AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA AUTOMATYKI I INŻYNIERII BIOMEDYCZNEJ

Praca dyplomowa magisterska

Asocjacyjny system wydajnej automatycznej klasteryzacji i eksploracji danych

Associative system of efficient automatic clusterization and data mining

Autor: *Agata Socha*
Kierunek studiów: *Automatyka i Robotyka*
Opiekun pracy: *dr hab. Adrian Horzyk*

Kraków, 2017

Uprzedzona o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».", oświadczam, że niniejszą pracę dyplomową wykonałam osobiście i samodzielnie i że nie korzystałam ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję promotorowi pracy, Panu Doktorowi Adrianowi Horzykowi, za pomoc w tworzeniu tej pracy dyplomowej, a także za cierpliwość i poświęcony czas.

Spis treści

1. Wstęp

Od wkroczenia w erę informacji ilość danych, które są gromadzone oraz przetwarzane rośnie z roku na rok. Jest to spowodowane rozpowszechnieniem systemów informatycznych, powszechną komputeryzacją i większymi możliwościami składowania danych w bazach, hurtowniach danych czy repozytoriach. Jednak w większości przypadków ilość informacji, jakie są gromadzone, znacznie przewyższa możliwości ich analizy. Firmy, przedsiębiorstwa czy organizacje stanęły przed problemem jak efektywnie przetwarzać, analizować i wyciągać wnioski ze zgromadzonych danych, aby czerpać z nich korzyści dla swoich działalności. To było impulsem do rozwoju technologii eksploracji danych, która umożliwia analizę i wydobycie z nich wiedzy.

Przy dużych wolumenach danych konieczne jest zastosowanie efektywnych struktur, które umożliwią szybkie wnioskowanie na temat relacji między danymi.

W niniejszej pracy zaprezentowano realizację algorytmów klasteryzacji danych przy użyciu efektywnych asocjacyjnych struktur grafowych do grupowania różnych zbiorów danych.

Motywacją do realizacji tego tematu pracy było zainteresowanie eksploracją danych a także strukturami grafowymi AGDS, które w łatwy sposób tworzą powiązania między danymi oraz szybko udostępniają te relacje.

1.1. Cel pracy

Celem pracy było zaprojektowanie i implementacja systemu automatycznej klasteryzacji i eksploracji danych w oparciu o asocjacyjny grafowy model danych wykorzystując relacje pomiędzy nimi, które on automatycznie udostępnia.

W zakres pracy wchodzi stworzenie panelu użytkownika, który pozwoli na wybór działania i prezentację danych oraz wizualizacja przebiegu procesów w grafie. Aby pokazać przewagę struktur grafowych nad strukturami tabelarycznymi pokazano także porównanie złożoności obliczeniowej i czasowej algorytmów zrealizowanych przy ich użyciu.

1.2. Zawartość pracy

Rozdział 2. jest rozdziałem wprowadzającym w tematykę niniejszej pracy magisterskiej - sztucznej inteligencji, eksploracji danych oraz struktur grafowych AGDS.

W rozdziale 3. opisano teoretyczne aspekty algorytmów klasteryzacji, które zostały zaimplementowane w pracy, sposób ich realizacji, a także opis zastosowanych struktur danych i wizualizacje.

Rozdział 4. zawiera eksperymenty przy użyciu zaimplementowanych metod oraz porównanie złożoności czasowej i obliczeniowej algorytmów. Eksperymenty powtórzono na zbiorach danych o różnej wielkości. Omówione zostały wyniki klasteryzacji oraz porównano je z rezultatami tych samych metod znajdujących się w bibliotekach języka PYTHON.

W rozdziale 5. podsumowano pracę i opisano wnioski.

2. Wprowadzenie

Niniejszy rozdział zawiera wprowadzenie do tematyki pracy - czym jest sztuczna inteligencja oraz jej zastosowanie w eksploracji danych. Przedstawiono także wstępny opis metod klasteryzacji oraz wprowadzenie do asocjacyjnych struktur grafowych AGDS.

2.1. Sztuczna Inteligencja

Najbardziej skomplikowaną siecią na świecie wciąż pozostaje ludzki mózg. Technologia w dalszym ciągu nie może z nim konkurować, mimo że dysponuje większą mocą obliczeniową. Istnieją jednak opinie, że człowiek zostanie w przyszłości zastąpiony przez roboty, a nawet, że SI będzie odpowiedzialna za zagładę ludzkości.

Sztuczna inteligencja otacza nas w życiu codziennym - jest w smartfonach, wyszukiwarkach internetowych, grach komputerowych i inteligentnych urządzeniach domowych. Coraz częściej obsługa klienta jest realizowana przy użyciu sztucznej inteligencji w postaci chatbotów. Innym przykładem są systemy rekomendujące, które zbierając dane użytkowników proponują mu najlepsze rozwiązanie. Jednym z ciekawszych zastosowań sztucznej inteligencji są autonomiczne samochody. Od kilku lat wiele firm stara się stworzyć system, który zastąpi człowieka na miejscu kierowcy. Przy pomocy czujników, kamer i radarów mogą one rozpoznać sytuację na drodze i odpowiednio zareagować, dużo szybciej niż zrobiłby to człowiek.

Przechodząc do teorii sztucznej inteligencji, pojęcie to do użycia wprowadził John McCarthy w 1956 roku na konferencji w Darthmouth Collage na temat inteligentnych maszyn. W literaturze można spotkać różne definicje SI [1]:

- *„Sztuczna inteligencja jest nauką o maszynach realizujących zadania, które wymagają inteligencji, kiedy są wykonywane przez człowieka. ”* - M. Minsky
- *„Sztuczna inteligencja obejmuje rozwiązywanie problemów sposobami wzorowanymi na naturalnych działaniach i procesach poznawczych człowieka za pomocą symulujących je programów komputerowych.”* - R.J. Schalkoff
- *„Sztuczna inteligencja stanowi dziedzinę informatyki dotyczącą metod i technik wnioskowania symbolicznego przez komputer oraz symbolicznej reprezentacji wiedzy stosowanej podczas takiego wnioskowania.”* - E. Feigenbaum

Jako zastosowania SI podaje się m.in. uczenie maszynowe, sieci neuronowe, rozpoznawanie mowy oraz eksplorację danych, na której skupia się niniejsza praca. Jest ona opisana w następnym podrozdziale.

2.2. Eksploracja danych

Eksploracja danych *(ang. data mining)*, nazywana odkrywaniem wiedzy z danych, jest jedną z najdynamiczniej rozwijających się dziedzin informatyki w ostatnich latach.

Przytaczając definicję [2], eksploracja danych *„jest procesem odkrywania znaczących nowych powiązań, wzorców i trendów przez przeszukiwanie dużych ilości danych zgromadzonych w skarbnicach danych, przy wykorzystaniu metod rozpoznawania wzorców, jak również metod statystycznych i matematycznych”*. Można spotkać także inne definicje. Jak podaje [3] *„eksploracja danych jest analizą zbiorów danych, w celu znalezienia nieoczekiwanych związków i podsumowania danych w oryginalny sposób, by były zrozumiałe i przydatne dla ich właściciela”*.

Początkowo eksplorację przeprowadzano na prostych typach danych - liczbowych czy tekstowych. Wraz z rozwojem tej technologii zaczęto analizować pliki multimedialne, mapy, sieci, a nawet struktury chemiczne, np. DNA [4].

2.2.1. Podstawowe zadania eksploracji danych

Metody eksploracji danych można podzielić na następujące klasy [5, 6, 2, 7, 4]:

– Klasyfikacja i predykcja - rozpoznawanie wzorca, który jest tworzony przez aktualne wartości parametrów wejściowych i przypisanie mu odpowiedniej etykiety klasy. Klasyfikacja dotyczy wartości dyskretnych, predykcja natomiast wartości ciągłych.

– Klastrowanie - dzielenie przestrzeni wielowymiarowej na grupy danych, które zawierają podobne do siebie elementy, gdzie podobieństwo może być mierzone przy użyciu różnych miar.

– Odkrywanie asocjacji - odkrywanie nieznanych zależności, poszukiwanie związków asocjacyjnych pomiędzy różnymi elementami bazy danych, np. *jeśli występuje X, to zachodzi też Y*. Najczęściej spotykane jest w świecie biznesu pod nazwą analizy podobieństw lub analizy koszyka sklepowego.

– Wykrywanie zmian i odchyleń - wyszukiwanie różnic między aktualnymi danymi a wartościami oczekiwanymi, np. wyszukiwanie anomalnych zachowań klientów.

– Redukcja danych - optymalne reprezentowane dużych ilości danych, np. zmniejszenie wymiaru wektorów przy niezmienionej ich liczbie. Jej celem jest zawarcie istotnych informacji w danych o mniejszej objętości.

2.2.2. Zastosowania eksploracji danych

Eksploracja danych ma zastosowanie w wielu różnych dziedzinach, m.in.:

– Medycyna: analizie można poddawać dane pacjentów chorych i zdrowych, aby stwierdzić, czy wyniki klasyfikują ich jako potencjalnie zagrożonych chorobą. Ciekawy przykład został opisany w [2], dotyczy on problemu rozpoznawania komórek krwiotwórczych szpiku kostnego na podstawie obrazu rozmazu szpiku. Jest to bardzo ważny proces na etapie diagnostyki zaawansowania białaczki.

– Marketing, handel: wiele przedsiębiorstw dysponuje zbiorami danych, które po właściwej analizie mogłyby wskazać odpowiednią dla nich ścieżkę rozwoju. Przykładowo dla branży handlowej można wyciągnąć wnioski dotyczące klientów, ich przedziału wiekowego czy płci, aby wiedzieć do jakiej grupy docelowej dostosować swoje produkty. Co w swoich koszykach mieli klienci, którzy kupili np. mleko? Co najlepiej sprzedawało się wiosną 2017 roku? Dzięki takiej wiedzy można odpowiednio tworzyć promocje i atrakcyjne oferty.

– Systemy rekomendujące: na podstawie danych użytkownika podpowiadane są najlepsze dla niego opcje, np. książki, filmy, zakupy, znajome osoby.

– Przewidywanie trzęsień ziemi: analizując i klasteryzując zaobserwowane epicentra trzęsień można zidentyfikować strefy zagrożone.

– Ubezpieczenia: badając dane klientów, np. kierowców z wysokimi stawkami odszkodowań, można zidentyfikować defraudacje.

2.3. Klasteryzacja

Klasteryzacja jest wyjątkowym rodzajem klasyfikacji [8]. Poniżej przedstawiono podział klasyfikacji na osobne problemy klasyfikacyjne (rys. 2.1).

Można wyróżnić dwa podstawowe rodzaje klasyfikacji [8, 9]:

– klasyfikacja wykluczająca się,

– klasyfikacja niewykluczająca się.

Przy klasyfikacji wykluczającej się obiekt należy tylko i wyłącznie do jednej klasy, natomiast przy niewykluczającej się obiekt może należeć do kilku klas.

Dalszy podział można przeprowadzić dla klasyfikacji wykluczającej się:

– klasyfikacja z nadzorem,

– klasyfikacja bez nadzoru.

Rys. 2.1. Podział klasyfikacji na podproblemy klasyfikacyjne.

W klasyfikacji bez nadzoru do przeprowadzenia klasyfikacji wykorzystuje się jedynie macierz odległości między rekordami. W drugim przypadku wykorzystuje się nie tylko macierz odległości, ale też etykiety kategorii.

Kolejnego podziału można dokonać dla klasyfikacji bez nadzoru:

– klasyfikacja hierarchiczna,

– klasyfikacja niehierarchiczna.

Metody hierarchiczne pozwalają wyodrębnić pełną hierarchię klas. Metody niehierarchiczne tworzą grupy ułożone niehierarchicznie, odrębne od siebie.

Klasteryzacja należy do grupy klasyfikacji bez nadzoru. Algorytm grupowania dzieli zbiór danych w grupy, przy maksymalizacji podobieństwa wewnątrz grupy i minimalizacji podobieństwa do rekordów spoza niej. Klasyfikuje się obserwacje, dane lub wektory cech w klastry, by zmniejszyć ilość danych. Klaster składa się z wielu podobnych obiektów zebranych i zgrupowanych razem. Podaje się kilka definicji klastra [8]:

– Klaster to zbiór elementów, które są podobne, a elementy spoza klastra są niepodobne.

– Klaster to agregacja punktów w przestrzeni, dla których odległość pomiędzy każdymi dwoma punktami w klastrze jest mniejsza niż odległość pomiędzy punktem w klastrze i poza nim.

– Klaster może być opisany jako połączone obszary w przestrzeni wielowymiarowej mającej stosunkowo dużą gęstość punktów, oddzielonych od innych takich regionów przez region mający małą gęstość punktów.

Dla zadania klasteryzacji można dobrać różne miary podobieństwa. W niniejszej pracy użyto miary euklidesowej:

$$d_{Euclidean}(x,y) = \sqrt{\sum_i (x_i - y_i)^2}, \tag{2.1}$$

gdzie x_1 , x_2,.., x_n i y_1, y_2,.., y_n reprezentują wartości *n* atrybutów dla dwóch wzorców.
Można wymienić także inne miary [7, 10]:

– odległość Manhattan (inaczej zwana metryką miejską):

$$d_{Manhattan}(x,y) = \sum_i |x_i - y_i|, \tag{2.2}$$

– odległość Minkowskiego - ogólny przypadek poprzednich miar dla ogólnego wykładnika q:

$$d_{Minkowski}(x,y) = \sqrt[q]{\sum_i |x_i - y_i|^q}, \tag{2.3}$$

– metryka maksimum (zwana także Czebyszewa):

$$d_{Czebyszew}(x,y) = \max_{i=1,..,n} |x_i - y_i|. \tag{2.4}$$

Przykładowe zastosowania klasteryzacji

Klasteryzacja może być wykorzystywana w wielu dziedzinach nauki. Poniżej opisano kilka przykładowych zastosowań [10]:

– Psychiatria: psychiatra Issy Pilowsky [11] przeprowadził klasteryzację na danych swoich dwustu pacjentów. Użył ich odpowiedzi na pytania zawarte w kwestionariuszu oraz informacji o ich płci, wieku i historii choroby. W wyniku otrzymał m.in. grupę pacjentów z wykrytą endogenną depresją. Innym ciekawym przykładem opisanym w [10] jest grupowanie pacjentów z zaburzeniami osobowości przy użyciu algorytmu hierarchicznego. Udało się wyodrębnić z badanej grupy pacjentów z czterema różnymi typami osobowości.

– Badanie pogody: Thomas Littmann [12] użył algorytmów klasteryzacji do grupowania dziennych wystąpień ciśnień powierzchniowych w basenie Morza Śródziemnego. Udało mu się podzielić je na 20 grup, które wyjaśniają wariancję opadów deszczu w regionach śródziemnomorskich.

– Archeologia: klasyfikacja artefaktów służy odkrywaniu ich różnorodnych użyć czy też okresu, w którym były użytkowane i przez jaką populację. Analiza skamieniałych materiałów pomaga odkrywać, jak żyły społeczności prehistoryczne.

– Bioinformatyka i genetyka: klasteryzacja danych może być użyta w celu identyfikacji grup genów z podobnymi wzorcami ekspresji. Pomaga to odpowiedzieć na pytanie, jak ekspresja genu wpływa na rozmaite choroby i które geny są odpowiedzialne za choroby dziedziczne.

– Astronomia: klasyfikacja obiektów astronomicznych często pomaga astronomom znajdować nietypowe obiekty w dużych zbiorach danych, przykładem są odkrycia kwazarów i brązowych karłów.

2.3.1. Algorytm hierarchicznej klasteryzacji

Algorytm hierarchiczny wielokrotnie dzieli przestrzeń danych przy założeniu różnej liczby klastrów. Można wyróżnić dwa podejścia [7, 8]:

– Aglomeracyjne - łączenie mniejszych klastrów w większe. Dla tego podejścia na starcie algorytmu liczba klastrów równa jest liczbie danych (rekordów). Następnie dwa klastry, które są najbliżej siebie, łączą się w jeden. Na końcu wszystkie rekordy znajdują się w jednym klastrze.

– Deaglomeracyjne - podział większych klastrów na mniejsze. Na początku wszystkie dane należą do jednego klastra i w kolejnych etapach dokonuje się ich podziału na osobne klastry.

