Sprawozdanie z ćwiczeń - Optymalizacja wielokryterialna dyskretna

Temat ćwiczenia:
Porównanie metod wyznaczania zbioru punktów niezdominowanych (frontu Pareto) oraz analiza wpływu liczby kryteriów na liczbę punktów niezdominowanych.

Wykonali:

Patryk Smajdor

Marek Mrówka

Data: 8.11.2025

1. Cel ćwiczenia

Celem ćwiczenia jest:

- implementacja trzech metod wyznaczania frontu Pareto:
    1. klasyczna (pełne porównania),
    2. z filtracją (sortowaniem wstępnym),
    3. metoda z punktem idealnym,
- porównanie ich efektywności obliczeniowej (czas, liczba porównań punktów i współrzędnych),
- opracowanie graficznego interfejsu użytkownika (GUI) umożliwiającego wybór metody, liczby kryteriów i generowanie danych,
- wykonanie serii eksperymentów na algorytmach,
- wyznaczenie zależności regresyjnych związanych z liczbą punktów niezdominowanych.

2. Część I – Implementacja metod

W ramach ćwiczenia zaimplementowano trzy różne podejścia do wyznaczania zbioru punktów niezdominowanych (frontu Pareto).
Wszystkie metody działają na wspólnych zasadach porównywania punktów wielokryterialnych przy zadanych kierunkach optymalizacji (minimalizacja lub maksymalizacja).
Każda z metod zwraca zestaw parametrów obliczeniowych:

- liczbę porównań punktów,
- liczbę porównań współrzędnych,
- całkowity czas obliczeń,
- oraz listę punktów niezdominowanych $P(X)$

Implementacje algorytmów znajdują się zarówno w notatniku lab2.ipynb oraz pliku metody_optymalizacji.py

Definicja dominacji i funkcje pomocnicze

Podstawą działania wszystkich metod jest funkcja **p2_dominuje_p1(p2, p1, kierunki)**, która sprawdza, czy punkt p_2**dominuje** punkt p_1.

Dominacja zachodzi wtedy, gdy:

- punkt $p_2$jest **nie gorszy** od $p_1$we wszystkich kryteriach,
- oraz **ściśle lepszy** w co najmniej jednym kryterium.

Aby obsłużyć zarówno minimalizację, jak i maksymalizację, wszystkie kryteria są przeskalowywane tak,
aby można je było traktować jako **problemy minimalizacji**.
Dla każdego punktu $p = (x_1, x_2, \dots, x_k)$obliczany jest wektor przeskalowany:

$$p' = (x_1 \cdot k_1,\ x_2 \cdot k_2,\ \dots,\ x_k \cdot k_k)$$

gdzie $k_i = 1$dla minimalizacji, a $k_i = -1$dla maksymalizacji.

Dodatkowo wykorzystano funkcję pomocniczą **odleglosc_kwadrat(p1, p2)**,
która oblicza kwadrat euklidesowej odległości między punktami i wykorzystywana jest w metodzie opartej o punkt idealny.

Algorytm klasyczny (znajdz_front_pareto)

Idea:

Metoda klasyczna polega na pełnym porównaniu każdego punktu z każdym innym punktem w zbiorze X.
Każdy punkt p_1 jest analizowany pod kątem dominacji przez inny punkt p_2.
Jeśli istnieje jakikolwiek punkt p_2, który dominuje p_1, to p_1 nie należy do zbioru niezdominowanych.

Opis działania:

1. Dla każdego punktu $p_1 \in X$:
   a. Dla każdego $p_2 \in X$, $p_2 \neq p_1$, sprawdź, czy p_2 dominuje p_1.
   b. Jeśli tak - punkt jest zdominowany i nie trafia na front Pareto.
   c. Jeśli nie - zostaje dodany do zbioru $P(X)$.
2. Po analizie wszystkich punktów zwracany jest zbiór niezdominowanych punktów.

```
znajdz_front_pareto(X, kierunki):
"""
Znajduje zbiór punktów niezdominowanych (front Pareto) z uwzględnieniem
kierunków optymalizacji i zlicza osobno:
    - liczbę porównań punktów,
    - liczbę porównań współrzędnych,
    - całkowity czas obliczeń.

Argumenty:
    X (list of list/tuple): Lista punktów wejściowych.
    kierunki (list/tuple): Lista z wartościami 1 (minimalizacja) lub -1 (maksymalizacja)
                           dla każdego kryterium.

Zwraca:
    tuple:
        - lista P punktów niezdominowanych,
        - liczba_porownan_punktow,
        - liczba_porownan_wspolrzednych,
        - czas_obliczen (sekundy)
"""
```

Rysunek 1 Opis funkcji realizującej klasyczny algorytm

Algorytm z filtracją (znajdz_front_z_filtracja)

Idea:

Metoda z filtracją jest modyfikacją klasycznego podejścia, w której eliminowane są punkty zdominowane już w trakcie obliczeń.
Pozwala to znacznie ograniczyć liczbę późniejszych porównań.

Opis działania:

1. Tworzona jest kopia zbioru X - lista punktów do przeglądu.
2. Wybierany jest pierwszy punkt jako **kandydat**.
3. Porównywany jest z pozostałymi punktami w celu znalezienia lepszego kandydata.
4. Po znalezieniu najlepszego kandydata (niezdominowanego) - dodawany jest do frontu Pareto.
5. Następuje **filtracja** - usunięcie z listy wszystkich punktów zdominowanych przez kandydata.
6. Proces powtarzany jest aż do wyczerpania zbioru.

```
znajdz_front_z_filtracja(X, kierunki):
"""
Znajduje front Pareto z agresywną filtracją, zliczając:
  - liczbę porównań punktów,
  - liczbę porównań współrzędnych,
  - całkowity czas działania.

Argumenty:
    X (list of list/tuple): Lista punktów wejściowych.
    kierunki (list/tuple): Lista z wartościami 1 (minimalizacja) lub -1 (maksymalizacja)
                           dla każdego kryterium.

Zwraca:
    tuple:
        - lista P punktów niezdominowanych,
        - liczba_porownan_punktow,
        - liczba_porownan_wspolrzednych,
        - czas_obliczen (sekundy)
"""
```

