POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRONIKI I TECHNIK INFORMACYJNYCH
INSTYTUT INFORMATYKI

Rok akademicki 2003/2004

PRACA DYPLOMOWA MAGISTERSKA

Andrzej Dominik

Analiza danych z zastosowaniem teorii zbiorów przybliżonych.

Opiekun pracy:
dr inż. Roman Podraza

Ocena …….

……

Podpis Przewodniczącego
Komisji Egzaminu Dyplomowego

Specjalność **Informatyka – Inżynieria oprogramowania i systemy informacyjne**

Data urodzenia **10 czerwca 1980**

Data rozpoczęcia studiów **01 października 1999**

ŻYCIORYS

Urodziłem się 10.06.1980 r. w Kielcach. Po ukończeniu szkoły podstawowej, kontynuowałem naukę w IV L.O. im. Hanki Sawickiej w Kielcach. W szkole średniej uczęszczałem do klasy o profilu matematyczno-fizycznym. W październiku 1999 r. rozpocząłem studia na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej.

…….

podpis studenta

EGZAMIN DYPLOMOWY

Złożył egzamin dyplomowy w dniu ……..…….2004 r.

z wynikiem ……..……..

Ogólny wynik studiów ……...…

Dodatkowe wnioski i uwagi Komisji ……

……

……

STRESZCZENIE

Niniejsza praca ma na celu przedstawienie możliwości wykorzystania teorii zbiorów przybliżonych do analizy różnego rodzaju danych. Obok aspektów czysto teoretycznych tej metodologii, zostały także zaprezentowane algorytmy służące do dyskretyzacji danych ciągłych oraz wyznaczające redukty względne tablicy decyzyjnej. Praca porusza także problem oceny wiarygodności zgromadzonych danych w kontekście zgodności pojedynczych obiektów z pozostałymi na podstawie odkrytych, zachodzących między nimi, zależności.

Integralną częścią pracy jest aplikacja ARES Rough Set Exploration System, w której zaimplementowano opisane algorytmy. Program umożliwia dyskretyzację danych ciągłych, wyznaczanie reduktów względnych oraz wyliczanie współczynników wiarygodności dla poszczególnych obiektów tablicy decyzyjnej.

Słowa kluczowe: tablica decyzyjna, zbiory przybliżone, dyskretyzacja, redukt, walidacja danych

---

Data Analysis Based on Rough Set Theory

ABSTRACT

The purpose of this thesis is to present possible application of the rough set theory in different data analysis. Theoretical aspects of this methodology and algorithms for real data discretization and finding relative reduct of decision table were presented. This thesis considers also the problem of estimating credibility of collected data in the context of consistency of single objects with others. Consistency concerns relationships explored between objects from a decision table

The integral part of the thesis is ARES Rough Set Expolration System. It includes implementation of all algorithms described in here. The program lets the user to discretize real data, find relative reducts and calculate credibility coefficients for all objects from a decision table.

Key words: decision table, rough sets, discretization, reduct, data validation

Spis treści.

1. Wstęp.

1.1. Wprowadzenie.

W ciągu ostatnich kilkunastu lat obserwujemy dość szybki rozwój technologii informatycznych. Jedną z przyczyn takiego stanu rzeczy jest ciągle rosnące zapotrzebowanie na przechowywanie i przetwarzanie ogromnej ilości danych (liczonych już w terabajtach) we wszystkich dziedzinach wiedzy (m.in. w finansach, medycynie, handlu, marketingu, itd…). Tak duża ilość zgromadzonych informacji bez odpowiedniej analizy jest jednak bezużyteczna. Tutaj pomocne okazują się nowe metody naukowe nazywane eksploracją danych (ang. data mining, data exploration) oraz odkrywaniem wiedzy (ang. knowledge discovery). Dzięki nim jesteśmy w stanie pozyskać użyteczną, z określonego punktu widzenia, wiedzę z posiadanych danych. Do procesu odkrywania wiedzy możemy użyć zbiorów przybliżonych, które umożliwiają m.in. indukcję reguł decyzyjnych czy też redukcję zbiorów danych.

1.2. Cel pracy.

Celem pracy jest przedstawienie teorii zbiorów przybliżonych oraz możliwości jej wykorzystania do analizy różnego rodzaju danych. Szczególny nacisk został położony na problem wyznaczania reduktów. Praca prezentuje ogólnie znane i wykorzystywane metody do wyznaczania maksymalnej liczby reduktów oraz minimalnego reduktu tablicy decyzyjnej, jak również nowe propozycje rozwiązywania tych problemów (np. algorytmy oparte na dekompozycji przestrzeni). Ponadto zaproponowałem także metody walidacji danych (wyznaczania współczynników wiarygodności dla poszczególnych obiektów tablicy decyzyjnej) wykorzystujące metodologię zbiorów przybliżonych. Praca zawiera szereg przykładów ilustrujących praktyczne wykorzystanie opisywanych metod. Sprawia to, że może zostać użyta jako materiał dydaktyczny dla osób chcących zapoznać się ze zbiorami przybliżonymi. Całość uzupełnia program ARES Rough Set Exploration System, w którym zaimplementowano opisane algorytmy.

1.3. Zawartość merytoryczna.

W rozdziale numer dwa zostały poruszone teoretyczne aspekty teorii zbiorów przybliżonych, których znajomość jest niezbędna do zrozumienia dalszej części pracy. Rozdział trzeci jest poświęcony ogólnemu spojrzeniu na różne reprezentacje i typy danych. Opisano w nim także podstawowe metody dyskretyzacji danych ciągłych. Rozdział numer cztery został w całości poświecony problemowi wyznaczania reduktów, zaś rozdział piąty wyznaczaniu współczynników wiarygodności dla obiektów. Rozważania na temat obu tych problemów prowadzone są w kontekście tablic decyzyjnych. W rozdziale szóstym został krótko opisany program ARES Rough Set Exploration System, który jest integralną częścią tej pracy. W kolejnym rozdziale został przedstawiony przykład wykorzystania tej aplikacji w procesie analizy danych. Rozdział ósmy jest podsumowaniem całej pracy zaś w dziewiątym rozdziale została zaprezentowana bibliografia.

2. Podstawy teorii zbiorów przybliżonych.

2.1. Wstęp.

Teoria zbiorów przybliżonych została sformułowana przez Zdzisława Pawlaka (biografia m.in. w [2.5], [2.8]) w 1982 roku w [1.21], [1.22]. Jest ona wykorzystywana jako narzędzie do syntezy zaawansowanych i efektywnych metod analizy oraz do redukcji zbiorów danych. Znalazła ona zastosowanie m.in. w eksploracji danych i odkrywaniu wiedzy, złożonych zadaniach klasyfikacji oraz w komputerowych systemach wspomagania decyzji. Przykłady konkretnego jej wykorzystania zostały przedstawione w [1.20] oraz [1.24].

Metodologia zbiorów przybliżonych zyskała sobie dużą popularność. Jest ona przedmiotem badań wielu osób na całym świecie. Poświęcono jej przeszło 2000 publikacji, w tym kilkanaście książek. Cyklicznie odbywają się na jej temat międzynarodowe konferencje i seminaria (m.in. w USA, Kanadzie i Japonii).

W kolejnych podrozdziałach zostały opisane charakterystyczne pojęcia związane z teorią zbiorów przybliżonych. Ich znajomość jest niezbędna do zrozumienia kolejnych rozdziałów pracy.

2.2. System informacyjny.

Istnieje szereg struktur, które mogą być wykorzystane do przechowywania danych. Sposób reprezentacji danych powinien jednak posiadać dwie podstawowe cechy: uniwersalność (powinien pozwalać na gromadzenie i przechowywanie zbiorów różnorodnych danych, opisujących badane zjawiska i procesy) oraz efektywność (powinien umożliwiać w łatwy sposób komputerową analizę tak zapisanych danych). Obie te cechy posiada znany i często wykorzystywany w praktyce tablicowy sposób reprezentacji danych. W tym podejściu zbiór danych przedstawiany jest w postaci tablicy, której kolumny są etykietowane przez atrybuty (parametry, własności, cechy), wiersze odpowiadają zaś obiektom (elementom, sytuacjom, stanom), a na przecięciu wierszy i kolumn znajdują się wartości odpowiednich atrybutów dla

poszczególnych obiektów. Tak zdefiniowaną strukturę nazywamy ***systemem informacyjnym (SI)*** (ang. information system) rzadziej zaś tablicą informacyjną lub tablicą typu atrybut-wartość. Formalnie systemem informacyjnym nazywamy uporządkowaną czwórkę:

$$SI = (U, A, V, f) \tag{2.1}$$

gdzie:

- U jest niepustym, skończonym zbiorem zwanym uniwersum, przy czym elementy zbioru U nazywamy obiektami
- A jest niepustym, skończonym zbiorem atrybutów
- $V = \bigcup_{a \in A} V_a$, przy czym V_a nazywamy dziedziną atrybutu $a \in A$
- $f : U \times A \to V$ jest funkcją informacji taką, że $\forall_{\substack{x \in U \\ a \in A}} f(x,a) \in V_a$

Przykład 2.1.

Pacjent	Ból głowy (g)	Ból mięsni (m)	Temperatura (t)	Grypa (c)
1	nie	tak	wysoka	tak
2	tak	nie	wysoka	tak
3	tak	tak	bardzo wysoka	tak
4	nie	tak	bardzo wysoka	tak
5	tak	nie	wysoka	nie
6	nie	tak	normalna	nie

Tabela 2.1. System informacyjny / tablica decyzyjna.

Tabela 2.1 przedstawia przykładowy system informacyjny zawierający wyniki badań przeprowadzonych dla grupy pacjentów ([2.1]). System ten składa się z sześciu obiektów (1, 2, .., 6) oraz czterech atrybutów (Ból głowy, Ból mięśni, Temperatura, Grypa). W dalszej części pracy kolejne atrybutu będą też oznaczane za pomocą liter: g, m, t, c. Zgodnie z definicją 2.1 rozpatrywany system informacyjny może zostać zapisany w następującej postaci: $SI = (U, A, V, f)$, gdzie:

- $U=\{1, 2, 3, 4, 5, 6\}$
- $A=\{Ból\ głowy, Ból\ mięśni, Temperatura, Grypa\}$
- $V= V_{Ból\ głowy} \cup V_{Ból\ mięśni} \cup V_{Temperatura} \cup V_{Grypa}$

  $V_{Ból\ głowy}=\{nie, tak\}$

$V_{Ból\ mięśni}=\{nie, tak\}$

$V_{Temperatura}=\{normalna, wysoka, bardzo\ wysoka\}$

$V_{Grypa}=\{nie, tak\}$

- $f : U \times A \rightarrow V$ *(np. f(1, Ból głowy)=nie; f(3, Grypa) = tak)*

2.2.1. Relacja nierozróżnialności.

Analizując poszczególne obiekty z tabeli 2.1 można zaobserwować, że obiekty o numerach 1, 4 i 6 mają te same wartości atrybutów: *ból głowy* oraz *ból mięśni* zaś obiekty o numerach 1 i 5 mają tę samą wartość atrybutu *temperatura*. O obiektach numer 1, 4 i 6 powiemy, że są nierozróżnialne ze względu na atrybuty: *ból głowy* oraz *ból mięśni*, zaś obiekty o numerach 1 i 5 są nierozróżnialne ze względu na atrybut: *temperatura*. Tę obserwację można uogólnić i wyrazić w sposób formalny stosując odpowiednio zdefiniowaną relację.

Niech $SI = (U, A, V, f)$ będzie systemem informacyjnym i niech $B \subseteq A$. ***Relację nierozróżnialności*** (ang. indiscernibility relation) na zbiorze obiektów U generowaną przez zbiór atrybutów B określamy jako:

$$IND_{SI}(B) = \{(x,y) \in U \times U : \underset{a \in B}{\forall} f(x,a) = f(y,a)\} \tag{2.2}$$

Poszczególne pary obiektów należą do relacji wtedy, gdy posiadają te same wartości dla wszystkich atrybutów ze zbioru B.

Relacja nierozróżnialności $IND_{SI}(B)$ jest relacją równoważności ([1.18]), gdyż jest relacją:

- zwrotną, gdyż:

  $\underset{u \in U}{\forall} (u,u) \in IND_{SI}(B)$

- symetryczną, gdyż:

  $\underset{u,v \in U}{\forall} \left((u,v) \in IND_{SI}(B) \Rightarrow (v,u) \in IND_{SI}(B)\right)$

- przechodnią, gdyż:

  $\underset{u,v,w \in U}{\forall} \left((u,v) \in IND_{SI}(B) \wedge (v,w) \in IND_{SI}(B) \Rightarrow (u,w) \in IND_{SI}(B)\right)$

Każda relacja równoważności dzieli zbiór, w którym jest określona, na rodzinę rozłącznych podzbiorów zwanych klasami abstrakcji (równoważności) lub zbiorami elementarnymi tej relacji. Klasa abstrakcji elementu $y \in X$ względem relacji równoważności R w zbiorze X to zbiór elementów $x \in X$, które są w relacji R z y ([2.9]).

Dla danej relacji nierozróżnialności $IND_{SI}(B)$ rodzinę wszystkich klas abstrakcji tej relacji oznacza się przez: $U/IND_{SI}(B)$. Poszczególne klasy nazywamy ***zbiorami B – elementarnymi***, zaś przez $I_{SI,B}(x)$ oznaczamy klasę tej relacji zawierającą obiekt x. Formalnie $I_{SI,B}(x)$ (oznaczane również: $[x]_{IND_{SI}(B)}$) można zdefiniować jako:

$$I_{SI,B}(x) = \{y \in U \mid (x,y) \in IND_{SI}(B)\} \tag{2.3}$$

Wszystkie elementy każdego zbioru B – elementarnego mają te same wartości wszystkich atrybutów należących do zbioru B (są nierozróżnialne względem tych atrybutów).
Zbiór $I_{SI,B}(x)$ zawiera zaś te wszystkie obiekty systemu informacyjnego SI, które są nierozróżnialne z obiektem x względem zbioru atrybutów B (mają te same wartości dla wszystkich atrybutów ze zbioru B).

Przykład 2.2.

Dla systemu informacyjnego przedstawionego w tabeli 2.1 można wyznaczyć relacje nierozróżnialności generowane przez różne zbiory atrybutów.
Niech:

$A_1=\{g, m, t\}$, $A_2=\{t\}$, $A_3=\{g, m\}$, $A_4=\{g, t, c\}$, $A=\{g, m, t, c\}$

$IND_{SI}(A_1) = \{ (1,1), (2,2), (3,3), (4,4), (5,5), (6,6), (2,5), (5,2) \}$

$IND_{SI}(A_2) = \{ (1,1), (2,2), (3,3), (4,4), (5,5), (6,6), (1,2), (2,1),$
$(1,5), (5,1), (2,5), (5,2), (3,4), (4,3) \}$

$IND_{SI}(A_3) = \{ (1,1), (2,2), (3,3), (4,4), (5,5), (6,6), (1,4), (4,1),$
$(1,6), (6,1), (4,6), (6,4), (2,5), (5,2) \}$

$IND_{SI}(A_4) = \{ (1,1), (2,2), (3,3), (4,4), (5,5), (6,6) \}$

$IND_{SI}(A) = IND_{SI}(A_4)$

Powyższe relacje dzielę zbiór obiektów systemu informacyjnego na następujące klasy abstrakcji (zbiory elementarne):

$$U / IND_{SI}(A_1) = \{ \{1\}, \{2,5\}, \{3\}, \{4\}, \{6\} \}$$

$$U / IND_{SI}(A_2) = \{ \{1,2,5\}, \{3,4\}, \{6\} \}$$

$$U / IND_{SI}(A_3) = \{ \{1,4,6\}, \{2,5\}, \{3\} \}$$

$$U / IND_{SI}(A_4) = \{ \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\} \}$$

$$U / IND_{SI}(A) = U / IND_{SI}(A_4)$$

Na tej podstawie można wyznaczyć przykładowe klasy abstrakcji zawierające poszczególne obiekty systemu informacyjnego:

$$I_{SI,A_3}(1) = \{1,4,6\}$$

$$I_{SI,A_3}(2) = \{2,5\}$$

$$I_{SI,A_3}(3) = \{3\}$$

2.2.2. Zbiór dokładny oraz zbiór przybliżony.

W celu zdefiniowania pojęcia zbioru przybliżonego i dokładnego wykorzystam relację nierozróżnialności. Takie podejście jest bardzo popularne, lecz warto podkreślić, że istnieje szereg uogólnień teorii zbiorów przybliżonych, w których punktem wyjścia nie jest relacja nierozróżnialności lecz np. relacja tolerancji lub relacja porządku.

Niech $SI = (U, A, V, f)$ będzie systemem informacyjnym i niech $B \subseteq A$. Mówimy, że zbiór $P \subseteq U$ jest ***zbiorem B – dokładnym*** (B – definiowalnym) wtedy, gdy jest on skończoną sumą zbiorów B – elementarnych. Każdy zbiór, który nie jest skończoną sumą zbiorów B – elementarnych jest ***zbiorem B – przybliżonym***.

Przykład 2.3.

