Adam Linke indeks: 121529
Mateusz Norel indeks: 126901

Sprawozdanie - Algorytmy Ewolucyjne i Metaheurystyczne

Repozytorium z kodem: https://github.com/imegirin/AEM

Część 1

1. Celem zajęć było wprowadzenie do algorytmów grupujących, które to pełnią dużą rolę w czynnościach związanych z szeroko rozumianym odkrywaniem danych. W ramach zajęć studenci przygotowywali proste algorytmy, których zadaniem było przyporządkować 201 punktów do 10, nieokreślonych grup. Jako miarę jakości algorytmu przyjęto sumę po długości minimalnych drzew rozpinających z każdej grupy.
2. Algorytm zachłanny:

   dla każdego wierzchołka w rozpatrywanej przestrzeni
   znajdź najbliższy mu klaster
   dodaj wierzchołek do znalezionego klastra
   koniec pętli

3. Algorytm heurystyka z *'żalem'*:

   dla każdej pary wylosowanych wierzchołków
   sprawdź, do której grupy najbliżej pierwszemu wierzchołkowi
   sprawdź, do której grupy najbliżej drugiemu wierzchołkowi
   tymczasowo dodaj drugi wierzchołek do znalezionej grupy
   sprawdź, do której grupy najbliżej pierwszemu wierzchołkowi
   jeżeli pierwszy wierzchołek zmienił grupę docelową po dodaniu drugiego wierzchołka do jednej z grup **to**
   zwróć pierwszy wierzchołek do rozpatrywanej przestrzeni
   dodaj drugi wierzchołek na stałe do najbliższej mu grupy
   w przeciwnym przypadku
   wycofaj dodanie drugiego wierzchołka do grupy
   zwróć drugi wierzchołek do rozpatrywanej przestrzeni
   dodaj pierwszy wierzchołek do najbliższej mu grupy
   koniec warunku
   koniec pętli

4. Eksperymenty przeprowadzono wykonując każdy algorytm po 100 razy. Wyniki prezentują się następująco:

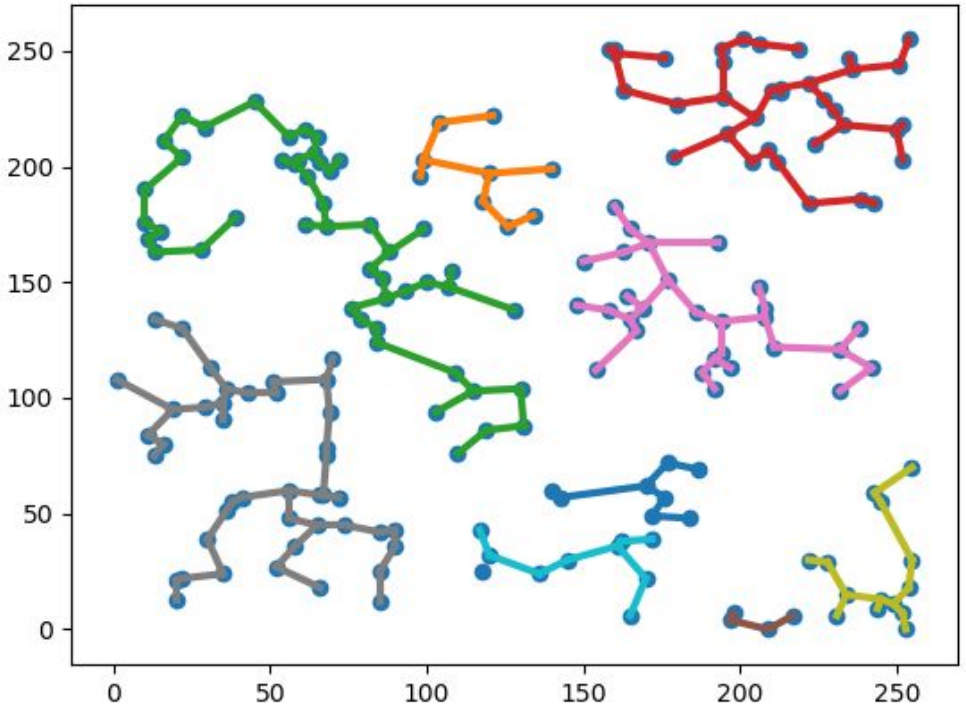
Metoda zachłanna:

Czas wykonania:

średni: 0.0001226,
najmniejszy: 0.00005211,
największy: 0.0065074

Wynik:

średni: 2128.2853,
najmniejszy: 2067.7949,
największy: 2204.6536



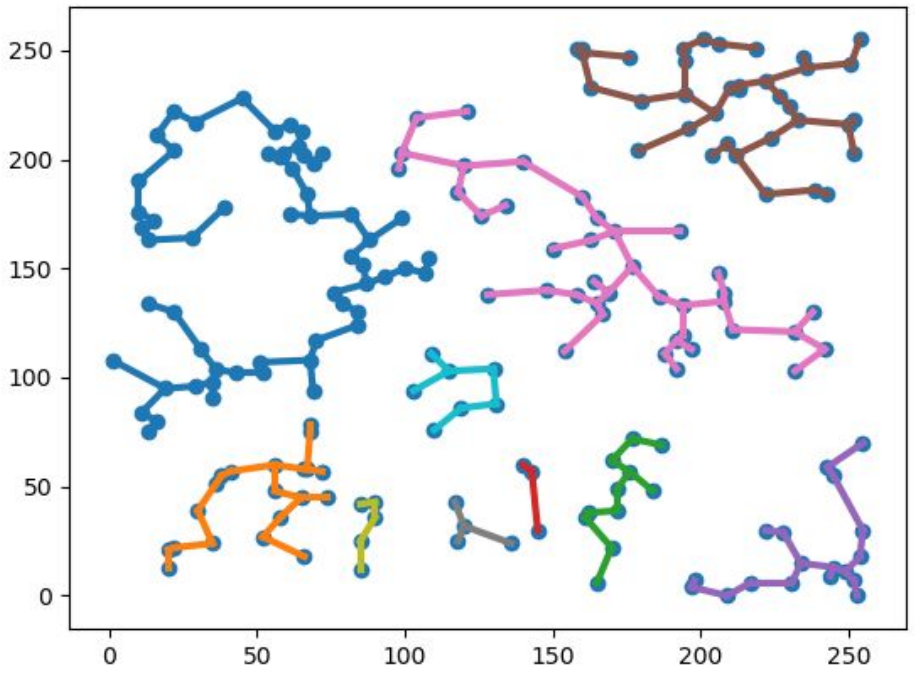
Metoda z ‘żalem’:

Czas wykonania:

średni: 0.0001027,
najmniejszy: 0.00005124,
największy: 0.005747

Wynik:

średni: 2128.676,
najmniejszy: 2053.2797,
największy: 2203.25924



Część 2

1. Program wykonujący grupowanie został wzbogacony o algorytmy lokalnego przeszukiwania w wersji zachłannej i stromej. Nową miarą jest średnia odległość między obiektami umieszczonymi w tej samej grupie. By zoptymalizować obliczenia, za każdym razem, gdy wywoływana jest funkcja, sprawdzane są zmiany między grupami, a wynikiem jest wartość wyznaczona w czasie poprzednich operacji rozszerzona o poprawki uwzględniające nowe lub brakujące obiekty w grupie. Jako rozwiązanie sąsiednie przyjęto takie, w którym jeden z obiektów przypisany jest do innej grupy niż rozwiązanie, którego sąsiedztwo jest rozpatrywane.
2. Pseudokod wersji stromej:
   powtarzaj dla każdego z nowych rozwiązań
   dla każdego obiektu
   sprawdź funkcję oceny po dołożeniu do każdej grupy
   zapisz najlepszy wynik
   koniec pętli
   jeśli znaleziono lepszy wynik
   wybierz najlepszy wynik i umieść w rozwiązaniu
   zwróć rozwiązanie i **wróć** do początku pętli
   w przeciwnym razie opuść wszystkie pętle i zakończ algorytm
   koniec pętli
3. Pseudokod wersji zachłannej:
   powtarzaj dla każdego z nowych rozwiązań

dla każdego obiektu
dla każdej grupy
jeśli przełożenie obiektu do tej grupy poprawi wynik
utwórz nowe rozwiązanie z nowym przypisaniem
zwróć rozwiązanie i **wróć** do początku pętli
w przeciwnym razie idź dalej
koniec pętli
koniec pętli
jeśli dotarłeś tutaj opuść wszystkie pętle i zakończ algorytm
koniec pętli