Rysunek 2 Opis funkcji realizującej algorytm z filtracją

Algorytm z punktem idealnym (algorytm_punkt_idealny)

Idea:

Trzecie podejście wykorzystuje pojęcie punktu idealnego, czyli teoretycznego punktu, który jest najlepszy we wszystkich kryteriach jednocześnie.
Dla każdego wymiaru i punkt idealny definiowany jest jako:

$$p_i^* = \begin{cases} \min\limits_{p \in X} p_i, & \text{jeśli } k_i = 1 \text{ (minimalizacja)} \\ \max\limits_{p \in X} p_i, & \text{jeśli } k_i = -1 \text{ (maksymalizacja)} \end{cases}$$

Punkty rzeczywiste porównywane są względem odległości od tego punktu - bliższe punktowi idealnemu mają większe szanse na bycie niezdominowanymi.

Opis działania:

1. Obliczenie punktu idealnego p^*.
2. Przeskalowanie punktów (dla kierunków minimalizacji).
3. Dla każdego punktu $p \in X$ wyznaczana jest kwadratowa odległość euklidesowa:

$$d(p) = \sum_i (p_i - p_i^*)^2$$

4. Punkty sortowane są rosnąco wg odległości $d(p)$.
5. Następnie punkty przeglądane są w tej kolejności, a każdy nowy punkt sprawdzany, czy nie jest zdominowany przez już znalezione.
   Zdominowane punkty są usuwane z dalszej analizy.
6. Zwracany jest zbiór punktów niezdominowanych oraz punkt idealny.

```
algorytm_punkt_idealny(X, kierunki):
"""
Algorytm znajdowania frontu Pareto oparty o punkt idealny,
zliczający porównania punktów, współrzędnych oraz czas wykonania.

Argumenty:
    X (list of list/tuple): Lista punktów wejściowych.
    kierunki (list/tuple): 1 = minimalizacja, -1 = maksymalizacja.

Zwraca:
    tuple:
        - lista P punktów niezdominowanych,
        - liczba_porownan_punktow,
        - liczba_porownan_wspolrzednych,
        - czas_obliczen (sekundy)
"""
```

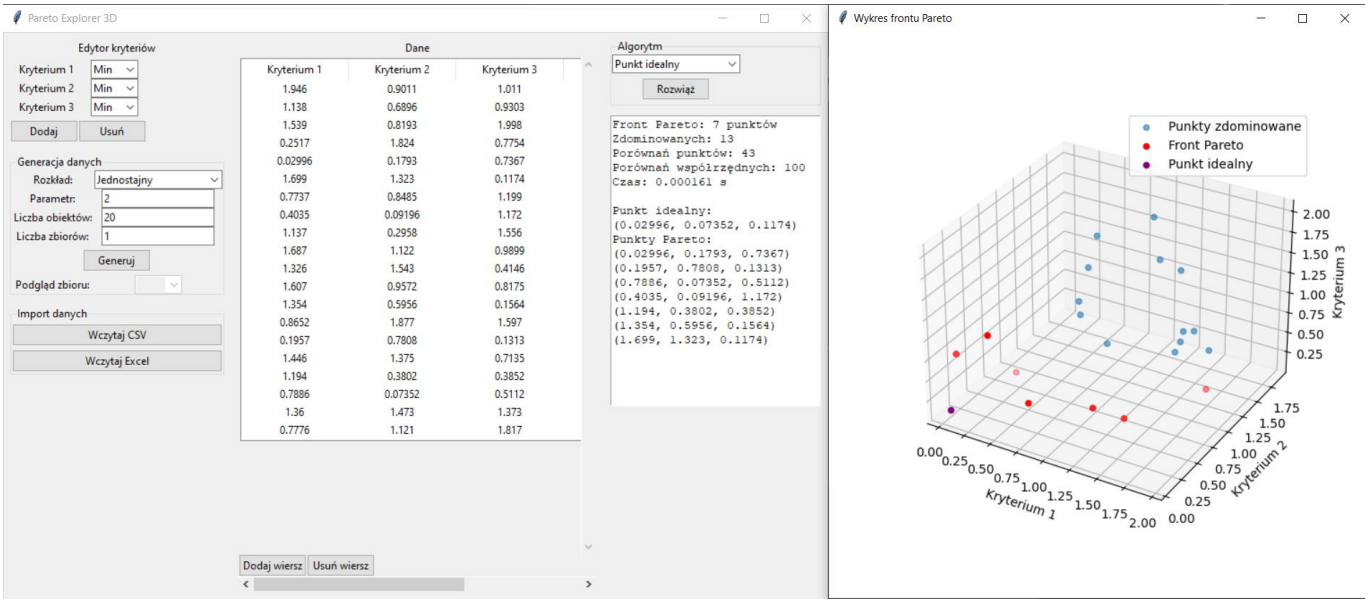
Rysunek 3 Opis funkcji realizującej algorytm oparty o punkt idealny

3. Część II – Stworzenie GUI

Na potrzeby ćwiczeń utworzono interfejs graficzny umożliwiający:

- wybór metody obliczeniowej,
- ustawienie liczby punktów i kryteriów,
- wybór rozkładu losowego (jednostajny, normalny, eksponencjalny),
- wybór liczby generowanych danych oraz liczbę generowanych zestawów danych,
- generację danych oraz import punktów z pliku .csv lub Excel,
- edycję danych w tabeli,
- wizualizację wyników (czas, liczba porównań, liczba punktów Pareto) wraz z wykresem,

Załączony zrzut ekranu GUI:

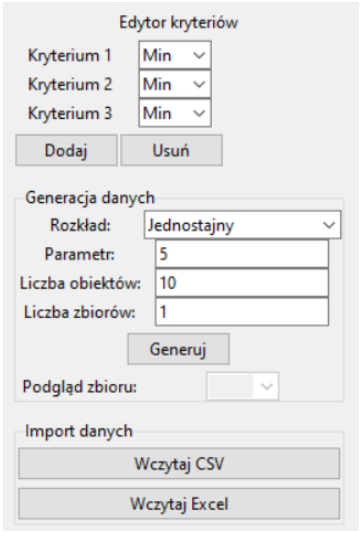


Rysunek 4 Widok GUI wraz z przedstawieniem wyników na przykładowych danych

Celem uruchomienia interfejsu graficznego należy uruchomić plik gui.py, ważne jest aby w środowisku roboczym zamieścić plik metody_optymalizacji.py zawierający definicje algorytmów wykorzystywanych przez interfejs.