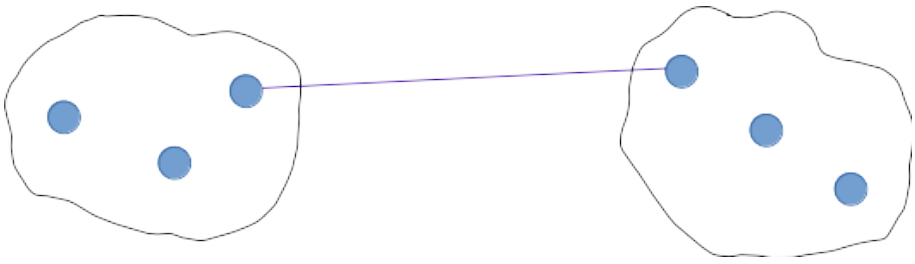
Proces podziału hierarchicznego zwykle jest przestawiany w postaci dendrogramu, grafu o strukturze drzewiastej [7].

Aby obliczyć odległość między dwoma klastrami należy zastosować jedną z poniższych metod:

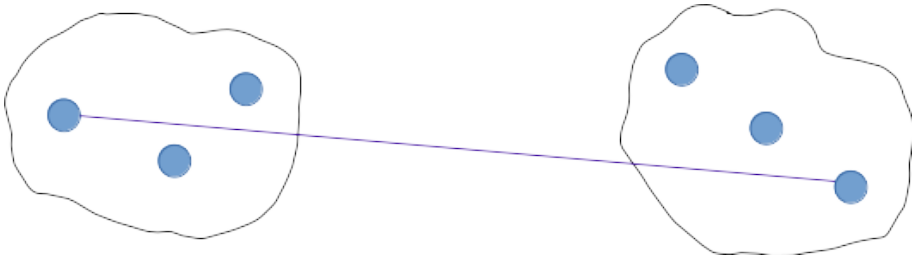
– Metoda pojedynczego połączenia: zwana metodą najbliższego sąsiedztwa. Odległość między dwoma grupami jest określona jako odległość między dwoma najbliższymi punktami z klastra A i z klastra B (rys. 2.2).

– Metoda całkowitego połączenia: zwana metodą najdalszego sąsiedztwa, oparta na maksymalnej odległości między dowolnym rekordem z klastra A i dowolnym rekordem z klastra B (rys. 2.3).

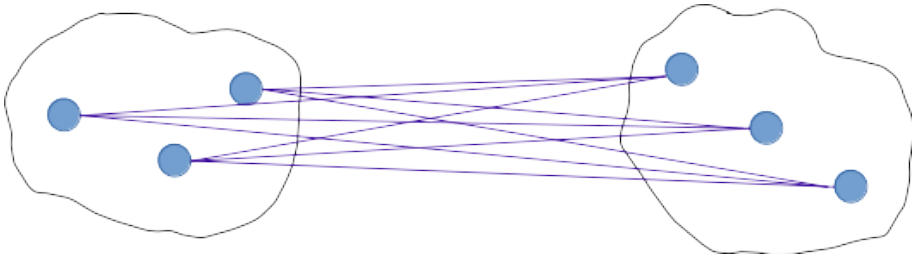
– Metoda średniego połączenia: odległość między dwoma klastrami jest obliczana jako średnia odległość wszystkich rekordów z klastra A do wszystkich rekordów z klastra B (rys. 2.4).



Rys. 2.2. Metoda pojedynczego połączenia.



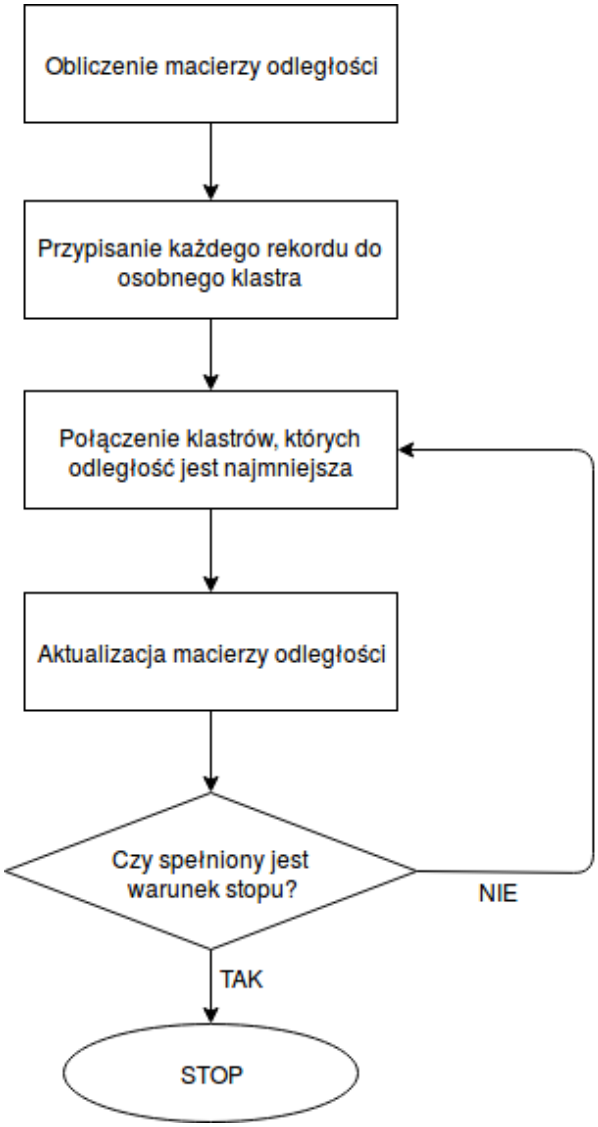
Rys. 2.3. Metoda całkowitego połączenia.



Rys. 2.4. Metoda średniego połączenia.

Algorytm aglomeracyjny z metodą pojedynczego połączenia postępuje w następujący sposób:

1. Liczba klastrów równa liczbie danych.
2. Określenie wartości miary podobieństwa pomiędzy wszystkimi rekordami.
3. Połączenie ze sobą klastrów o najmniejszej odległości.
4. Uaktualnienie macierzy odległości między poszczególnymi klastrami.
5. Powtarzanie powyższych kroków do uzyskania jednego klastra.



Rys. 2.5. Schemat blokowy algorytmu klasteryzacji hierarchicznej z metodą pojedynczego połączenia.

2.3.2. Algorytm klasteryzacji niehierarchicznej

Przykładowe algorytmy niehierarchiczne to: [7, 8, 2]:

– Metody podziału (ang. *partitioning methods*): dzielą przestrzeń na k klastrów, środkiem każdego klastra jest *centroid*. Każdy punkt jest przypisany do klastra, którego centroid jest dla niego najbliższy. Wyróżnia się:

  – algorytm k-średnich,

  – algorytm k-medoidów.

– Metody rozmytej analizy skupień (ang. *fuzzy clustering*) - przy podziale przestrzeni danych na klastry korzysta z macierzy odległości pomiędzy rekordami oraz z tzw. miary przynależności rozmy-

tej. Wartość współczynnika przynależności należy do przedziału $[0, 1]$, gdy dla systemów klasycznych wartość ta wynosiła 1 przy przynależności i 0 przy braku przynależności. W podejściu rozmytym rekord należy do wszystkich klastrów z różnym współczynnikiem przynależności.

Wyróżnia się algorytmy:

- algorytm grupowania górskiego,
- algorytm c-means,
- algorytm Gustafsona-Kessela.

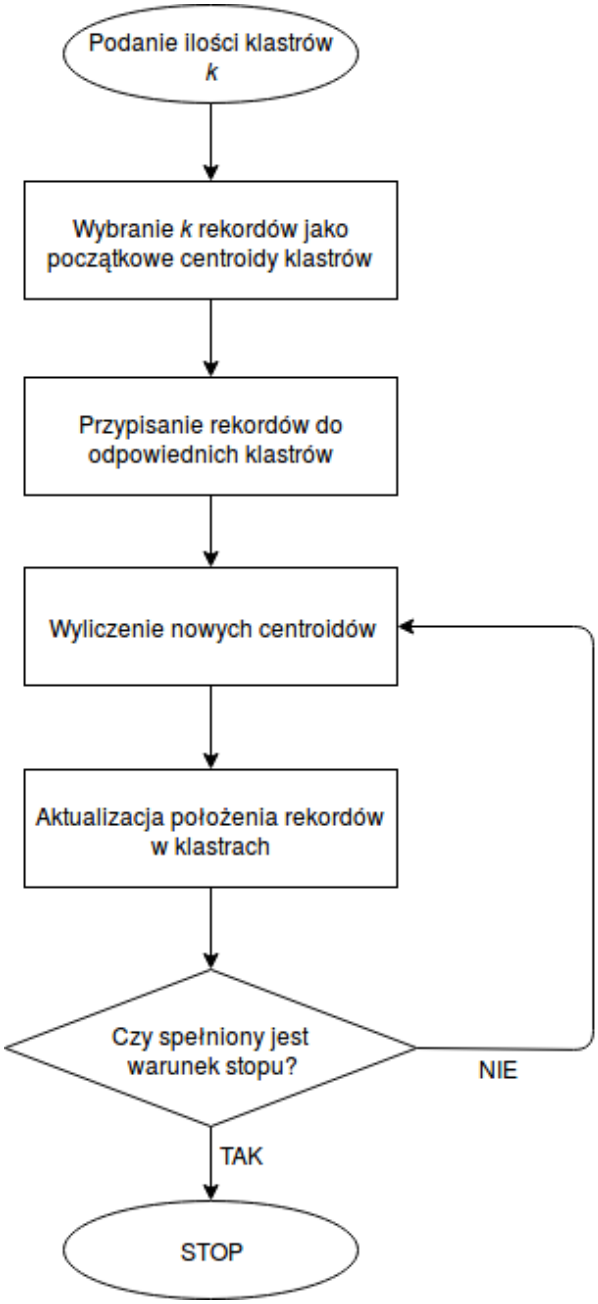
W niniejszej pracy zaimplementowano metodę k-średnich i zostanie ona szerzej omówiona.

2.3.2.1. Metoda k-średnich

Metoda k-średnich to prosty i efektywny algorytm, jedna z wersji algorytmu Lindego-Buzo-Graya [2]. Algorytm postępuje w następujący sposób:

Dany jest zbiór *n* obserwacji opisanych wektorami x_1, x_2, ... , x_n każdy o rozmiarze *N*. Algorytm k-średnich dzieli te *n* obserwacje na *k* klastrów S_1, S_2, ..., S_k przy *k* < *n*.

1. Określenie ilości klastrów *k*.
2. Losowe przypisanie *k* rekordów jako początkowe środki klastrów S_1, S_2, ..., S_k.
3. Dla każdego wektora x_1, x_2, ... , x_n znaleźć najbliższy środek grupy c_i.
4. Obliczenie sumarycznego błędu kwadratowego - sumy kwadratów odległości między centroidem c_i a wektorami x_j.
5. Zmiana centroidów - nowa wartość jest średnią wszystkich elementów należących do klastra.
6. Powtarzanie kroków od 3 do 5 aż do zbieżności lub zakończenia.



Rys. 2.6. Schemat blokowy algorytmu k-średnich.

2.4. Grafowa asocjacyjna struktura danych AGDS

Struktury danych są bardzo istotnym elementem przy implementacji algorytmu, ponieważ znacząco wpływają na szybkość dostępu do danych i wykonywania wszystkich operacji oraz mogą ułatwiać odczytywanie relacji pomiędzy danymi. Odpowiedni dobór struktury może także zmniejszyć złożoność obliczeniową algorytmu. Dane mogą być przechowywane w niezmienionej formie w strukturze pasywnej, wtedy nie mają na siebie żadnego wpływu. Przy operacji wykonywanej na jednej komórce pamięci wartości pozostałych komórek nie zmieniają się [13]. Komórki pamięci są więc niezależne. Ma to miej-

sce w klasycznych tabelarycznych strukturach danych. Innym sposobem przechowywania danych jest struktura aktywna – skojarzeniowa. Jej działanie jest podobne do działania ludzkiego mózgu. Przy zapisywaniu w nich danych bądź operacjach na nich następuje pobudzenie aktywności i reakcji skojarzeniowych innych powiązanych rekordów.

Grafowa asocjacyjna struktura danych AGDS (ang. *associative graph data structure*) jest strukturą dynamiczną i pasywną. Jest to graf pozwalający na przechowywanie wartości danych i ich kombinacji oraz relacji je łączących [13]. Struktura AGDS skutecznie zastępuje mało efektywne struktury tabelaryczne. Dane i kombinacje w strukturze AGDS są reprezentowane w postaci węzłów oraz krawędzi odwzorowujących zależności pomiędzy nimi. Struktura grafowa oparta na drzewie pozwala uzyskać bardzo szybki dostęp do dowolnych danych oraz reprezentowanych relacji pomiędzy tymi danymi. Struktury te są oszczędne, gdyż nie przechowują duplikatów danych ani duplikatów ich kombinacji.

AGDS jest opisane jako:

$$AGDS = (VV, VR, VS, VC, ESIM, ESEQ, EDEF), \tag{2.5}$$

gdzie VV, VR, VS, VC są zbiorami wierzchołków, a ESIM, ESEQ, EDEF są zbiorami krawędzi o następującym znaczeniu:

– VV - zbiór wierzchołków reprezentujących pojedynczą wartość,

– VR - zbiór wierzchołków reprezentujących przedział wartości,

– VS - zbiór wierzchołków reprezentujących podzbiór wartości,

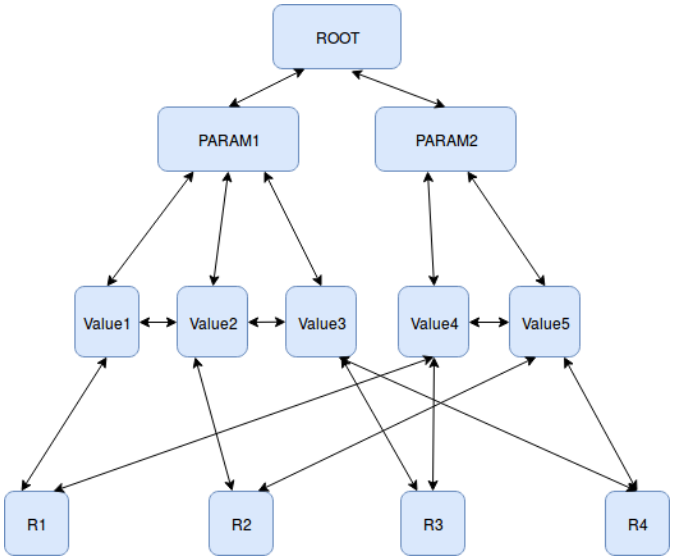
– VC - zbiór wierzchołków reprezentujących kombinację wartości,

– ESIM - zbiór krawędzi nieskierowanych łączących asocjacyjnie następne wierzchołki,

– ESEQ - zbiór krawędzi skierowanych łączących asocjacyjnie następne wierzchołki,

– EDEF - zbiór krawędzi dwustronnie skierowanych łączących wierzchołek definiujący z wierzchołkiem definiowanym w taki sposób że etykieta (waga) określa przejście od jednego do drugiego wierzchołka.

Rysunek 2.7 przedstawia grafową strukturę AGDS, natomiast tabela 2.1 przedstawia jej tabelaryczny odpowiednik.



Rys. 2.7. Struktura grafowa AGDS.

Rekord	PARAM1	PARAM2
R1	Value1	Value4
R2	Value2	Value5
R3	Value3	Value4
R4	Value3	Value5

Tabela 2.1. Struktura tabelaryczna dla danych przedstawionych na rys. 2.7.

Struktury AGDS zapewniają:

– bezstratną kompresję danych poprzez eliminację duplikatów,

– szybki dostęp do danych,

– szybkie wyszukiwanie powiązanych danych i obiektów,

– redukcję złożoności obliczeniowej,

– przechowywanie posortowanych wartości dla wszystkich atrybutów.

Asocjacyjne struktury danych są w stanie zastąpić mało efektywne struktury tabelaryczne oraz część operacji na danych związanych z przeszukiwaniem tabel, jak np. wyszukiwanie obiektów podobnych, duplikatów, przeszukiwanie klas względem wybranych wartości cech, porządkowanie obiektów według wybranych grup atrybutów [13]. Implementacja tych struktur danych opisana została w podrozdziale 3.1.