Kontynuując przykład z poprzedniego podrozdziału niech:

$X_1 = \{1, 2, 3, 5\}$, $X_2 = \{3, 4, 5, 6\}$

Możemy stwierdzić, że:

- Zbiór X_1 jest zbiorem A_1 – dokładnym, gdyż jest skończoną sumą zbiorów A_1 – elementarnych: $X_1=\{\{1\} \cup \{2,5\} \cup \{3\}\}$
- Zbiór X_2 jest zbiorem A_1 – przybliżonym, gdyż nie jest skończoną sumą zbiorów A_1 – elementarnych (obiekty 2 i 5 należą do jednego zbioru B – elementarnego, zaś zbiór X_2 zawiera tylko obiekt numer 5, a nie zawiera obiektu numer 2)
- Zbiór X_1 jest zbiorem A_2 – przybliżonym, gdyż nie jest skończoną sumą zbiorów A_2 – elementarnych (obiekty 3 i 4 należą do jednego zbioru C – elementarnego, zaś zbiór X_1 zawiera tylko obiekt numer 3, a nie zawiera obiektu numer 4)
- Zbiór X_2 jest zbiorem A_2 – przybliżonym, gdyż nie jest skończoną sumą zbiorów A_2 – elementarnych (obiekty 1, 2 i 5 należą do jednego zbioru C – elementarnego, zaś zbiór X_2 zawiera tylko obiekt numer 5, a nie zawiera obiektów numer 1 i 2)

2.2.3. Aproksymacja zbioru.

Jeśli $SI = (U, A, V, f)$ jest systemem informacyjnym takim, że $B \subseteq A$ oraz $X \subseteq U$ to:

- ***B – dolnym przybliżeniem*** (aproksymacją) zbioru X w systemie informacyjnym SI nazywamy zbiór:

$$\underline{B}X = \{x \in U : I_{SI,B}(x) \subseteq X\} \tag{2.4}$$

- ***B – górnym przybliżeniem*** (aproksymacją) zbioru X w systemie informacyjnym SI nazywamy zbiór:

$$\overline{B}X = \{x \in U : I_{SI,B}(x) \cap X \neq \emptyset\} \tag{2.5}$$

- ***B – pozytywnym obszarem*** (ang. positive area) zbioru X w systemie informacyjnym SI nazywamy zbiór:

$$POS_B(X) = \underline{B}X \tag{2.6}$$

- ***B – brzegiem*** (granicą) (ang. boundary) zbioru X w systemie informacyjnym SI nazywamy zbiór:

$$BN_B(X) = \overline{B}X - \underline{B}X \tag{2.7}$$

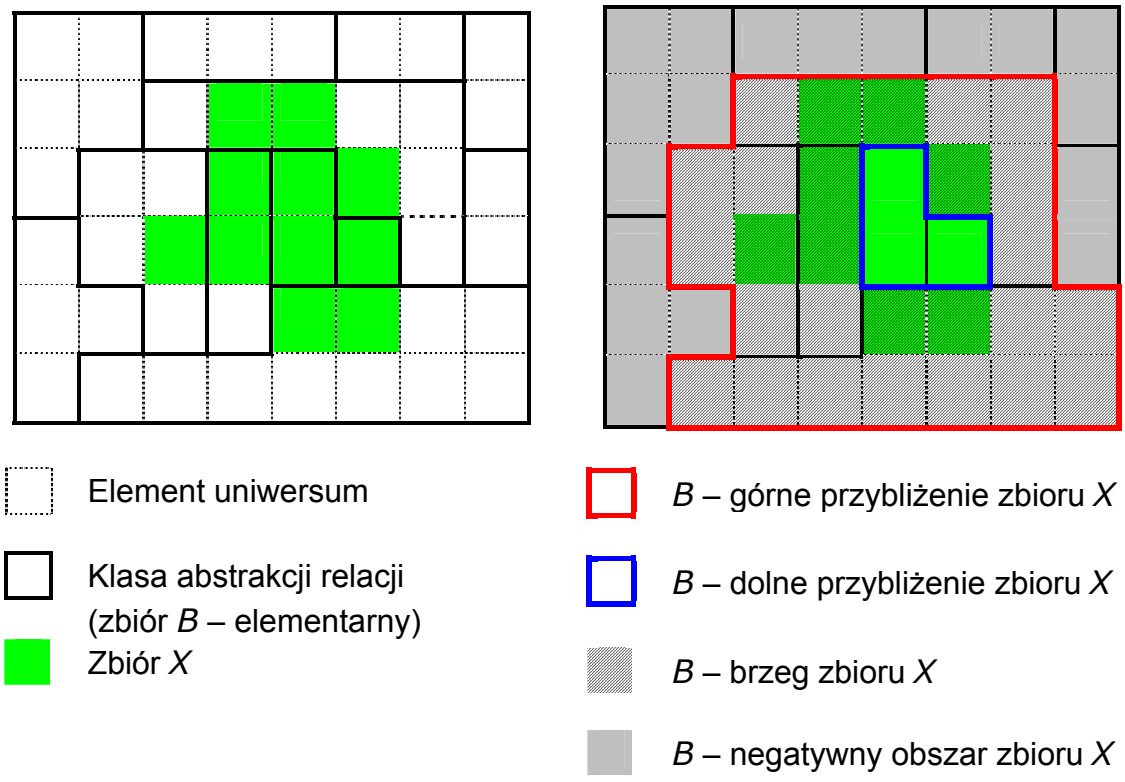
- ***B – negatywnym obszarem*** (ang. negative area) zbioru X w systemie informacyjnym SI nazywamy zbiór:

$$NEG_B(X) = U - \overline{B}X \tag{2.8}$$

Z definicji (2.4 – 2.8) możemy wysnuć następujące wnioski:

- $\underline{B}X \subseteq X \subseteq \overline{B}X$
- zbiór X jest B – dokładny, gdy: $\underline{B}X = \overline{B}X \Leftrightarrow BN_B(X) = \emptyset$
- zbiór X jest B – przybliżony, gdy: $\underline{B}X \neq \overline{B}X \Leftrightarrow BN_B(X) \neq \emptyset$

W dalszej części przedstawię intuicyjną oraz graficzną interpretację powyższych pojęć.



Rysunek 2.1. Graficzna interpretacja pojęć związanych z aproksymacją zbioru.

Obszar pozytywny zbioru ($POS_B(X)$), jest zbiorem tych elementów uniwersum U, które na pewno mogą być zidentyfikowane jako elementy zbioru X przy wykorzystaniu wartości atrybutów ze zbioru B.

Brzeg zbioru ($BN_B(X)$) jest zbiorem tych elementów uniwersum U, które być może mogą być zidentyfikowane jako elementy X przy wykorzystaniu wartości atrybutów ze zbioru B.
Obszar negatywny zbioru ($NEG_B(X)$) jest zbiorem tych elementów uniwersum U, które nie mogą być zidentyfikowane jako elementy X przy wykorzystaniu wartości atrybutów ze zbioru B.

Przykład 2.4.

Kontynuując przykład z poprzedniego podrozdziału wyznaczę aproksymacje dla przykładowych zbiorów obiektów:

$\underline{A_1}X_1 = \{1,2,3,5\}$

$\overline{A_1}X_1 = \{1,2,3,5\}$

$POS_{A_1}(X_1) = \underline{A_1}X_1 = \{1,2,3,5\}$

$BN_{A_1}(X_1) = \overline{A_1}X_1 - \underline{A_1}X_1 = \emptyset$ (potwierdzenie faktu, że zbiór X_1 jest zbiorem A_1 – dokładnym)

$NEG_{A_1}(X_1) = U - \overline{A_1}X_1 = \{4,6\}$

$\underline{A_1}X_2 = \{3,4,6\}$

$\overline{A_1}X_2 = \{2,3,4,5,6\}$

$POS_{A_1}(X_2) = \underline{A_1}X_2 = \{3,4,6\}$

$BN_{A_1}(X_2) = \overline{A_1}X_2 - \underline{A_1}X_2 = \{2,5\}$ (potwierdzenie faktu, że zbiór X_2 jest zbiorem A_1 – przybliżonym)

$NEG_{A_1}(X_1) = U - \overline{A_1}X_1 = \{1\}$

2.2.4. Liczbowa charakterystyka aproksymacji zbioru.

Każdy zbiór (przybliżony lub dokładny) można scharakteryzować ilościowo za pomocą współczynnika dokładności aproksymacji (przybliżenia). Współczynnik dokładności

aproksymacji zbioru X w systemie informacyjnym SI względem zbioru atrybutów B wyraża się wzorem:

$$\alpha_B(X) = \frac{card(POS_B(X))}{card(\overline{B}X)} = \frac{card(\underline{B}X)}{card(\overline{B}X)} \tag{2.9}$$

gdzie $card(X)$ oznacza liczność zbioru X.

Łatwo zauważyć, że:

- $0 \leq \alpha_B(X) \leq 1$
- jeżeli X jest zbiorem dokładnym to: $\alpha_B(X) = 1$
- jeżeli X jest zbiorem przybliżonym to: $0 \leq \alpha_B(X) < 1$

Przykład 2.5.

Kontynuując przykład z poprzedniego podrozdziału policzymy dokładność aproksymacji dla zbiorów X_1 oraz X_2 względem zbioru atrybutów A_1:

$$\alpha_{A_1}(X_1) = \frac{card(\underline{A_1}X_1)}{card(\overline{A_1}X_1)} = \frac{4}{4} = 1$$

$$\alpha_{A_1}(X_2) = \frac{card(\underline{A_1}X_2)}{card(\overline{A_1}X_2)} = \frac{3}{5}$$

2.2.5. Klasyfikacja zbiorów przybliżonych.

Niech $X \subseteq U$ będzie zbiorem B – przybliżonym. Taki zbiór może należeć do jednej z czterech klas zbiorów przybliżonych:

- klasy zbiorów w ***przybliżeniu B – definiowalnych***, gdy

$$\underline{B}X \neq \emptyset \wedge \overline{B}X \neq U \tag{2.10}$$

- klasy zbiorów ***wewnętrznie B – niedefiniowalnych***, gdy

$$\underline{B}X = \emptyset \wedge \overline{B}X \neq U \tag{2.11}$$

- klasy zbiorów ***zewnętrznie B – niedefiniowalnych***, gdy

$$\underline{B}X \neq \emptyset \wedge \overline{B}X = U \tag{2.12}$$

- klasy zbiorów ***całkowicie B – niedefiniowalnych***, gdy

$$\underline{B}X = \emptyset \wedge \overline{B}X = U \tag{2.13}$$

Intuicyjna interpretacja powyższej klasyfikacji jest następująca:

- jeżeli zbiór X jest w przybliżeniu B – definiowalny, to dla niektórych elementów z U można jednoznacznie rozstrzygnąć, wykorzystując relację nierozróżnialności $IND_{SI}(B)$, czy należą one do X czy do jego dopełnienia $\sim X$
- jeżeli zbiór X jest wewnętrznie B – niedefiniowalny, to można jednoznacznie rozstrzygnąć, wykorzystując relację nierozróżnialności $IND_{SI}(B)$, czy pewne elementy U należą do $\sim X$, natomiast nie można jednoznacznie rozstrzygnąć czy jakiś element U należy do X
- jeżeli zbiór X jest zewnętrznie B – niedefiniowalny, to można wykorzystując relacje nierozróżnialności, jednoznacznie rozstrzygnąć, czy niektóre elementy U należą do X, natomiast nie można rozstrzygnąć dla dowolnego elementu U jego przynależności do $\sim X$
- jeżeli zbiór X jest całkowicie B – niedefiniowalny, to nie można jednoznacznie rozstrzygnąć, wykorzystując relacje nierozróżnialności, o żadnym elemencie U, czy należy on do X czy do $\sim X$

Przykład 2.6.

Sklasyfikujemy teraz zbiór z poprzedniego przykładu:

- zbiór X_2 jest w przybliżeniu A_1 – definiowalny, bo:

  $\underline{A_1}X_2 = \{3,4,6\} \neq \emptyset \wedge \overline{A_1}X_2 = \{2,3,4,5,6\} \neq U$

2.2.6. Aproksymacja rodziny zbiorów.

Problem aproksymacji pojedynczego zbioru obiektów można uogólnić na problem aproksymacji rodziny zbiorów. Niech: $F = \{X_1, X_2, X_3, .., X_n\}$ będzie rodziną podzbiorów U ($\underset{i \in 1..n}{\forall} X_i \subseteq U$) pewnego systemu informacyjnego $SI = (U, A, V, f)$ oraz niech $B \subseteq A$:

- ***B – dolnym przybliżeniem*** (aproksymacją) rodziny zbiorów F w systemie informacyjnym SI nazywamy zbiór:

$$\underline{B}F = \{\underline{B}X_1, \underline{B}X_2, \underline{B}X_3, .., \underline{B}X_n\} \tag{2.14}$$

- ***B – górnym przybliżeniem*** (aproksymacją) rodziny zbiorów F w systemie informacyjnym SI nazywamy zbiór:

$$\overline{B}F = \{\overline{B}X_1, \overline{B}X_2, \overline{B}X_3, .., \overline{B}X_n\} \tag{2.15}$$

- ***B – pozytywnym obszarem*** rodziny zbiorów F w systemie informacyjnym SI nazywamy zbiór:

$$POS_B(F) = \bigcup_{X_i \in F} POS_B(X_i) = \bigcup_{X_i \in F} \underline{B}X_i \tag{2.16}$$

- ***B – brzegiem*** (granicą) rodziny zbiorów F w systemie informacyjnym SI nazywamy zbiór:

$$BN_B(F) = \bigcup_{X_i \in F} BN_B(X_i) = \bigcup_{X_i \in F} (\overline{B}X_i - \underline{B}X_i) \tag{2.17}$$

- ***B – negatywnym obszarem*** rodziny zbiorów F w systemie informacyjnym SI nazywamy zbiór:

$$NEG_B(F) = \bigcup_{X_i \in F} NEG_B(X_i) = U - \bigcup_{X_i \in F} \overline{B}X_i \tag{2.18}$$

Przykład 2.7.

Kontynuując przykład z poprzedniego podrozdziału policzmy teraz aproksymację dla rodziny zbiorów Z.

$Z = \{X_1, X_2\}$

$\underline{A_1}Z = \{\underline{A_1}X_1, \underline{A_1}X_2\} = \{\{1,2,3,5\}, \{3,4,6\}\}$

$\overline{A_1}Z = \{\overline{A_1}X_1, \overline{A_1}X_2\} = \{\{1,2,3,5\}, \{2,3,4,5,6\}\}$

$POS_{A_1}(Z) = POS_{A_1}(X_1) \cup POS_{A_1}(X_2) = \{1,2,3,5\} \cup \{3,4,6\} = \{1,2,3,4,5,6\}$

$BN_{A_1}(Z) = BN_{A_1}(X_1) \cup BN_{A_1}(X_2) = \emptyset \cup \{2,5\} = \{2,5\}$

$NEG_{A_1}(Z) = U - (\overline{A_1}X_1 \cup \overline{A_1}X_2) = \{1,2,3,4,5,6\} - (\{1,2,3,5\} \cup \{2,3,4,5,6\}) = \emptyset$

2.2.7. Liczbowe charakterystyki aproksymacji rodziny zbiorów.

Aproksymację rodzin zbiorów, podobnie jak aproksymację pojedynczych zbiorów, można scharakteryzować liczbowo przy użyciu pewnych współczynników. W przypadku rodzin zbiorów mamy do dyspozycji następujące miary:

- jakość aproksymacji rodziny F w systemie informacyjnym SI względem zbioru atrybutów B:

$$\gamma_B(F) = \frac{card(POS_B(F))}{card(U)} \tag{2.19}$$

- dokładność aproksymacji rodziny F w systemie informacyjnym SI względem zbioru atrybutów B:

$$\alpha_B(F) = \frac{card(POS_B(F))}{\sum_{X_k \in F} card(\overline{B}X_k)} \tag{2.20}$$

Dla powyższych współczynników zachodzą następujące prawidłowości:

- $0 \le \alpha_B(F) \le \gamma_B(F) \le 1$
- jeżeli wszystkie elementy rodziny F są zbiorami B – dokładnymi to: $\alpha_B(F) = \gamma_B(F) = 1$

Przykład 2.8.

Posługując się wynikami z poprzedniego przykładu można wyznaczyć jakość oraz dokładność aproksymacji dla rodziny zbiorów Z:

$$\gamma_{A_1}(Z) = \frac{card(POS_{A_1}(Z))}{card(U)} = \frac{6}{6} = 1$$

$$\alpha_{A_1}(Z) = \frac{card(POS_{A_1}(Z))}{\sum_{X_k \in F} card(\overline{A_1}X_k)} = \frac{6}{9} = \frac{2}{3}$$

2.2.8. Macierz, tablica, funkcja oraz wektor odróżnialności dla systemu informacyjnego.

Jeśli $SI = (U, A, V, f)$ jest systemem informacyjnym takim, że $U = \{u_1, .., u_n\}$ i $A = \{a_1, .., a_m\}$, to ***macierz odróżnialności (rozróżnialności) systemu informacyjnego SI M(SI)*** (ang. discernibility matrix) definiujemy następująco:

$$M(SI) = (H_{i,j})_{i,j=1,..,n} = \{a \in A : f(u_i, a) \neq f(u_j, a)\}, \text{ dla } i, j = 1, .., n$$
$$\text{gdzie : } n = |U| \tag{2.21}$$

Macierz odróżnialności jest dwuwymiarową macierzą kwadratową o wymiarach: $|U| \times |U|$. Komórka $M(SI)[i,j]$ zawiera zbiór tych atrybutów, dla których obiekty uniwersum u_i i u_j mają różne wartości (są rozróżnialne przy pomocy tych atrybutów).

Przykład 2.9.

Tabela 2.2 przedstawia macierz odróżnialności dla systemu informacyjnego z tabeli 2.1.

U\U	1	2	3	4	5	6
1	{Ø}	{g,m}	{g,t}	{t}	{g,m,c}	{t,c}
2	{g,m}	{Ø}	{m,t}	{g,m,t}	{c}	{g,m,t,c}
3	{g,t}	{m,t}	{Ø}	{g}	{m,t,c}	{g,t,c}
4	{t}	{g,m,t}	{g}	{Ø}	{g,m,t,c}	{t,c}
5	{g,m,c}	{c}	{m,t,c}	{g,m,t,c}	{Ø}	{g,m,t}
6	{t,c}	{g,m,t,c}	{g,t,c}	{t,c}	{g,m,t}	{Ø}

Tabela 2.2. Macierz odróżnialności dla system informacyjnego.

Własności macierzy odróżnialności:

- macierz $M(SI)$ ma zawsze na przekątnej zbiory puste $(Ø)$
- macierz $M(SI)$ jest symetryczna względem przekątnej
- każdy element macierzy $M(SI)$ jest zbiorem
- rozmiar macierzy rośnie w sposób kwadratowy wraz ze wzrostem liczby obiektów w systemie informacyjnym

Powyższe cechy sprawiają, że taka reprezentacja macierzy, jest bardzo niewygodna z programistycznego punktu widzenia. Macierz zawiera redundantne informacje, zawartości komórek nie są typami prostymi a ponadto nie mają stałej wielkości (liczby elementów w zbiorze). W efekcie struktura ta ma bardzo dużą złożoność pamięciową, która dla systemu informacyjnego $SI = (U, A, V, f)$ wynosi: $|U|^2 * |A|$.