4. Wyniki eksperymentów:
   a. Stromy algorytm lokalnego przeszukiwania zainicjalizowany zachłannym algorytmem budowania klastrów

	średni	najniższy	najwyższy
czas	185 s	101 s	358 s
wynik	558.31	545.32	572.63

   b. Zachłanny algorytm lokalnego przeszukiwania zainicjalizowany zachłannym algorytmem budowania klastrów

	średni	najniższy	najwyższy
czas	0.811 s	0.45 s	6.10 s
wynik	557.9	545.3	572.33

   c. Stromy algorytm lokalnego przeszukiwania zainicjalizowany algorytmem losowego budowania klastrów

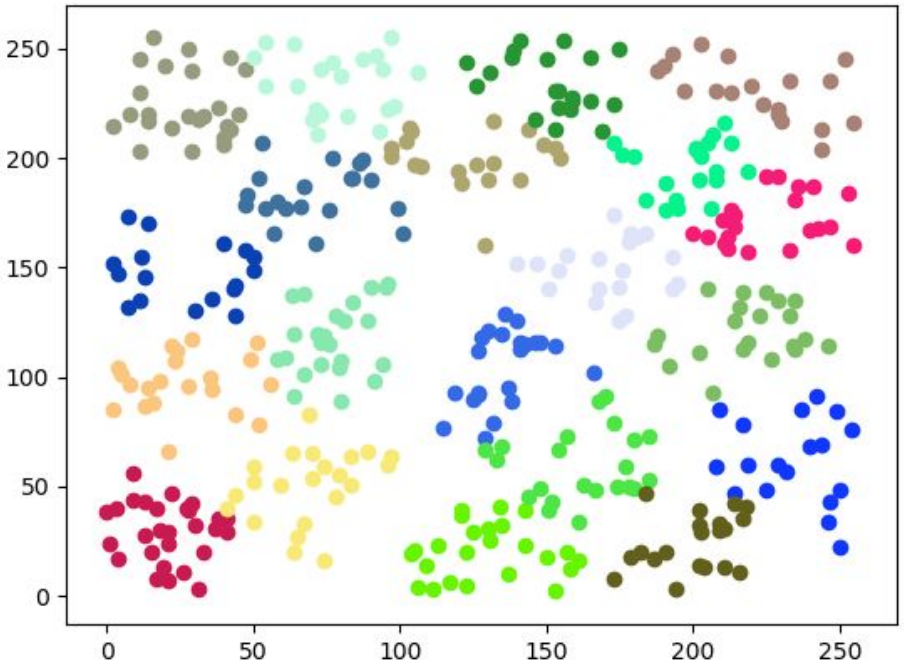
	średni	najniższy	najwyższy
czas	317.82 s	228.63 s	479.96 s
wynik	557.36	544.08	579.97

   d. Zachłanny algorytm lokalnego przeszukiwania zainicjalizowany algorytmem losowego budowania klastrów

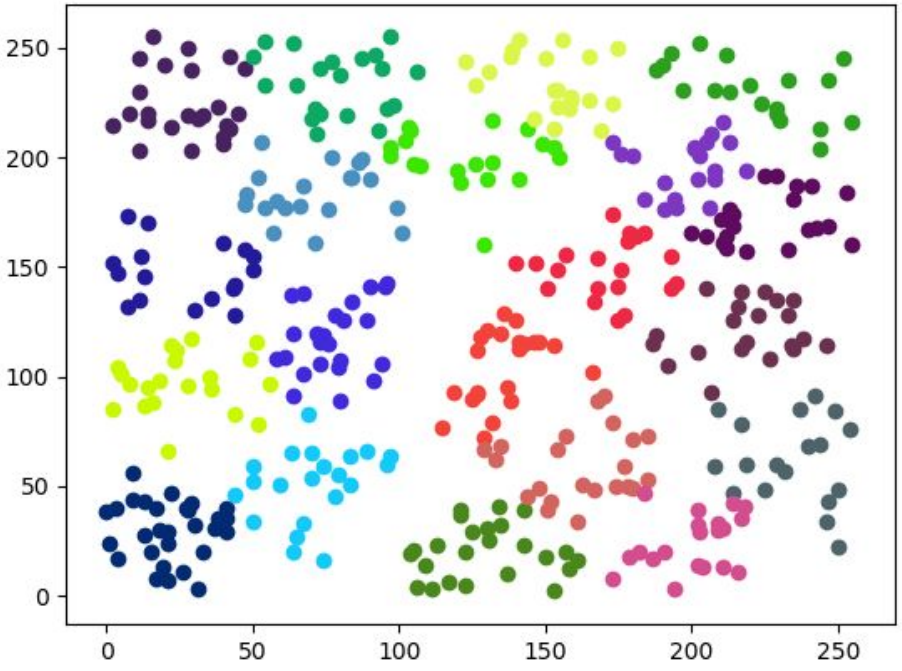
	średni	najniższy	najwyższy
czas	0.59 s	0.45 s	1.46 s
wynik	557.36	544.08	579.96

