych potrzeb. Wygenerowane punkty są wyświetlane w formie tabeli, którą można sortować według wartości poszczególnych kryteriów.

Zarządzanie punktami

Program oferuje funkcje dodawania, aktualizowania i usuwania punktów. Użytkownik może ręcznie wprowadzać nowe punkty lub edytować istniejące poprzez wybór odpowiedniego punktu w tabeli. Aktualizacja odbywa się poprzez wpisanie nowych wartości dla wybranego punktu, a usuwanie za pomocą jednego kliknięcia.

Wybór algorytmu optymalizacji

Użytkownik może wybrać kryterium optymalizacji (minimalizacyjne lub maksymalizacyjne), a następnie uruchomić jeden z dostępnych algorytmów:

- "Algorytm bez filtracji"
- "Algorytm z filtracją punktów zdominowanych"
- "Algorytm oparty o punkt idealny".

Każdy z tych algorytmów różni się metodą znajdowania punktów niezdominowanych, co umożliwia porównanie efektywności. Wyniki są wyświetlane w oknie tekstowym, zawierającym listę punktów niezdominowanych.

Funkcje wizualizacji i benchmarkingu

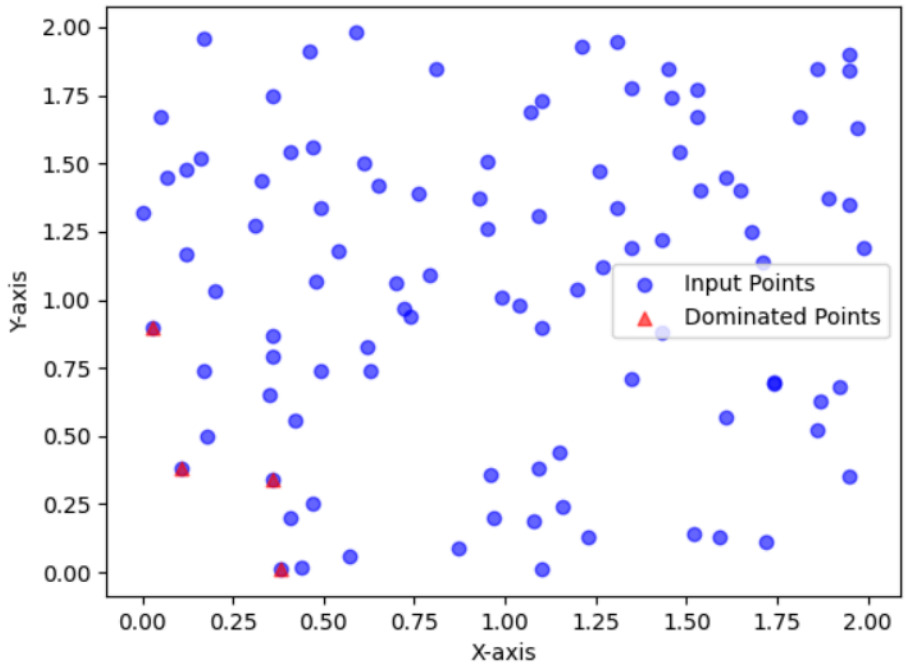
Aplikacja oferuje możliwość wizualizacji punktów w 2D lub 3D, z podziałem na punkty wejściowe i wyjściowe, co ułatwia analizę wyników optymalizacji. Dodatkowo dostępna jest funkcja benchmarkingu, która mierzy czas wykonania i liczbę porównań dla każdego algorytmu, co pozwala na ocenę ich wydajności.