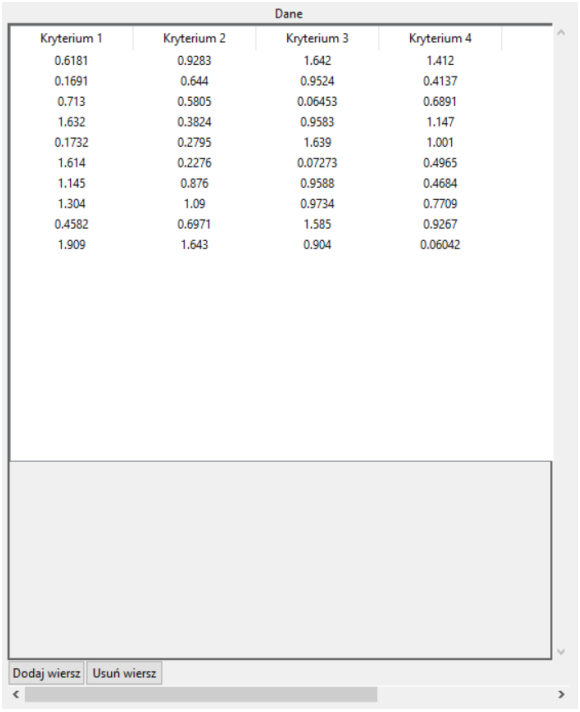
W oknie edytora kryteriów możliwe jest dodanie lub usunięcie kryterium oraz specyfikacja, czy dane kryterium powinno zostać poddane maksymalizacji bądź minimalizacji.

Okno generacji danych pozwala na wybór rodzaju rozkładu, wprowadzenie parametru rozkładu, aż w końcu liczbę obiektów i liczbę generowanych zbiorów danych(dla więcej niż jednego wykonywane są zbiorcze obliczenia). Po naciśnięciu przycisku generuj, rekordy zostaną wygenerowane i będą widoczne w oknie ‘Dane’.



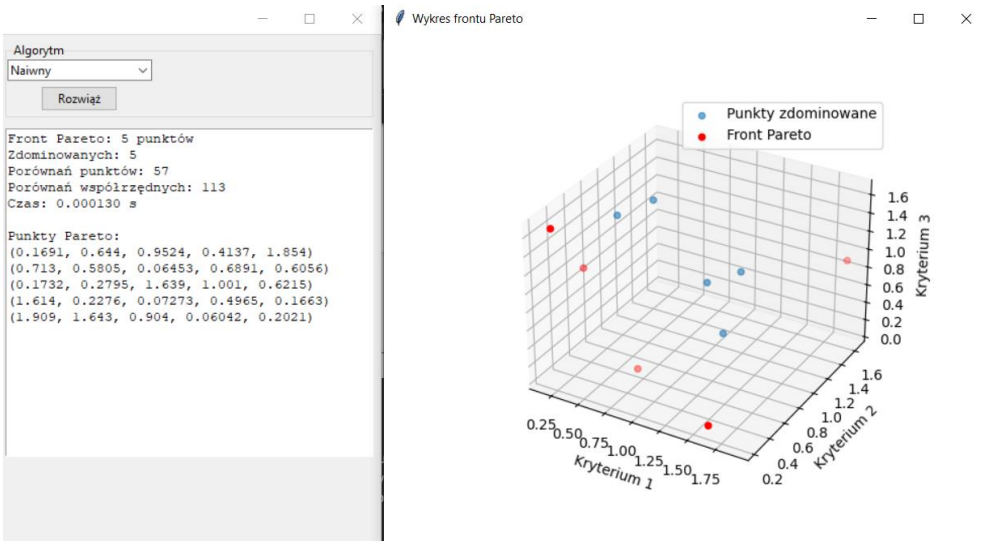
Rysunek 5 Okno edytora kryteriów i generacji danych

W oknie danych widoczne są zaimportowane lub wygenerowane punkty (wiersze) z widocznymi kryteriami (kolumnami). W oknie tym możliwa jest edycja danych poprzez kliknięcie w wybraną komórkę tabeli i wprowadzenie nowej wartości. W dodatku można dodać lub usunąć określony wiersz(punkt) za pomocą przycisków. Celem sprawdzenia wartości punktów i kryteriów wykraczających poza rozmiar okienka, wprowadzono suwaki umożliwiające swobodny przegląd punktów.



Rysunek 6 Widok okna danych

W trzeciej części, okienku wynikowym możliwy jest wybór algorytmu do przeprowadzenia obliczeń. Po naciśnięciu przycisku ‘rozwiąż’ wyświetlone zostaną wyniki oraz wykres z oznaczonymi punktami niezdominowanymi i punktem idealnym w przypadku użycia algorytmu opartego o punkt idealny.



Rysunek 7 Okno wyników działania algorytmu

4. Eksperymenty i wyniki

Eksperyment 1 - porównanie trzech metod

Celem eksperymentów było porównanie trzech metod wyznaczania frontu Pareto uprzednio opisanych.

Eksperymenty przeprowadzono dla losowo wygenerowanych danych o liczbie punktów: **n = 100, 1000, 10000**, oraz liczbie kryteriów **k = 2, 3, 4, 5, 10**.