Wizualizacja najlepszego wyniku dla każdego algorytmu:

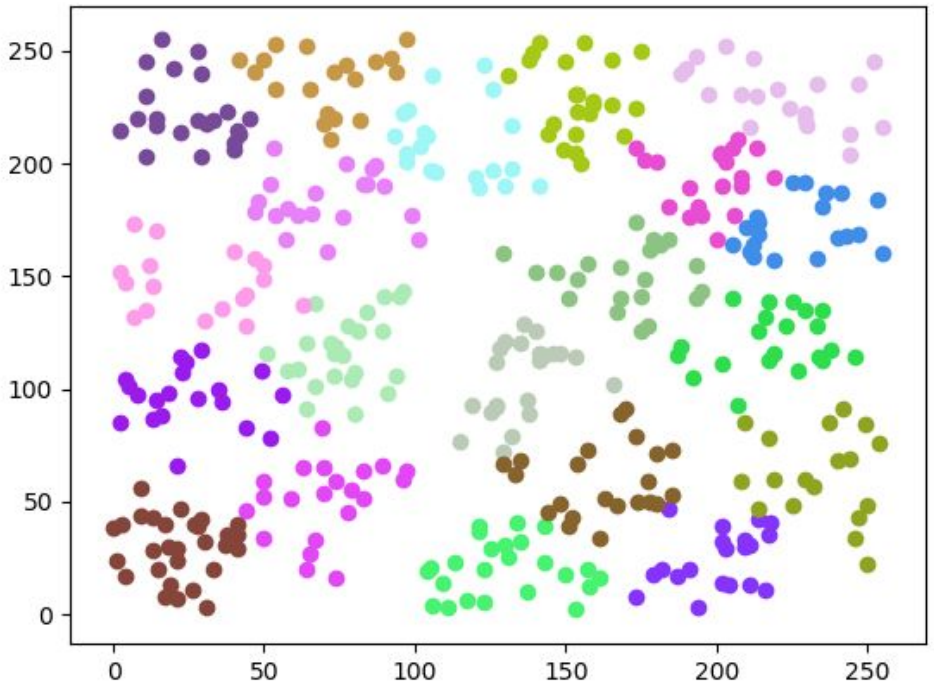
A. Stromy algorytm lokalnego przeszukiwania zainicjalizowany zachłannym algorytmem budowania klastrów