2.5. Złożoność obliczeniowa algorytmu

Dobór odpowiedniego algorytmu dla danego problemu jest ważnym zadaniem. Programista powinien odpowiedzieć sobie na pytanie, czy jego rozwiązanie powinno być łatwe i szybkie do zrozumienia dla innych programistów, proste do zaimplementowania i weryfikacji działania, czy ważniejszym aspektem jest efektywne wykorzystanie zasobów komputera i szybkość działania [14].

Złożoność algorytmu może być rozpatrywana pod kątem czasu bądź zasobów, jakie potrzebne są do jego wykonania [15].

Parametrem, który najczęściej decyduje o czasie wykonania algorytmu, jest rozmiar danych wejściowych. Czas potrzebny na wykonanie programu przy danych wejściowych o rozmiarze *n* przyjęto oznaczać *T(n)* [14]. Sposobem wyrażania funkcyjnej złożoności czasowej jest notacja „dużego O". Wyróżnia się notacje takie jak:

- *O(1)* - stała złożoność: stała ilość operacji, niezależna od danych wejściowych,
- *O(n)* - złożoność liniowa: ilość operacji wprost proporcjonalna do ilości danych wejściowych,
- $O(n^x)$ - złożoność wielomianowa: ilość operacji wprost proporcjonalna do ilości danych wejściowych podniesionych do potęgi *x*,
- $O(log_2 n)$ - złożoność logarytmiczna: logarytmiczna ilość operacji w stosunku do danych wejściowych.

Przy obliczaniu złożoności rozpatrywane mogą być trzy przypadki [14]:

- pesymistyczny - maksymalna złożoność dla danych o rozmiarze *n*,
- średni - uśredniony czas wykonywania dla danych o rozmiarze *n*,
- optymistyczny - minimalna złożoność dla danych o rozmiarze *n*.

W niniejszej pracy rozpatrywany jest przypadek uśredniony.

3. Opis stworzonego rozwiązania

Niniejszy rozdział zawiera opis sposobu implementacji algorytmów klasteryzacji na grafowych strukturach AGDS. Został również zaprezentowany panel użytkownika oraz sposób prezentacji danych.

Implementacja struktur danych i algorytmów została wykonana w języku C++. Aby umożliwić przyjmowanie przez parametry wartości typu całkowitoliczbowego, zmiennoprzecinkowego czy tekstowego użyto biblioteki BOOST::VARIANT [16]. GUI zostało zaimplementowane przy użyciu programu QTCREATOR. Wizualizację grafów wykonano z wykorzystaniem biblioteki GRAPHVIZ [17] i języka PYTHON.

Przy implementacji przyjęto założenie, że wczytywane dane będą w pełni poprawne i kompletne. Format pliku wejściowego to plik tekstowy lub o rozszerzeniu *CSV, (comma-separated values)*, przy czym dane oddzielone są od siebie znakiem tabulacji.

Algorytm został napisany w sposób umożliwiający wczytywanie danych z dowolną liczbą parametrów i różnej wielkości.

3.1. Struktury danych

Wstęp teoretyczny do asocjacyjnych grafowych struktur danych został zaprezentowany w podrozdziale 2.4, a wygląd grafu przedstawiono na rysunku 2.7. Struktura grafu składa się z takich elementów jak:

- *Korzeń (`ROOT`)*: klasa zawierająca w sobie wskaźniki do wszystkich parametrów, jakie ma dany obiekt. Pozwala na przechodzenie pomiędzy parametrami iteracyjnie po tych wskaźnikach.

- *Parametr (`PARAMETER`)*: klasa posiadająca wskaźnik do obiektu klasy *korzeń* (jeden poziom wyżej w hierarchii) oraz do listy wskaźników do wartości `VALUE` jakie przyjmuje (poziom niżej w hierarchii).

- *Wartość (`VALUE`)*: klasa przechowująca m.in. wskaźnik do elementu typu liczbowego lub łańcucha znaków. Posiada wskaźnik do parametru, do którego przynależy oraz do rekordów, które mają tę wartość dla owego parametru.

- *Rekord (`RECORD`)*: klasa posiadająca wskaźniki do wartości, które go definiują.

3.2. Realizacja podstawowych operacji na strukturach AGDS

Budowa grafu umożliwia bardzo szybkie wykonywanie podstawowych obliczeń takich jak szukanie minimum czy maksimum danego parametru. Listy wartości są posortowane, dlatego wystarczy pobrać odpowiednio pierwszy lub ostatni element listy. Schemat grafu z zaznaczonymi elementami minimalnymi i maksymalnymi przedstawiony jest na rysunku 3.1. Wizualizację wykonano na zbiorze *Irysy* [20] zawierającym 10 elementów (tabela 3.1). Parametry rekordów to długość i szerokość płatka kwiatu *(petal-length/width)*, przyjęto oznaczenie PLE i PWI, oraz długość i szerokość liścia *(leaf-length/width)*, przyjęto oznaczenie SLE i SWI, a także klasa (class).

Rekord	leaf-length	leaf-width	petal-length	petal-width	class
R1	5.1	3.5	1.4	0.2	Setosa
R2	5.8	2.7	5.1	1.9	Virginica
R3	7.1	3	5.9	2.1	Virginica
R4	5.5	2.3	4	1.3	Versicolor
R5	6.3	2.9	5.6	1.8	Virginica
R6	4.9	3	1.4	0.2	Setosa
R7	4.7	3.2	1.3	0.2	Setosa
R8	5.2	3.5	1.5	0.2	Setosa
R9	6.9	3.1	4.9	1.5	Virginica
R10	6.5	2.8	4.6	1.5	Virginica

Tabela 3.1. Wybrane rekordy ze zbioru *Irysów*.

Wagi połączeń w strukturach AGDS

Połączenia w grafie AGDS posiadają przypisane wagi [18]. Węzły grafu, reprezentujące sąsiednie wartości, połączone są krawędzią o wadze wyliczanej w oparciu o wzór 3.1.

$$w_{ab}^{P_i} = 1 - \frac{|a-b|}{MAX_{P_i} - MIN_{P_i}}, \tag{3.1}$$

gdzie a, b to elementy z listy wartości danego atrybutu, a $MAX_{P_i} - MIN_{P_i}$ to zakres wartości tego atrybutu.

Wartość wagi łączącej węzeł wartości danego atrybutu z węzłami obiektów R_m obliczana jest na podstawie ilości wystąpień tej wartości we wszystkich obiektach (wzór 3.2). Wartość ta jest przechowywana w węzłach wartości atrybutów. W podstawowej wersji implementacji struktury AGDS wartości wag pomiędzy węzłami wartości i węzłami obiektów przyjmują wartość $w_{a,P_i} = 1$.

$$w_{a,P_i} = \frac{1}{N_i}, \tag{3.2}$$

gdzie N_i to ilość wystąpień danej wartości.

Wagi pomiędzy węzłami obiektów a węzłami wartości są zawsze równe 1 (wzór 3.3):

$$w_{P_i,a} = 1. \tag{3.3}$$

Przy obliczaniu podobieństwa obiektów w węzłach R_i obliczana jest suma ważonych wejść przekazanych przez inne węzły. Proces ten można opisać w kilku krokach [19]:

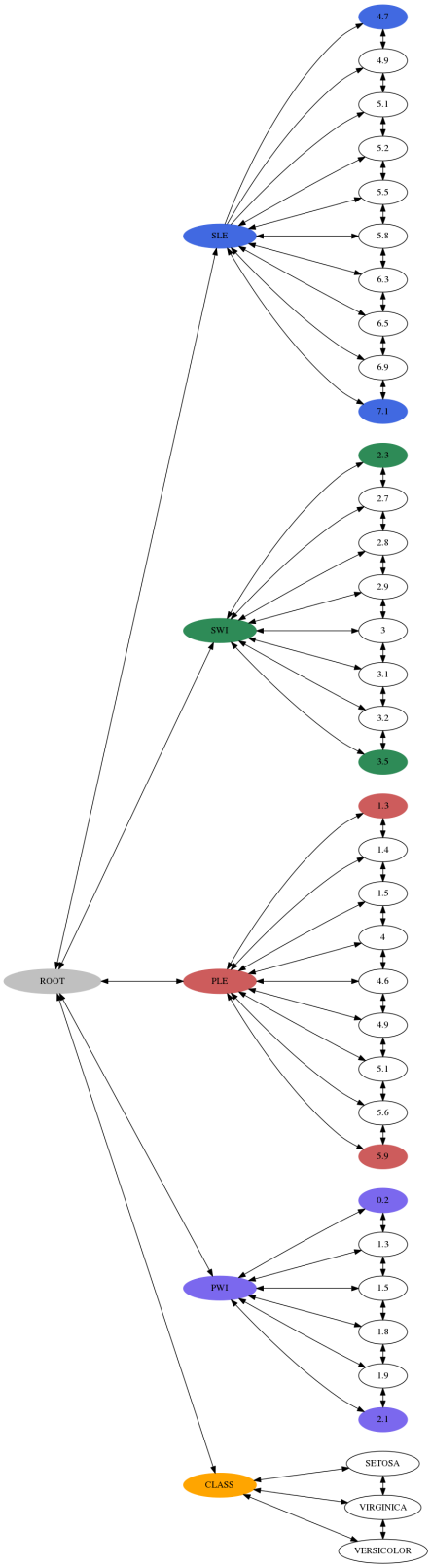
1. Rekord R_n wysyła żądanie wyznaczenia podobieństw do węzłów definiujących jego wartości.
2. Do wszystkich węzłów R_i, z którymi połączone są te wartości, dodawana jest wartość 1.
3. Do węzłów sąsiednich wartości dodawana jest wartość obliczona ze wzoru 3.1. Następnie węzły sąsiednich wartości dodają tę wartość, pomnożoną przez wagę połączenia (wzór 3.2), do wszystkich połączonych z nimi węzłów R_i.

Działanie jest powtarzane tylko dla wartości w najbliższym sąsiedztwie lub do osiągnięcia minimum bądź maksimum danego atrybutu.

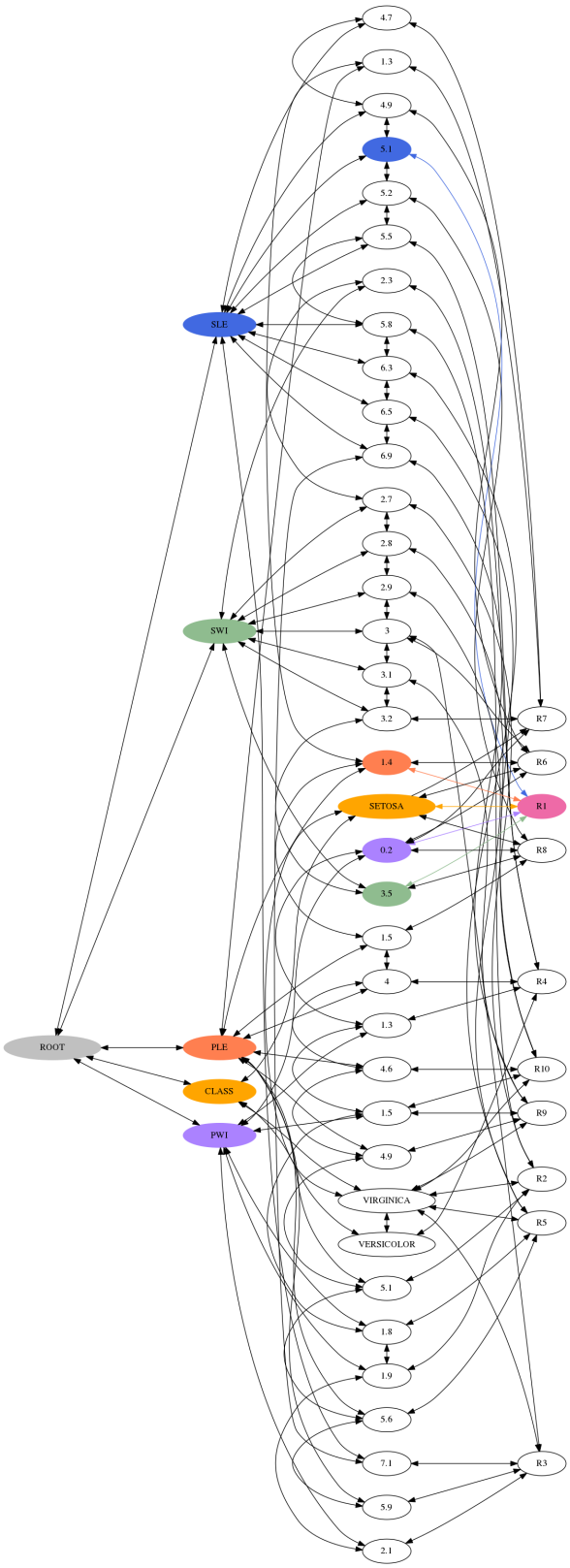
Proces poszukiwania rekordów podobnych przeprowadzono na zbiorze *Irysy*. Szukano rekordów podobnych do pierwszego rekordu – R1. Na rysunku 3.2 wartości rekordu R1 zaznaczono kolorami. Rekordy posiadające te same wartości co rekord R1 zaznaczono jasnoróżowym kolorem na rysunku 3.3. W następnych krokach sprawdzone zostały wartości najbliższe wartościom R1, zaznaczono je kolorami na rysunku 3.4. W tym przykładzie sprawdzono najbliższe sąsiedztwo wartości rekordu R1. Jaśniejsze kolory przedstawione na wykresach przedstawiają mniejsze podobieństwo rekordów do R1.

Można odczytać, że w kolejności najbardziej podobne do rekordu R1 są: R6, R7, R8 R10, R9, R4, R3, R2 i R5. Tabela 3.1 pokazuje, że najbardziej podobne rekordy należą do tej samej klasy Irysów.

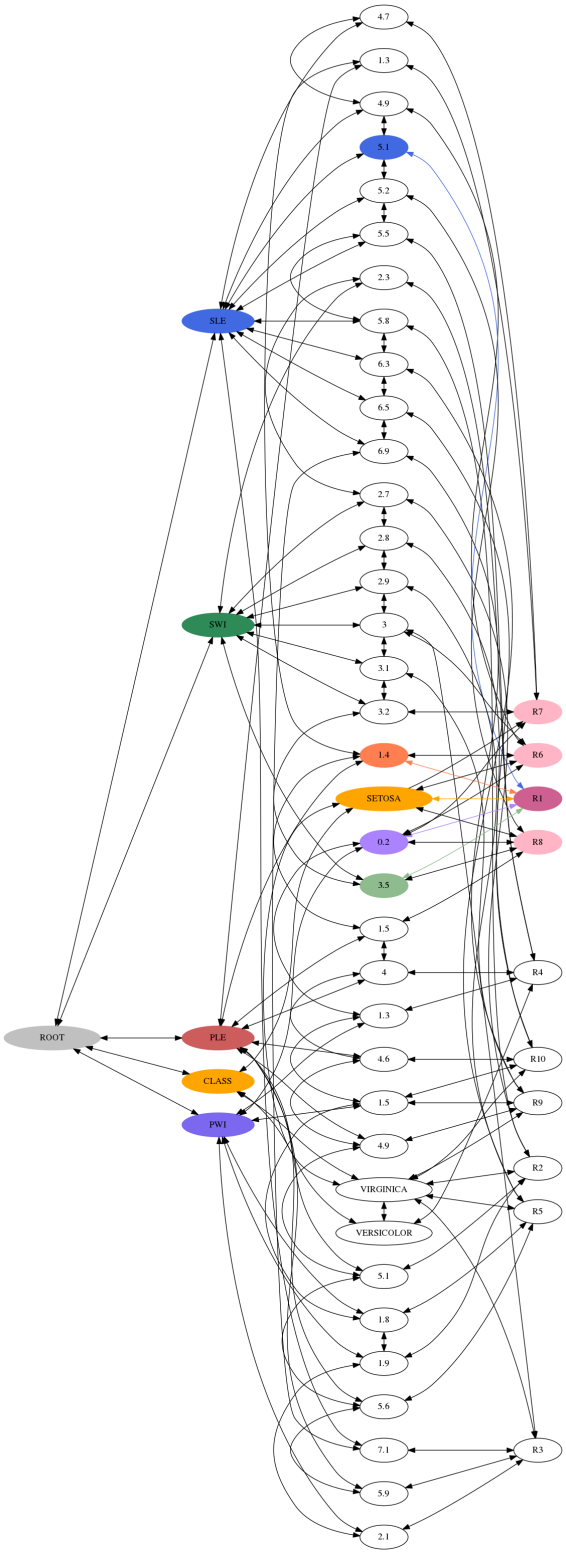
Wyszukanie rekordów podobnych do grupy jest analogiczne jak poszukiwanie podobnych do jednego rekordu, jedynie powtórzone dla każdego z rekordów należących do grupy i sumowaniu wartości podobieństwa.