Macierz odróżnialności, ze względu na swoje ograniczenia, nie jest szeroko używana w rzeczywistych zastosowaniach. Zamiast niej można wykorzystywać ***tablicę odróżnialności T(SI)*** (ang. discernibility table), którą dla systemu informacyjnego $SI = (U, A, V, f)$ określa się następująco:

$$T(SI) = T[(i,j),k] = \begin{cases} 0, f(u_i, a_k) = f(u_j, a_k) \\ 1, f(u_i, a_k) \neq f(u_j, a_k) \end{cases} \qquad (2.21)$$

$$\text{gdzie}: i, j = 1,..,|U|; j > i; k = 1,..,|A|$$

W stosunku do macierzy odróżnialności tablica:

- nie zawiera redundantnych informacji o tych samych parach obiektów
- jest typową dwuwymiarową strukturą o stałych wymiarach
- poszczególne elementy tablicy mają wartość: 0 lub 1

Cechy te sprawiają, że poszczególne wiersze tabeli mogą być reprezentowana w postaci bitowej (każda komórka to jeden bit) z użyciem tzw. zbiorów bitowych (ang. BitSet, typ którego implementacja jest dostępna w większości języków programowania (np. JAVA, C++)), co zapewni optymalne zużycie pamięci.

Przykład 2.10.

Tabela 2.3 zawiera tablicę odróżnialności, dla systemu informacyjnego z tabeli 2.1, która jest równoważna macierzy odróżnialności z tabeli 2.2.

U	U	g	m	t	c
1	**2**	1	1	0	0
1	**3**	1	0	1	0
1	**4**	0	0	1	0
1	**5**	1	1	0	1
1	**6**	0	0	1	1
2	**3**	0	1	1	0
2	**4**	1	1	1	0
2	**5**	0	0	0	1
2	**6**	1	1	1	1
3	**4**	1	0	0	0
3	**5**	0	1	1	1
3	**6**	1	0	1	1
4	**5**	1	1	1	1
4	**6**	0	0	1	1
5	**6**	1	1	1	0

Tabela 2.3. Tablica odróżnialności dla systemu informacyjnego.

Wprowadźmy jeszcze pojęcie ***wektora odróżnialności dla systemu informacyjnego SI***. Wektor ten dla systemu $SI = (U, A, V, f)$ określa się następująco:

$$W(SI) = W(SI)[n * (i-1) + j] = M(SI)[i, j]$$
$$\text{gdzie : } i, j = 1,..,n; j > i; \quad n = |U| \tag{2.22}$$

Wektor odróżnialności jest więc wektorem zbiorów atrybutów zawierającym wszystkie elementy jednej części macierzy odróżnialności (tej poniżej przekątnej). Poprzez ***optymalną postać wektora odróżnialności*** rozumiemy wektor odróżnialności, który spełnia następujące warunki:

1. $\underset{X \in W(SI)}{\forall} X \neq \emptyset$
2. $\underset{X,Y \in W(SI)}{\forall} X \neq Y$ (2.23)
3. $\underset{X,Y \in W(SI)}{\forall} X \not\subset Y \wedge Y \not\subset X$

Optymalna postać wektora odróżnialności zawiera więc tylko elementy niepuste oraz takie, które nie są nadzbiorami innych elementów z wektora.

Wektor w optymalnej postaci będzie także nazywany ***zbiorem odróżnialności systemu informacyjnego*** i oznaczany symbolem $Z(SI)$.

Przykład 2.11.

Wyznaczmy wektor odróżnialności i jego optymalną postać dla systemu informacyjnego z tabeli 2.1 (korzystając z macierzy odróżnialności z tabeli 2.2):

$W(SI)=$ $(\{g,m\},\ \{g,t\},\ \{t\},\ \{g,m,c\},\ \{t,c\},\ \{m,t\},\ \{g,m,t\},\ \{c\},$
$\{g,m,t,c\},\ \{g\},\ \{m,t,c\},\ \{g,t,c\},\ \{g,m,t,c\},\ \{t,c\},\ \{g,m,t\})$

$Z(SI)=$ $(\{t\},\ \{c\},\ \{g\})$

Wiedzę zawartą w macierzy odróżnialności (tablicy odróżnialności) można także przedstawić w postaci funkcji odróżnialności. ***Funkcją odróżnialności systemu informacyjnego SI*** (ang. discernibility function) nazywamy funkcję boolowską f_{SI} zmiennych $a_1^*,..,a_m^*$ odpowiadających odpowiednio atrybutom (systemu informacyjnego) $a_1,..,a_m$ zdefiniowaną następująco:

$$f_{SI}(a_1^*,..,a_m^*) = \bigcap\{\bigcup(H_{i,j} : 1 \le j < i \le n \wedge H_{i,j} \neq \emptyset)\} \qquad (2.24)$$

gdzie:

- $n=|U|,\ m=|A|$
- $\bigcup(H_{i,j})$ jest alternatywą wszystkich zmiennych $a^* \in \{a_1^*,..,a_m^*\}$ takich, że $a \in H_{i,j}$

Przykład 2.12.

Obliczmy funkcję odróżnialności dla macierzy odróżnialności z tabeli 2.2:

$$\begin{aligned} f_{SI}(g^*,m^*,t^*,c^*) = {} & (g^* \vee m^*) \wedge (g^* \vee t^*) \wedge (t^*) \wedge (g^* \vee m^* \vee c^*) \wedge (t^* \vee c^*) \wedge \\ & (m^* \vee t^*) \wedge (g^* \vee m^* \vee t^*) \wedge (c^*) \wedge (g^* \vee m^* \vee t^* \vee c^*) \wedge \\ & (g^*) \wedge (m^* \vee t^* \vee c^*) \wedge (g^* \vee t^* \vee c^*) \wedge (g^* \vee m^* \vee t^* \vee c^*) \wedge \\ & (t^* \vee c^*) \wedge (g^* \vee m^* \vee t^*) \end{aligned}$$

Wyrażenie to można uprościć stosując m.in. prawo pochłaniania ($a \wedge (a \vee b) = a$) do postaci:

$$f_{SI}(g^*,m^*,t^*,c^*) = t^* \wedge g^* \wedge c^*$$

2.2.9. Zbiór pokrywający dla systemu informacyjnego.

Niech $SI = (U, A, V, f)$ będzie systemem informacyjnym i $U = \{u_1, .., u_n\}$ oraz niech $B \subseteq A$. Oznaczmy przez $CS(B)$ zbiór tych wszystkich par obiektów systemu informacyjnego SI, które mają różne wartości dla co najmniej jednego z atrybutów ze zbioru B (są rozróżnialne względem tego atrybutu). Formalnie $CS(B)$ będziemy nazywać ***zbiorem par obiektów systemu informacyjnego pokrytych przez zbiór atrybutów B***[1] i definiować jako:

$$CS(B) = \{(u_i, u_j) : i, j = 1, .., n \wedge j > i \wedge \underset{a \in A}{\exists}(a \in B \wedge a \in M(SI)[i, j])\} \qquad (2.25)$$

Zbiór pokrytych obiektów jest nierozerwalnie związany z tablicą oraz wektorem odróżnialności. $CS(B)$ może być interpretowany jako zbiór numerów tych wierszy tablicy odróżnialności, które w co najmniej jednej z kolumn, odpowiadających atrybutom z rozpatrywanego zbioru B, mają wartość 1. $CS(B)$ można także interpretować jako zbiór tych elementów wektora odróżnialności $W(SI)$, które mają niepuste przecięcie ze zbiorem atrybutów B. Należy mieć jednak świadomość, że w tej interpretacji elementami zbioru $CS(B)$ nie będą pary obiektów typu (u_i, u_j), tylko zbiory atrybutów względem których oba te obiekty są rozróżnialne. Niemniej jednak, w kontekście tej pracy, można używać zamiennie wszystkich tych trzech interpretacji pod warunkiem, że konsekwentnie, cały czas będziemy korzystać z tej samej. Jest to możliwe, gdyż w dalszej części będziemy się raczej skupiać na mocy zbioru $CS(B)$ niż na jego pojedynczych elementach.

Przez $|CS(B)|$ będziemy rozumieć ilość par obiektów systemu informacyjnego SI, które mają różne wartości dla co najmniej jednego z atrybutów ze zbioru B (są rozróżnialne względem tego atrybutu) .

Mówimy, że zbiór $B \subseteq A$ jest ***zbiorem odróżniającym (rozróżniającym) obiekty systemu informacyjnego*** wtedy, gdy spełnia on następującą równość:

$$CS(B) = CS(A) \Leftrightarrow |CS(B)| = |CS(A)| \qquad (2.26)$$

Zbiór odróżniający obiekty systemu informacyjnego jest tzw. zbiorem trafieniowym (ang. hitting set [1.16], [1.17], [2.4]) dla wektora odróżnialności.

[1] Dla *CS(B)* zbiór atrybutów *B* nazywamy zbiorem pokrywającym (ang. cover set) obiekty systemu informacyjnego.

Warto podkreślić, że jeśli system informacyjny zawiera tylko obiekty unikatowe (każdy obiekt różni się od pozostałych przynajmniej wartością jednego atrybutu) to: $|CS(A)| = \frac{|U|^2 - |U|}{2}$

Przykład 2.13.

Policzmy CS oraz $|CS|$ dla przykładowych zbiorów atrybutów systemu informacyjnego z tabeli 2.1. Do obliczeń wykorzystamy macierz odróżnialności dla tego systemu (tabelę 2.2).

$CS(\{g, m, t, c\})=\{ (1,2), (1,3), (1,4), (1,5), (1,6), (2,3), (2,4), (2,5), (2,6), (3,4), (3,5), (3,6), (4,5), (4,6), (5,6) \}$

$|CS(\{g, m, t, c\})|=15$

$CS(\{g, m, t\})=\{ (1,2), (1,3), (1,4), (1,5), (1,6), (2,3), (2,4), (2,6), (3,4), (3,5), (3,6), (4,5), (4,6), (5,6) \}$

$|CS(\{g, m, t\})|=14$

$CS(\{g, t, c\})=\{ (1,2), (1,3), (1,4), (1,5), (1,6), (2,3), (2,4), (2,5), (2,6), (3,4), (3,5), (3,6), (4,5), (4,6), (5,6) \}$

$|CS(\{g, t, c\})|=15$

$CS(\{g\})=\{ (1,2), (1,3), (1,5), (2,4), (2,6), (3,4), (3,6), (4,5), (5,6) \}$

$|CS(\{g\})|=9$

2.2.10. Pojęcie reduktu.

Niech $SI = (U, A, V, f)$ będzie systemem informacyjnym oraz $B \subseteq A$. Atrybut a nazywamy ***zbędnym*** w B, gdy: $IND_{SI}(B) = IND_{SI}(B \setminus \{a\})$[2] w przeciwnym przypadku atrybut a nazywamy ***niezbędnym*** w B.

Zbiór atrybutów B nazywamy ***niezależnym*** w systemie informacyjnym SI, gdy każdy atrybut należący do B jest niezbędny w B, w przeciwnym przypadku zbiór B nazywamy ***zależnym***.

[2] $B \setminus \{a\} \Leftrightarrow B - \{a\}$

Zbiór atrybutów Q ($Q \subseteq B$) nazywamy ***reduktem*** zbioru atrybutów B w systemie informacyjnym SI i oznaczamy $R_{SI}(B)$, gdy:

1. zbiór atrybutów Q jest niezależny
2. $IND_{SI}(B) = IND_{SI}(Q)$ (2.27)

Zbiór wszystkich reduktów zbioru atrybutów B w systemie informacyjnym SI oznaczamy przez $RED_{SI}(B)$.

W celu zdefiniowania reduktu możemy się także posłużyć pojęciem zbioru odróżniającego obiekty systemu informacyjnego. Zbiór Q jest reduktem zbioru atrybutów B w systemie informacyjnym SI, gdy:

$$\begin{cases} CS(Q) = CS(B) \\ \neg \underset{C \subset Q}{\exists} CS(C) = CS(Q) \end{cases} \iff \begin{cases} |CS(Q)| = |CS(B)| \\ \neg \underset{C \subset Q}{\exists} |CS(C)| = |CS(Q)| \end{cases} \quad (2.28)$$

Pierwszy warunek jest równoważny warunkowi równości relacji nierozróżnialności zaś drugi warunkowi niezależności zbioru Q z klasycznej definicji reduktu (definicji 2.27).

W rzeczywistych zastosowaniach (gdy chcemy przeprowadzić redukcję systemu informacyjnego) szukamy reduktów pełnego zbioru atrybutów systemu informacyjnego tzn. $R_{SI}(A)$ dla $SI = (U, A, V, f)$.

Rdzeniem (ang. core) zbioru reduktów $RED_{SI}(B)$ nazywamy zbiór określony wzorem:

$$CORE_{SI}(B) = \bigcap_{R \in RED_{SI}(B)} R \quad (2.29)$$

Rdzeń zbioru reduktów $RED_{IS}(B)$ zawiera wszystkie atrybuty niezbędne w zbiorze B.

Przykład 2.14.

Zbiór wszystkich reduktów zbioru atrybutów $\{g,m,t,c\}$ systemu informacyjnego z tabeli 2.1 wynosi: $RED_{IS}(\{g, m, t, c\})=\{g, t, c\}$[3].

Aby udowodnić, że zbiór $\{g,t,c\}$ jest reduktem należy pokazać, że zachodzą warunki z definicji 2.27:

[3] wynik ten został uzyskany za pomocą jednej z metod wyznaczania reduktów systemu informacyjnego (tablicy decyzyjnej), które zostały opisane w kolejnym rozdziale

- $IND_{SI}(\{g,m,t,c\}) = IND_{SI}(\{g,t,c\})$

  Równość tego warunku została pokazana w jednym z poprzednich przykładów.

- zbiór $\{g,t,c\}$ jest niezależny

  Możemy to pokazać, usuwając z tego zbioru kolejne atrybuty i sprawdzając czy relacja nierozróżnialności względem takiego okrojonego zbioru jest różna od relacji nierozróżnialności względem całego zbioru atrybutów**.** Jeżeli tak będzie, to zbiór $\{g,t,c\}$ będzie reduktem.

Możemy także skorzystać z definicji 2.28 i pokazać, że:

- $|CS(\{g,m,t,c\})| \neq |CS(\{g,t,c\})|$

  Fakt ten został pokazany w przykładzie 2.13.

  $|CS(\{g, m, t, c\})| = |CS(\{g, t, c\})| = 15$

- $\neg \exists_{C \subset \{g,t,c\}} |CS(C)| \neq |CS(\{g,t,c\})|$

  Wystarczy pokazać, że $|CS(C)| \neq |CS(\{g,t,c\})|$ dla każdego dwuelementowego podzbioru zbioru $\{g,t,c\}$:

  $|CS(\{g,\ t\})| = 14$

  $|CS(\{g, c\})| = 13$

  $|CS(\{c, t\})| = 13$

Udowodniliśmy, że istotnie zbiór $\{g,t,c\}$ jest reduktem zbioru atrybutów $\{g,m,t,c\}$ w systemie informacyjnym z tabeli 2.1.

2.3. Tablica decyzyjna.

Szczególnym rodzajem systemów informacyjnych są ***tablice decyzyjne (TD)***. Tablicą decyzyjną nazywamy uporządkowaną piątkę:

$$TD = (U, C, D, V, f) \tag{2.30}$$

gdzie:

- $C, D \subset A$; $C \neq \emptyset, D \neq \emptyset$; $C \cup D = A$; $C \cap D = \emptyset$

- elementy zbioru C nazywamy atrybutami warunkowymi
- elementy zbioru D nazywamy atrybutami decyzyjnymi
- f nazywamy funkcją decyzyjną
- interpretacja U oraz V jest taka sama jak w przypadku systemu informacyjnego (definicja 2.1), ponadto poszczególne wartości v dziedzin atrybutów D ($v \in V_D$) będziemy nazywać ***klasami decyzyjnymi***

Podstawowa różnica między tablicą decyzyjną a systemem informacyjnym polega więc na tym, że część atrybutów traktujemy jako atrybuty warunkowe (C) a część jako decyzyjne (D).

Przykład 2.15.

W dalszej części rozważań tabelę 2.1 będziemy traktować jako tablicę decyzyjną. Zbiór atrybutów systemu informacyjnego dzielimy na dwa podzbiory: podzbiór atrybutów warunkowych (C) oraz podzbiór atrybutów decyzyjnych (D) w następujący sposób:

- $C=\{Ból\ głowy,\ Ból\ mięśni,\ Temperatura\}=\{g,\ m,\ t\}$
- $D=\{Grypa\ \}=\{c\}$

2.3.1. Tablice decyzyjne deterministyczne i niedeterministyczne.

Każdy obiekt $u \in U$ tablicy decyzyjnej $TD = (U, C, D, V, f)$ może zostać zapisany w postaci zdania warunkowego (postaci: *jeżeli* warunki *to* decyzja) i być traktowany jako reguła decyzyjna. ***Regułą decyzyjną*** w tablicy decyzyjnej TD nazywamy funkcje: $g : C \cup D \to V$ jeżeli istnieje $x \in U$ taki, że $g = f_x$. Obcięcie g do C ($g|C$) oraz g do D ($g|D$) nazywamy odpowiednio warunkami oraz decyzjami reguły decyzyjnej g.

Przykład 2.16.