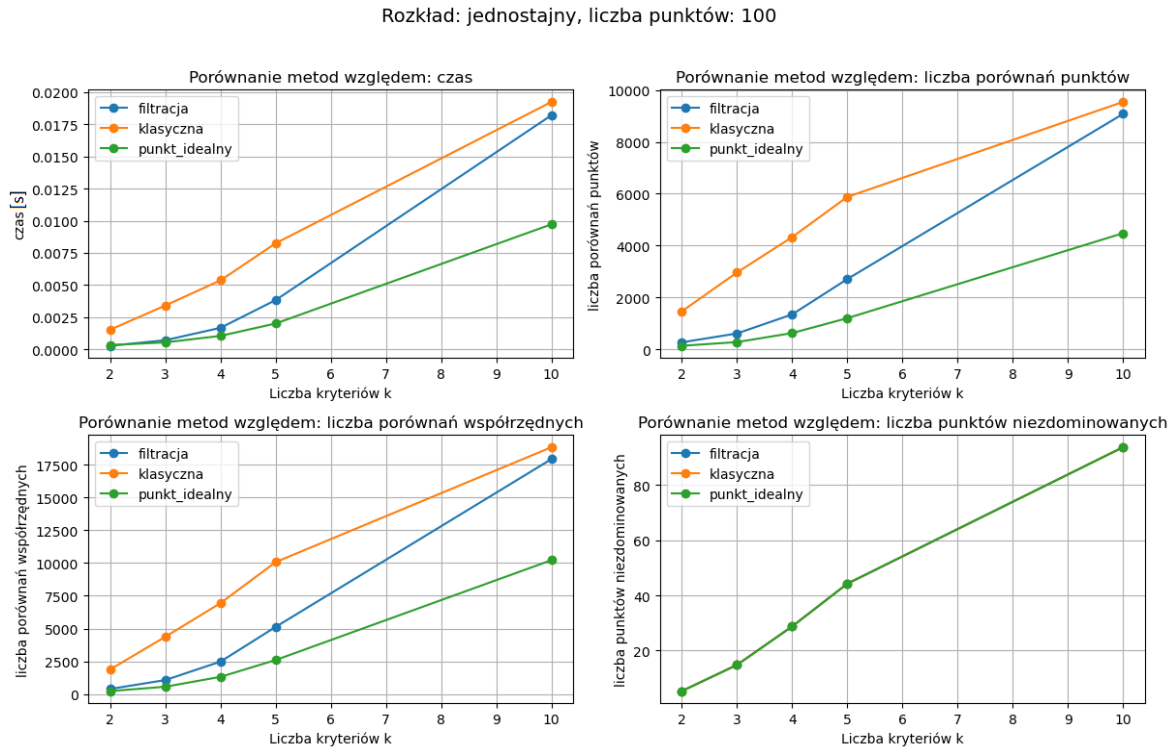
Uwzględniono dwa rozkłady:

- **Jednostajny** na przedziale [0, 2],
- **Poissona(λ=2)**.

Każdy eksperyment powtórzono $\mathbf{K_1 = 50}$, $\mathbf{K_2 = 25}$, $\mathbf{K_3 = 20}$ razy odpowiednio dla zbiorów 100-, 1000- i 10000-elementowych.
Dla każdej kombinacji (rozkład, n, k, metoda) obliczano:

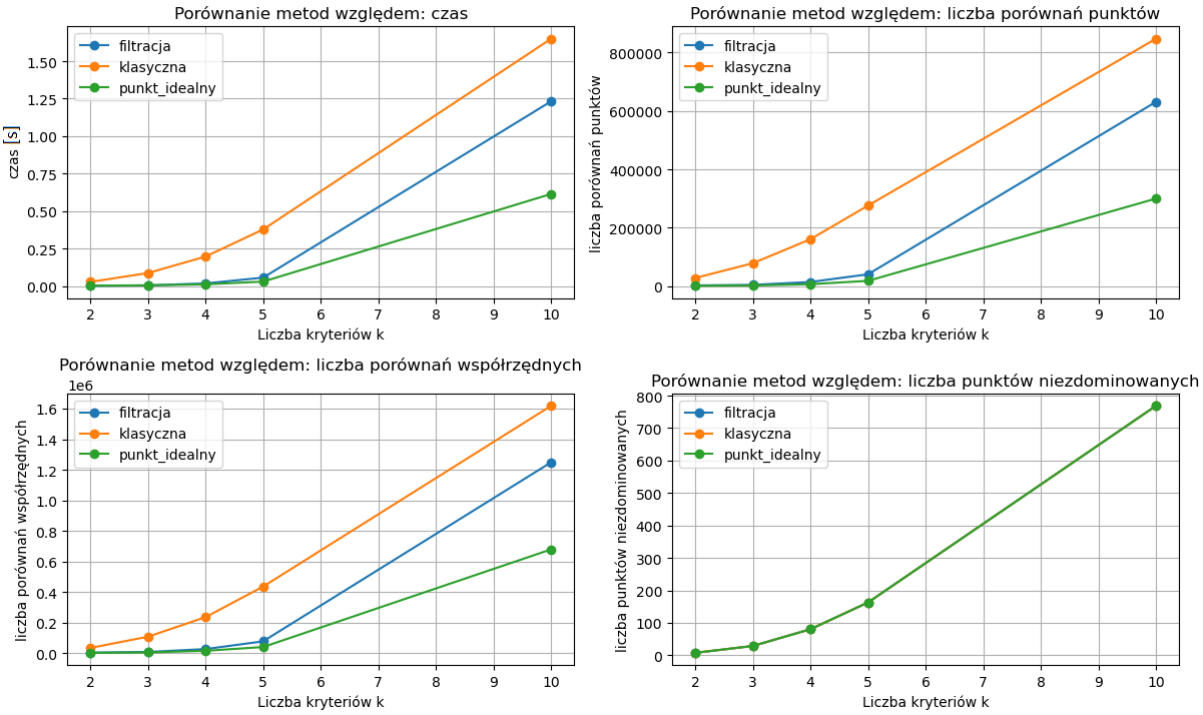
- czas obliczeń,
- liczbę porównań punktów i współrzędnych,
- liczbę punktów niezdominowanych.