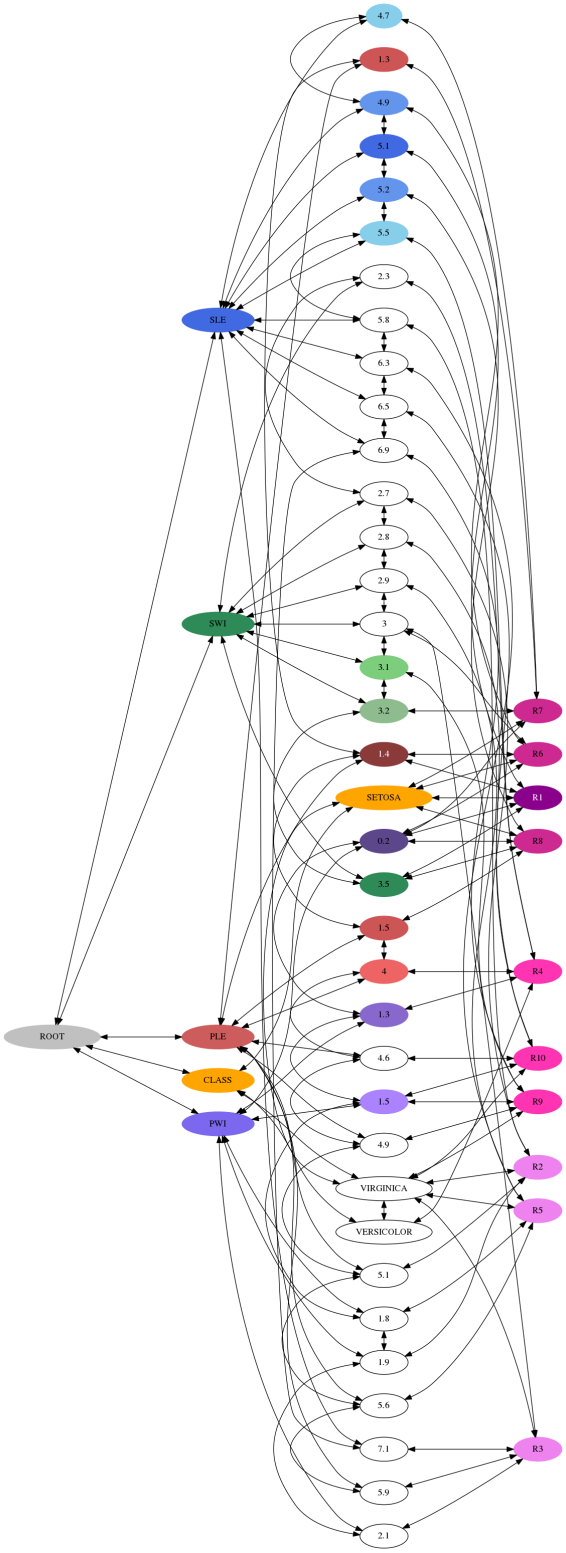
Rys. 3.1. Struktura grafowa AGDS z zaznaczonymi wartościami minimalnymi i maksymalnymi parametrów.



Rys. 3.2. Graf AGDS z wartościami rekordu R1 zaznaczonymi kolorem.



Rys. 3.3. Graf AGDS z zaznaczonymi wartościami rekordu R1 oraz rekordami mającymi te same wartości co R1.



Rys. 3.4. Graf AGDS z pobudzanymi rekordami podobnymi do R1.

3.3. Realizacja algorytmu klasteryzacji hierarchicznej

Teoria algorytmu hierarchicznej klasteryzacji została opisana w podrozdziale 2.3.1. Zaimplementowany został algorytm z metodą pojedynczego połączenia (rys. 2.2).

Pierwszym krokiem implementacji jest wyliczenie macierzy odległości pomiędzy wszystkimi rekordami. Zostało to zrealizowane poprzez utworzenie klasy *Distance*, która przechowuje wskaźniki do rekordu pierwszego i drugiego oraz wartości odległości pomiędzy nimi według miary euklidesowej (wzór 2.1). Struktura AGDS umożliwia bardzo prosty i szybki dostęp do wszystkich wartości rekordów poprzez łączące je powiązania. Następnie wszystkie obiekty klasy *Distance*, umieszczone w kontenerze typu lista, zostały posortowane według odległości – od najmniejszej do największej. Wskaźniki na elementy typu *Rekord* zostały umieszczone w kontenerze *map*. Do każdego rekordu przypisany jest parametr informujący, w którym klastrze się znajduje. Algorytm przechodząc po kolei przez listę odległości ustawia dla rekordów z najmniejszą odległością ten sam numer klastra. Działanie to jest powtarzane do spełnienia kryterium stopu. Jako wynik zwracany jest kontener *map*, mówiący o przynależności poszczególnych rekordów do klastrów.

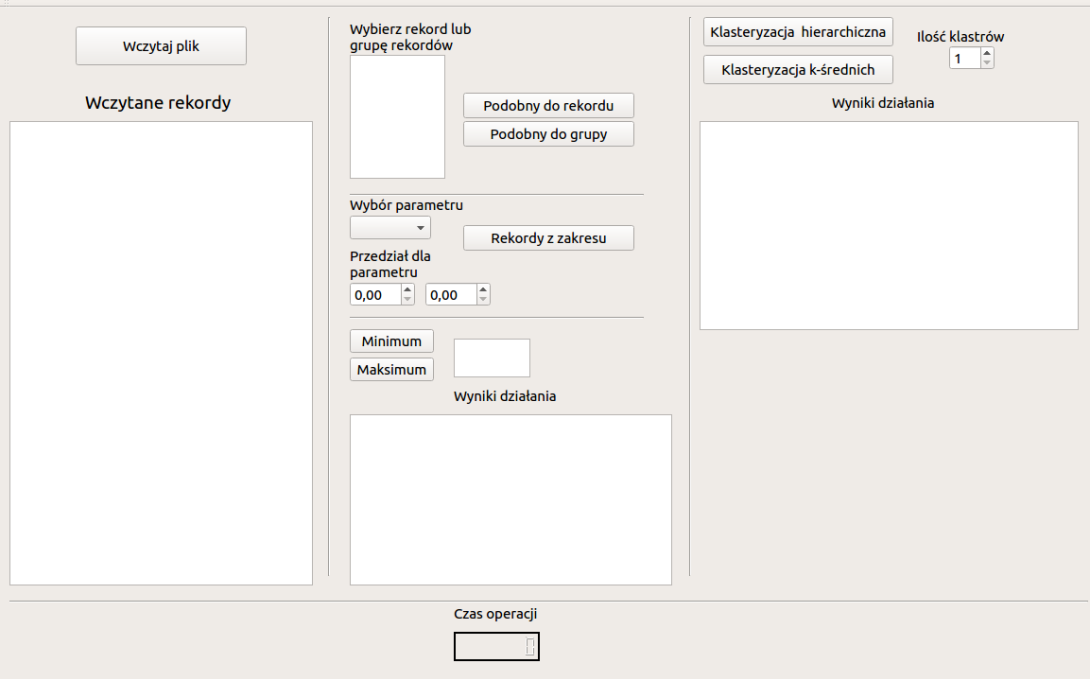
3.4. Realizacja algorytmu k-średnich

Teoretyczny opis tego algorytmu został ujęty w podrozdziale 2.3.2.1.

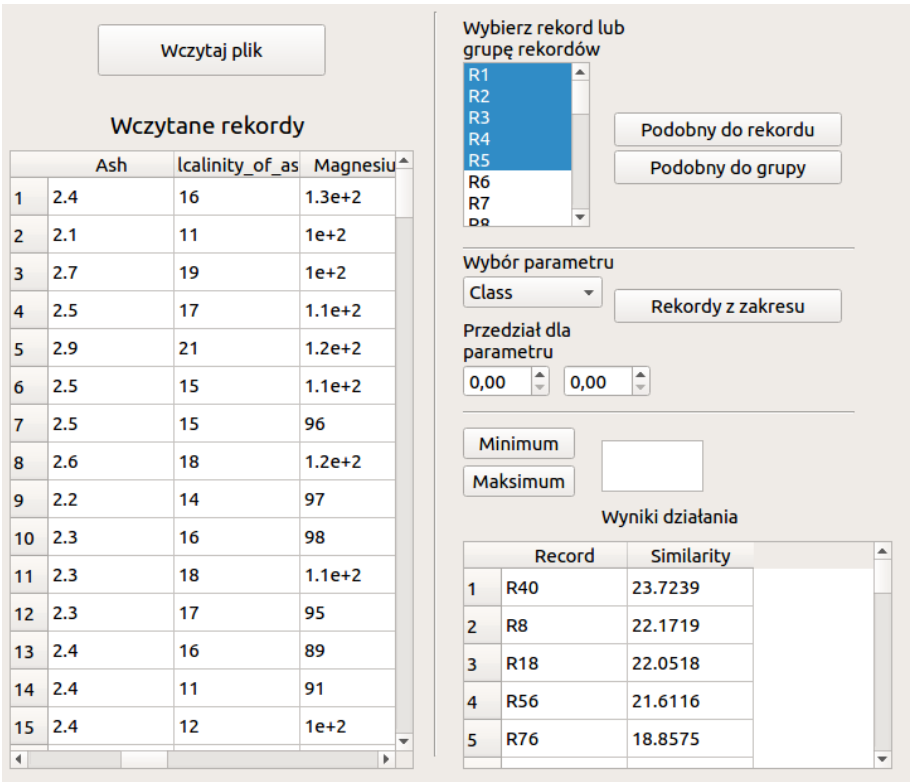
W pierwszym kroku wybrano losowo k centroidów spośród wszystkich rekordów. Centroidy te zostały dodane jako pierwsze elementy osobnych klastrów. Klastry są elementami typu *list* przechowywanymi w elemencie typu *vector*. W następnych krokach rekordy zostają dodane do odpowiednich list – klastrów, dla których wyliczane są nowe centroidy. Po tym kroku następuje przesuwanie rekordów pomiędzy listami dopóki centroidy będą zmieniać swoje wartości.

3.5. Panel użytkownika i prezentacja danych

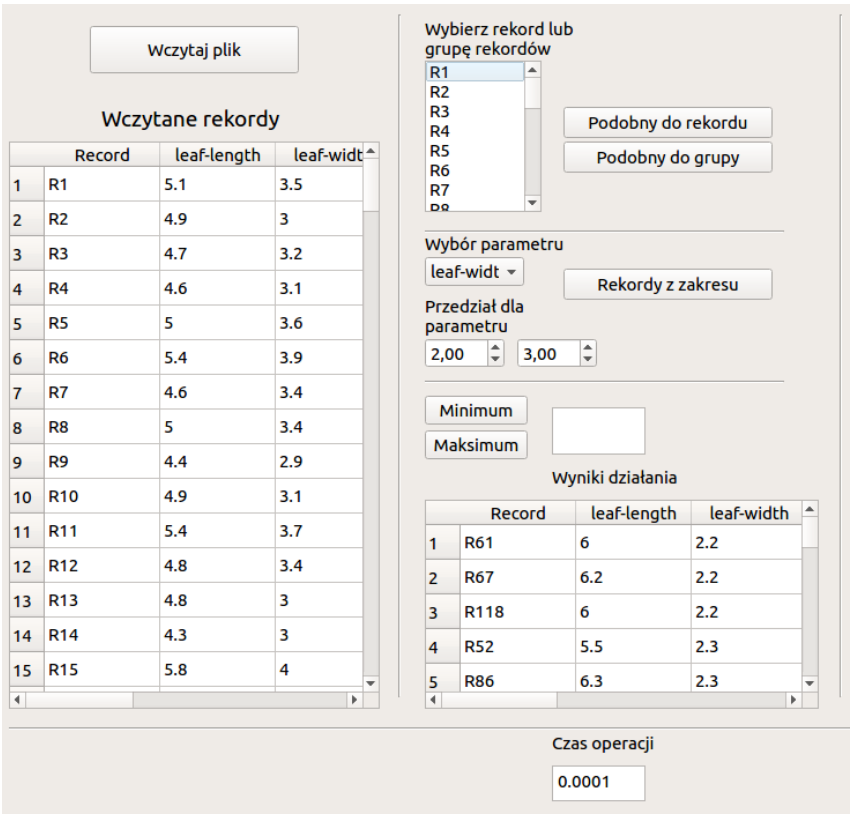
Panel użytkownika został stworzony przy użyciu programy QT CREATOR. Na rysunku 3.5 przedstawiono wygląd po uruchomieniu, rysunku 3.6, 3.7, 3.8 przedstawiają wygląd panelu po wczytaniu danych i wykonaniu operacji.



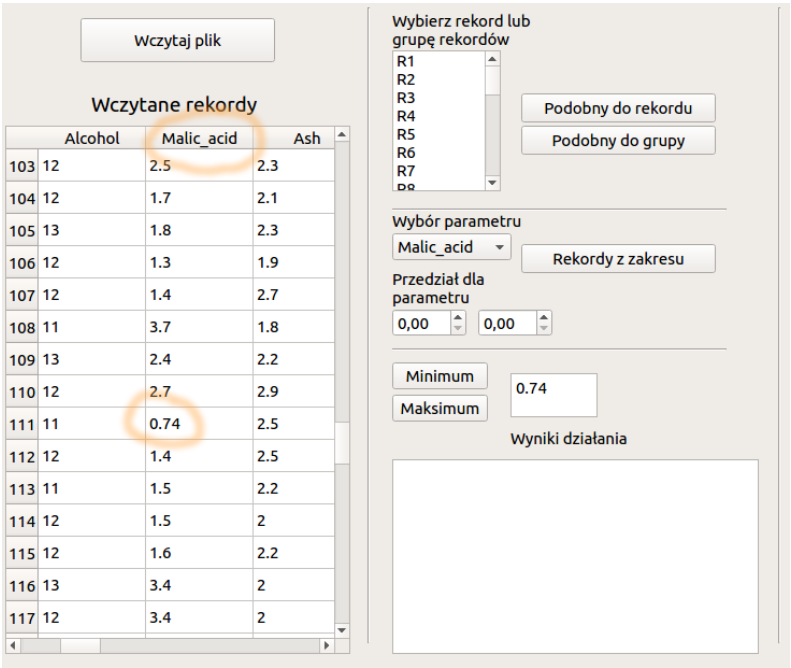
Rys. 3.5. Panel użytkownika.



Rys. 3.6. Panel użytkownika z wykonaną operacją wyszukiwania rekordów podobnych do grupy.

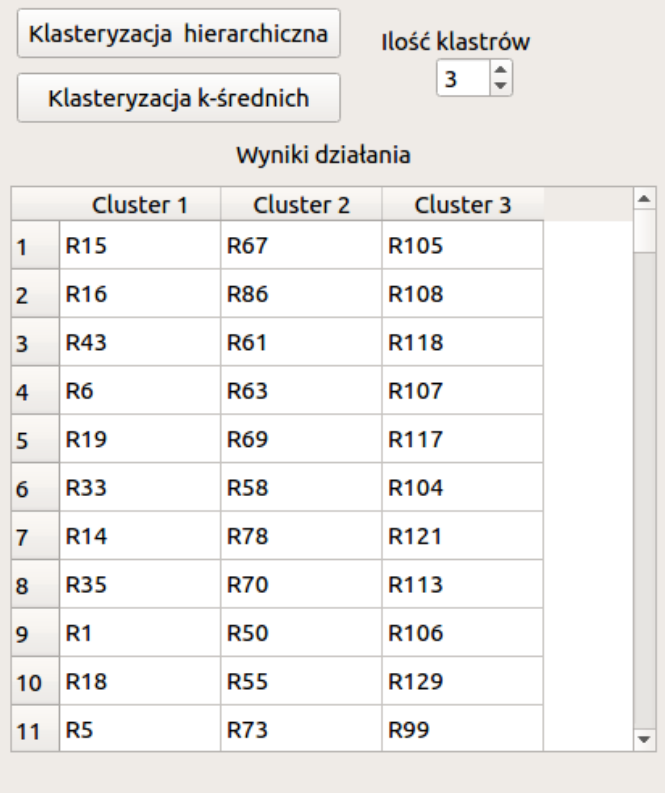


Rys. 3.7. Panel użytkownika z wykonaną operacją wyszukiwania rekordów o parametrze *leaf-width* w zakresie (2,3).