Z przykładowej tablicy decyzyjnej z tabeli 2.1 możemy wyprowadzić następujące reguły (odpowiadające konkretnym obiektom):

(1) ***jeżeli*** (g="nie") ***i*** (m="tak") ***i*** (t="wysoka") ***to*** (c="tak")

(2) ***jeżeli*** (g="tak") ***i*** (m="nie") ***i*** (t="wysoka") ***to*** (c="tak")

(3) ***jeżeli*** (g="tak") ***i*** (m="tak") ***i*** (t="bardzo wysoka") ***to*** (c="tak")

(4) ***jeżeli*** (g="nie") ***i*** (m="tak") ***i*** (t="bardzo wysoka") ***to*** (c="tak")

(5) ***jeżeli*** (g="tak") ***i*** (m="nie") ***i*** (t="wysoka") ***to*** (c="nie")

(6) ***jeżeli*** (g="nie") ***i*** (m="tak") ***i*** (t="normalna") ***to*** (c="nie")

Reguły decyzyjne można dzielić na wiele różnych grup biorąc pod uwagę różne kryteria. Jeden z podziałów wyróżnia dwie grupy reguł:

- ***reguły deterministyczne***

  Reguła w tablicy decyzyjnej TD jest deterministyczna, gdy równość atrybutów warunkowych implikuje równość atrybutów decyzyjnych. Fakt ten możemy wyrazić przy pomocy następującej zależności dla obiektów tablicy decyzyjnej:

$$\underset{\substack{x,y\in U\\ x\neq y}}{\forall}\left(\underset{c\in C}{\forall}(f(x,c)=f(y,c))\Rightarrow\underset{d\in D}{\forall}(f(x,d)=f(y,d))\right) \tag{2.31}$$

- ***reguły niedeterministyczne***

  Reguła w tablicy decyzyjnej TD jest niedeterministyczna, gdy równość atrybutów warunkowych nie implikuje równości atrybutów decyzyjnych, co można wyrazić następującą zależnością dla obiektów tablicy decyzyjnej:

$$\underset{\substack{x,y\in U\\ x\neq y}}{\exists}\left(\underset{c\in C}{\forall}(f(x,c)=f(y,c))\wedge\underset{d\in D}{\exists}(f(x,d)\neq f(y,d))\right) \tag{2.32}$$

Tablica decyzyjna jest deterministyczna (dobrze określona, spójna), gdy wszystkie reguły w niej zawarte są deterministyczne, w przeciwnym przypadku jest niedeterministyczna (źle określona, niespójna).

Przykład 2.17.

Tablica decyzyjna z tabeli 2.1 jest niedeterministyczna, gdyż reguły pochodzące z obiektów: 2 i 5 są niedeterministyczne (patrz przykład 2.16).

2.3.2. Relacja nierozróżnialności względem decyzji.

Z uwagi na rzeczywiste zastosowania tablice decyzyjne najczęściej posiadają tylko jeden atrybut decyzyjny, dlatego w dalszej części rozważań przyjmiemy, że $D=\{d\}$. Wszystkie definicje mogą jednak w prosty sposób zostać uogólnione na przypadek, kiedy zbiór atrybutów decyzyjnych posiada więcej niż jeden element.

Niech $TD=(U,C,\{d\},V,f)$ będzie tablicą decyzyjną i niech $B \subseteq C$. ***Relację nierozróżnialności względem decyzji d*** na zbiorze obiektów U generowaną przez zbiór atrybutów B definiujemy jako:

$$IND_{TD}(B,d)=\{(x,y)\in U\times U : (x,y)\in IND_{SI}(B)\vee f(x,d)=f(y,d)\} \qquad (2.33)$$

Relacja nierozróżnialności względem decyzji różni się od relacji nierozróżnialności tym, że nie rozróżnia obiektów mających takie same wartości decyzji nawet wtedy, gdy obiekty te różnią się na rozważanym podzbiorze atrybutów warunkowych B. Warto podkreślić, że relacja nierozróżnialności względem decyzji nie jest relacją równoważności. Jest co prawda zwrotna i symetryczna, ale nie jest przechodnia. Fakt, że relacja ta nie jest przechodnia postaram się teraz pokazać.

Niech: $u,v,w\in U\wedge(u,v)\in IND_{TD}(B,d)\wedge(v,w)\in IND_{TD}(B,d)$ oraz

$f(u,d)\neq f(w,d)\wedge f(u,b)\neq f(w,b)$ dla $b\in B$ wtedy $(u,w)\notin IND_{TD}(B,d)$, więc relacja nierozróżnialności względem decyzji nie jest przechodnia.

Przykład 2.18.

Wyznaczymy teraz relację nierozróżnialności dla tablicy decyzyjnej z tabeli 2.1 względem decyzji d generowaną przez zbiory atrybutów: C_1, C_2, C_3:

$C_1=\{g, m, t\}$, $C_2=\{g, m\}$, $C_3=\{g, t\}$, $d=\{c\}$

$$IND_{TD}(C_1,d)=\{\ (1,1),\ (2,2),\ (3,3),\ (4,4),\ (5,5),\ (6,6),\ (2,5),\ (5,2),\ (1,2),\ (2,1),\ (1,3),\ (3,1),\ (1,4),\ (4,1),\ (2,3),\ (3,2),\ (2,4),\ (4,2),\ (3,4),\ (4,3),\ (5,6),\ (6,5)\ \}$$

$$IND_{TD}(C_2,d) = \{ (1,1), (2,2), (3,3), (4,4), (5,5), (6,6), (1,4), (4,1), (1,6), (6,1), (4,6), (6,4), (2,5), (5,2), (1,2), (2,1), (1,3), (3,1), (2,3), (3,2), (2,4), (4,2), (3,4), (4,3), (5,6), (6,5) \}$$

$$IND_{TD}(C_3,d) = \{ (1,1), (2,2), (3,3), (4,4), (5,5), (6,6), (2,5), (5,2), (1,2), (2,1), (1,3), (3,1), (1,4), (4,1), (2,3), (3,2), (2,4), (4,2), (3,4), (4,3), (5,6), (6,5) \}$$

2.3.3. Macierz, tablica, funkcja oraz wektor odróżnialności dla tablicy decyzyjnej.

Podobnie jak dla systemów informacyjnych w podrozdziale 2.2.8 zdefiniowaliśmy macierz, tablicę, funkcję oraz wektor odróżnialności bazując na relacji nierozróżnialności, tak teraz zdefiniujemy te same pojęcia dla tablicy decyzyjnej bazując na relacji nierozróżnialności względem decyzji.

Jeśli $TD = (U, C, \{d\}, V, f)$ jest tablicą decyzyjną taką, że $U = \{u_1, .., u_n\}$ i $C = \{c_1, .., c_m\}$, to ***macierz odróżnialności tablicy decyzyjnej TD M(TD,d)*** definiujemy następująco:

$$M(TD,d) = (H_{i,j})_{i,j=1,..,n} = \{c \in C : f(u_i,c) \neq f(u_j,c) \wedge f(u_i,d) \neq f(u_j,d)\} \quad \text{dla } i,j = 1,..,n \text{ gdzie: } n = |U| \tag{2.34}$$

Przykład 2.19.

Obliczmy macierz odróżnialności dla tablicy decyzyjnej z tabeli 2.1.

U\U	1	2	3	4	5	6
1	{Ø}	{Ø}	{Ø}	{Ø}	{g,m}	{t}
2	{Ø}	{Ø}	{Ø}	{Ø}	{Ø}	{g,m,t}
3	{Ø}	{Ø}	{Ø}	{Ø}	{m,t}	{g,t}
4	{Ø}	{Ø}	{Ø}	{Ø}	{g,m,t}	{t}
5	{g,m}	{Ø}	{m,t}	{g,m,t}	{Ø}	{Ø}
6	{t}	{g,m,t}	{g,t}	{t}	{Ø}	{Ø}

Tabela 2.4. Macierz odróżnialności dla tablicy decyzyjnej.

Analogicznie, jak dla systemu informacyjnego, zdefiniujemy teraz ***tablicę odróżnialności T(TD,d)*** dla tablicy decyzyjnej $TD = (U, C, \{d\}, V, f)$. Określa się ją następująco:

$$T(DT,d) = T[(i,j),k] = \begin{cases} 0, f(u_i, c_k) = f(u_j, c_k) \vee f(u_i, d) = f(u_j, d) \\ 1, f(u_i, c_k) \neq f(u_j, c_k) \wedge f(u_i, d) \neq f(u_j, d) \end{cases} \quad (2.35)$$

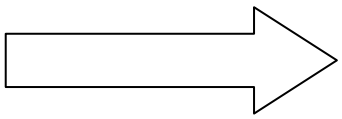
$$\text{gdzie}: i, j = 1,..,|U|; j > i; k = 1,..,|C|$$

Przykład 2.20.

Tabela 2.5 zawiera tablicę odróżnialności (pełną oraz zredukowaną) dla tablicy decyzyjnej z tabeli 2.1. Wersję zredukowaną można uzyskać usuwając z wersji pełnej wszystkie te wiersze, dla których wartości w kolumnach: g, m, t są równe 0.

U	U	g	m	t
1	**2**	0	0	0
1	**3**	0	0	0
1	**4**	0	0	0
1	**5**	1	1	0
1	**6**	0	0	1
2	**3**	0	0	0
2	**4**	0	0	0
2	**5**	0	0	0
2	**6**	1	1	1
3	**4**	0	0	0
3	**5**	0	1	1
3	**6**	1	0	1
4	**5**	1	1	1
4	**6**	0	0	1
5	**6**	0	0	0

Usuwamy wiersze, które dla wszystkich atrybutów mają wartość:0



U	U	g	m	t
1	**5**	1	1	0
1	**6**	0	0	1
2	**6**	1	1	1
3	**5**	0	1	1
3	**6**	1	0	1
4	**5**	1	1	1
4	**6**	0	0	1

Tabela 2.5. Tablica odróżnialności dla tablicy decyzyjnej.

Wektor odróżnialności dla tablicy decyzyjnej TD ($TD = (U, C, \{d\}, V, f)$) określa się następująco:

$$W(TD,d) = W(TD,d)[n*(i-1)+j] = M(TD,d)[i,j]$$
$$\text{gdzie}: i, j = 1,..,n; j > i; \; n = |U| \quad (2.36)$$

Przykład 2.21.

Wyznaczmy wektor odróżnialności i jego optymalną postać dla tablicy decyzyjnej z tabeli 2.1 (korzystając z macierzy odróżnialności z tabeli 2.4):

$W(TD,d) = (\{\emptyset\}, \{\emptyset\}, \{\emptyset\}, \{g,m\}, \{t\}, \{\emptyset\}, \{\emptyset\}, \{\emptyset\}, \{g,m,t\},$
$\{\emptyset\}, \{m,t\}, \{g,t\}, \{g,m,t\}, \{t\}, \{\emptyset\})$

$Z(TD,d) = (\{t\}, \{g,m\})$

Funkcję odróżnialności dla tablicy decyzyjnej TD ($TD = (U,C,\{d\},V,f)$ nazywamy funkcję boolowską $f_{TD,d}$ zmiennych $c_1^*,..,c_m^*$ odpowiadających odpowiednio atrybutom warunkowym (tablicy decyzyjnej) $c_1,..,c_m$ zdefiniowaną następująco:

$$f_{TD,d}(c_1^*,..,c_m^*) = \bigcap\{\bigcup(H_{i,j} : 1 \le j < i \le n \wedge H_{i,j} \neq \emptyset)\} \qquad (2.37)$$

gdzie:

- $n=|U|$, $m=|C|$
- $\bigcup(H_{i,j})$ jest alternatywą wszystkich zmiennych $c^* \in \{c_1^*,..,c_m^*\}$ takich, że $c \in H_{i,j}$

Funkcja ta zdefiniowana jest dokładnie tak samo jak w przypadku systemu informacyjnego, z tym że pomijamy w niej atrybut decyzyjny.

Przykład 2.22.

Funkcja odróżnialności dla tablicy decyzyjnej z tabeli 2.1 określona jest wyrażeniem:

$$f_{TD,c}(g^*,m^*,t^*) = (g^* \vee m^*) \wedge (t^*) \wedge (g^* \vee m^* \vee t^*) \wedge (m^* \vee t^*) \wedge (g^* \vee t^*) \wedge$$
$$(g^* \vee m^* \vee t^*) \wedge (t^*) = (g^* \vee m^*) \wedge (t^*) = (g^* \wedge t^*) \vee (m^* \wedge t^*)$$

2.3.4. Zbiór pokrywający dla tablicy decyzyjnej.

Zbiór par obiektów tablicy decyzyjnej $TD = (U, C, \{d\}, V, f)$ pokrytych przez zbiór atrybutów B definiujemy podobnie jak w przypadku systemu informacyjnego:

$$CS(B,d) = \{(u_i, u_j) : i, j = 1,..,n \wedge j > i \wedge \underset{a \in C}{\exists}(a \in B \wedge a \in M(TD,d)[i,j])\} \quad (2.38)$$

Wszystkie własności jakie posiadał zbiór pokrywający w przypadku systemu informacyjnego są prawdziwe w kontekście tablicy decyzyjnej.

Przykład 2.23.

Policzmy CS oraz $|CS|$ dla przykładowych zbiorów atrybutów tablicy decyzyjnej z tabeli 2.1. Do obliczeń wykorzystamy macierz odróżnialności dla tej tablicy (tabelę 2.4).

$CS(\{g, m, t\},\{c\})=\{ (1,5), (1,6), (2,6), (3,5), (3,6), (4,5), (4,6) \}$

$|CS(\{g, m, t\},\{c\})|=7$

$CS(\{g, t\},\{c\})=\{ (1,5), (1,6), (2,6), (3,5), (3,6), (4,5), (4,6) \}$

$|CS(\{g, t\},\{c\})|=7$

$|CS(\{g, t\},\{c\})|=|CS(\{g, m, t\},\{c\})|$

$CS(\{m, t\},\{c\})=\{ (1,5), (1,6), (2,6), (3,5), (3,6), (4,5), (4,6) \}$

$|CS(\{m, t\},\{c\})|=7$

$|CS(\{m, t\},\{c\})|=|CS(\{g, m, t\},\{c\})|$

$CS(\{t\},\{c\})=\{ (1,6), (2,6), (3,5), (3,6), (4,5), (4,6) \}$

$|CS(\{t\},\{c\})|=6$

$CS(\{g\},\{c\})=\{ (1,5), (2,6), (3,6), (4,5) \}$

$|CS(\{g\},\{c\})|=4$

$CS(\{m\},\{c\})=\{ (1,5), (2,6), (3,5), (4,5) \}$

$|CS(\{m\},\{c\})|=4$

2.3.5. Pojęcie reduktu względnego.

Niech $TD = (U, C, \{d\}, V, f)$ będzie tablicą decyzyjną oraz niech $B \subseteq C$. Zbiór atrybutów Q ($Q \subseteq B$) nazywamy ***reduktem zbioru atrybutów B względem decyzji d***, gdy:

1. zbiór atrybutów Q jest niezależny
2. $IND_{TD}(B,d) = IND_{TD}(Q,d)$ (2.39)

Zbiór wszystkich reduktów zbioru atrybutów B względem decyzji d oznaczamy przez: $RED_{TD}(B,d)$ zaś rdzeń: $CORE_{TD}(B,d)$.

Zależności między reduktem względnym oraz zbiorem pokrywającym dla tablicy decyzyjnej są takie same jak w przypadku reduktu oraz zbioru pokrywającego dla systemu informacyjnego.

Przykład 2.24.

Zbiór wszystkich reduktów względnych zbioru atrybutów $\{g,m,t\}$ tablicy decyzyjnej z tabeli 2.1 wynosi:

$RED_{TD}(\{g, m, t\},\{c\})=\{ \{g,m\}, \{g,t\}\}.$

Można to udowodnić korzystając z przykładu z poprzedniego podrozdziału, gdzie pokazano, że:

$|CS(\{m, t\},\{c\})|=|CS(\{g, t\},\{c\})|=|CS(\{g, m, t\},\{c\})|=7$

oraz

$|CS(\{m\},\{c\})|=4,\ |CS(\{t\},\{c\})|=6,\ |CS(\{g\},\{c\})|=4$

Wykorzystana została tutaj definicja 2.28 w kontekście zbioru pokrywającego dla tablicy decyzyjnej.

3. Charakterystyka badanych danych.

3.1. Wstęp.

Jak wspomniałem wcześniej system informacyjny jest taką strukturą, w której można przechowywać dane różnej postaci. Ściślej mówiąc atrybuty mogą przyjmować wartości różnych typów np. liczby (1; 4; 5,7), napisy („mały”, „duży”) czy też wartości logiczne (TAK, NIE). Taki sposób opisu można określić mianem naturalnego, gdyż jest on łatwo interpretowany przez człowieka. Algorytmy przeprowadzające różnego rodzaju analizy danych zawartych w tablicach decyzyjnych będą natomiast głównie operować na wartościach typu całkowitoliczbowego (m.in. ze względu na łatwość reprezentacji takich danych w pamięci komputera). Potrzebny więc jest pewien mechanizm, który umożliwi przekształcenie danych różnych typów na dane całkowitoliczbowe. W następnym podrozdziale przedstawię podstawowe typy danych wraz z przykładowymi metodami postępowania z nimi w kontekście komputerowej analizy.

3.2. Typy wartości atrybutów.

3.2.1. Typ wyliczeniowy.

Dziedzina atrybutu wyliczeniowego jest zdefiniowana poprzez zbiór konkretnych wartości, jakie ten atrybut może przyjmować. Wartości te mogą być dowolnego typu (tekstowego, liczbowego, itd…). Bardzo często poszczególnym wartościom typu wyliczeniowego przypisywane są unikalne w obrębie całej dziedziny liczby całkowite. Taka transformacja umożliwia przechowywanie danych w tablicy decyzyjnej (w pamięci komputera) w pożądanej postaci całkowitoliczbowej, przy równoczesnym zachowaniu reprezentacji wygodnej z perspektywy użytkownika (mapowanie odwrotne liczb całkowitych na wartości typów docelowych).

Typ ten jest ponadto bardzo często wykorzystywany jako prosty sposób na zapewnienie poprawności wprowadzanych przez użytkownika danych. Użytkownik wybiera jedną z proponowanych (z góry ustalonych) dla danego atrybutu wartości.

Przykład 3.1.