Wyniki zostały zapisane w ramce danych i zapisane do pliku CSV, a następnie średnie wyniki zwizualizowano w postaci wykresów liniowych:



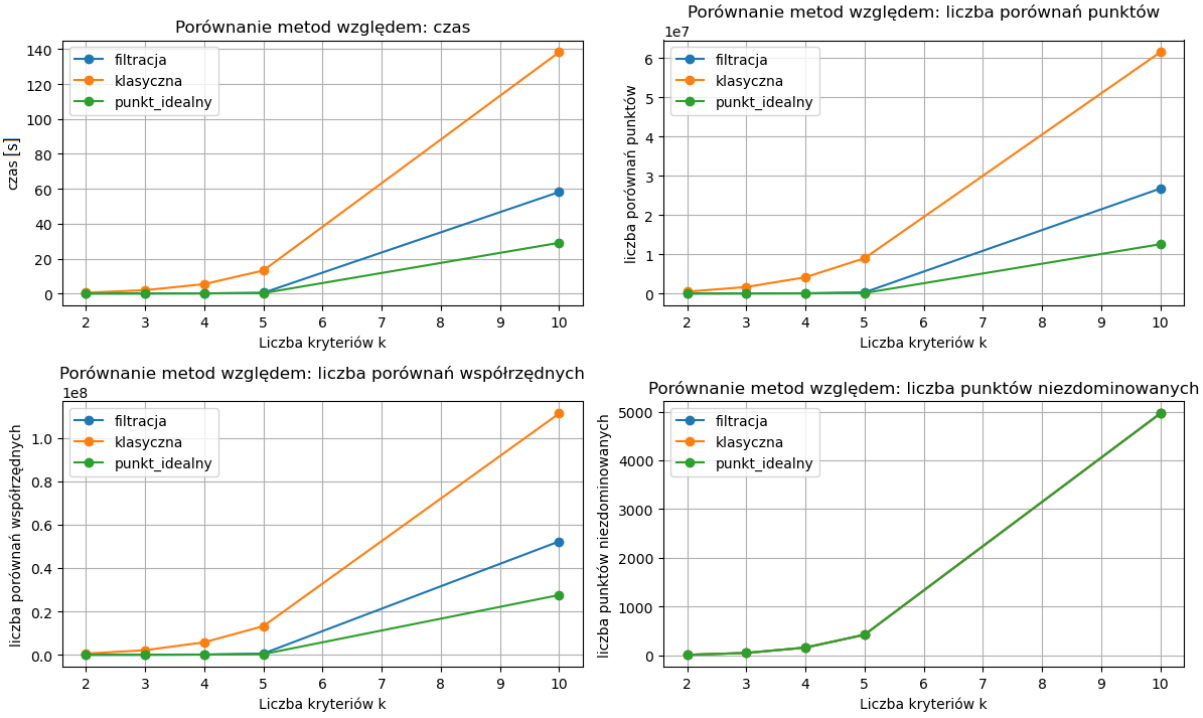
Rysunek 8 Wizualizacja wyników dla: rozkładu jednostajnego, liczba punktów: 100

Rozkład: jednostajny, liczba punktów: 1000



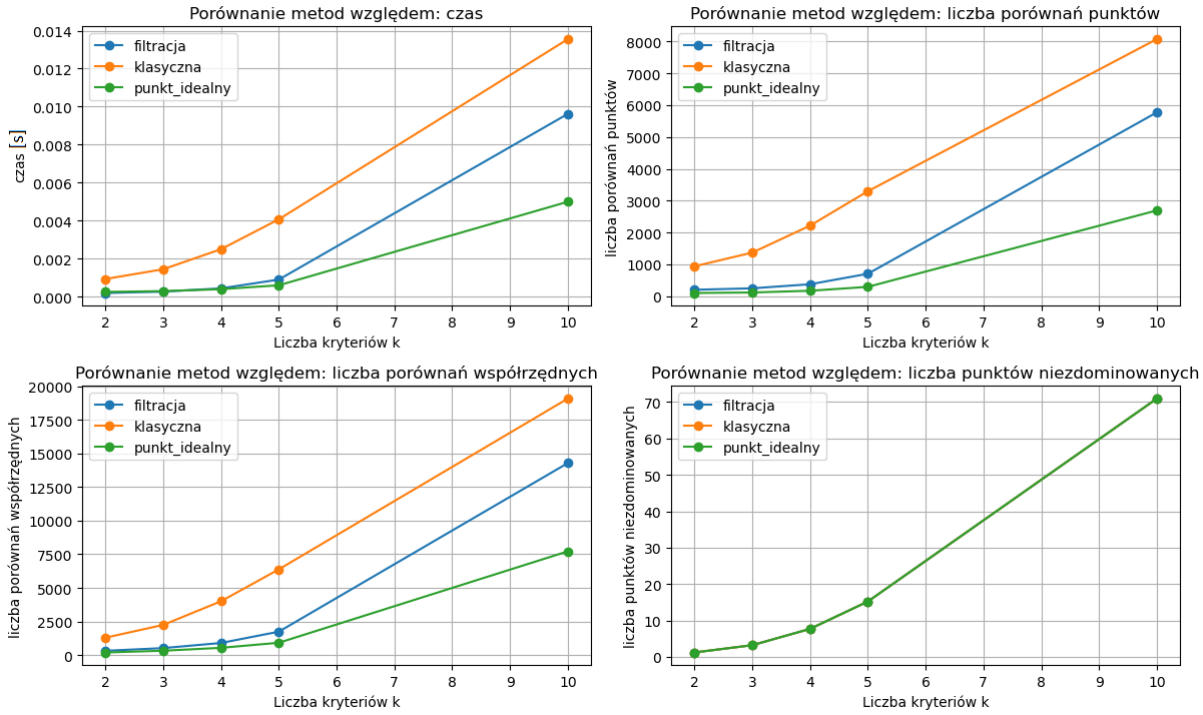
Rysunek 9 Wizualizacja wyników dla: rozkładu jednostajnego, liczba punktów: 1000