Przykładowe użycie

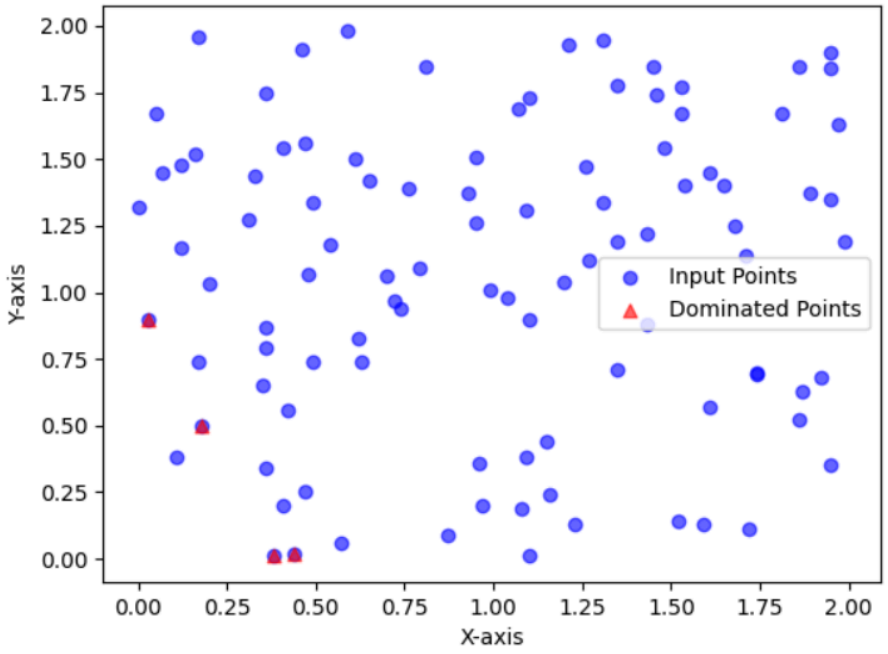
1. Wprowadzenie punktów ręcznie lub losowe wygenerowanie zbioru o rozkładzie Gaussa.
2. Wybranie kryterium maksymalizacji i uruchomienie algorytmu bez filtracji.
3. Wyświetlenie wyników oraz ich wizualizacja.
4. Przeprowadzenie benchmarkingu, aby porównać wydajność trzech algorytmów dla określonego zbioru punktów.

Eksperymenty obliczeniowe

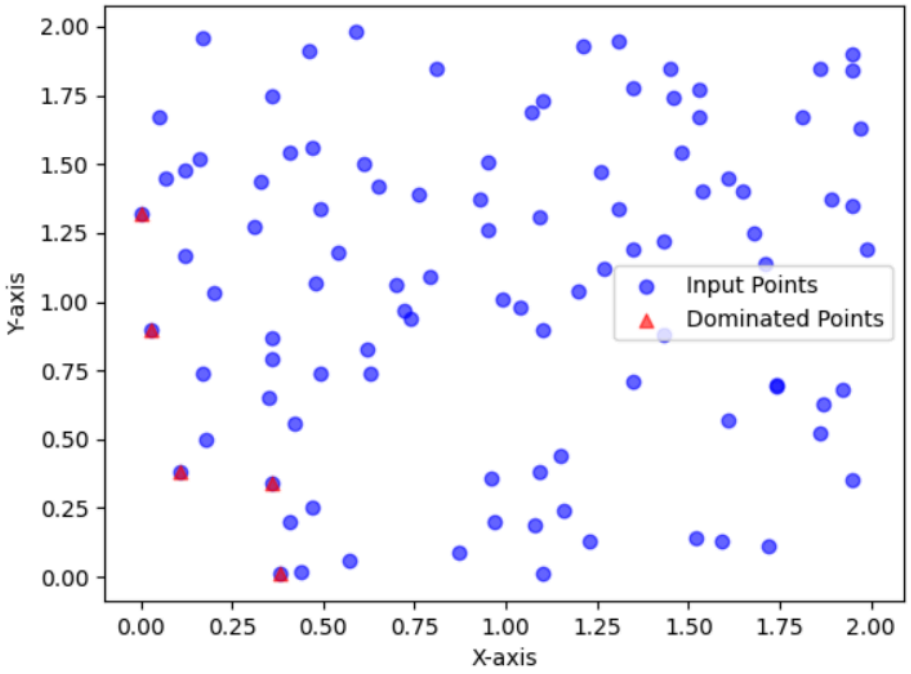
Generowanie losowo 100 elementowego zbioru przy użyciu rozkładu jednostajnego [0, 2], liczba kryteriów = 2, kryterium min:
Algorytm bez filtracji:
(0.11, 0.38)
(0.38, 0.01)
(0.03, 0.9)
(0.36, 0.34)