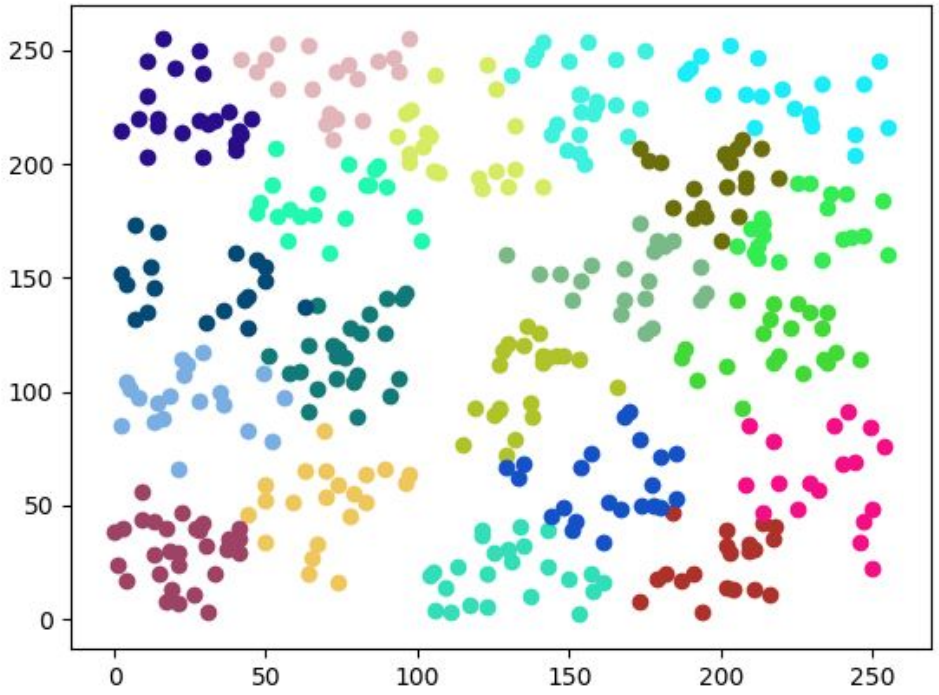
B. Zachłanny algorytm lokalnego przeszukiwania zainicjalizowany zachłannym algorytmem budowania klastrów



C. Stromy algorytm lokalnego przeszukiwania zainicjalizowany algorytmem losowego budowania klastrów



D. Zachłanny algorytm lokalnego przeszukiwania zainicjalizowany algorytmem losowego budowania klastrów



Część 3

1. Celem zadanie była poprawa efektywności lokalnego poszukiwania poprzez cachowanie obliczeń z poprzednich iteracji i zastosowanie ruchów kandydackich. Stworzono implementacja wykorzystuje do tego celu słownik, w którym zapamiętywany jest zysk informacyjny przy przeniesieniu konkretnego punktu z określonej grupy do innej określonej grupy oraz pomocnicza tablica zawierająca *k* najbliższych sąsiadów dla rozpatrywanego punktu, na podstawie której wybierane są grupy kandydackie dla przeniesienia.
   Do ruchów kandydackich wykorzystywane są krótsze listy najbliższych punktów. Sprawdzane jest w jakich zbiorach znajdują się punkty.
   Wyznaczając funkcję celu, decydującą o przeniesieniu punktu do innej grupy, zdecydowano się na na liczenie jedynie zmiany w pełnej funkcji celu pod wpływem przeniesienia punktu. Wyznaczana jest suma odległości punktów do każdego punktu w grupie do której należy oraz suma odległości do punktów w porównywanej grupie. Następnie punkt jest umieszczany w grupie której suma odległości ma najniższą wartość.

2. Pseudokod

```
powtarzaj dla każdego z nowych rozwiązań
        dla każdego obiektu
                jeżeli uwzględniać ruchy kandydackie
                        znajdź grupy do których należą sąsiedzi
                w przeciwnym razie
                        bierz pod uwagę wszystkie grupy
                dla każdej grupy
                        wyznacz ruch
                        jeżeli obecny ruch znajduje się w pamięci podręcznej
                                pobierz jego wartość zysku
                        w przeciwnym razie
                                wyznacz wartość zysku
                                zapisz wartość zysku do pamięci podręcznej
                zapisz najlepszy wynik
        koniec pętli
        jeśli znaleziono lepszy wynik
                wybierz najlepszy wynik i umieść w rozwiązaniu
                zwróć rozwiązanie i wróć do początku pętli
        w przeciwnym razie opuść wszystkie pętle i zakończ algorytm
koniec pętli
```

3. Wyniki

a) bez żadnego z tych mechanizmów

	średni	najniższy	najwyższy
czas	28.828 s	15.717 s	50.584 s
wynik	27.216	26.592	28.21

b) z cache

	średni	najniższy	najwyższy
czas	30.889 s	18.873 s	47.784 s
wynik	27.297	26.571	28.881

c) z ruchami kandydackimi

	średni	najniższy	najwyższy
czas	16.878 s	8.746 s	31.092 s
wynik	27.265	26.493	28.553

d) z cache i ruchami kandydackimi

	średni	najniższy	najwyższy
czas	16.845 s	9.530 s	35.694 s
wynik	27.288	26.585	28.179