Rozkład: jednostajny, liczba punktów: 10000



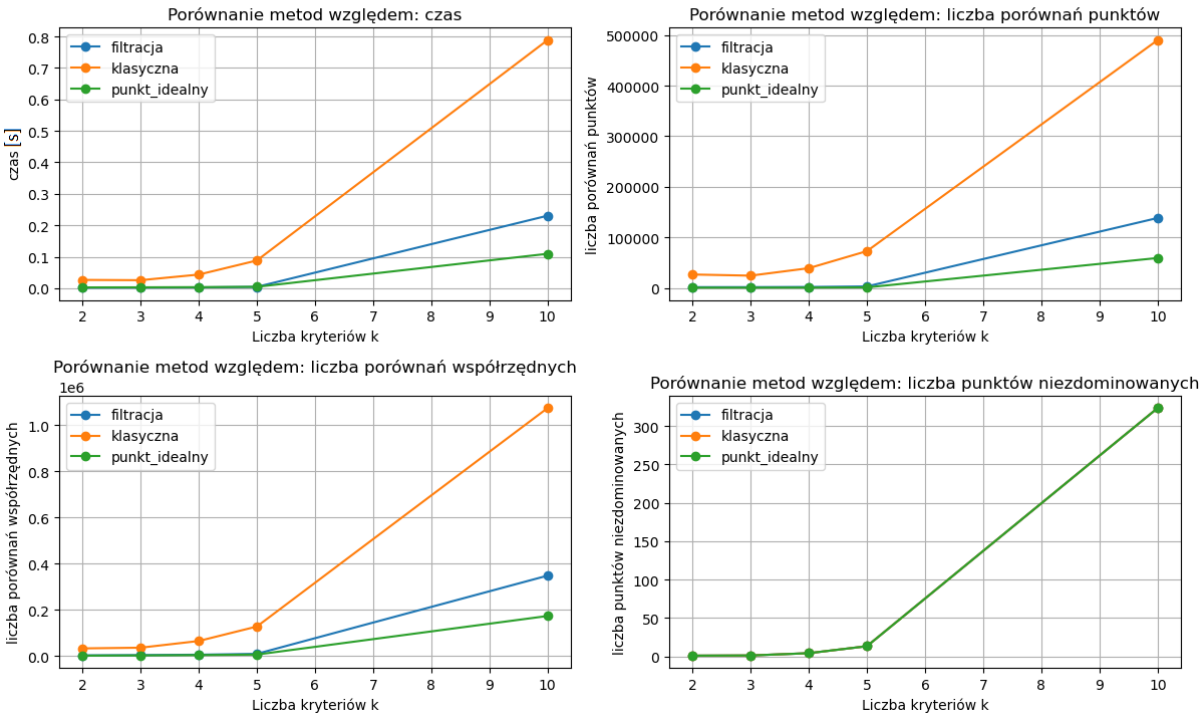
Rysunek 10 Wizualizacja wyników dla: rozkładu jednostajnego, liczba punktów: 10000

Rozkład: poissona, liczba punktów: 100



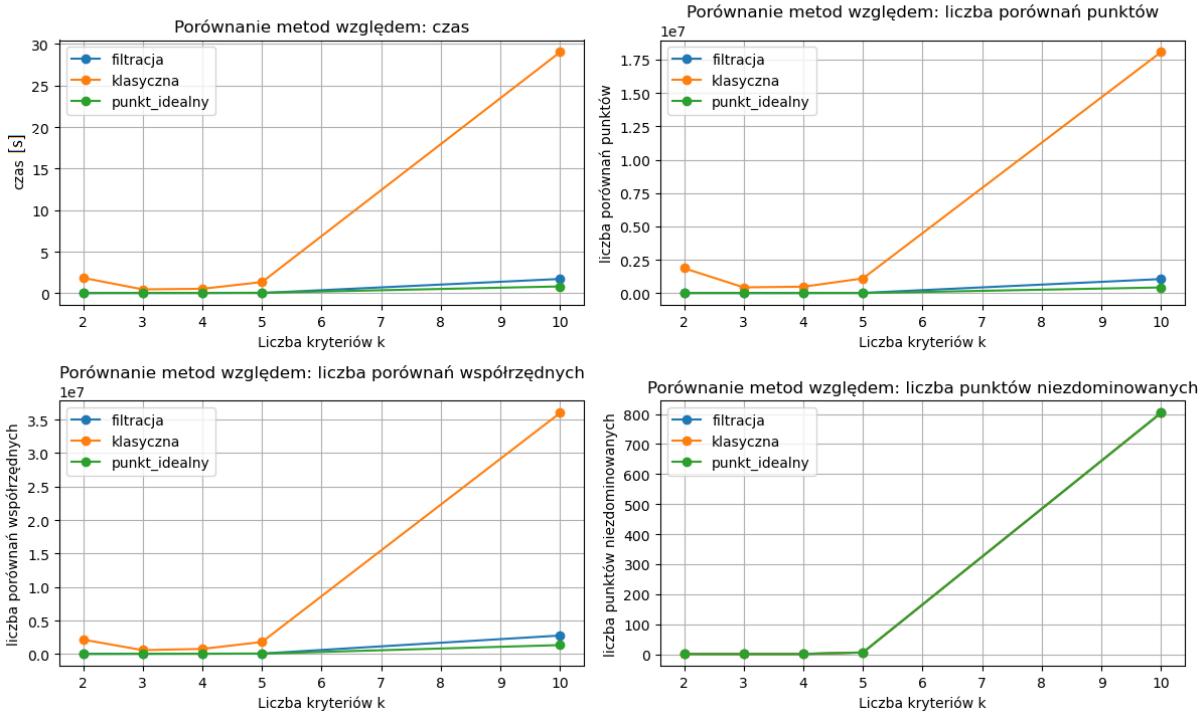
Rysunek 11 Wizualizacja wyników dla: rozkładu Poissona , liczba punktów: 100

Rozkład: poissona, liczba punktów: 1000



Rysunek 12 Wizualizacja wyników dla: rozkładu Poissona , liczba punktów: 1000

Rozkład: poissona, liczba punktów: 10000



Rysunek 13 Wizualizacja wyników dla: rozkładu Poissona , liczba punktów: 10000

Eksperyment 2 - analiza regresyjna zależności P(X)

Drugi zestaw eksperymentów miał na celu zbadanie, jak **liczba punktów niezdominowanych** zależy od liczby kryteriów w zbiorze danych.