Algorytm z filtracją punktów zdominowanych:
(0.18, 0.5)
(0.44, 0.02)
(0.03, 0.9)
(0.38, 0.01)



Algorytm oparty o punkt idealny:
(0.38, 0.01)
(0.11, 0.38)
(0.36, 0.34)
(0.03, 0.9)
(0.0, 1.32)

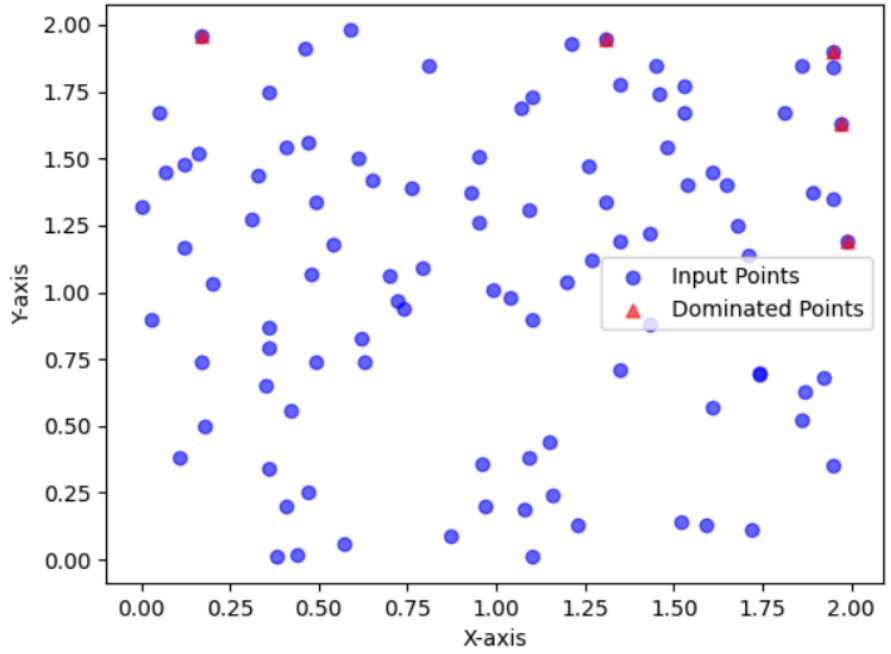


Przy ocenie czasu działania algorytmów należy posłużyć się liczbą porównań, czas działania był zbyt krótki dla wykorzystywanej biblioteki, żeby go zapisać do zmiennej.