Rys. 3.8. Panel użytkownika z operacją wyszukiwania minimum parametru *Malic*.

Wyniki klasteryzacji przedstawiane są w ostatniej kolumnie panelu użytkownika. Rysunek 3.9 przedstawia sposób prezentacji wyodrębnionych klastrów.



	Cluster 1	Cluster 2	Cluster 3
1	R15	R67	R105
2	R16	R86	R108
3	R43	R61	R118
4	R6	R63	R107
5	R19	R69	R117
6	R33	R58	R104
7	R14	R78	R121
8	R35	R70	R113
9	R1	R50	R106
10	R18	R55	R129
11	R5	R73	R99

Rys. 3.9. Klasteryzacja hierarchiczna przeprowadzona na zbiorze Irysów.

4. Eksperymenty

W niniejszym rozdziale zaprezentowano wyniki podstawowych operacji eksploracji danych porównując ich działanie na strukturach tabelarycznych i grafowych AGDS. Przedstawiono także efekty uzyskane z algorytmów klasteryzacji dla małych i dużych zbiorów danych. Do eksperymentów zawartych w tym rozdziale użyto kilku zbiorów. Zostały opisane przy eksperymentach, do których ich użyto. Dobierano je pod kątem ilości danych i liczby parametrów.

4.1. Porównanie działań na strukturach tabelarycznych i AGDS

Do eksperymentów zawartych w tym podrozdziale użyto zbioru danych *Bike*, który pochodzi z [20]. Składa się on z 5 tysięcy rekordów i 5 parametrów. Podzielono go na części, aby sprawdzać czasowe wyniki dla mniejszych zbiorów, tj. od 500 do 5 tysięcy.

4.1.1. Minimalna i maksymalna wartość wybranego parametru

Czasy w poniższej tabeli są wynikiem najmniejszym uzyskanym w 5 próbach.

Działanie	**Rozmiar danych wejściowych**	**Czas działania na strukturach tabelarycznych [s]**	**Czas działania na struktur AGDS [s]**
Wyszukiwanie minimum /maksimum dla parametru	500	6.6e-05	7e-6
Wyszukiwanie minimum /maksimum dla parametru	3000	0.000473	7e-6
Wyszukiwanie minimum /maksimum dla parametru	5000	0.000748	8e-6

Tabela 4.1. Porównanie czasów operacji na tabelach i strukturach AGDS - wartość minimalna i maksymalna parametru.

Porównując czasy wyszukiwania minimum i maksimum widoczne jest, że w przypadki struktur AGDS niezależnie od wielkości danych czas jest stały i wynosi w przybliżeniu 8e-6. Jest to spowo-

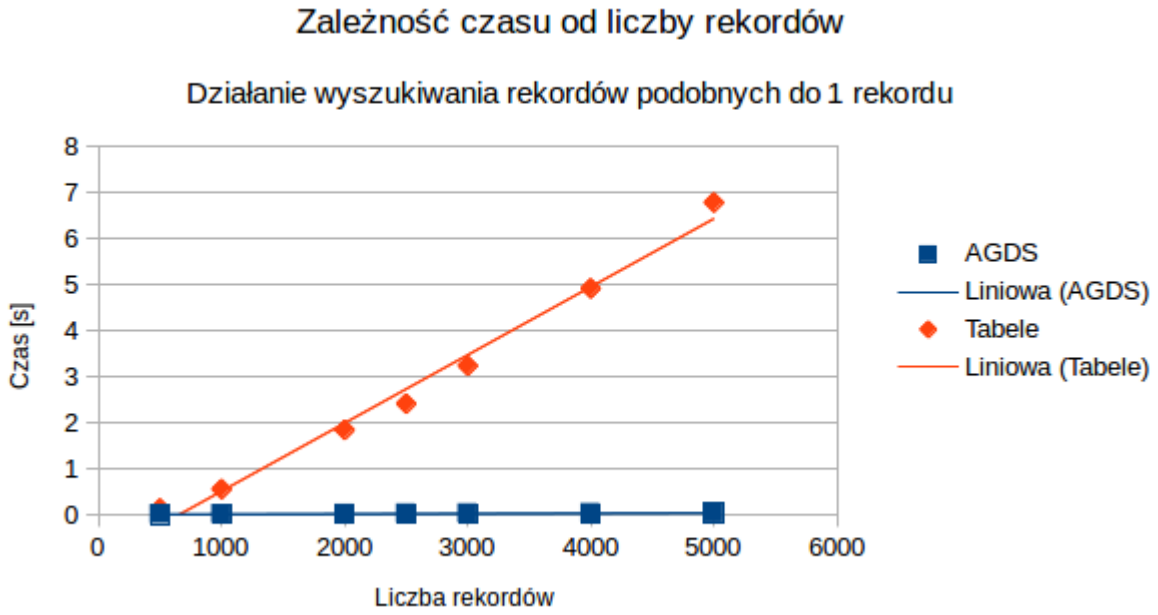
dowane tym, że parametr ma zawartą informację o swojej minimalnej i maksymalnej wartości. W przypadku tabel wartość minimalną i maksymalną należy wyszukać, nie jest ona nigdzie zachowana.

4.1.2. Poszukiwanie rekordów podobnych do wybranego rekordu

Procedura wyszukiwania rekordów podobnych opisana została w podrozdziale 3.2.

Rozmiar danych wejściowych	**Czas działania na strukturach tabelarycznych [s]**	**Czas działania na struktur AGDS [s]**
500	0.130546	0.004282
1000	0.5518	0.0079
2000	1.84039	0.01
2500	2.4102	0.01371
3000	3.23475	0.023135
4000	4.9195	0.026746
5000	6.78749	0.030543

Tabela 4.2. Porównanie czasów operacji na tabelach i strukturach AGDS - lista rekordów podobnych do wybranego.

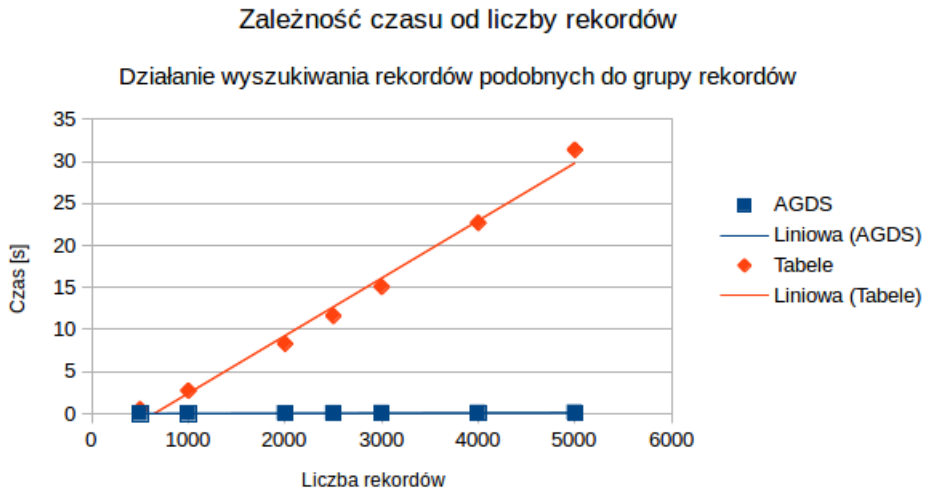


Rys. 4.1. Wykres zależności czasu od wielkości zbioru danych dla poszukiwania rekordów podobnych do wybranego rekordu.

4.1.3. Poszukiwanie rekordów podobnych do grupy rekordów

Rozmiar danych wejściowych	Czas działania na strukturach tabelarycznych [s]	Czas działania na struktur AGDS [s]
500	0.5193497	0.01276
1000	2.72629	0.0225039
2000	8.31123	0.041098
2500	11.6229	0.048912
3000	15.07335	0.058066
4000	22.689	0.0784816
5000	31.36399	0.096354

Tabela 4.3. Porównanie czasów operacji na tabelach i strukturach AGDS - lista rekordów podobnych do wybranej grupy rekordów.



Rys. 4.2. Wykres zależności czasu od wielkości zbioru danych dla poszukiwania rekordów podobnych do grupy.

Po analizie algorytmów i wykonaniu obliczeń otrzymano złożoność obliczeniową $O(n)$. Zależności czasu wykonywania algorytmów od wielkości zbioru danych widoczne na wykresach w podrozdziałach 4.1.2 i 4.1.3 potwierdzają przeprowadzone obliczenia. Oba algorytmy działają w ten sam sposób i mają złożoność liniową, ale struktury tabelaryczne mają dużo większy współczynnik kierunkowy linii trendu. Znaczna różnica czasów pomiędzy strukturami tabelarycznymi i grafowymi wynika z powiązań pomiędzy danymi w strukturach AGDS. Pozwala to uniknąć wielu czasochłonnych pętli obliczeniowych.

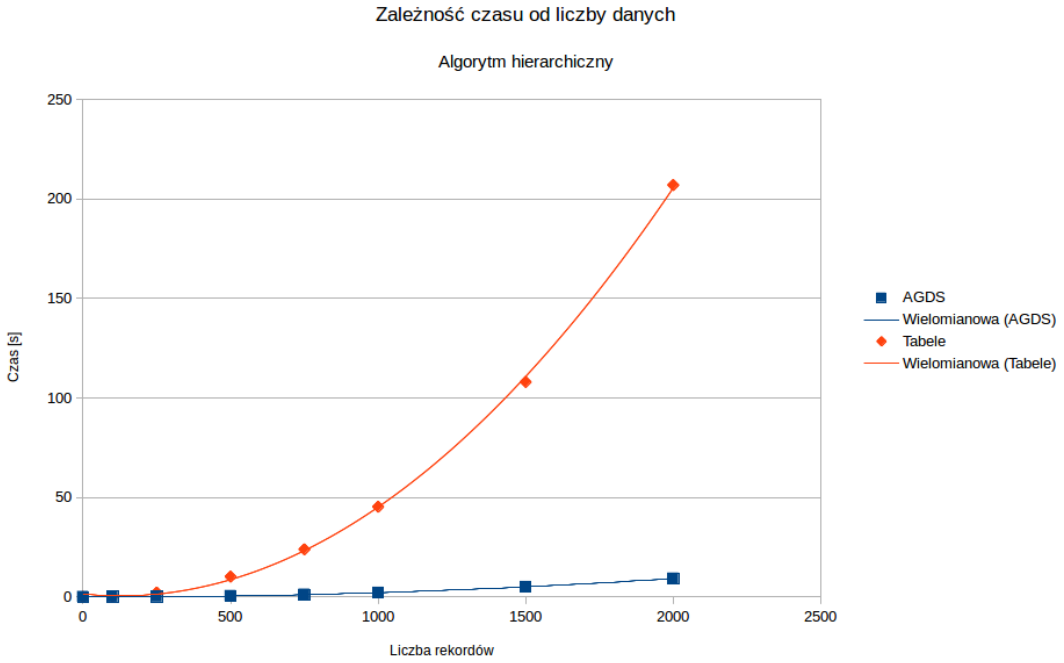
4.1.4. Algorytmy klasteryzacji

Algorytm klasteryzacji hierarchicznej

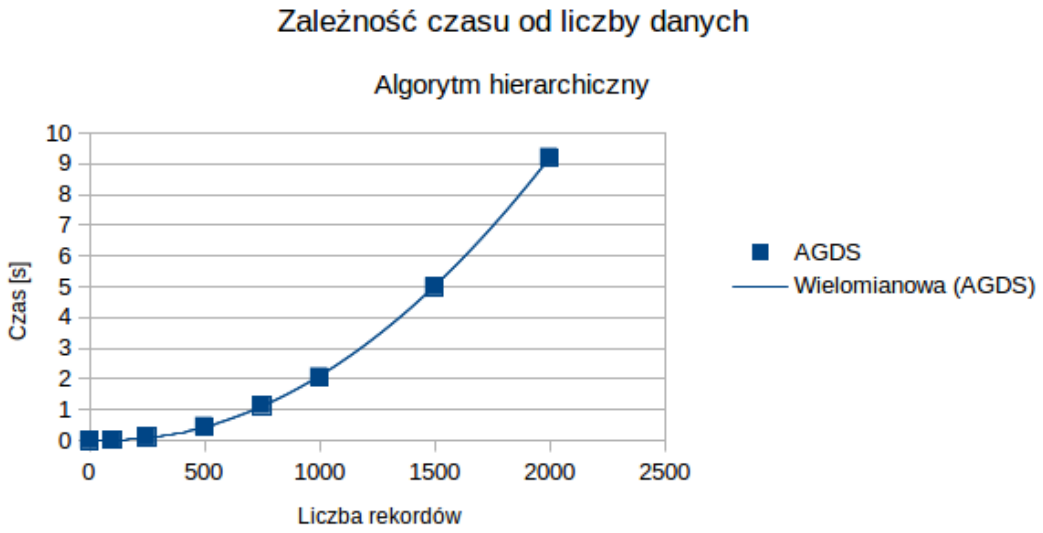
Porównano czasy działania algorytmu dla zbioru *Bike* podzielonego na mniejsze zbiory od 250 do 2000 elementów. Parametr wejściowy określający ilość klastrów został ustawiony na wartość 3. Wyniki przedstawiono w tabeli 4.4.

Rozmiar danych wejściowych	**Czas działania na strukturach tabelarycznych [s]**	**Czas działania na strukturach AGDS [s]**
100	0.3	0.017
250	2.13	0.11
500	10.14	0.46
750	23.86	1.14
1000	45.6	2.07
1500	108	5
2000	207	9.22

Tabela 4.4. Porównanie czasów działania algorytmu klasteryzacji hierarchicznej.



Rys. 4.3. Wykres zależności czasu działania algorytmu od wielkości zbioru danych (algorytm hierarchiczny).



Rys. 4.4. Wykres zależności czasu działania algorytmu od wielkości zbioru danych (algorytm hierarchiczny).

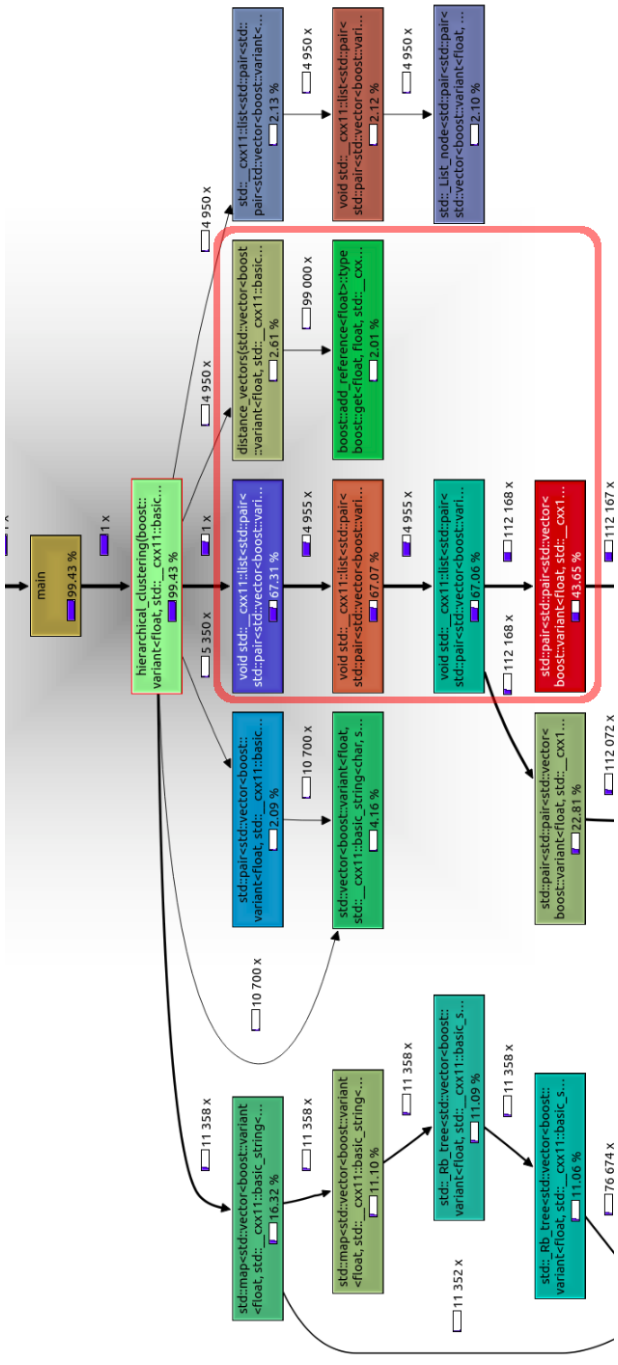
Rys. 4.5. Wykres zależności czasu działania algorytmu od wielkości zbioru danych (algorytm hierarchiczny).