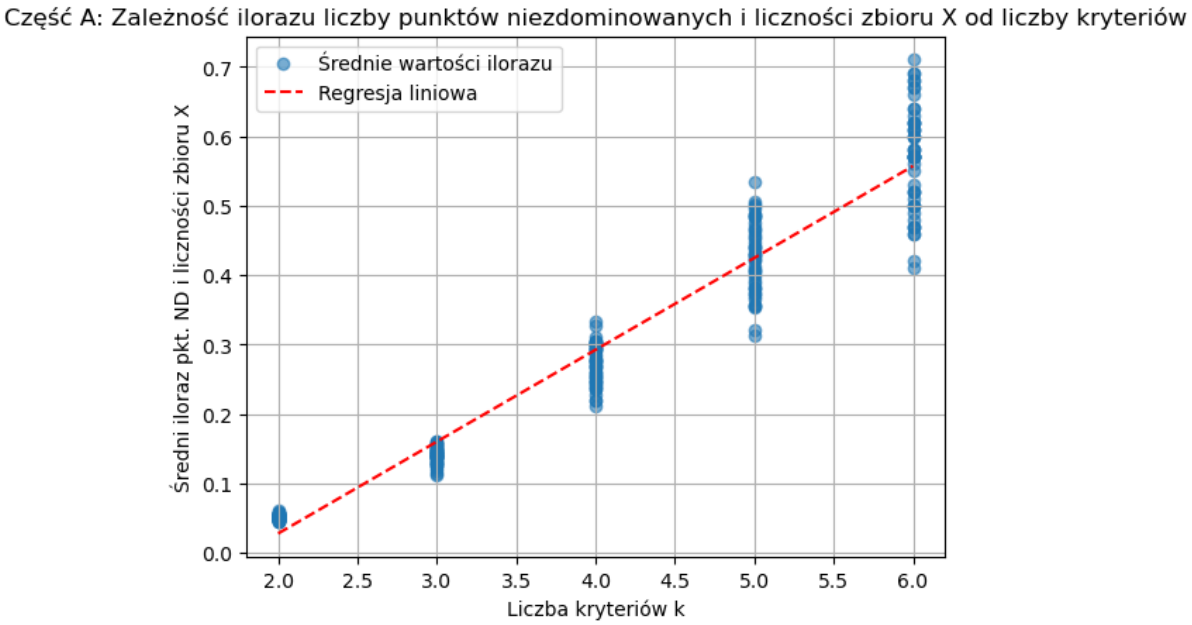
Zrealizowano dwie analizy regresyjne:

A) Zależność liczby punktów niezdominowanych od liczby kryteriów (k)

Dla $k_0 = 6$ i 50 zbiorów 100-elementowych wygenerowano wszystkie kombinacje problemów k-kryterialnych (dla $2 \leq k \leq k_0$).
Dla każdej kombinacji wyznaczono iloraz:

$$\frac{|\ P(X_k)\ |}{|\ X\ |}$$

i wyznaczono regresję liniową zależności tego ilorazu od liczby kryteriów *k*.



Rysunek 14 Wizualizacja wyników dla: zależność ilorazu liczby punktów i liczności zbioru od liczby kryteriów

B) Zależność liczby punktów niezdominowanych przy zmniejszaniu liczby kryteriów

Dla $k_0 = 7$ i 50 zbiorów 100-elementowych wyznaczano front Pareto dla pełnego zestawu (k_0 kryteriów), a następnie dla wszystkich podzestawów mniejszych ($k < k_0$).

Dla każdej kombinacji obliczano:

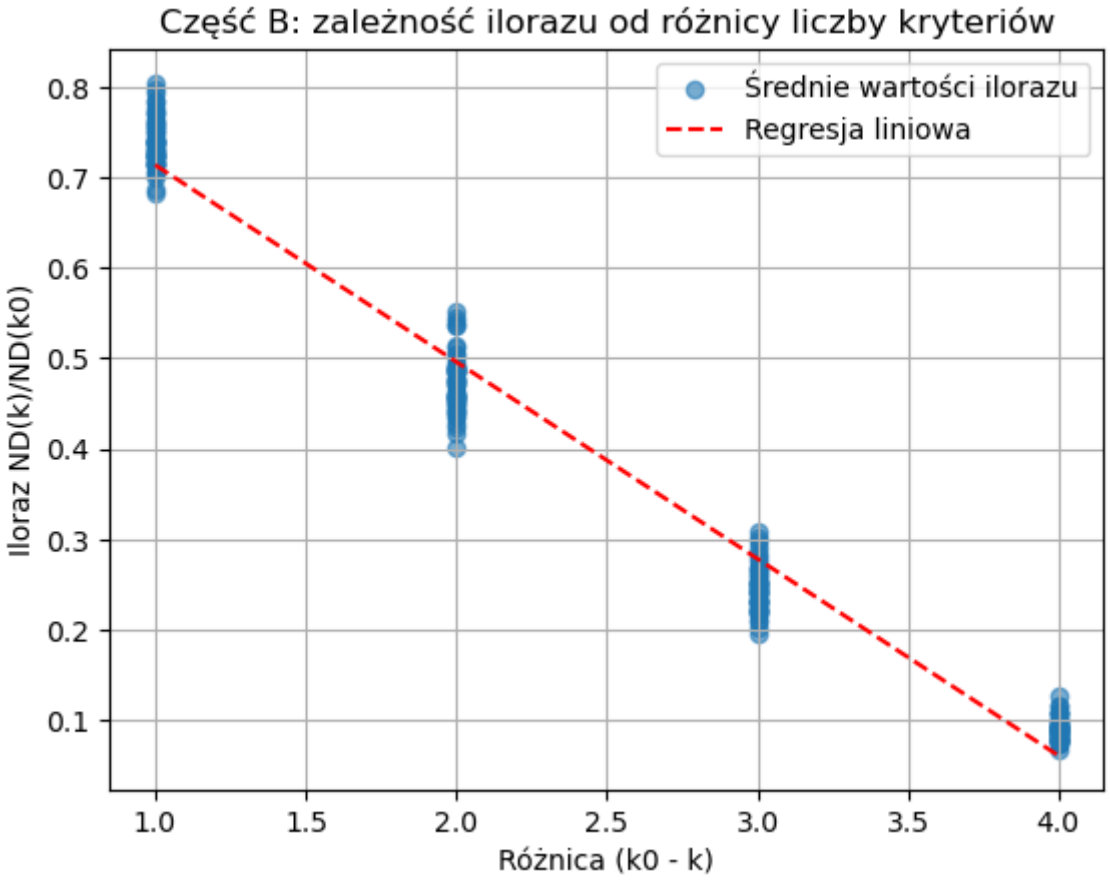
$$\frac{| P(X_k) |}{| P(X_{k_0}) |}$$

i analizowano regresję liniową tej wartości w zależności od różnicy liczby kryteriów ($k_0 - k$).