Atrybut temperatura w tablicy decyzyjnej z tabeli 2.1 jest typu wyliczeniowego. Dziedziną tego atrybutu jest: $V_{Temperatura}$={normalna, wysoka, bardzo wysoka}. Kolejnym wartością z dziedziny możemy przypisać kolejne liczby naturalne: 0, 1, 2 i tak reprezentować je w pamięci komputera.

3.2.2. Typ liczb całkowitych.

Dziedziną tego typu jest cały zbiór liczb całkowitych. Służy on do wyrażania tych wielkości badanych parametrów, które da się opisać przy pomocy liczby całkowitej. Większość algorytmów analizy danych operuje właśnie na danych tego typu.

3.2.3. Typ liczb rzeczywistych.

Dziedziną tego typu jest zbiór liczb rzeczywistych. Rozszerzają one typ liczb całkowitych, umożliwiając opisanie cech przy pomocy wartości ułamkowych. Dane rzeczywiste rzadko podlegają bezpośredniej analizie (m.in. ze względu na ich drobną granulację). Z reguły są one na początku poddawane procesowi dyskretyzacji, czyli przekształceniu na liczby typu całkowitego.

3.2.4. Typ logiczny.

Dziedzina typu logicznego składa się z dwóch wartości: *{tak, nie}*. Typ ten może być traktowany jako podzbiór dziedziny typu liczb całkowitych, przy założeniu że wartość prawdy logicznej reprezentujemy liczbą *„1”* zaś fałszu liczbą *„0”*.

3.2.5. Tekst.

Dane typu tekstowego są trudne w analizie (m.in. z tego względu, że nie ma możliwości przeprowadzenia walidacji takiego pola) i z reguły jej nie podlegają. Są za to wykorzystywane jako komentarze, uwagi itd...

3.2.6. Inne dane.

Do danych innego typu, niż wymienione powyżej, możemy zaliczyć wszelkiego rodzaju dane multimedialne tzn. obrazy, sekwencje video i dźwięki. Takie dane, podobnie jak dane tekstowe, są trudne w analizie i bardzo często są traktowane tylko jako pewne dodatkowe informacje nie podlegające jej (np. zdjęcia pacjentów, zeskanowane dokumenty prezentujące wyniki badań, itd…)

3.3. Algorytmy dyskretyzacji danych ciągłych.

3.3.1. Wstęp.

Dyskretyzacja wartości atrybutów ciągłych polega na zastąpieniu każdej wartości atrybutu wartością dyskretną, odpowiadającą pewnemu przedziałowi ciągłych wartości oryginalnego atrybutu. Przedziały te są uporządkowane, co sprawia, że w wyniku dyskretyzacji otrzymujemy zamiast atrybutu ciągłego atrybut porządkowy o skończonej liczbie wartości. Metody dyskretyzacji można podzielić, biorąc pod uwagę różne kryteria, na następujące grupy (podział z zaproponowany w [1.7]):

- metody prymitywne i zaawansowane

  Metody prymitywne nie uwzględniają rozkładu wartości atrybutów i klas decyzyjnych (dzielą zakres wartości atrybutu na ustaloną w pewien arbitralny sposób liczbę równych

przedziałów). Metody zaawansowane zaś dopasowują sposób dyskretyzacji do zbioru przetwarzanych danych.

- metody globalne i lokalne

  Metody globalne dyskretyzują każdy atrybut ciągły w sposób niezależny od innych atrybutów (tzn. wartości dyskretne atrybutu zależą tylko i wyłącznie od wartości ciągłych tego samego atrybutu). W przypadku metod lokalnych zakres wartości atrybutu ciągłego może być dzielony na przedziały na różne sposoby w różnych obszarach dziedziny, wyznaczonych przez wartości innych atrybutów (tzn. wartości dyskretne atrybutu zależą także od wartości innych atrybutów).

- metody bez nadzoru i z nadzorem

  W przypadku dyskretyzacji z nadzorem znane są klasy decyzyjne poszczególnych obiektów i są one uwzględniane podczas dyskretyzacji. Pozwala to tak dobrać przedziały dyskretyzacji, aby było możliwe jak najlepsze reprezentowanie klasy docelowej za pomocą nowego zestawu atrybutów. Metody bez nadzoru zaś nie uwzględniają informacji o klasach decyzyjnych.

- metody zstępujące i wstępujące

  Podział ten ma charakter czysto techniczny i prezentuje dwa różne algorytmicznie podejścia do procesu dyskretyzacji. Dyskretyzacja zstępująca polega na tym, że cała dziedzina dyskretyzowanego atrybutu jest dzielona na podprzedziały za pomocą wybieranych kolejno wartości progowych. W przypadku dyskretyzacji wstępujące rozpoczynamy od podziału dziedziny atrybutu na wiele małych przedziałów, z których każda zawiera jedną wartość tego atrybutu, a następnie wybrane przedziały są łączone w większe.

W dalszej części zostaną przedstawione cztery proste algorytmy dyskretyzacji danych rzeczywistych ([1.7], [1.15], [1.31]). Wszystkie te metody są przykładem dyskretyzacji: prostej, lokalnej, bez nadzoru oraz zstępującej. Pomimo swojej prostoty, niektóre z nich, są nadal używane w zastosowaniach praktycznych (m.in. mogą służyć jako narzędzie do wstępnej analizy danych przed wykorzystaniem bardziej wyrafinowanych algorytmów).

Przykład 3.2.

Tabela 3.1 przedstawia system informacyjny z atrybutami, których dziedziny są typu rzeczywistego (ciągłego). System ten posłuży jako przykład do zaprezentowania działania poszczególnych algorytmów dyskretyzacji danych w kolejnych podrozdziałach.

Pacjent	Waga (g)	Wzrost (z)	Temperatura (t)
1	72,0	170,0	37,9
2	75,5	170,0	37,4
3	80,5	182,0	38,5
4	65,0	165,5	39,2
5	86,5	166,0	38,0
6	74,5	165,0	36,6

Tabela 3.1. System informacyjny z atrybutami o wartościach ciągłych.

3.3.2. Dyskretyzacja naiwna.

Dyskretyzacja naiwna jest jednym z najprostszych sposobów przekształcenia danych z postaci rzeczywistej na postać całkowitą. Polega ona na tym, że każdej nowej wartości rzeczywistej danego atrybutu przyporządkowujemy nową wartość naturalną. Schematyczny zapis tego algorytmu w kontekście dyskretyzacji systemu informacyjnego przedstawiony jest poniżej:

Algorytm 3.1: Naiwny algorytm do dyskretyzacji danych ciągłych.

Wejście: system informacyjny $SI = (U, A, V, f)$

Wyjście: zdyskretyzowany system informacyjny $SI^* = (U^*, A, V^*, f^*)$

Start procedury:

[1] **Dla każdego** atrybutu $a \in A$ **powtarzaj**

[2] Posortuj obiekty z systemu informacyjnego SI w kolejności rosnącej względem atrybutu a

[3] $i:=0,\ v:=-1$

[4] **Dopóki** $i <| U |$ **powtarzaj**

[5] **Jeżeli** $i \neq 0 \wedge f(u_i, a) = f(u_{i-1}, a)$ **to**

[6] $f^*(u_i, a) := f^*(u_{i-1}, a)$

[7] **Inaczej**

[8] $v:=v+1$

[9] $f^*(u_i, a) := v$

[10] $i:=i+1$

Koniec procedury

Istotną wadą tego algorytmu jest fakt, iż nie uwzględnia on specyfiki przetwarzanych danych. Każde dwie, rożne wartości jednego atrybutu zostaną przekształcone w dwie inne liczby naturalne, bez względu na fakt jak bardzo te wartości były oddalone od siebie (algorytm ten nie grupuje wartości ciągłych atrybutu, tak aby odwzorować je w jedną dyskretną wartość).

Ponadto po dokonaniu dyskretyzacji tracimy istotne informacje o pierwotnych danych ciągłych tzn. nie wiemy jakiej wartości rzeczywistej odpowiada dana wartość dyskretna. Jest to istotny problem w przypadku, gdy do dziedziny atrybutu dyskretnego będziemy chcieli dodać nowe wartości poprzez dyskretyzacje następnych (nowych) wartości rzeczywistych. W efekcie może się okazać, że dwie takie same wartości rzeczywiste mogły zostać odwzorowane na dwie różne wartości dyskretne. Aby temu zapobiec należałoby pamiętać wszystkie pary postaci *(wartość rzeczywista, wartość dyskretna)*, co z kolei angażuje duże zasoby pamięci komputera.

Przykład 3.3.

Tabela 3.2 przedstawia system informacyjny z tabeli 3.1 przed oraz po dyskretyzacji z użyciem naiwnego algorytmu dyskretyzacji.

Pacjent	Waga (g)		Wzrost (z)		Temperatura (t)	
1	72,0	1	170,0	3	37,9	2
2	75,5	3	170,0	3	37,4	1
3	80,5	4	182,0	4	38,5	4
4	65,0	0	165,5	1	39,2	5
5	86,5	5	166,0	2	38,0	3
6	74,5	2	165,0	0	36,6	0

Tabela 3.2. System informacyjny ze zdyskretyzowanymi wartościami atrybutów ciągłych przy pomocy dyskretyzatora naiwnego.

3.3.3. Dyskretyzacja według równej szerokości.

Idea tego algorytmu polega na podziale całej dziedziny atrybutu na n równych przedziałów i przyporządkowaniu każdemu z nich jednej wartości dyskretnej. Schemat tego algorytmu zastosowanego do dyskretyzacji systemu informacyjnego znajduje się poniżej:

Algorytm 3.2: Algorytm dyskretyzacji według równej szerokości.

Wejście: system informacyjny $SI = (U, A, V, f)$, wektor S z liczbami określającymi ilość przedziałów dla poszczególnych atrybutów

Wyjście: zdyskretyzowany system informacyjny $SI^* = (U^*, A, V^*, f^*)$

Start procedury:

[1] **Dla każdego** atrybutu $a \in A$ **powtarzaj**

[2] Znajdź minimalną oraz maksymalną wartość atrybutu a i podstaw je odpowiednio pod zmienne: $minVal$ oraz $maxVal$

[3] $range := \frac{maxVal - minVal}{S[a]}$

[4] **Dla każdego** obiektu $u \in U$ **powtarzaj**

[5] $i:=1$

[6] **Dopóki** $f(u,a)>(minVal+range*i)$ **powtarzaj**

[7] $i:=i+1$

[8] $f^*(u,a) = i - 1$

Koniec procedury

Metoda ta, w odróżnieniu od poprzedniej, dokonuje grupowania oryginalnych wartości atrybutów. Cała grupa wartości rzeczywistych jest przekształcana w jedną wartość dyskretną.
W rzeczywistych zastosowaniach brzegowe przedziały dyskretyzacji (tzn. pierwszy i ostatni) modyfikuje się w taki sposób, że kres dolny przedziału pierwszego zastępuję się znakiem: $-\infty$, zaś kres górny ostatniego przedziału zastępuje się znakiem: $+\infty$. Ten prosty zabieg powoduje, że dyskretyzacja nowych wartości tego samego atrybutu nie nastręcza trudności. Wystarczy zbadać przynależność wartości do kolejnych przedziałów dyskretyzacji. W odróżnieniu od poprzedniego algorytmu wystarczy że będziemy pamiętać dla atrybutu tylko przedziały dyskretyzacji i odpowiadające im wartości dyskretne a nie wszystkie pary postaci: *(wartość rzeczywista, wartość dyskretna)*.

Przykład 3.4.

Tabela 3.3 zawiera parametry dla algorytmu dyskretyzacji według równej szerokości (liczba przedziałów dla poszczególnych atrybutów) oraz dodatkowe informacje dotyczące poszczególnych atrybutów wyznaczane podczas działania tej metody (wartość minimalna oraz maksymalna atrybutu jak również długość przedziału). Tabela 3.4 przedstawia system informacyjny z tabeli 3.1 przed oraz po dyskretyzacji z użyciem tego algorytmu oraz parametrów zawartych w tabeli 3.3.

Atrybut	Liczba przedziałów (S)	Wartość minimalna (minVal)	Wartość maksymalna (maxVal)	Rozmiar przedziału (range)
Waga	2	65,0	86,5	10,75
Wzrost	4	165,0	182,0	4,25
Temperatura	4	36,6	39,2	0,65

Tabela 3.3. Parametry oraz informacje dla algorytmu dyskretyzacji według równej szerokości.

Pacjent	Waga (g)		Wzrost (z)		Temperatura (t)	
1	72,0	0	170,0	1	37,9	1
2	75,5	0	170,0	1	37,4	1
3	80,5	1	182,0	3	38,5	2
4	65,0	0	165,5	0	39,2	3
5	86,5	1	166,0	0	38,0	2
6	74,5	0	165,0	0	36,6	0

Tabela 3.4. System informacyjny ze zdyskretyzowanymi wartościami atrybutów ciągłych przy pomocy dyskretyzatora według równej szerokości.

3.3.4. Dyskretyzacja według równej częstości.

Algorytm ten dzieli dziedzinę atrybutu na n przedziałów o różnej szerokości tak, aby każdy z nich zawierał taką samą, z góry zadaną liczbę przykładów. Istotną cechą tej metody jest fakt, iż wartości dyskretne atrybutu będą miały rozkład równomierny.

Algorytm 3.3: Algorytm dyskretyzacji według równej częstości.

Wejście: system informacyjny $SI = (U, A, V, f)$, wektor P z liczbami określającymi ilość obiektów trafiających do poszczególnych przedziałów

Wyjście: zdyskretyzowany system informacyjny $SI^* = (U^*, A, V^*, f^*)$

Start procedury:

[1] **Dla każdego** atrybutu $a \in A$ **powtarzaj**

[2] Posortuj obiekty uniwersum U w kolejności rosnącej względem wartości atrybutu a

[3] $objects := P[a]$; $i:=0$, $v:=-1$

[4] **Dopóki** $i <| U |$ **powtarzaj**

[5] **Jeżeli** $i \bmod objects=0$ **to**

[6] $v:=v+1$

[7] **Jeżeli** $i \neq 0 \wedge f(u_i, a) = f(u_{i-1}, a)$ **to**

[8] $f^*(u_i, a) := f^*(u_{i-1}, a)$

[9] **Inaczej**

[10] $f^*(u_i, a) := v$

[11] $i := i+1$

Koniec procedury

Podobnie jak poprzednia metoda ten algorytm także odwzorowuje całe grupy wartości rzeczywistych w jedną wartość dyskretną. Dodanie nowych wartości do dziedziny także nie nastręcza trudności pod warunkiem, że po procesie pierwotnej dyskretyzacji utworzymy odpowiednie przedziały (na podstawie efektów dyskretyzacji) i zapamiętamy odpowiadające im wartości naturalne. Warto jednak podkreślić, że dodając nowe wartości do dziedziny stracimy równomierny rozkład wartości dla atrybutu.

Przykład 3.5.

Tabela 3.5 zawiera parametry dla algorytmu dyskretyzacji według równej częstości tzn. liczbę obiektów w przedziale dla poszczególnych atrybutów oraz wynikającą z niej liczbę przedziałów. Tabela 3.6 przedstawia system informacyjny z tabeli 3.1 przed oraz po dyskretyzacji z użyciem tego algorytmu oraz parametrów zawartych w tabeli 3.5.

Atrybut	Liczba obiektów w przedziale (P)	Liczba przedziałów
Waga	2	3
Wzrost	3	2
Temperatura	3	2

Tabela 3.5. Parametry oraz informacje dla algorytmu dyskretyzacji według równej częstości.

Pacjent	Waga (g)		Wzrost (z)		Temperatura (t)	
1	72,0	0	170,0	1	37,9	0
2	75,5	1	170,0	1	37,4	0
3	80,5	2	182,0	1	38,5	1
4	65,0	0	165,5	0	39,2	1
5	86,5	2	166,0	0	38,0	1
6	74,5	1	165,0	0	36,6	0

Tabela 3.6. System informacyjny ze zdyskretyzowanymi wartościami atrybutów ciągłych przy pomocy dyskretyzatora według równej częstości.

3.3.5. Dyskretyzacja z wykorzystaniem wiedzy eksperta.

Metoda ta jest pewnym rozszerzeniem metody dyskretyzacji według równej szerokości. W tym przypadku dziedzina atrybutu nie jest dzielona na równe przedziały, tylko na przedziały, których charakter jest określony przez eksperta (tzn. osobę, która dobrze zna specyfikę i charakter problemu, z którego pochodzą dane). Ekspert określając parametry dyskretyzacji będzie się kierował swoją wiedzą oraz doświadczeniem uzyskanymi w rzeczywistych warunkach. Tak sprecyzowane przedziały dyskretyzacji będą bardziej odzwierciedlać prawdziwy charakter danych niż metody dzielące przestrzeń atrybutu na arbitralnie narzuconą liczbę równych przedziałów (jak to miało miejsce w poprzednich metodach).

Algorytm 3.4: Algorytm dyskretyzacji wykorzystujący wiedzę eksperta.

Wejście: system informacyjny $SI = (U, A, V, f)$, wektor wektorów $P = (P_1, .., P_{|A|})$, elementy każdego z podwektorów zawierają przedziały dyskretyzacji dla poszczególnych atrybutów (w formie*:* $minValue,\ maxValue$)

Wyjście: zdyskretyzowany system informacyjny $SI^* = (U^*, A, V^*, f^*)$

Start procedury:

[1] **Dla każdego** atrybutu $a \in A$ **powtarzaj**

[2] **Dla każdego** *obiektu* $u \in U$ **powtarzaj**

[3] **Dla każdego** *przedziału* $p_i \in P_a$ **powtarzaj**

[4] **Jeżeli** $p.minValue < f(u,a) \leq p.maxValue$ **to**

[5] $f^*(u,a) = i$

Koniec procedury

Należy pamiętać, że przedziały zaproponowane przez eksperta powinny być poprawne w sensie logicznym tzn. powinny pokrywać całą dziedzinę dopuszczalnych wartości dla atrybutu, jak również powinny być rozłączne. Przy tak zdefiniowanych przedziałach algorytm ten, w

kontekście dodawania nowych wartości do dziedziny, wykazuje takie same cechy jak metoda względem równej szerokości.

Przykład 3.6.