Na wykresie 4.3 widoczna jest duża różnica czasów działania algorytmów. Znacznie lepsze wyniki otrzymano dla struktur AGDS. W strukturach tabelarycznych wielokrotnie wykonywane są pętle, w których wyszukiwane są rekordy o najmniejszych odległościach do siebie nawzajem, co znacząco wpływa na czas działania.

Obliczanie macierzy odległości sprawia, że złożoność obliczeniowa jest rzędu $O(n^2)$. Sortowanie, którego złożoność obliczeniowa wynosi $O(log_n)$ zwiększa skomplikowanie całego algorytmu. Powyższe

zależności zostały zobrazowane na wykresach 4.4 i 4.5. W obu przypadkach dodana została krzywa regresji - wielomian drugiego stopnia, co potwierdza wyliczenia o złożoności obliczeniowej $O(n^2)$.

Działanie algorytmu na obu strukturach zostało przeanalizowane w aplikacji *Valgrind*, służącej do debugowania pamięci i wykrywania wycieków. Przy użyciu aplikacji *KCachegrind* przeanalizowano czas działania poszczególnych operacji (również wewnątrz bibliotek wykorzystywanych w projekcie). Analiza wykazała duży narzut obecny przy konwersji typu BOOST::VARIANT na typ liczbowy (rys. 4.6). Oznacza to, że istnieje dodatkowe pole do optymalizacji czasowej zaimplementowanych algorytmów, ale nie zmieni to ich złożoności obliczeniowej.

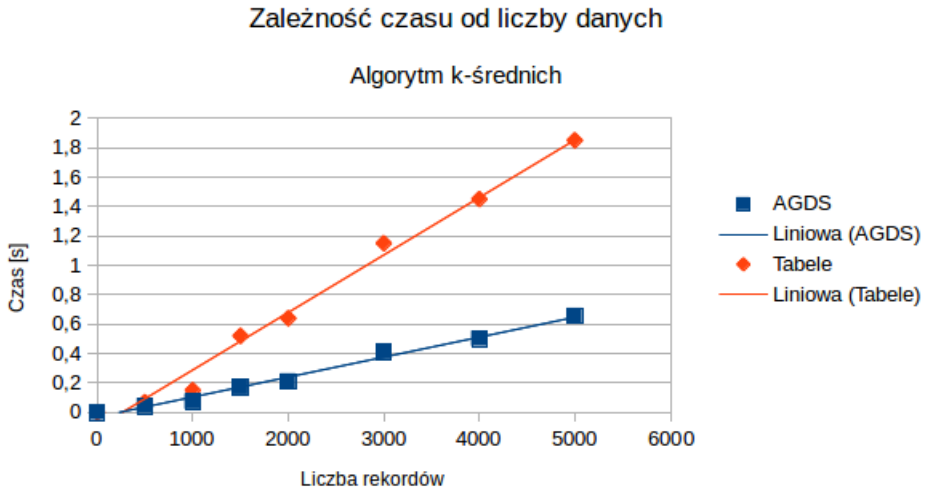


Rys. 4.6. Graf z programu *KCachegrind* wygenerowany za pomocą programu *Valgrind* z zaznaczonymi wywołaniami odpowiadającymi za konwersję typu BOOST::VARIANT.

Algorytm k-średnich

Rozmiar danych wejściowych	Czas działania na strukturach tabelarycznych [s]	Czas działania na strukturach AGDS [s]
500	0.07	0.04
1000	0.15	0.07
1500	0.52	0.17
2000	0,64	0.21
3000	1.15	0.41
4000	1.45	0.5
5000	1.85	0.65

Tabela 4.5. Porównanie czasów działania algorytmu k-średnich.



Rys. 4.7. Wykres zależności czasu działania algorytmu od wielkości zbioru danych (algorytm k-średnich).

Można zaobserwować, że czasy działania algorytmu na strukturach AGDS są kilkukrotnie szybsze niż na strukturach tabelarycznych. Oba algorytmy przechodzą te same kroki: przesuwanie rekordów pomiędzy kolejnymi klastrami, aktualizacja centroidów. Są to operacje czasochłonne. Po analizie złożoności algorytmu otrzymano wynik $O(n)$, przy czym dla struktur AGDS współczynnik kierunkowy linii trendu jest niższy i złożoność czasowa jest dużo mniejsza niż dla tabel.

4.2. Eksperymenty - klasteryzacja

Rezultaty otrzymane jako wyniki zaimplementowanych metod zostały porównane ze sobą wzajemnie oraz do rezultatów tych samych metod języka PYTHON. Dla algorytmu hierarchicznego użyto metody *linkage* z bibliotekiSCI-PY [21] a dla algorytmu k-średnich funkcji *KMeans* z biblioteki *Scikit* [22].

4.2.1. Przykład 1

Do pierwszego eksperymentu wybrano 20 rekordów ze zbioru *Irysy* [20], aby działanie metody było lepiej widoczne na małym zbiorze. W tabeli 4.6 zaprezentowano wartości rekordów. Elementy ze zbioru *Irysy* mają parametr tekstowy *class*, obliczenia zostały przeprowadzone bez jego uwzględnienia, aby nie wpływał on na grupowanie.

nr rekordu	leaf-length	leaf-width	petal-length	petal-width	class
1	5.10	3.50	1.40	0.20	Iris-setosa
2	4.90	3.00	1.40	0.20	Iris-setosa
3	4.70	3.20	1.30	0.20	Iris-setosa
4	4.60	3.10	1.50	0.20	Iris-setosa
5	5.00	3.60	1.40	0.20	Iris-setosa
6	4.60	3.40	1.40	0.30	Iris-setosa
7	7.00	3.20	4.70	1.40	Iris-versicolor
8	6.40	3.20	4.50	1.50	Iris-versicolor
9	6.90	3.10	4.90	1.50	Iris-versicolor
10	5.50	2.30	4.00	1.30	Iris-versicolor
11	6.50	2.80	4.60	1.50	Iris-versicolor
12	5.70	2.80	4.50	1.30	Iris-versicolor
13	6.30	3.30	6.00	2.50	Iris-virginica
14	5.80	2.70	5.10	1.90	Iris-virginica
15	7.10	3.00	5.90	2.10	Iris-virginica
16	6.30	2.90	5.60	1.80	Iris-virginica
17	6.50	3.00	5.80	2.20	Iris-virginica
18	7.60	3.00	6.60	2.10	Iris-virginica
19	7.30	2.90	6.30	1.80	Iris-virginica
20	6.40	3.20	5.30	2.30	Iris-virginica

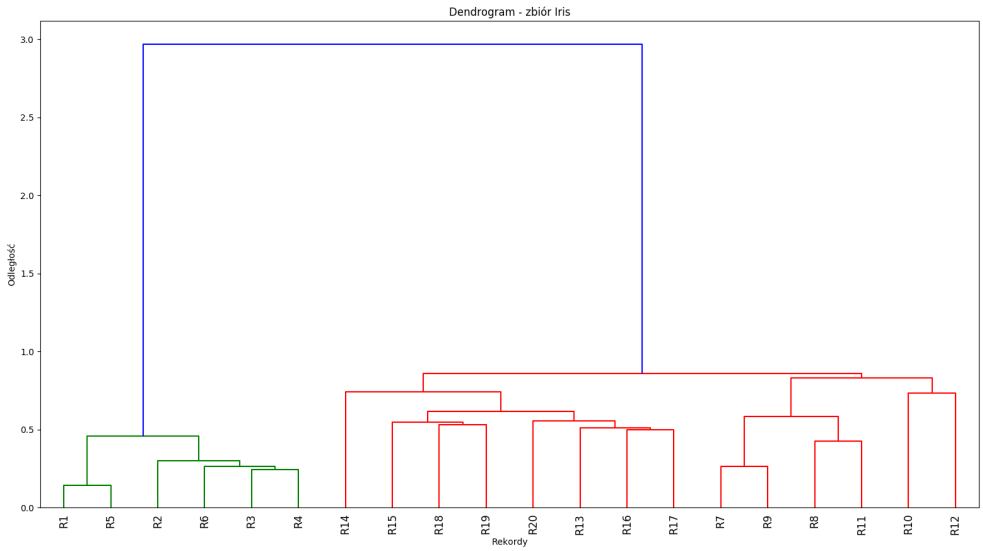
Tabela 4.6. 20 rekordów ze zbiory *Irysy*.

4.2.1.1. Wyniki - algorytm hierarchiczny

Wyniki działania algorytmu hierarchicznego przedstawiono w tabeli 4.7. Porównano macierz odległości z wynikiem zwróconym przez metodę *linkage* z biblioteki SCI-PY i otrzymane wyniki były takie same. Na rysunku 4.8 zaprezentowano wykres przedstawiający hierarchiczne połączenie rekordów ze sobą. Można z niego odczytać sposób grupowania elementów na dowolnym poziomie.

klaster nr 1	klaster nr 2	klaster nr 3
R1	R13	R7
R2	R14	R8
R3	R15	R9
R4	R16	R10
R5	R17	R11
R6	R18	R12
	R19	
	R20	

Tabela 4.7. Wynik algorytmu hierarchicznego, podział na 3 klastry.



Rys. 4.8. Dendrogram przedstawiający połączenia klastrów dla zbioru 20 Irysów.

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20
R1		0,88	0,87	0,96	0,72	0,87	4,06	3,68	4,22	3,17	3,85	3,48	5,33	4,26	5,34	4,74	5,1	6,13	5,68	4,67
R2	0,88		0,76	0,78	0,93	0,81	4,15	3,75	4,29	3,05	3,87	3,46	5,38	4,24	5,4	4,76	5,14	6,2	5,73	4,73
R3	0,87	0,76		0,74	0,87	0,75	4,33	3,91	4,47	3,23	4,05	3,62	5,51	4,39	5,57	4,91	5,29	6,37	5,9	4,87
R4	0,96	0,78	0,74		0,95	0,78	4,23	3,8	4,36	3,06	3,9	3,46	5,38	4,23	5,45	4,77	5,15	6,25	5,78	4,75
R5	0,72	0,93	0,87	0,95		0,85	4,12	3,73	4,27	3,22	3,91	3,52	5,36	4,3	5,39	4,78	5,14	6,18	5,73	4,71
R6	0,87	0,81	0,75	0,78	0,85		4,28	3,85	4,42	3,2	4,01	3,56	5,42	4,32	5,51	4,85	5,21	6,31	5,85	4,79
R7	4,06	4,15	4,33	4,23	4,12	4,28		0,95	0,75	2,01	0,96	1,54	1,97	1,61	1,57	1,43	1,62	2,23	1,84	1,42
R8	3,68	3,75	3,91	3,8	3,73	3,85	0,95		0,95	1,55	0,82	1,09	1,94	1,27	1,83	1,37	1,65	2,59	2,17	1,33
R9	4,22	4,29	4,47	4,36	4,27	4,42	0,75	0,95		1,98	0,91	1,49	1,76	1,43	1,38	1,21	1,4	2,06	1,65	1,25
R10	3,17	3,05	3,23	3,06	3,22	3,2	2,01	1,55	1,98		11,46	1,01	2,75	1,52	2,79	2,07	2,45	3,57	3,1	2,19
R11	3,85	3,87	4,05	3,9	3,91	4,01	0,96	0,82	0,91	11,46		1,09	1,93	1,18	1,71	1,28	1,57	2,47	2,03	1,34
R12	3,48	3,46	3,62	3,46	3,52	3,56	1,54	1,09	1,49	1,01	1,09		2,19	1,11	2,25	1,52	1,91	3,03	2,56	1,67
R13	5,33	5,38	5,51	5,38	5,36	5,42	1,97	1,94	1,76	2,75	1,93	2,19		1,5	1,18	1,14	0,87	1,67	1,49	1,02
R14	4,26	4,24	4,39	4,23	4,3	4,32	1,61	1,27	1,43	1,52	1,18	1,11	1,5		1,72	1,02	1,28	2,47	2,05	1,14
R15	5,34	5,4	5,57	5,45	5,39	5,51	1,57	1,83	1,38	2,79	1,71	2,25	1,18	1,72		1,15	0,93	1,11	0,89	1,19
R16	4,74	4,76	4,91	4,77	4,78	4,85	1,43	1,37	1,21	2,07	1,28	1,52	1,14	1,02	1,15		0,86	1,81	1,41	0,69
R17	5,1	5,14	5,29	5,15	5,14	5,21	1,62	1,65	1,4	2,45	1,57	1,91	0,87	1,28	0,93	0,86		1,53	1,24	0,9
R18	6,13	6,2	6,37	6,25	6,18	6,31	2,23	2,59	2,06	3,57	2,47	3,03	1,67	2,47	1,11	1,81	1,53		0,88	1,92
R19	5,68	5,73	5,9	5,78	5,73	5,85	1,84	2,17	1,65	3,1	2,03	2,56	1,49	2,05	0,89	1,41	1,24	0,88		1,62
R20	4,67	4,73	4,87	4,75	4,71	4,79	1,42	1,33	1,25	2,19	1,34	1,67	1,02	1,14	1,19	0,69	0,9	1,92	1,62	

Tabela 4.8. Macierz odległości pomiędzy rekordami ze zbioru *Irysy*.

4.2.1.2. Wyniki - algorytm k-średnich

Wylosowano 3 początkowe centroidy, są to rekordy numer: 5, 11, 12 (tab. 4.6). Warunkiem stopu jest niezmienność centroidów w kolejnych iteracjach. Centroidy są liczone jako średnie wartości wszystkich rekordów należących do klastra. W tym przykładzie w 4. iteracji centroidy zostały ustalone. Ich końcowe wartości przedstawiono w tabeli 4.9.

	leaf-length	leaf-width	petal-length	petal-width
Centroid 1	4.82	3.3	1.4	0.22
Centroid 2	6.78	3.04	5.92	2.11
Centroid 3	6.25	2.87	4.61	1.48

Tabela 4.9. Końcowe wartości centroidów dla początkowych centroidów R5, R11 i R12.

klaster nr 1	klaster nr 2	klaster nr 3
R1	R13	R7
R2	R15	R8
R3	R16	R9
R4	R17	R10
R5	R18	R11
R6	R19	R12
	R20	R14

Tabela 4.10. Wynik algorytmu hierarchicznego, podział na 3 klastry.

Aby sprawdzić wpływ doboru centroidów na przebieg algorytmu, przeprowadzono eksperyment dla innych rekordów początkowych: 8, 14 i 15. Centroidy pozostały niezmienione po 4. iteracji. Ich wartości przedstawiono w tabeli 4.11.

	leaf-length	leaf-width	petal-length	petal-width
Centroid 1	4.81	3.3	1.4	0.21
Centroid 2	6.27	2.91	4.8	1.61
Centroid 3	6.96	3.04	6.12	2.14

Tabela 4.11. Końcowe wartości centroidów dla początkowych centroidów R8, R14 i R15.

Wyniki w tym przypadku są podobne do poprzedniego przykładu, ale kilka rekordów zmieniło miejsce. Klaster o numerze 1 pozostał niezmieniony natomiast R14 i R20 zostały przesunięte do klastra numer 3. Wyniki zaprezentowano w tabeli 4.12.

klaster nr 1	klaster nr 2	klaster nr 3
R1	R13	R7
R2	R15	R8
R3	R17	R9
R4	R18	R10
R5	R19	R11
R6		R12
		R14
		R16
		R20

Tabela 4.12. Wynik algorytmu hierarchicznego, podział na 3 klastry.

Powyższe eksperymenty pokazują, że podział rekordów na klastry jest niedeterministyczny, tzn. zależy w dużej mierze od początkowego doboru centroidów.

Działanie metody porównano z wynikami otrzymanymi z algorytmu znajdującego się w bibliotece SCIKIT-LEARN. Metoda *sklearn.cluster.KMeans* losowo dobiera centroidy początkowe z podanego zbioru danych, nie ma możliwości ich ustawienia, aby porównać działanie metod dla tych samych centroidów początkowych.