W obu przypadkach przeprowadzono eksperymenty dla każdej z trzech metod:

- klasycznej,
- z filtracją,
- z punktem idealnym.

Wyniki eksperymentu zostały przedstawione w postaci wykresów liniowych:



Rysunek 15 Wizualizacja wyników dla: zależności ilorazu od różnicy liczby kryteriów

5. Analiza i interpretacja wyników

Wyniki eksperymentu 1

Czas obliczeń

- czas obliczeń rośnie **wraz z liczbą punktów (n)** oraz **liczbą kryteriów (k)**,
- **metoda klasyczna** charakteryzuje się najwyższym czasem obliczeń - zwłaszcza dla n = 10 000, gdzie złożoność kwadratowa staje się bardzo widoczna,
- **metoda z filtracją** skraca czas o ok. 40–60% w stosunku do klasycznej,
- **metoda punktu idealnego** jest najszybsza dla małych wartości k, jednak jej przewaga maleje przy większej liczbie kryteriów, co sugeruje wzrost kosztu porównywania odległości do punktu idealnego w wyższych wymiarach.

Liczba porównań

- Metoda klasyczna wykonuje największą liczbę porównań punktów i współrzędnych.
- Filtracja i punkt idealny znacznie redukują liczbę niepotrzebnych operacji – nawet sześciokrotnie.

Liczba punktów niezdominowanych

- Liczba punktów niezdominowanych rośnie wraz z liczbą kryteriów.

Wykresy wskazują, że metoda z filtracją zapewnia najlepszy kompromis między dokładnością a szybkością.

Wyniki eksperymentu 2A - zależność ilorazu $|P(X_k)| / |X|$ od liczby kryteriów k

Dla każdej metody uzyskano liniową zależność w postaci:

$$\frac{|\ P(X_k)\ |}{|\ X\ |} = a \cdot k + b$$

Dla metody klasycznej:

y = 0.1323 * k -0.2365

- Udział punktów niezdominowanych w zbiorze rośnie średnio o ok. 13% przy każdym dodatkowym kryterium.
- Dla małej liczby kryteriów (k = 2–3) liczba punktów niezdominowanych stanowi jedynie ok. 5–10% zbioru.
- Dla większych k (np. k = 6) udział ND przekracza 50%, co wskazuje, że wzrost wymiarowości znacznie utrudnia dominację punktów.
- Wartość ujemnego wyrazu wolnego (-0.2365) oznacza, że dla bardzo małego k proporcja ND jest bliska zera - co jest zgodne z intuicją.

Wykresy regresji potwierdzają trend liniowy w całym zakresie analizowanych k.

Wyniki eksperymentu 2B - zależność $|P(X_k)|$ / $|P(X_{k_0})|$ od różnicy $(k_0 - k)$

Dla każdej metody dopasowano model regresji:

$$\frac{|\,P(X_k)\,|}{|\,P(X_{k_0})\,|} = c \cdot (k_0 - k) + d$$

Dla metody klasycznej:

y = -0.2178 * (k0 - k) + 0.9317

- Dla każdego zmniejszenia liczby kryteriów o 1, liczba punktów niezdominowanych maleje o ok. 22% względem zbioru odniesienia (pełnego k_0).
- Dla $(k_0 - k) = 0$, czyli dla pełnej liczby kryteriów, wartość ilorazu wynosi ≈ 0.93, co oznacza, że front Pareto pozostaje duży i stabilny.
- Malejący trend potwierdza intuicję: im mniej kryteriów, tym łatwiej o dominację między punktami.

Wykresy regresji pokazują wyraźny spadek wartości ilorazu wraz ze wzrostem $(k_0 - k)$.

6. Wnioski

Porównanie metod

- Metoda klasyczna potwierdza poprawność, lecz jest najwolniejsza ($O(n^2)$).
- Metoda z filtracją oferuje najlepszy kompromis - podobną dokładność przy znacznym skróceniu czasu.
- Metoda z punktem idealnym jest najszybsza w niskich wymiarach, ale traci względną efektywność dla dużych k.

Zależność od liczby kryteriów

- Liczba punktów niezdominowanych rośnie liniowo wraz z k

  (A: $y = 0.1323k - 0.2365$).

- Oznacza to, że każde nowe kryterium znacząco poszerza front Pareto i zmniejsza stopień dominacji.

Wpływ redukcji liczby kryteriów

- Redukcja liczby kryteriów powoduje spadek liczby punktów ND

  (B: $y = -0.2178(k_0 - k) + 0.9317$).

- Średni spadek wynosi ok. 20% na każde usunięte kryterium, co potwierdza silny wpływ wymiarowości na strukturę frontu.

Modelowanie regresyjne

- W obu częściach (A i B) model liniowy dobrze opisuje obserwowane zależności.
- W przyszłości warto sprawdzić modele potęgowe lub wykładnicze, które mogą lepiej oddać zachowanie frontu Pareto dla dużych k.

Wnioski praktyczne

- W zastosowaniach wielokryterialnych liczba kryteriów ma kluczowy wpływ na strukturę wyników.
- Dla wysokich wymiarów konieczne jest stosowanie metod redukujących złożoność, np. filtracji, dominacji częściowej lub aproksymacji frontu Pareto.