Atrybut	Przedziały zaproponowane przez eksperta
Waga	*(0;65>,(65;90>,(90;150>*
Wzrost	*(0;150>,(150;250>*
Temperatura	*(30;35>,(35;37>,(37;39>,(39;44>*

Tabela 3.7. Parametry dla algorytmu dyskretyzacji wykorzystującego wiedzę eksperta.

Pacjent	Waga (g)		Wzrost (z)		Temperatura (t)	
1	72,0	1	170,0	1	37,9	2
2	75,5	1	170,0	1	37,4	2
3	80,5	1	182,0	1	38,5	2
4	65,0	0	165,5	1	39,2	3
5	86,5	1	166,0	1	38,0	2
6	74,5	1	165,0	1	36,6	1

Tabela 3.8. System informacyjny ze zdyskretyzowanymi wartościami atrybutów ciągłych przy pomocy dyskretyzatora wykorzystującego wiedzę eksperta.

4. Metody wyznaczania reduktów.

4.1. Wstęp.

W literaturze bardzo często poruszane są następujące dwie klasy problemów związane z wyznaczaniem reduktów systemu informacyjnego (tablicy decyzyjnej):

- wyznaczenie minimalnego reduktu (reduktu o minimalnej długości, minimalnej liczbie atrybutów)
- wyznaczenie maksymalnej liczby reduktów (w idealnym przypadku wszystkich reduktów)

Problem wyznaczenia minimalnego reduktu można sprowadzić wielomianowo do zagadnienia znajdowania minimalnego pokrycia wierzchołkowego w grafie, czyli klasycznego problemu ***NP-trudnego***. Odpowiedni dowód znajduje się w [1.8]. Świadczy to o tym, że oba zaprezentowane zagadnienia, dotyczące wyznaczania reduktów, należą do klasy problemów NP-trudnych, czyli takich dla których nie istnieje algorytm o wielomianowej złożoności na komputer deterministyczny. Problemy tej klasy złożoności przy aktualnie posiadanej wiedzy oraz możliwościach technicznych komputerów są w zasadzie nierozwiązywalne metodami dokładnymi. Z tego względu obok algorytmu dokładnego znajdującego wszystkie redukty (czyli także ten minimalny redukt) istnieje szereg algorytmów heurystycznych (przybliżonych), o znacznie mniejszej złożoności obliczeniowej, rozwiązujących te problemy. Warto jednak już teraz wspomnieć, iż wynik uzyskany przy pomocy algorytmu heurystycznego wcale nie musi być wynikiem optymalnym. I tak redukt otrzymany przy pomocy algorytmu przybliżonego do wyznaczania minimalnego reduktu nie musi być tym minimalnym, jak również zbiór reduktów uzyskany przy pomocy algorytmu przybliżonego do wyznaczania wszystkich reduktów nie musi zawierać wszystkich reduktów danej tablicy decyzyjnej czy też systemu informacyjnego.
W dalszej części tego rozdziału zostaną przedstawione metody, które są wykorzystywane w problemach wyznaczania minimalnego oraz maksymalnej liczby reduktów tablicy decyzyjnej.

4.2. Dekompozycja przestrzeni w problemie wyznaczania reduktów.

W rozdziale numer 2 zostało sformułowane pojęcie reduktu oraz reduktu względnego. Intuicyjnie reduktem R zbioru atrybutów B nazwiemy taki podzbiór zbioru B, który zapewnia nam takie samo rozróżnianie obiektów jak zbiór atrybutów B.

Wiedząc, że redukt jest podzbiorem pewnego zbioru atrybutów spróbujmy scharakteryzować oraz oszacować przestrzeń potencjalnych rozwiązań, jaką należy przeszukać w problemach wyznaczania minimalnego oraz maksymalnej liczby reduktów zbioru atrybutów B. Przestrzeń ta powinna zawierać wszystkie możliwe podzbiory rozpatrywanego zbioru B. Do jej reprezentacji najczęściej wykorzystywane są dwa różne podejścia:

- podejście bazujące na zbiorach atrybutów (ang. attribute-set based)
  Przestrzeń rozwiązań Atr jest reprezentowana w postaci zbioru, którego elementami są wszystkie możliwe podzbiory zbioru atrybutów B. Rozmiar tej przestrzeni wynosi więc: $2^{|B|}$.
- podejście bazujące na permutacjach atrybutów (ang. attribute-permutation based)
  Przestrzeń rozwiązań Per zawiera wszystkie możliwe permutacje atrybutów należące do zbioru B. Każda permutacja zawiera dokładnie $|B|$ elementów – wszystkie elementy ze zbioru B. Rozmiar takiej przestrzeni wynosi więc: $|B|!$. Pomimo tego, że jest to znacznie więcej niż w poprzednim przypadku (dla odpowiednio dużego $|B|$) to przestrzeń ta, w kontekście poszukiwania reduktów, jest równoważna poprzedniej. Każda permutacja reprezentuje pewną kolejność w jakiej należy dodawać atrybuty do zbioru pustego (odejmować od zbioru pełnego) aby otrzymać zbiór rozróżniający. Łatwo więc zauważyć, że pewna ilość różnych elementów z przestrzeni Per odpowiada dokładnie jednemu elementowi z przestrzeni Atr. Problemem tym zajmiemy się szarzej w kolejnych podrozdziałach.

Przykład 4.1.

Posłużmy się systemem informacyjnym z tabeli 2.1. Niech $B=\{g, m, t, c\}$. W przypadku poszukiwania reduktów zbioru atrybutów B odpowiednie przestrzenie Atr i Per będą opisane przez następujące elementy:

$Atr=\{ \{\emptyset\}, \{g\}, \{m\}, \{t\}, \{c\}, \{g, m\}, \{g, t\}, \{g, c\}, \{m, t\}, \{m, c\}$
$\{t, c\}, \{g, m, t\}, \{g, m, c\}, \{g, t, c\}, \{m, t, c\}, \{g, m, t, c\} \}$

$Per=\{ (g, m, t, c), (m, g, t, c), (m, t, g, c), (m, t, c, g), (g, t, m, c),$
$(t, g, m, c), (t, m, g, c), (t, m, c, g), (g, t, c, m), (t, g, c, m),$
$(t, c, g, m), (t, c, m, g), (g, m, c, t), (m, g, c, t), (m, c, g, t),$
$(m, c, t, g), (g, c, m, t), (c, g, m, t), (c, m, g, t), (c, m, t, g),$
$(g, c, t, m), (c, g, t, m), (c, t, g, m), (c, t, m, g) \}$

W dalszych rozważaniach skupimy się na przestrzeni Atr – bazującej na reprezentacji przy pomocy zbioru atrybutów. Jak wspomniałem wcześniej rozmiar tak zdefiniowanej przestrzeni wynosi: $2^{|B|}$. Można łatwo zauważyć, że dla odpowiednio dużego $|B|$ efektywne przejrzenie otrzymanej przestrzeni nie jest możliwe. Przestrzeń tę można jednak podzielić na mniejsze, rozłączne podprzestrzenie. Dekompozycja będzie polegała na wyodrębnieniu $n=|B|$ podprzestrzeni takich, że: $B = \bigcup_{k=1}^{|B|} B_k$ i $B_k = \{b \in B : card(b) = k\}$. Rozmiar podprzestrzeni k wynosi: $\binom{|B|}{k}$. Każda podprzestrzeń jest więc znacznie mniejsza od oryginalnej. Poszczególne podprzestrzenie będą przeszukiwane niezależnie.

Dekompozycję przestrzeni można wykorzystać w następujących okolicznościach:

- gdy chcemy wyznaczyć redukt o jakiejś konkretnej, z góry zadanej wielkości (ilości atrybutów). Przeszukujemy wtedy tylko tą podprzestrzeń, która zawiera zbiory atrybutów o żądanej ilości elementów
- gdy chcemy wyznaczyć minimalny redukt. Przeszukujemy kolejno podprzestrzenie od $B_{|B|}$ do B_1 (lub też od B_1 do $B_{|B|}$) szukając w każdej z nich jednego zbioru rozróżniającego i usuwając z niego atrybuty zbędne.

- gdy chcemy wyznaczyć wszystkie redukty. Przeszukujemy kolejno wszystkie podprzestrzenie szukając w każdej z nich wszystkich zbiorów rozróżniających i usuwając z nich atrybutu zbędne.

Dekompozycja przestrzeni w problemie znajdowania reduktów oraz korzyści płynące z jej zastosowania zostały szerzej omówione w [1.35].

Idea dekomponowania przestrzeni została z powodzeniem wykorzystana w wielu problemach optymalizacyjnych np. w problemie kolorowania grafu ([1.6]).

4.3. Zaawansowane techniki heurystyczne.

Stosowanie metod dokładnych, czyli takich, które dają optymalny wynik do rozwiązywania problemów należących do klasy *NP* jest w rzeczywistości bardzo często niemożliwe, ze względu na ograniczenie czasowe. Duża złożoność algorytmów do rozwiązywania tych problemów powoduje, że w realnych zastosowaniach wykorzystuje się tzw. algorytmy heurystyczne. Stosując algorytm heurystyczny, nie mamy gwarancji, że otrzymany wynik będzie optymalnym rozwiązaniem. Otrzymamy go jednak w znacznie krótszym czasie niż rozwiązanie pochodzące z wykonania algorytmu dokładnego.

Algorytmy heurystyczne możemy podzielić na dwie podstawowe grupy:

- algorytmy specjalizowane – rozwiązują one jeden konkretny problem bądź pewną klasę problemów (np. algorytm Johnsona do znajdowania minimalnego reduktu)
- algorytmy ogólne – rozwiązują one w zasadzie dowolny problem (po odpowiednim sparametryzowaniu)

Do ogólnych metod heurystycznych zaliczamy m.in.:

- algorytmy genetyczne (ewolucyjne)
- symulowane wyżarzanie
- poszukiwanie tabu

W dalszej części przedstawię ogólne informacje dotyczące algorytmów genetycznych, które będą wykorzystywane przy prezentacji metod wyznaczania reduktów z użyciem tej właśnie metody heurystycznej.

Algorytmy genetyczne zostały wynalezione przez Johna Hollanda oraz jego kolegów z Uniwersytetu Michigan w 1975 roku. Szerokie studium nad tym problemem prowadził Dawid E. Goldberg (m.in. [1.9]) i to głównie jego osoba jest kojarzona z tymi metodami.

Algorytmy genetyczne są to algorytmy poszukiwania oparte na mechanizmach doboru naturalnego oraz dziedziczności (Darwinowskiej teorii ewolucji). Ich działanie łączy w sobie dwa istotne elementy: eksploatację (czyli znajdowanie nowych wyników na podstawie poprzednio osiągniętych rezultatów) oraz eksplorację (czyli przeszukiwanie nowych, jeszcze nie odwiedzanych, obszarów przestrzeni rozwiązań). Odpowiednio sterując parametrami algorytmu genetycznego możemy sprawić aby algorytm skupiał się bardziej na jednym z tych dwóch elementów.

W wielu przypadkach algorytmy genetyczne sprawdzają się lepiej niż klasyczne algorytmy optymalizacyjne. Spowodowane to jest specyficznymi cechami algorytmów genetycznych, takimi jak:

- nie przetwarzają one bezpośrednio parametrów zadania, lecz ich zakodowaną postać
- prowadzą poszukiwania, wychodząc nie z pojedynczego punktu (np. jak w metodzie największego spadku), lecz z pewnej ich ilości (tyle punktów ile osobników w populacji)
- korzystają tylko z funkcji celu (przystosowania), nie zaś jej pochodnych lub innych pomocniczych informacji (np. jak w metodach gradientowych), co powoduje, że algorytm genetyczny nie musi znać optymalizowanej funkcji
- stosują metody probabilistyczne, a nie deterministyczne reguły wyboru (tak jak typowe, klasyczne metody optymalizacji), co powoduje że algorytm genetyczny można uruchamiać parokrotnie mając nadzieję, że w kolejnej iteracji otrzymamy lepszy wynik niż w poprzednich

Warto także wspomnieć, że metoda ta jest uniwersalna (można ją stosować w szerokiej gamie różnych problemów) oraz stosunkowo szybka (w porównaniu z algorytmami znajdującymi rozwiązanie optymalne) .

Algorytmy genetyczne posiadają jednak szereg wad. Do najistotniejszych można zaliczyć:

- uniwersalność (nie jest tak skuteczna jak specjalizowane algorytmy dla konkretnych problemów)

- fakt, iż uzyskanie dobrego wyniku jest możliwe przy prawidłowym zakodowaniu problemu, dobraniu funkcji celu, zdefiniowaniu operatorów oraz dobraniu parametrów algorytmu. Dotychczas nie udało się sformułować żadnych konkretnych twierdzeń, jak to należy robić, aby uzyskać satysfakcjonujący rezultat. W procesie projektowania algorytmu istotne znaczenie odgrywają tutaj intuicja i doświadczenie projektanta jak również jego cierpliwość (czasami algorytm trzeba uruchamiać kilkunastokrotnie, aby dobrać dla niego poprawne parametry)
- algorytm genetyczny nie daje rozwiązania optymalnego (uzyskujemy rozwiązanie przybliżone), co więcej nie ma żadnych dowodów, które mówią, ile razy uzyskane rozwiązanie jest gorsze od rozwiązania optymalnego

Podstawowym elementem algorytmu ewolucyjnego jest populacja, czyli zbiór potencjalnych rozwiązań problemu. Każde z nich nazywamy osobnikiem bądź chromosomem. Chromosomy składają się z jednostek elementarnych – genów. Ogólny schemat działania algorytmu genetycznego wygląda następująco:

[1] Kodowanie problemu
[2] Inicjalizacja populacji bazowej T_b
[3] T_t – populacja tymczasowa
[4] **Powtarzaj dopóki** nie zostanie spełniony warunek stopu
[5] Reprodukcja ($T_t \mapsto T_b$)
[6] Krzyżowanie ($T_t \mapsto T_t$)
[7] Mutacja ($T_t \mapsto T_t$)
[8] Sukcesja ($T_b \mapsto T_t$)
[9] Wybierz najlepszego osobnika z T_b

Aby rozwiązać konkretne zadanie przy pomocy algorytmu genetycznego, musimy zakodować przestrzeń stanów (czyli wszystkie potencjalne rozwiązania) w języku binarnym oraz zaprojektować funkcję przystosowania osobnika, która ma za zadanie ocenić jakość znajdowanych przez algorytm rozwiązań. Następnie tworzymy populację bazową T_b zawierająca pewną ilość osobników (dopuszczalnych rozwiązań problemu). W etapie reprodukcji wybieramy pewną ilość najlepszych (w sensie wartości funkcji przystosowania) osobników, które następnie

poddajemy procesom krzyżowania oraz mutacji. Osobniki te tworzą nową populację (populację tymczasową) T_t. Następnie tworzymy nową populację bazową poprzez wybranie osobników z populacji tymczasowej oraz starej populacji bazowej (selekcję osobników dokonujemy zgodnie ze znanymi schematami np. wybieramy najlepsze chromosomy z populacji T_t oraz T_b). Krok ten powtarzamy, aż zostanie spełniony warunek stopu (np. ogólna ilość iteracji algorytmu lub też brak poprawy rozwiązania w ciągu ostatnich k iteracji).

4.4. Metody wyznaczania reduktów o zadanej długości.

Zdefiniujmy teraz problem pomocniczy polegający na wyznaczeniu n reduktów względnych tablicy decyzyjnej $TD = (U, C, \{d\}, V, f)$ o długości nie większej niż k. Rozwiązaniem tego problemu będzie więc zbiór reduktów $RED*_{TD}(C,d)$ spełniający następujące warunki:

$$\begin{cases} card(RED*_{TD}(C,d)) = n \\ \forall_{R \in RED*_{TD}(C,d)} \; card(R) \le k \end{cases}$$

Algorytmy będą działać zgodnie z określonym schematem. Będą one poszukiwać zbiorów o długości k, które rozróżniają obiekty tablicy decyzyjnej. Tak znaleziony zbiór, po usunięciu atrybutów zbędnych, będzie reduktem o długości mniejszej bądź równej k. Z tego właśnie względu problem został skonstruowany w ten sposób, aby nie odrzucać rozwiązań o ilości atrybutów mniejszej niż k.

Poniżej zostaną zaprezentowane dwa przykładowe algorytmy rozwiązujące tak określony problem. Zostaną one wykorzystane w dalszej części pracy poświęconej metodom wyznaczania minimalnego reduktu oraz wszystkich reduktów względnych tablicy decyzyjnej.

4.4.1. Algorytm losowy.

Algorytm ten przeszukuje przestrzeń reprezentowaną w postaci zbioru atrybutów (Atr) w sposób całkowicie losowy. Pojedyncza iteracja algorytmy polega na wylosowaniu k różnych atrybutów i

utworzeniu z nich zbioru. Następnie sprawdzane jest, czy taki zbiór jest zbiorem rozróżniającym obiekty tablicy decyzyjnej, Jeżeli tak, to usuwane są z niego atrybuty zbędne. Tak zmodyfikowany zbiór zostaje dodany do zwracanego rozwiązania. Algorytm wykonuje N takich iteracji.

Algorytm 4.1: Algorytm losowy wyznaczający redukty o zadanej długości.

Wejście: tablica decyzyjna $TD = (U, C, \{d\}, V, f)$, ilość iteracji N, wielkość zbioru rozróżniającego k, ilość reduktów do wyznaczenia n

Wyjście: zbiór reduktów TD: $RED^*_{TD}(C,d)$

Start procedury:

[1] Wyznacz macierz odróżnialności $M(TD,d)$ dla tablicy decyzyjnej TD

[2] Wyznacz wektor odróżnialności $W(TD,d)$ i usuń z niego elementy powtarzające się.