W wyniku działania tej metody otrzymano następujące centroidy końcowe (tab. 4.13):

	leaf-length	leaf-width	petal-length	petal-width
Centroid 1	4.82	3.3	1.4	0.22
Centroid 2	6.78	3.04	5.92	2.11
Centroid 3	6.25	2.87	4.61	1.58

Tabela 4.13. Końcowe wartości centroidów dla początkowych centroidów dla metody *sklearn.cluster.KMeans*.

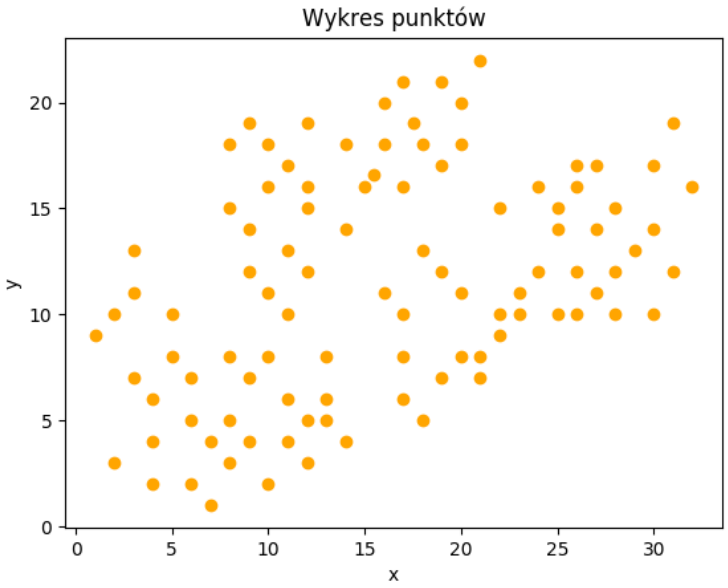
Centroidy są takie same jak w poprzednim przypadku (tab. 4.9), ale nieco inne niż w przypadku 2. (tab. 4.11). Otrzymane przypisanie poszczególnych rekordów do grup otrzymuje się w formie wektora, z numerami klastrów, do których przypisane są rekordy. Podział na klastry jest identyczny jak w przypadku 1. (tab. 4.10).

4.2.2. Przykład 2

Przykład 2 został wykonany na zbiorze *XY* (tab. 4.14) stworzonym na potrzeby pracy. Składa się ze 100 rekordów, każdy z nich ma współrzędne *(x,y)*. Przykład z danymi dwuwymiarowymi dobrze pokazuje działanie algorytmu k-średnich i przemieszczanie się centroidów w kolejnych iteracjach. Rysunek 4.9 przedstawia rozkład punktów w układzie współrzędnych.

Rekord	x	y	Rekord	x	y	Rekord	x	y	Rekord	x	y
R1	3	7	R26	1	9	R51	18	18	R76	22	10
R2	4	4	R27	2	10	R52	17.5	19	R77	23	10
R3	4	6	R28	3	11	R53	17	16	R78	23	11
R4	5	8	R29	5	10	R54	15.5	16.6	R79	24	12
R5	6	5	R30	3	13	R55	20	18	R80	24	16
R6	6	2	R31	6	7	R56	19	17	R81	25	15
R7	7	1	R32	8	15	R57	17	21	R82	25	14
R8	7	4	R33	8	18	R58	16	20	R83	26	16
R9	8	3	R34	9	12	R59	20	20	R84	26	12
R10	8	5	R35	9	14	R60	19	21	R85	26	17
R11	8	8	R36	9	19	R61	21	22	R86	27	17
R12	9	4	R37	10	18	R62	17	8	R87	27	14
R13	9	7	R38	10	16	R63	16	11	R88	28	15
R14	10	8	R39	10	11	R64	17	6	R89	29	13
R15	11	4	R40	11	10	R65	17	10	R90	30	14
R16	11	6	R41	11	17	R66	18	5	R91	25	10
R17	12	5	R42	12	16	R67	18	13	R92	30	17
R18	13	5	R43	11	13	R68	19	7	R93	31	19
R19	13	8	R44	12	12	R69	19	12	R94	30	10
R20	13	6	R45	12	15	R70	20	8	R95	26	10
R21	14	4	R46	12	19	R71	20	11	R96	27	11
R22	2	3	R47	15	16	R72	21	7	R97	28	12
R23	4	2	R48	14	14	R73	21	8	R98	28	10
R24	10	2	R49	14	18	R74	22	9	R99	31	12
R25	12	3	R50	16	18	R75	22	15	R100	32	16

Tabela 4.14. Zbiór *XY*.



Rys. 4.9. Punkty zbioru *XY* przedstawione w układzie współrzędnym.

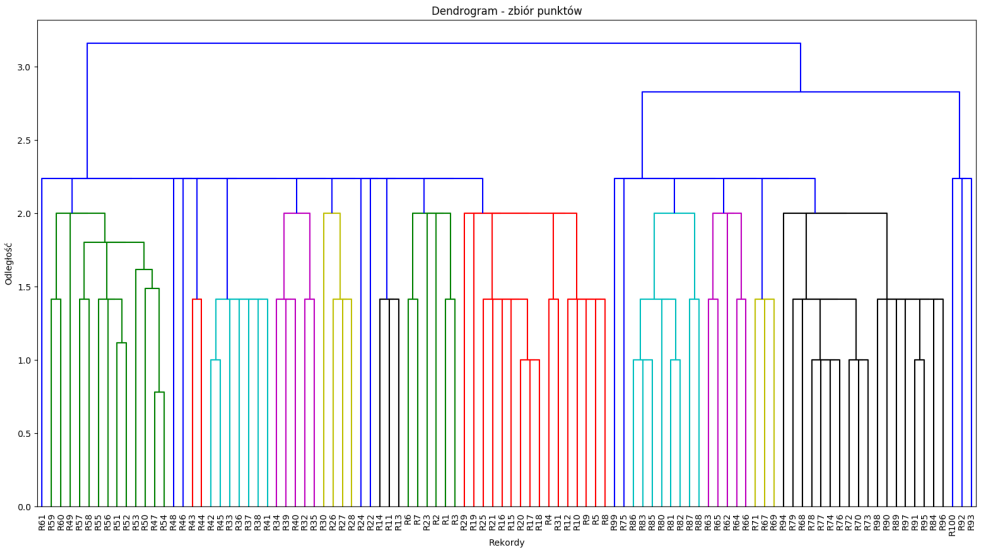
4.2.2.1. Wyniki - algorytm hierarchiczny

W tym przykładzie, podobnie jak w poprzednim, porównano otrzymane wyniki z rezultatami algorytmu z biblioteki SCI-PY. W tym przypadku macierze odległości również się pokrywają i klasyfikacja rekordów do grup jest identyczna.

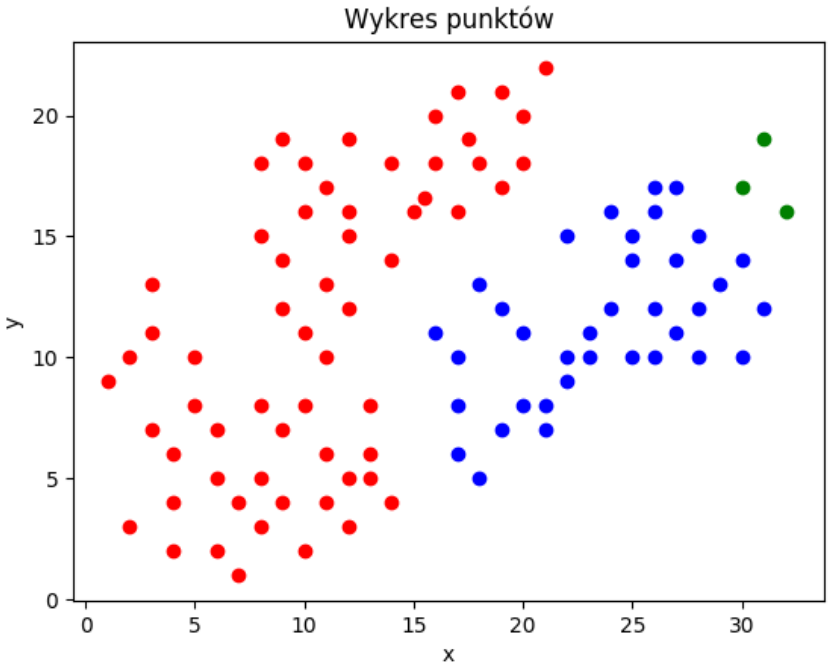
Otrzymany podział na klastry zaprezentowano w tabeli 4.15.

klaster 1						klaster 2				klaster 3
R52	R56	R57	R26	R27	R28	R67	R64	R97	R90	R92
R58	R59	R60	R30	R6	R7	R69	R66	R68	R72	R93
R54	R55	R51	R2	R1	R3	R71	R63	R96	R89	R100
R53	R50	R47	R23	R19	R5	R62	R65	R95	R84	
R46	R48	R49	R8	R9	R10	R94	R99	R79	R91	
R45	R43	R44	R12	R4	R31	R88	R87	R77	R78	
R37	R41	R42	R29	R16	R15	R86	R85	R74	R76	
R38	R33	R36	R17	R18	R20	R83	R82	R70	R73	
R40	R32	R35	R25	R21	R22	R81	R80			
R14	R34	R39	R24	R11	R13	R75	R98			

Tabela 4.15. Podział na klastry (algorytm hierarchiczny).



Rys. 4.10. Dendrogram pokazujący działanie algorytmu hierarchicznego na zbiorze *XY*.

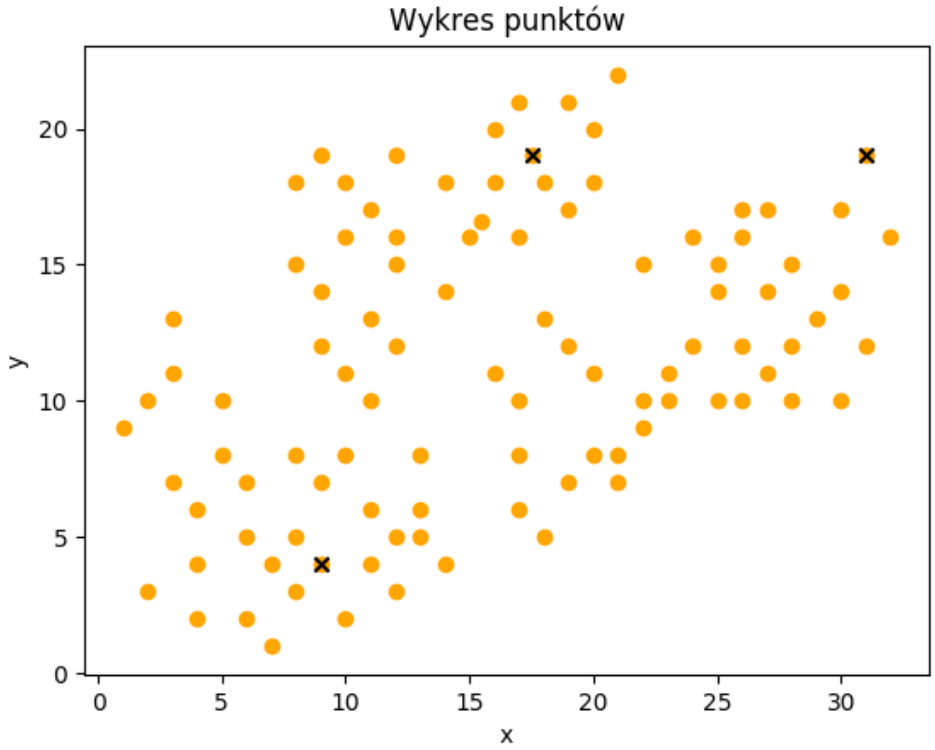


Rys. 4.11. Klasteryzacja punktów ze zbioru *XY* metodą hierarchiczną.

4.2.2.2. Wyniki - algorytm k-średnich

Dla tego przykładu początkowymi centroidami było rekordy R12, R53 i R92. Ich wartości umieszczono w tabeli 4.16. W tabeli 4.17 zaprezentowano wartości końcowe centroidów. Algorytm zakończył

się po 6 krokach - centroidy i rekordy nie zmieniały swoich położeń. Rysunek 4.12 przedstawia położenie centroidów na początku, a rysunek 4.13 klasyfikację rekordów oraz położenie centroidów na końcu algorytmu (oznaczone X).



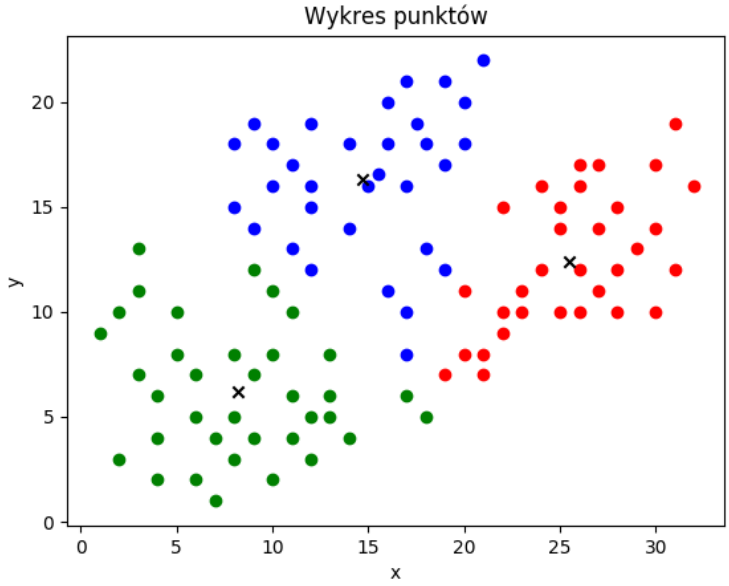
Rys. 4.12. Położenie centroidów początkowych R12, R53 i R92.

	x	y
R12	31	19
R53	17.5	19
R92	9	4

Tabela 4.16. Początkowe wartości centroidów, R12, R53 i R92.

	x	y
Centroid 1	25.5	12.4
Centroid 2	14.7	16.3
Centroid 3	8.2	6.2

Tabela 4.17. Końcowe wartości centroidów dla początkowych centroidów R12, R53 i R92.

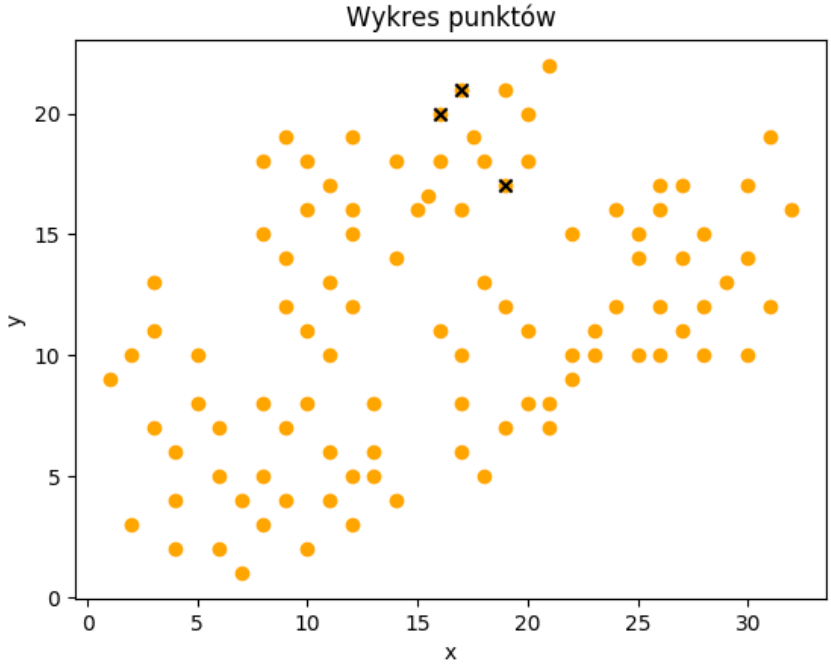


Rys. 4.13. Końcowe centroidy grup dla początkowych centroidów R12, R53 i R92.