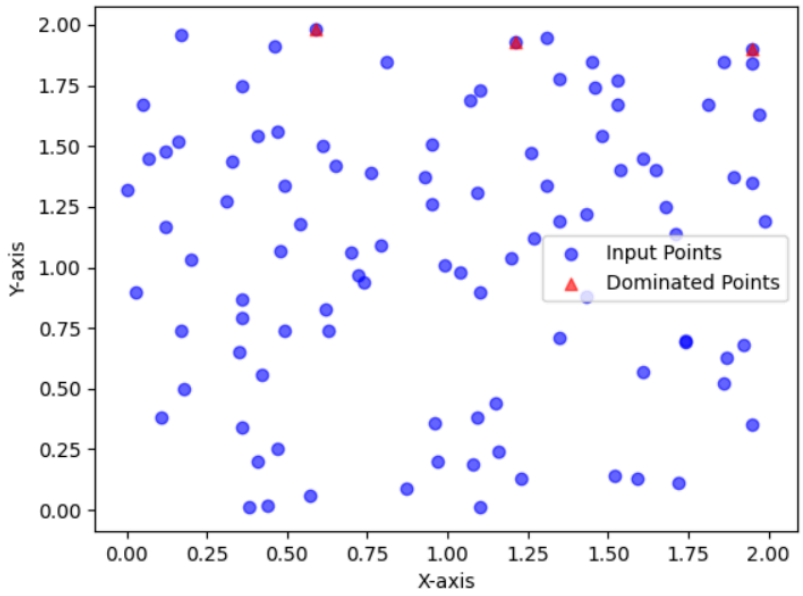
Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	0.00	135
Algorytm z filtracją punktów zdomin	0.00	185
Algorytm oparty o punkt idealny	0.00	124

Ten sam zbiór punktów, ale dla kryterium max:

Algorytm bez filtracji:
(1.95, 1.9)
(1.31, 1.95)
(0.17, 1.96)
(1.97, 1.63)
(1.99, 1.19)

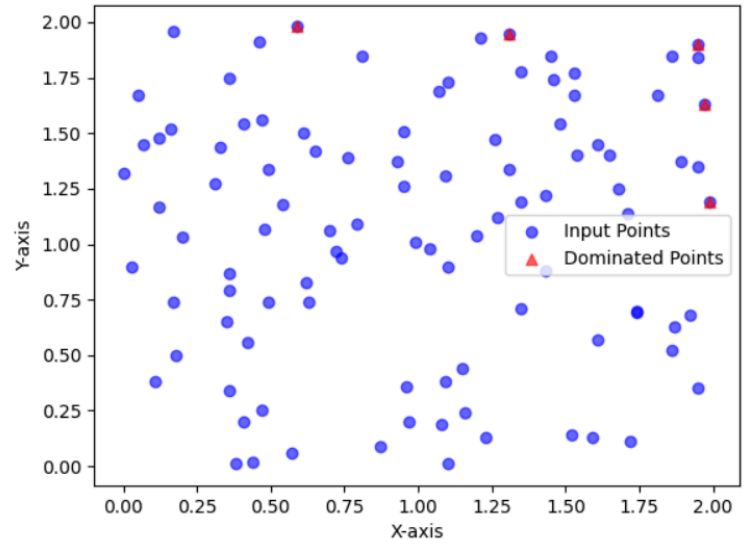


Algorytm z filtracją punktów zdominowanych:
(1.95, 1.9)
(0.59, 1.98)
(1.21, 1.93)



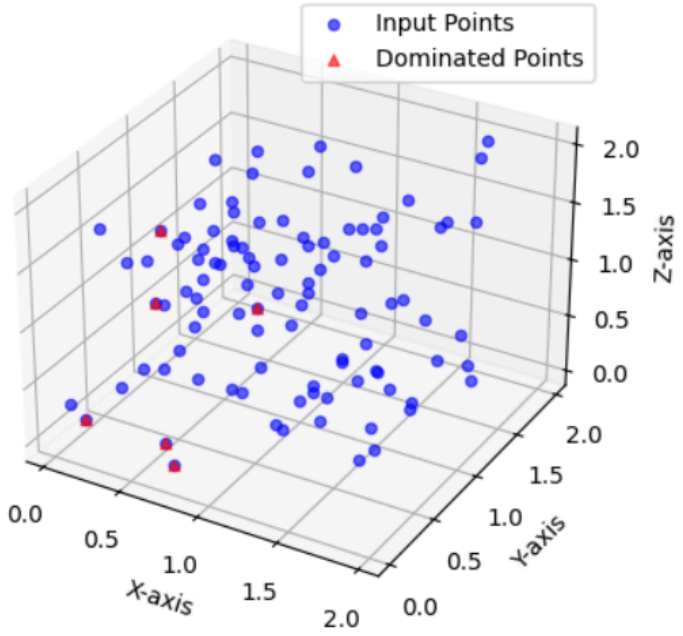
Algorytm oparty o punkt idealny:

(1.95, 1.9)
(1.97, 1.63)
(1.31, 1.95)
(1.99, 1.19)
(0.59, 1.98)

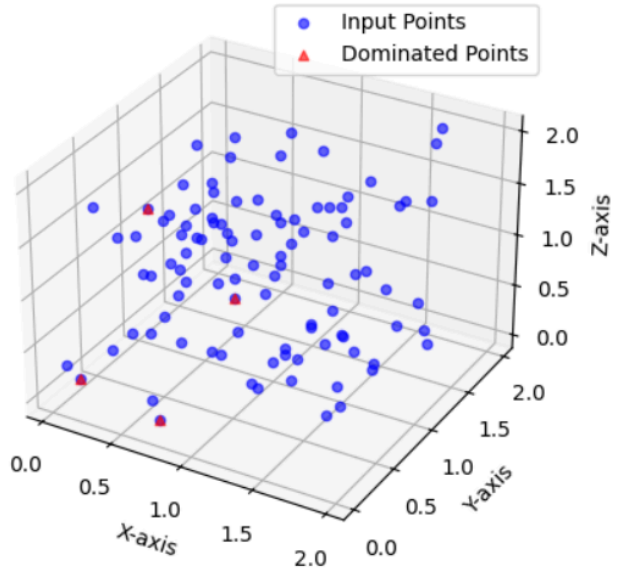


Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	0.00	128
Algorytm z filtracją punktów zdomin	0.00	122
Algorytm oparty o punkt idealny	0.00	113

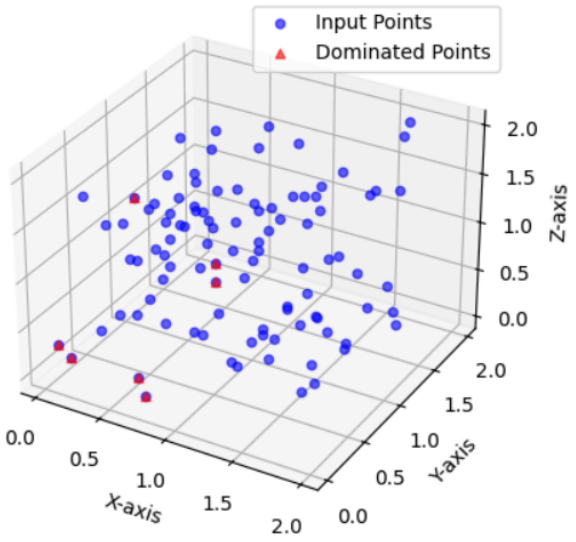
Liczba kryteriów = 3, kryterium min
Algorytm bez filtracji:
(0.76, 0.03, 0.1)
(0.03, 0.26, 0.02)
(1.3, 0.02, 1.68)
(0.55, 0.21, 1.29)
(0.65, 0.12, 2.0)
(0.7, 0.05, 0.26)



Algorytm z filtracją punktów zdominowanych:
(0.76, 0.03, 0.1)
(0.03, 0.26, 0.02)
(0.36, 1.61, 0.01)
(0.65, 0.12, 2.0)



Algorytm oparty o punkt idealny:
(0.03, 0.26, 0.02)
(0.03, 0.13, 0.26)
(0.7, 0.05, 0.26)
(0.76, 0.03, 0.1)
(0.36, 1.61, 0.01)
(0.65, 0.12, 2.0)
(1.3, 0.02, 1.68)



Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	0.00	264
Algorytm z filtracją punktów zdomin	0.00	191
Algorytm oparty o punkt idealny	0.00	125

Liczba kryteriów = 4:
Dobór liczby kryteriów powyżej 3 spowodował, że biblioteka zliczająca czas działania algorytmu wskazała określony czas działania.