[3] $RED^*_{TD}(C,d) := \emptyset$

[4] **Powtarzaj** N razy

[5] Utwórz zbiór R poprzez wybranie w sposób losowy k elementów ze zbioru C

[6] **Jeżeli** $CS(R,d)=CS(C,d)$ **to**

[7] Usuń atrybuty zbędne z R

[8] $RED^*_{TD}(C,d) := RED^*_{TD}(C,d) \cup R$

[9] **Jeżeli** $card(RED^*_{TD}(C,d)) >= n$ **to przerwij** pętlę 4

Koniec procedury

Specyfika tego algorytmu, polegające na błądzeniu przypadkowym, sprawia, że jest on użyteczny tylko i wyłącznie w wyznaczaniu reduktów względnych dla tablic decyzyjnych o małej liczbie atrybutów. Dla bardziej złożonych przypadków, kiedy przestrzeń potencjalnych rozwiązań staje się bardzo duża, prawdopodobieństwo trafienia w rozwiązanie jest bliskie zeru. Algorytm ten nie jest więc w praktyce wykorzystywany, może jednak służyć jako pewien poziom odniesienia dla innych, bardziej wyrafinowanych metod.

4.4.2. Algorytm ewolucyjny.

Bardziej zaawansowaną metodą w stosunku do algorytmu losowego jest algorytm ewolucyjny. Niewątpliwą jego zaletą jest fakt, że w odróżnieniu od poprzednika nie prowadzi on poszukiwań na ślepo, lecz w pewien uporządkowany sposób. Poniżej scharakteryzują poszczególne elementy tego algorytmu.

4.4.2.1. Reprezentacja chromosomu.

Chromosom jest reprezentowany w postaci wektora binarnego o długości równej ilości atrybutów w tablicy decyzyjnej ($card(C)$). Wartość na i-tym genie (elemencie wektora) określa, czy dany atrybut z tablicy decyzyjnej należy („1”), czy też nie należy („0”) do zbioru atrybutów. Dla zbioru atrybutów B odpowiedni chromosom S posiada reprezentację:

$$S[i] = \begin{cases} 0 & gdy \quad b_i \notin B \\ 1 & gdy \quad b_i \in B \end{cases}$$

Ponieważ algorytm ma szukać zbiorów rozróżniających o zadanej długości: k, więc wektor będzie miał dokładnie k genów z wartością „1”. Pozostałe geny będą miały wartość „0”.

4.4.2.2. Inicjalizacja populacji bazowej.

Populacja bazowa jest inicjalizowana przy pomocy osobników wygenerowanych w sposób losowy z rozkładem równomiernym. Każdy osobnik z populacji spełnia oczywiście ograniczenie związane z konkretną liczbą genów z wartością „1” w chromosomie.

4.4.2.3. Operator krzyżowania.

Podstawowy problem z operatorem krzyżowania, jak również mutacji, jest taki, że z dozwolonych osobników (stała liczba genów z wartością „1') muszą one wytworzyć także dozwolone osobniki.

Zaproponowany operator krzyżowania jest pewnym wariantem klasycznego krzyżowania równomiernego. Krzyżowanie z dwóch chromosomów rodzicielskich tworzy dwoje dzieci. Wyznaczanie chromosomów potomnych (P_1, P_2) z chromosomów rodzicielskich (R_1, R_2) odbywa się zgodnie ze schematem:

[1] $i=0$

[2] **Dopóki** $i<|R_1|$ **powtarzaj**

[3] $Q[i]:=R_1[i]+R_2[i]$

[4] $i:=i+1$

[5] $i=0$

[6] **Dopóki** $i<|R_1|$ **powtarzaj**

[7] **Jeżeli** $(Q[i]=0)$ **to** $P_1[i] := 0 \wedge P_2[i] := 0$

[8] **Jeżeli** $(Q[i]=1)$ **to** $P_1[i] := 0 \wedge P_2[i] := 0$

[9] **Jeżeli** $(card(P_1)=k)$ **to** $P_1[i] := 0 \wedge P_2[i] := 1$

[10] **Inaczej Jeżeli** $(card(P_2)=k)$ **to** $P_1[i] := 1 \wedge P_2[i] := 0$

[11] **Inaczej** wylosuj cyfrę $s \in <0,1>$ **i** $P_s[i] := 1 \wedge P_s[i] := 0$

[12] **Jeżeli** $Q[i]=2$ **to** $P_1[i] := 1 \wedge P_2[i] := 1$

Jeżeli na pozycji i-tej każdy z rodziców miał gen z wartością „1" („0") to oba osobniki potomne będą miały na genie i-tym wartość „1" („0"). Jeżeli zaś jeden z rodziców miał na i-tej pozycji gen z wartością „1" zaś drugi z wartością „0" to wtedy losowany jest osobnik potomny, który będzie miał gen o wartości „1" (zachowując warunek na maksymalną liczbę wartości „1" w każdym chromosomie).

Osobniki potomne mogą więc mieć i-ty gen ustawiony na wartość „1" tylko wtedy, gdy jeden z rodziców miał wartość i-tego genu równą „1".

4.4.2.4. Operator mutacji.

Mutacja ma charakter dyskretny. Chromosomy, które podlegają mutacji losujemy z pewnym prawdopodobieństwem. Następnie dla każdego wybranego osobnika losujemy dwa geny, z których jeden ma wartość „0" a drugi ma wartość „1" i zamieniamy je miejscami.

4.4.2.5. Funkcja przystosowania.

Funkcja przystosowania dla chromosomu B zależy tylko i wyłącznie od ilości wierszy tablicy odróżnialności (komórek macierzy odróżnialności), którą pokrywa odpowiedni zbiór atrybutów i wyraża się wzorem:

$$fitness(B) = \frac{card(CS(B,d))}{card(CS(C,d))}$$

Łatwo zauważyć, że $0 \leq fitness(B) \leq 1$.

Można także zastosować modyfikator promujący te chromosomy, które kodują zbiory rozróżniające dla danej tablicy decyzyjnej tzn.

$$fitness_{mod}(B) = fitness(B) + g(B), \quad gdzie$$
$$g(B) = \begin{cases} 0 & dla \quad CS(B,d) \neq CS(C,d) \\ 0{,}5 & dla \quad CS(B,d) = CS(C,d) \end{cases}$$

W takim przypadku zachodzi następująca zależność: $0 \leq fitness_{mod}(B) \leq 1{,}5$.

4.5. Metody wyznaczania maksymalnej liczby reduktów.

4.5.1. Algorytm dokładny.

Standardowy algorytm wyznaczający wszystkie redukty danej tablicy decyzyjnej działa zgodnie z następującym schematem ([1.2]):

- wyznacza macierz odróżnialności oraz funkcję odróżnialności tablicy decyzyjnej
- przeprowadza minimalizację funkcji odróżnialności oraz zamienia jej postać z koniunkcji alternatyw na alternatywę koniunkcji.

Algorytm 4.2: Algorytm dokładny (wyczerpujący) wyznaczający wszystkie redukty względne tablicy decyzyjnej.

Wejście: tablica decyzyjna $TD = (U, C, \{d\}, V, f)$

Wyjście: zbiór reduktów względnych TD: $RED_{TD}(C, d)$

Start procedury:

[1] Wyznacz macierz odróżnialności $M(TD,d)$ dla tablicy decyzyjnej TD

[2] Wyznacz optymalną postać wektora odróżnialności W(TD,d) (usuwając powtarzające się elementy oraz stosując zasadę pochłaniania)

[3] $LR = \emptyset$ //przechowuje listę reduktów

[4] $F = \emptyset$ //przechowuje pośrednie wyniki algorytmu

[5] **Dla każdego** elementu H z wektora odróżnialności W(TD,d) **powtarzaj**

[6] $S := \{R \in LR : R \cap H \neq \emptyset\}$

[7] $$T := \begin{cases} \{a\} \cup R : a \in H, R \cap H = \emptyset, R \in F & gdy\ F \neq \emptyset \\ \{a\} : a \in H & gdy\ F = \emptyset \end{cases}$$

[8] $F := R$

[9] $LR := S \cup T$

[10] Ze zbioru LR usuń elementy powtarzające się

[11] Z każdego z elementów zbioru LR usuń atrybuty zbędne

[12] $RED_{TD}\{C,d\} := LR$

Koniec procedury

Złożoność obliczeniowa: na złożoność algorytmu wpływają głównie kroki od [5] do [7].

Złożoność algorytmu jest wykładnicza i wynosi: $O\left(|U| - \frac{1 - |U|^w}{1 - |U|}\right)$, gdzie w jest ilością elementów wektora $W(TD,d)$.

Algorytm ten jest jedynym algorytmem do wyznaczania reduktów z prezentowanych w tej pracy, który daje nam pewność, że zwrócony wynik będzie zawierał wszystkie redukty względne danej tablicy decyzyjnej.
Należy tutaj dodać, że algorytm w zaprezentowanej tutaj postaci nie daje możliwości uzyskania częściowego wyniku. Jeżeli chcemy uzyskać tylko kilka reduktów, to i tak musimy czekać aż algorytm zostanie całkowicie zakończony. Jest to niewątpliwie jedna z istotnych wad tego rozwiązania. Istnieje zmodyfikowana wersja algorytmu wyczerpującego, przedstawiona w [1.8], która nie ma już tej niedogodności. Przy jej użyciu można dość szybko otrzymać pierwsze redukty i wstrzymać dalsze wykonywanie algorytmu.

4.5.2. Algorytmy ewolucyjne.

W tym podrozdziale przedstawię dotychczas stosowane (dobrze ugruntowane i opisane w literaturze) algorytmy ewolucyjne służące do wyznaczania maksymalnej liczby reduktów. Dokładny ich opis znajduje się w [1.37] oraz w [1.4].

4.5.2.1. Wariant z kodowaniem binarnym.

Scharakteryzuję teraz krótko jeden z algorytmów bazujących na reprezentacji przestrzeni w postaci zbioru atrybutów (przestrzeń Atr).

4.5.2.1.1. Reprezentacja chromosomu.

Reprezentacja chromosomu jest taka sama jak przedstawiona w punkcie 4.4.2.1 z tym, że chromosom może zawierać dowolną liczbę genów z wartością równą „1”. Tzn. zbiór atrybutów reprezentowany przez chromosom nie posiada ograniczenia na stałą wielkość w obrębie jednej populacji. Szukamy więc zbiorów rozróżniających o dowolnej wielkości.

4.5.2.1.2. Inicjalizacja populacji bazowej.

Populacja bazowa jest inicjalizowana w sposób losowy z użyciem rozkładu równomiernego. Dla każdego osobnika losujemy liczbę n, która określa ilość atrybutów należących do zbioru a następnie losujemy n różnych liczb odpowiadającym poszczególnym atrybutom tablicy decyzyjnej.

4.5.2.1.3. Operator krzyżowania.

Zastosowany operator krzyżowania jest standardowym jednopunktowym krzyżowaniem. Polega ono na tym, że dla każdego z osobników rodzicielskich losowany jest punkt rozcięcia, który dzieli chromosom na dwie części: część A i część B. Dwa chromosomy potomne tworzone są w następujący sposób:

- pierwszy chromosom potomny dziedziczy część A chromosomu pierwszego rodzica i część B chromosomu drugiego rodzica
- drugi chromosom potomny dziedziczy część B chromosomu pierwszego rodzica i część A chromosomu drugiego rodzica,

4.5.2.1.4. Operator mutacji.

Operator mutacji jest taki sam jak operator omówiony w punkcie 4.4.2.1.

4.5.2.1.5. Funkcja przystosowania.

Funkcja przystosowania jest taka sama jak funkcja omówiona w punkcie 4.4.2.2.

4.5.2.1.6. Podsumowanie.

Jest to najmniej złożony algorytm ewolucyjny służący do znajdowania reduktów. Jego niewątpliwą zaletą jest właśnie prostota oraz szybkość działania. Algorytm ten jest skuteczny tylko dla tablic decyzyjnych o małej liczbie atrybutów. Dla złożonych problemów nie znajduje on jednak w ogóle rozwiązania.

4.5.2.2. Wariant z kodowaniem permutacyjnym.

Istnieją także algorytmy, które przeszukują przestrzeń bazującą na reprezentacji permutacyjnej (Per). Przedstawię teraz opis takiego rozwiązania.

4.5.2.2.1. Reprezentacja chromosomu.

Chromosom jest reprezentowany w postaci permutacji liczb naturalnych $[1,..N]$, gdzie $N=|A|$. Każda liczba oznacza jeden atrybut ze zbioru atrybutów tablicy decyzyjnej. Permutacja odzwierciedla pewną sekwencję atrybutów. Można ją interpretować na dwa sposoby:

- kolejność, w jakiej będą dodawane atrybuty (od lewej do prawej) do zbioru pustego ($\emptyset$) tak, aby otrzymać zbiór rozróżniający obiekty tablicy decyzyjnej (zgodnie z zachłannym algorytmem dodającym atrybuty wykorzystywanym do znajdowania pojedynczego reduktu opisanym w podrozdziale 4.6.2.1)
- kolejność, w jakiej będą odejmowane atrybuty (od lewej do prawej) od zbioru wszystkich atrybutów (A) tak, aby otrzymać zbiór rozróżniający obiekty tablicy decyzyjnej (zgodnie z zachłannym algorytmem odejmującym atrybuty wykorzystywanym do znajdowania pojedynczego reduktu opisanym w podrozdziale 4.6.2.2)

4.5.2.2.2. Inicjalizacja populacji bazowej.

Każdy osobnik populacji bazowej jest początkowo inicjalizowany posortowaną tablicą liczb od 1 do N. Następnie dla każdego chromosomu wybieramy losowo pewną ilość par genów, które są ze sobą przestawiane (tak aby uzyskać losową permutację).

4.5.2.2.3. Operator krzyżowania.

Krzyżowane zachodzi zgodnie ze schematem PMX (krzyżowania z częściowym odwzorowaniem), którego dokładny opis można znaleźć w [1.19]. Ogólna idea tego procesu polega na tym, że chromosomy rodzicielskie są cięte w dwóch losowo wybranych miejscach, a następnie, powstałe fragmenty chromosomów są łączone i traktowane jako transpozycje. Osobniki potomne powstają poprzez przestawienie genów osobników rodzicielskich właśnie zgodnie z tymi transpozycjami. Z dwóch osobników rodzicielskich dostajemy dwa osobniki potomne, które podobnie jak rodzice, są także permutacjami.

4.5.2.2.4. Operator mutacji.

Mutacja chromosomu polega na wylosowaniu z rozkładem równomiernym dwóch różnych genów a następnie na zamianie ich miejscami.

4.5.2.2.5. Funkcja przystosowania.

Funkcja przystosowania zależy tylko i wyłącznie od wielkości (ilości elementów) zbioru rozróżniającego i wyraża się wzorem: $fitness(B) = \frac{1}{card(Q)}$, gdzie Q jest zbiorem rozróżniającym utworzonym z permutacji B (przy użyciu odpowiedniego algorytmu zachłannego).

4.5.2.2.6. Podsumowanie.

Algorytm ten, w porównaniu do poprzednika, ma znacznie większą złożoność obliczeniową, na którą w największym stopniu wpływa funkcja przystosowania, a konkretnie tworzenie zbioru rozróżniającego z permutacji atrybutów. Niewątpliwą zaletą tego algorytmu jest fakt, iż dla dowolnej tablicy decyzyjnej, zawsze znajdzie on przynajmniej jeden redukt (dokładnie jeden zbiór rozróżniający, z którego następnie można otrzymać redukt usuwając atrybuty zbędne).

4.5.3. Algorytm wykorzystujący dekompozycję przestrzeni.

Korzyści płynące z dekompozycji przestrzeni przedstawionej w punkcie 4.2 można wykorzystać w problemie wyznaczania maksymalnej liczby (wszystkich) reduktów. Ogólna idea algorytmu polega na tym, że w każdej iteracji poszukujemy reduktów o konkretnej długości równej k.

Parametr k przyjmuje kolejno wartości od 1 do $|C|$. Każda iteracji polega więc na wykonaniu algorytmu, który wyznacza redukty o zadanej liczbie atrybutów. W tym celu można wykorzystać przykładowe metody zaprezentowane w podrozdziale 4.4.

Algorytm 4.3: Algorytm wyznaczający wszystkie redukty względne tablicy decyzyjnej oparty na dekompozycji przestrzeni.

Wejście: tablica decyzyjna: $TD = (U, C, \{d\}, V, f)$, ilość iteracji N

Wyjście: zbiór reduktów TD: $RED_{TD}(C, d)$

Start procedury:

[1] Wyznacz macierz odróżnialności $M(TD,d)$ dla tablicy decyzyjnej TD

[2] Wyznacz wektor odróżnialności $W(TD,d)$ i usuń z niego elementy powtarzające się.

[3] $k := 1$

[4] **Dopóki** $k < |C|$ **powtarzaj**

[5] Wykonaj algorytm znajdujący redukty o długości k i zapisz rezultat do G_k

[6] $RED_{TD}(C,d) := \bigcup_{k=1..|C|} G_k$

Koniec procedury

Niewątpliwą zaletą tego algorytmu jest fakt, iż nie przeszukujemy całej przestrzeni za jednym razem. Poszukiwania prowadzimy niezależnie w każdej podprzestrzeni, co drastycznie zmniejsza ilość potencjalnych rozwiązań w każdej iteracji. Dodatkowo algorytm ten jest łatwo parametryzowany i może być wykorzystany do różnych celów (np. wyznaczenia m najkrótszych reduktów). Ponadto charakteryzuje się dowolnością w zastosowaniu metody do znajdowania reduktów o zadanej długości k. Zaproponowane zostały przykładowe dwie metody, ale można zastosować także inne.

4.6. Metody wyznaczania minimalnego reduktu.

4.6.1. Algorytm dokładny.

Algorytm dokładny przedstawiony w podrozdziale 4.5.1. znajduje wszystkie redukty względne dla zadanej tablicy decyzyjnej. Mając zbiór wszystkich reduktów możemy więc wyznaczyć redukt najkrótszy. Warto podkreślić, że ta metoda to jedyny sposób na znalezienie optymalnego rozwiązania rozpatrywanego problemu. Inne algorytmy zaprezentowane w tym podrozdziale są heurystykami. Nie mamy więc żadnej gwarancji, że wyznaczone przez te metody redukty będą rzeczywiście najkrótsze.