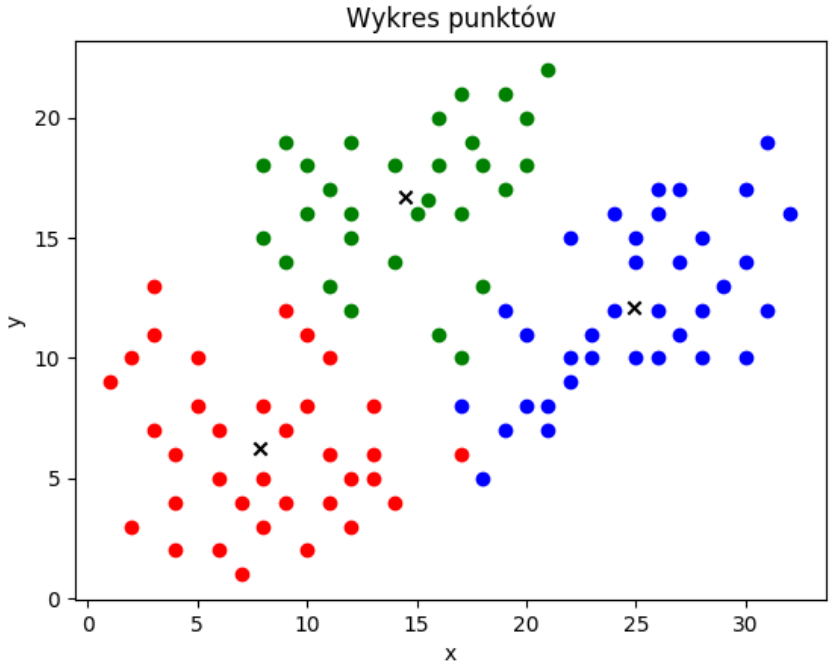
Dla porównania wylosowano inne centroidy początkowe: R56, R57 i R58 (tab. 4.20). Po 18. iteracji centroidy nie zmieniały swoich położeń i miały wartości przedstawione w tab. 4.20 oraz na wykresie 4.15.

	leaf-length	leaf-width
R56	19	17
R57	17	21
R58	16	20

Tabela 4.18. Początkowe wartości centroidów: R56, R57 i R58.



Rys. 4.14. Położenie centroidów początkowych R56, R57 i R58.

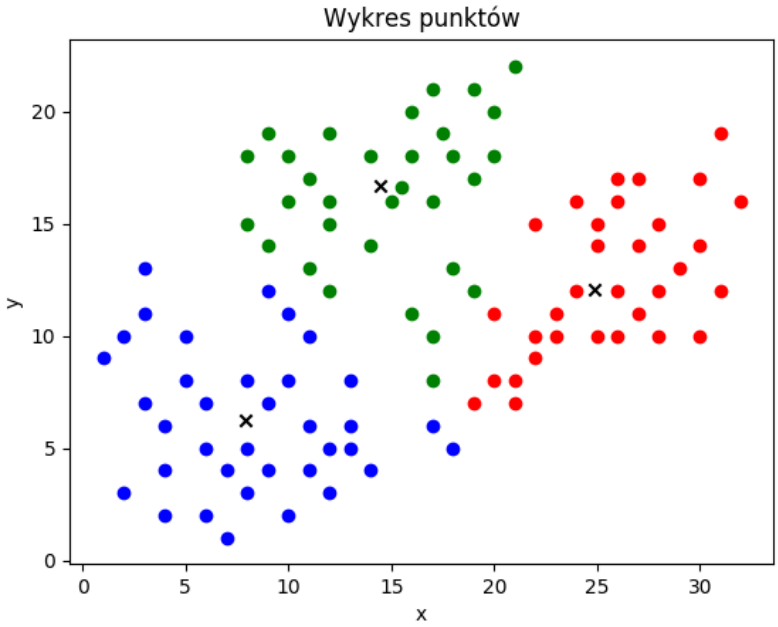


Rys. 4.15. Położenie końcowe centroidów dla centroidów początkowych R56, R57 i R58.

	x	y
Centroid 1	24.9	12.08
Centroid 2	7.88	6.25
Centroid 3	14.46	16.7

Tabela 4.19. Końcowe wartości centroidów dla centroidów początkowych R56, R57 i R58.

Rezultaty porównano z wynikami z funkcji *sklearn.cluster.KMeans*.



Rys. 4.16. Wynik klasteryzacji algorytmu *sklearn.cluster.KMeans*.

	x	y
Centroid 1	25.56	12.43
Centroid 2	8.16	6.22
Centroid 3	14.68	16.27

Tabela 4.20. Końcowe wartości centroidów - metoda *sklearn.cluster.KMeans*.

4.2.3. Wnioski płynące z eksperymentów

Po analizie przykładu pierwszego (rozdział 4.2.1) i drugiego (rozdział 4.2.2) można zauważyć, że wyniki pochodzące z metody hierarchicznej i k-średnich są do siebie bardzo zbliżone.

W przykładzie pierwszym (zbiór *Irysy*) otrzymana klasteryzacja pokrywa się w pełni dla klastra nr 1, natomiast dla klastra nr 2 i klastra nr 3 jest różnica kilku rekordów. Wynika to z podobieństwa pomiędzy elementami - Irysami z gatunku *Versicolor* oraz *Virginica* (tab. 4.6). W przypadku algorytmu k-średnich otrzymano dla dwóch prób zbliżone wartości centroidów dla różnych rekordów początkowych.

W przykładzie drugim (zbiór *XY*) podział na klastry w algorytmie hierarchicznym wygląda inaczej niż w k-średnich. Widać dużą dysproporcję w rozmiarach klastrów. Wynika to z charakteru algorytmu - mniejsze klastry łączone są w kolejnych krokach z większymi. Metoda k-średnich dała dobre rezultaty klasteryzacji, nawet pomimo tego, że w drugim przypadku początkowe centroidy były bardzo blisko siebie (rys. 4.14).

Zestawiając działanie zaimplementowanych metod z algorytmami pochodzącymi z bibliotek języka PYTHON można zauważyć, że dla obu algorytmów wyniki pokrywają się. Najbardziej wiarygodne zestawienie można uzyskać dla algorytmu hierarchicznego, ze względu na zwracaną przez funkcję *linkage* macierz odległości. Po porównaniu macierzy pochodzących z obu funkcji jednoznacznie stwierdzono, że wyniki są identyczne i grupowanie przebiegało w ten sam sposób.

Rezultaty algorytmu k-średnich były porównywane do funkcji *KMeans*. Metoda ta losowo przyjmuje wartości centroidów i nie ma możliwości wpłynięcia na dobór centroidów początkowych tak, aby w obu przypadkach były takie same. Funkcja zwraca wektor mówiący o przynależności rekordów do klastrów oraz końcowe wartości centroidów. Wyniki pochodzące z obu metod są do siebie bardzo zbliżone.

4.3. Porównanie algorytmów klasteryzacji

Wnioski otrzymane z analizy eksperymentów i działania algorytmu podsumowano w tabeli 4.21. Pierwszym porównywanym aspektem jest struktura powstałych klastrów. W wyniku zastosowania algorytmu hierarchicznego można otrzymać klastry mocno różniące się ilością elementów. W przykładzie pierwszym, w przypadku podziału na więcej niż 4 klastry rekord R14 stanowiłby osobny klaster, ponieważ jego odległość do pozostałych w grupie jest największa (rys. 4.8). W przykładzie drugim dysproporcję pomiędzy klastrami widać na wykresie 4.10. Klaster numer 3 zawiera jedynie 3 elementy, a pozostałe klastry są o wiele liczniejsze. Przy algorytmie k-średnich wielkość i kształt klastrów zależy od wyboru początkowych centroidów.

Kolejną różnicą pomiędzy algorytmami jest konieczność podawania liczby klastrów do algorytmu k-średnich, przy algorytmie hierarchicznym nie ma potrzeby podawania tego parametru.

Szybkość działania przemawia na korzyść algorytmu k-średnich. Algorytm hierarchiczny wymaga wyliczenia macierzy odległości oraz wielokrotnego łączenia klastrów ze sobą. Niesie to za sobą większą złożoność obliczeniową oraz spowalnia działanie. Ten aspekt wpływa również na wielkość zbiorów, dla których powinno się stosować te algorytmy. Dla dużych zbiorów danych przy algorytmie hierarchicznym czas wykonywania będzie długi, dlatego lepiej sprawdzi się algorytm k-średnich [23]. Jeżeli natomiast dane wejściowe będą ze sobą powiązane, korzystniej będzie zastosować algorytm hierarchiczny.

Cecha	Algorytm hierarchiczny	Algorytm k-średnich
Struktura powstałych klastrów	Hierarchiczna, zależne i powiązane ze sobą.	Niezależne od siebie, odrębne.
Ilość klastrów	Nie musi być z góry określona, można określić na podstawie struktury.	Ilość klastrów musi zostać określona przed wykonaniem algorytmu.
Szybkość działania	Powolny - dużo operacji przeszukiwania i łączenia klastrów.	Szybki, bardziej wydajny niż algorytm hierarchiczny.
Dla jakich typów danych	Polecany do działań na małych i średnich zbiorach danych [23, 10], zbiory hierarchiczne.	Duże zbiory, niezależne dane.
Złożoność obliczeniowa	$O(n^2)$	$O(n)$

Tabela 4.21. Porównanie algorytmów k-średnich oraz hierarchicznego pod względem wybranych cech.

5. Podsumowanie

W ramach niniejszej pracy magisterskiej udało się zrealizować asocjacyjny system automatycznej klasteryzacji i eksploracji danych. Po wczytaniu dowolnego zbioru system przeprowadza jego analizę. Odkrywane są cechy, takie jak minimum czy maksimum danego parametru, podobieństwo rekordów, a także sposób ich łączenia w grupy przy określeniu dowolnej liczby klastrów. Wszystkie powyższe operacje wykonywane są przy użyciu panelu użytkownika, który również wyświetla wyniki. W ramach pracy udało się również zwizualizować proces wyszukiwania podobnego rekordu w grafie, pokazać strukturę, jaką buduje algorytm hierarchiczny oraz zaprezentować graficznie przebieg algorytmu k-średnich w kolejnych iteracjach. Przedstawiono także porównanie złożoności obliczeniowej i czasowej powyższych działań.

Dane do eksperymentów dobierano pod kątem liczby parametrów i rekordów. Porównano wyniki działania algorytmów na zbiorach o różnych wielkościach.

Algorytmy zostały zaimplementowane w sposób umożliwiający obsługę danych różnego typu. Był to jeden z większych problemów, na które natrafiono przy implementacji. Naturalne jest, że bazy danych informacji zawierają rekordy o mieszanych typach pól, np. liczbowe, tekstowe czy binarne. Obsługa takiej bazy w języku *C++* tworzy problem programistyczny powodowany cechą tego języka, znaną jako statyczna typowalność [24]. Kompilator C++ w czasie kompilacji programu musi znać typy wszystkich danych. Oczywistym jest, że typy danych zawartych w bazie znane są dopiero po uruchomieniu programu i wczytaniu zbioru. W kontenerze takim jak *list* czy *vector* przechowywany może być tylko jeden typ danych. Zatem, aby obsługiwać dane różnego typu, użyto struktury udostępnianej przez bibliotekę BOOST::VARIANT. Obiekt tego typu może przyjmować wartość zarówno liczbową, tekstową jak i binarną.

Zaprezentowane wyniki wykazują, że struktury AGDS są efektywniejsze od struktur tabelarycznych. Możliwy jest dostęp w czasie stałym do informacji zawartych w obiektach, a wyszukiwanie zależności pomiędzy rekordami jest niezwykle szybkie. Największa różnica czasów wykonywania się algorytmów pomiędzy strukturami tabelarycznymi i AGDS widoczna jest przy wyszukiwaniu rekordów podobnych. Dla 5 tysięcy rekordów czas działania metody dla struktur tabelarycznych jest ponad trzystukrotnie większy niż dla AGDS. Przy zadaniu eksploracji danych, gdzie rozpatruje się głównie duże zbiory danych, czas operacji jest niezwykle ważnym czynnikiem.

W pracy pokazane zostały także cechy algorytmu hierarchicznego i algorytmu k-średnich oraz różnice w ich działaniu. Algorytm k-średnich działa szybciej niż hierarchiczny, ale jego działanie jest w du-

żej mierze zależne od wyboru centroidów początkowych. Algorytm hierarchiczny zawsze daje ten sam rezultat dla tych samych danych. Wykonuje on więcej operacji i jest polecany dla mniejszych zbiorów. Wyniki czasowe działania algorytmów napisanych przy użyciu AGDS są wielokrotnie lepsze niż tych samych algorytmów napisanych przy użyciu tabel. Porównanie otrzymanych wyników z wynikami algorytmów pochodzących z bibliotek języka PYTHON dowodzi poprawności zaimplementowanych algorytmów.

Jako propozycje dalszego rozwoju można wskazać implementację dodatkowych algorytmów klasteryzacji, np. sieci Kohonena czy algorytmów rozmytych grupowania, w celu porównania otrzymanych wyników i dalszego wnioskowania na temat różnic i podobieństw między nimi. Kolejną propozycją jest stworzenie uniwersalnej biblioteki zawierającej zaimplementowaną strukturę grafową AGDS, tak aby umożliwić innym osobom łatwiejsze korzystanie z tych struktur i tym samym wpłynąć na ich popularyzację.

Bibliografia

[1] L. Rutkowski. *Wybrane zagadnienia sztucznej inteligencji*. Wydawnictwo Naukowe PWN, 2005.

[2] S. Osowski. *Metody i narzędzia eksploracji danych*. Wydawnictwo BTC, 2013.

[3] Hand D.; Mannila H; Smyth P. *Principles of Data Mining*. MIT Press, 2001.

[4] Morzy T. „Eksploracja danych: problemy i rozwiązania". W: paź. 1999.

[5] Morzy T. „Eksploracja danych". W: *NAUKA* (mar. 2007).

[6] Han J.; Kamber M. *Data Mining: Concepts and Techniques*. Morgan Kaufman Publishers, 2006.

[7] D. T. Larose. *Odkrywanie wiedzy z danych. Wprowadzenie do eksploracji danych*. Wydawnictwo Naukowe PWN, 2006.

[8] A.Jain; R. Dubes. *Algorithms for clustering data*. Prentice-Hall, 1988.

[9] Wawrzyniak A. *Klasyfikacje rozmyte w modelowaniu zjawisk ekonomicznych*. Uniwersytet Szczeciński.

[10] Everitt B. i in. *Cluster Analysis*. John Wiley and Sons, Ltd, 2011.

[11] Pilowsky I. i in. „The classification of depression by numerical taxonomy". W: *British Journal of Psychiatry* ().

[12] Littmann T. „An empirical classification of weather types in the Mediterranean Basin and their interrelation with rainfall". W: *Theoretical and Applied Climatology* ().

[13] A. Horzyk. *Sztuczne systemy skojarzeniowe i asocjacyjna sztuczna inteligencja*. Akademicka Oficyna Wydawnicza EXIT, 2003.

[14] Aho A. V.; Hopcroft A. E.; Ullman J. D. *Algorytmy i struktury danych*. Helion, 2003.

[15] Wroblewski P. *Algorytmy, struktury danych i techniki programowania*. Helion, 2003.

[16] *Boost.Variant documentation*. URL: *http://www.boost.org/doc/libs/1_64_0/doc/html/variant.html*.

[17] *Graphviz - Graph Visualization Software*. URL: *http://www.graphviz.org/*.

[18] *Struktury asocjacyjne oraz asocjacyjne grafy neuronowe do eksploracji wiedzy z danych - Adrian Horzyk*. URL: *http://home.agh.edu.pl/~horzyk/lectures/miw/MIW-AsocjacyjneGrafyNeuronowe.pdf*.

[19] *Asocjacyjne grafowe struktury danych AGDS - Adrian Horzyk*. URL: *http://home.agh.edu.pl/~horzyk/lectures/miw/MIW-AGDS-%C4%86wiczenia.pdf*.

[20] *UCI Machine Learning Repository*. URL: *http://archive.ics.uci.edu/ml/datasets.html*.

[21] *Hierarchical clustering (scipy.cluster.hierarchy)*. URL: *https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html*.

[22] *Scikit learn documentation*. URL: *http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html*.

[23] Kaur U. „Comparison between K-Mean and Hierarchical algorithm using query redirection". W: *International Journal of Advanced Research in Computer Science and Software Engineering* ().

[24] Eckel B. *Thinking in C++*. Helion, 2002.

Spis rysunków

Spis tablic