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	3.41	671
Algorytm z filtracją punktów zdomin	2.06	631
Algorytm oparty o punkt idealny	1.54	259

Liczba kryteriów = 5:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	0.00	1230
Algorytm z filtracją punktów zdomin	0.00	1199
Algorytm oparty o punkt idealny	0.00	760

Liczba kryteriów = 10:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	13.11	4406
Algorytm z filtracją punktów zdomin	9.93	5754
Algorytm oparty o punkt idealny	14.11	4158

Generowanie losowo 1000 elementowego zbioru przy użyciu rozkładu jednostajnego [0, 2], liczba kryteriów = 2:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	15.71	2003
Algorytm z filtracją punktów zdomin	15.55	1133
Algorytm oparty o punkt idealny	15.60	1063

Liczba kryteriów = 3:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	14.51	2712
Algorytm z filtracją punktów zdomin	15.63	3983
Algorytm oparty o punkt idealny	21.63	1708

Liczba kryteriów = 4:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	15.57	9088
Algorytm z filtracją punktów zdomin	17.35	3566
Algorytm oparty o punkt idealny	14.59	2723

Liczba kryteriów = 5:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	62.27	30947
Algorytm z filtracją punktów zdomin	31.25	20707
Algorytm oparty o punkt idealny	31.25	12350

Liczba kryteriów = 10:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	705.90	321708
Algorytm z filtracją punktów zdomin	470.78	399939
Algorytm oparty o punkt idealny	520.31	297005

Generujemy losowo K1 (K1≥50) zbiorów 100 elementowych o liczbie kryteriów 2,3,4,5,10 oraz K2≥25 zbiorów 1000-elementowych, liczba kryteriów jw. Wyznaczamy parametry obliczeniowe metody jak w p. (A) i porównujemy z innymi metodami z sortowaniem wstępnym i bez:

100 elementów, 2 kryteria:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	3.62	213
Algorytm z filtracją punktów zdc	2.57	208
Algorytm oparty o punkt idealny	2.24	161

100 elementów, 3 kryteria:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	3.57	378
Algorytm z filtracją punktów zdc	2.46	371
Algorytm oparty o punkt idealny	2.45	253

100 elementów, 4 kryteria:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	6.41	1170
Algorytm z filtracją punktów zdc	4.33	1373
Algorytm oparty o punkt idealny	4.00	728

100 elementów, 5 kryteriów:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	9.95	1768
Algorytm z filtracją punktów zdc	5.81	2140
Algorytm oparty o punkt idealny	5.18	1343

100 elementów, 10 kryteriów:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	18.40	4543
Algorytm z filtracją punktów zdominowanych	10.47	5949
Algorytm oparty o punkt idealny	12.80	4442

1000 elementów, 2 kryteria:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	18.68	1441
Algorytm z filtracją punktów zdc	11.65	1138
Algorytm oparty o punkt idealny	10.65	1193

1000 elementów, 3 kryteria:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	28.90	4531
Algorytm z filtracją punktów zdc	14.24	3425
Algorytm oparty o punkt idealny	11.68	1841

1000 elementów, 4 kryteria:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	36.50	11101
Algorytm z filtracją punktów zdc	19.43	11344
Algorytm oparty o punkt idealny	13.59	4901

1000 elementów, 5 kryteriów:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	61.07	23835
Algorytm z filtracją punktów zdc	23.91	22674
Algorytm oparty o punkt idealny	21.17	13665

1000 elementów, 10 kryteriów:

Algorithm	Execution Time (ms)	Number of Comparisons
Algorytm bez filtracji	403.59	346703
Algorytm z filtracją punktów zdominowanych	243.08	444368
Algorytm oparty o punkt idealny	397.28	326943