4.6.2. Proste algorytmy heurystyczne.

Zostaną tutaj zaprezentowane dwie proste metody zachłanne. Ich niewątpliwą zaletą jest mała złożoność obliczeniowa. Otrzymywane wyniki mogą jednak dość znacznie odbiegać od optimum.

4.6.2.1. Algorytm zachłanny dodający atrybuty

Jest to prosty algorytm heurystyczny, który uzyskuje redukt dodając do pustego zbioru atrybutów atrybuty zgodnie z pewną permutacją P, będącą permutacją zbioru wszystkich atrybutów tablicy decyzyjnej. (Typowo permutacja P zawiera atrybuty w takiej kolejności, w jakiej występują one w tablicy decyzyjnej.) Algorytm kończy działanie, gdy tworzony zbiór będzie zbiorem rozróżniającym wszystkie obiekty tablicy decyzyjnej.

Algorytm 4.4: Algorytm wyznaczający jeden redukt względny tablicy decyzyjnej dodający atrybuty.

Wejście: tablica decyzyjna: $TD = (U, C, \{d\}, V, f)$, permutacja atrybutów tablicy decyzyjnej P

Wyjście: jeden redukt: $R_{TD}(C,d)$

Start procedury:

[1] Wyznacz macierz odróżnialności $M(TD,d)$ dla tablicy decyzyjnej TD

[2] Wyznacz wektor odróżnialności $W(TD,d)$ i usuń z niego elementy powtarzające się.

[3] $R := \emptyset$

[4] **Dla każdego** $a \in P$ **powtarzaj**:

[5] $R := R \cup \{a\}$

[6] **Jeżeli** $CS(R,d)=CS(C,d)$ **to przerwij** pętlę [4]

[7] Usuń atrybuty zbędne z R

[8] $R_{TD}(C,d) := R$

Koniec procedury

Złożoność obliczeniowa: $O(card(W(TD,d)) * card(C))$

4.6.2.2. Algorytm zachłanny odejmujący atrybuty

Działanie tego algorytmu jest zbliżone do działania poprzedniego algorytmu. W tym przypadku algorytm startuje ze zbiorem zawierającym wszystkie atrybuty i odejmuje od niego kolejne elementy zgodnie z permutacją P, usuwając w ten sposób atrybuty zbędne. Algorytm kończy działanie po usunięciu wszystkich atrybutów zbędnych z pierwotnego zbioru.

Algorytm 4.4: Algorytm wyznaczający jeden redukt względny tablicy decyzyjnej dodający atrybuty.

Wejście: tablica decyzyjna: $TD = (U, C, \{d\}, V, f)$, permutacja atrybutów tablicy decyzyjnej P

Wyjście: jeden redukt: $R_{TD}(C,d)$

Start procedury:

[1] Wyznacz macierz odróżnialności $M(TD,d)$ dla tablicy decyzyjnej TD

[2] Wyznacz wektor odróżnialności $W(TD,d)$ i usuń z niego elementy powtarzające się.

[3] $R := C$

[4] **Dla każdego** $a \in P$ **powtarzaj**:

[5] **Jeżeli** $CS(R \setminus \{a\}, d) = CS(C,d)$ **to** $R := R \setminus \{a\}$

[6] Usuń atrybuty zbędne z R

[7] $R_{TD}(C,d) := R$

Koniec procedury

Złożoność obliczeniowa: $O(card(W(TD,d)) * card(C))$

4.6.3. Algorytm Johnsona.

Algorytm Johnsona jest jednym z najpopularniejszych algorytmów aproksymacyjnych służących do wyznaczania minimalnego („prawie” minimalnego) reduktu.

Metoda ta stara się uzyskać rozwiązanie dodając do pustego zbioru atrybutów te atrybuty, które w danym kroku rozróżniają jak największą liczbę par obiektów (tzn. te atrybuty, które jak najczęściej występują w elementach wektora odróżnialności tablicy decyzyjnej).

Algorytm Johnsona można sprowadzić do algorytmu Greedy-Set-Cover znajdującego minimalne pokrycie zbiorami, który to w każdym kroku do pokrycia dodaje zbiór zawierający jak najwięcej jeszcze niepokrytych elementów.

Algorytm 4.6: Algorytm Johnsona wyznaczający jeden redukt względny tablicy decyzyjnej.

Wejście: tablica decyzyjna: $TD = (U, C, \{d\}, V, f)$

Wyjście: jeden redukt: $R_{TD}(C,d)$

Start procedury:

[1] Wyznacz macierz odróżnialności $M(TD,d)$ dla tablicy decyzyjnej TD

[2] Wyznacz wektor odróżnialności $W(TD,d)$ i usuń z niego elementy powtarzające się.

[3] **Dla każdego** elementu H z wektora odróżnialności $W(TD,d)$ **powtarzaj**:

[4] **Jeżeli** $H=\{a\}$ dla pewnego $a \in C$ **to** $R := R \cup \{a\}$

[5] Usuń z W(TD,d) elementy mające niepuste przecięcie z R

[6] **Dopóki** $W(TD,d) \neq \emptyset$ **powtarzaj**:

[7] Niech a będzie atrybutem należącym do największej liczby elementów z $W(TD,d)$

[8] $R := R \cup \{a\}$

[9] Usuń z $W(TD,d)$ elementy mające niepuste przecięcie z R

[10] Usuń atrybuty zbędne z R

[11] $R_{TD}(C,d) := R$

Koniec procedury

Złożoność obliczeniowa: pętla [6] powoduje, że złożoność obliczeniowa tego algorytmu wynosi:

$$O(card(W(TD,d))^2 * card(C) * card(R_{TD}(C,d)))$$

W porównaniu do dwóch poprzednich algorytmów heurystycznych ta metoda jest bardziej zaawansowana. Nie dodaje ona atrybutów na ślepo (od lewej do prawej) tylko kieruje się pewnymi racjonalnymi przesłankami w celu wyznaczenia optymalnego rozwiązania.

4.6.4. Ulepszony algorytm Johnsona.

Klasyczny algorytm Johnsona wyznacza minimalny redukt w stosunkowo krótkim czasie, jednak jakość tego rozwiązania dość często może nie być zadowalająca. Dlatego chciałbym zaproponować pewne modyfikacje do tej metody, które powodują, że otrzymywany wynik jest lepszy od tego uzyskiwanego przy pomocy klasycznego algorytmu i mniej odbiega od rozwiązania optymalnego. Ulepszenie polega na tym, że w kroku numer 2 algorytmu 4.6 dodatkowo wyznaczamy optymalną postać wektora odróżnialności (czyli de facto obliczamy zbiór odróżnialności). Zabieg ten co prawda poprawia rozwiązanie ale powoduje także wzrost złożoności obliczeniowej algorytmu. Na złożoność ulepszonego algorytmu wpływa głównie proces minimalizacji wektora. Złożoność algorytmu wynosi więc: $O(card(U)^2 * card(C)^2)$.

4.6.5. Algorytmy ewolucyjne.

Do wyznaczenia minimalnego reduktu względnego danej tablicy decyzyjnej można wykorzystać także algorytmy przedstawione w rozdziale 4.5.2. Znajdują one maksymalną liczbę reduktów, więc wystarczy przejrzeć otrzymane zbiory i wybrać z nich element minimalny.

4.6.6. Algorytm wykorzystujący dekompozycję przestrzeni.

Dekompozycja przestrzeni poszukiwań w tym przypadku powoduje, że problem optymalizacyjny zostaje przekształcony w serię problemów decyzyjnych. Czyli zamiast szukać odpowiedzi na pytanie: „Jaki jest minimalny redukt względny danej tablicy decyzyjnej ?” szukamy odpowiedzi na serię pytań postaci: „Czy istnieje redukt względny dla danej tablicy decyzyjnej o długości k (zawierający dokładnie k atrybutów) ?”. Problem decyzyjny ma znacznie mniejszą przestrzeń poszukiwań niż problem optymalizacyjny, co powoduje zmniejszenie złożoności czasowej tego algorytmu.

Algorytm kolejno poszukuje reduktu o długości k, przy czym k przyjmuje wartości od 1 do $|C|$. W każdej iteracji uruchamiany jest algorytm, który wyznacza jeden redukt o długości k lub mniejszej z użyciem np. jednej z metod z punkty 4.4. Każda następna wartość parametru k jest o jeden mniejsza od długości reduktu znalezionego w ostatnim kroku.

Algorytm 4.7: Algorytm wyznaczający wszystkie redukty względne tablicy decyzyjnej oparty na dekompozycji przestrzeni.

Wejście: tablica decyzyjna: $TD = (U, C, \{d\}, V, f)$, ilość iteracji N

Wyjście: jeden redukt TD: $R_{TD}(C,d)$

Start procedury:

[1] Wyznacz macierz odróżnialności $M(TD,d)$ dla tablicy decyzyjnej TD

[2] Wyznacz wektor odróżnialności $W(TD,d)$ i usuń z niego elementy powtarzające się.

[3] $R := C$

[4] $k := |C|$

[5] **Dopóki** $k>0$ **powtarzaj**

[6] Wykonaj algorytm znajdujący jeden redukt o długości k i zapisz rezultat do R

[7] **Jeżeli** $R = \emptyset$ **to przerwij** pętlę [5]

[8] $k := len(R)-1$

[9] $R_{TD}(C,d) := R$

Koniec procedury

4.7. Porównanie algorytmów wyznaczających redukty.

4.7.1. Wstęp.

W tej części pracy zostaną porównane algorytmy służące do wyznaczania reduktów, zaprezentowane w poprzednich podrozdziałach. Porównanie ma na celu przedstawienie charakterystycznych cech poszczególnych metod oraz znalezienie najefektywniejszej z nich.
Eksperymenty zostały przeprowadzone na kilku zbiorach danych, których właściwości oraz krótki opis znajduje się w tabeli 4.1.

Nazwa	Liczba obiektów / atrybutów	Opis
ZOO	101 / 17	o dane dotyczące zwierząt o źródło: UCI Repository ([2.2]) o dyskretne wartości atrybutów, siedem klas decyzyjnych o wszystkie obiekty są unikalne
PRINCE-TON	62 / 2000	o dane dotyczące ekspresji genów ludzkich komórek o źródło: University of Princeton o ciągłe wartości atrybutów, przed testami zostały poddane binarnej dyskretyzacji, dwie klasy decyzyjne o wszystkie obiekty są unikalne
UNI_1	1200 / 100	o dane wygenerowane z rozkładem równomiernym o binarnych wartościach atrybutów i klas decyzyjnych
UNI_2	250 / 1500	o dane wygenerowane z rozkładem równomiernym o binarnych wartościach atrybutów i klas decyzyjnych o wszystkie obiekty są unikalne

SONAR	150	o dane pochodzące z pomiarów sonarem o źródło: University of Princeton o ciągłe wartości atrybutów z przedziału $(0;1)$, przed testami zostały poddane binarnej dyskretyzacji; dwie klasy decyzyjne o wszystkie obiekty są unikalne
	61	

Tabela 4.1. Właściwości badanych zbiorów danych w problemie wyznaczania reduktów tablicy decyzyjnej.

Warto podkreślić, że testy z użyciem algorytmów deterministycznych (np. dokładnego, czy też Johnsona) zostały przeprowadzone jednokrotnie, zaś testy z użyciem algorytmów niedeterministycznych (algorytmów genetycznych i losowych) zostały powtórzone 10 krotnie dla różnych ziaren generatora liczb losowych. Prezentowane wyniki są średnią arytmetyczną ze wszystkich powtórzeń danego algorytmu. Zabieg ten jest konieczny, gdyż rezultat pojedynczego wykonania algorytmu niedeterministycznego jest zmienną losową i nie powinien być traktowany jako wiarygodne rozwiązanie.

W przypadku algorytmów genetycznych została przeprowadzona dodatkowa seria testów w celu dobrego ich sparametryzowania. Uzyskana w ten sposób wiedza pozwoliła na dobranie parametrów typu: prawdopodobieństwo mutacji, krzyżowania czy tez liczność populacji w taki sposób, aby algorytmy dawały jak najlepsze rozwiązania.

4.7.2. Problem wyznaczania wszystkich reduktów.

Algorytmy wyznaczające wszystkie redukty względne tablicy decyzyjnej zostały porównane pod kątem ilości znalezionych reduktów oraz czasu działania. Opis badanych algorytmów znajduje się w tabeli 4.2.

Lp.	Nazwa algorytmu	Opis
1.	Wyczerpujący	Algorytm przedstawiony w podrozdziale: 4.5.1.
2.	Genetyczny binarny	Algorytm przedstawiony w podrozdziale: 4.5.2.1.
3.	Genetyczny permutacyjny	Algorytm przedstawiony w podrozdziale: 4.5.2.2. Reprezentacja chromosomu dodająca atrybuty.
4.	Genetyczny binarny z dekompozycją	Algorytm wykorzystujący dekompozycję przestrzeni przedstawiony w podrozdziale 4.5.3. W każdym kroku (do znajdowania reduktów o długości k) uruchamiany jest algorytm genetyczny przedstawiony w podrozdziale 4.4.2.

Tabela 4.2. Opis badanych algorytmów dla problemu wyznaczania wszystkich reduktów względnych tablicy decyzyjnej.

Wyniki eksperymentów dla badanych zbiorów danych znajdują się w tabeli 4.3. Znak '*' został użyty, gdy dany algorytm nie znalazł rozwiązania w akceptowalnym czasie.

Zbiór danych	Algorytm	Liczba reduktów	Czas działania [s]
ZOO	Wyczerpujący	33,0	2
	Genetyczny binarny	23,7	2
	Genetyczny permutacyjny	31,9	99
	Genetyczny binarny z dekompozycją	32,5	33
PRINCETON	Wyczerpujący	*	*
	Genetyczny binarny	0,0	140
	Genetyczny permutacyjny	279,5	142
	Genetyczny binarny z dekompozycją	1186,0	154
SONAR	Wyczerpujący	*	*
	Genetyczny binarny	3,5	120
	Genetyczny permutacyjny	46,2	800
	Genetyczny binarny z dekompozycją	50,4	170

Tabela 4.3. Wyniki testów algorytmów wyznaczających wszystkie redukty względne tablicy decyzyjnej.

Otrzymane wyniki pokazują, że algorytm wyczerpujący jest zupełnie nieprzydatny w przypadku, gdy mamy do czynienia z dużymi zbiorami danych. Jego duża złożoność obliczeniowa praktycznie uniemożliwia rozwiązania bardziej złożonych problemów. W takich przypadkach niezastąpione okazują się być algorytmy przybliżone, które znajdują rozwiązania w akceptowalnym czasie.

Warto zauważyć, że dla zbioru danych PRINCETON algorytm genetyczny binarny nie znalazł żadnego reduktu. Jest to spowodowane faktem, iż przeszukuje on całą przestrzeń na raz. W przypadku zbioru z tak dużą ilością atrybutów (2000) taka metodologia okazuje się być zupełnie nieefektywna. Dla tego zbioru algorytm genetyczny permutacyjny, znajduje w tym samym czasie, prawie cztery razy mniej reduktów niż algorytm, który wykorzystuje dekompozycję przestrzeni.

W przypadku zbioru danych SONAR uwidaczniają się różnice czasowe. Algorytm genetyczny permutacyjny potrzebował ponad 5 razy więcej czasu na znalezienie porównywalnej liczby reduktów co algorytm z dekompozycją przestrzeni. Tak duża różnica czasowa wynika z charakteru algorytmu permutacyjnego. W każdym kroku wykonuje on kilkukrotnie więcej obliczeń wartości $|CS(B,d)|$ (gdzie B jest zbiorem atrybutów warunkowych zaś d atrybutem decyzyjnym, zgodnie z definicją 2.25) niż algorytm z kodowaniem binarnym. Warto wspomnieć, że koszt czasowy obliczenia $|CS(B)|$ rośnie w sposób kwadratowy wraz ze wzrostem ilości obiektów tablicy decyzyjnej.

4.7.3. Problem wyznaczania minimalnego reduktu.

Algorytmy wyznaczające minimalny redukt względny tablicy decyzyjnej zostały porównane pod kątem długości najkrótszego znalezionego reduktu oraz czasu działania.

Lp.	Nazwa algorytmu	Opis
1.	Wyczerpujący	Algorytm przedstawiony w podrozdziale: 4.5.1.
2.	Johnsona	Algorytm przedstawiony w podrozdziale: 4.6.3.
3.	Johnsona rozszerzony	Algorytm przedstawiony w podrozdziale: 4.6.4.
4.	Zachłanny dodający	Algorytm przedstawiony w podrozdziale: 4.6.2.1.

5.	Zachłanny odejmujący	Algorytm przedstawiony w podrozdziale: 4.6.2.2.
6.	Losowy z dekompozycją	Algorytm wykorzystujący dekompozycję przestrzeni przedstawiony w podrozdziale 4.6.6.. W każdym kroku uruchamiany jest algorytm losowy przedstawiony w podrozdziale 4.4.1.
7.	Genetyczny binarny	Algorytm przedstawiony w podrozdziale: 4.5.2.1.
8.	Genetyczny permutacyjny	Algorytm przedstawiony w podrozdziale: 4.5.2.2. Reprezentacja chromosomu dodająca atrybuty.
9.	Genetyczny binarny z dekompozycją	Algorytm wykorzystujący dekompozycję przestrzeni przedstawiony w podrozdziale 4.6.6.. W każdym kroku uruchamiany jest algorytm genetyczny przedstawiony w podrozdziale 4.4.2..

Tabela 4.4. Opis badanych algorytmów dla problemu wyznaczania minimalnego reduktu względnego tablicy decyzyjnej.

Wyniki eksperymentów dla badanych zbiorów danych znajdują się w tabeli 4.5. Podobnie jak w poprzednich testach znak '*' użyto w celu zaznaczenia, że rozpatrywany algorytm nie znalazł rozwiązania w rozsądnym czasie.

Zbiór danych	**Algorytm**	**Długość najkrótszego**	**Czas działania [s]**
ZOO	Wyczerpujący	5,0	2
	Johnsona	6,0	1
	Johnsona rozszerzony	5,0	1
	Zachłanny dodający	5,0	1
	Zachłanny odejmujący	6,0	1
	Losowy z dekompozycją	5,0	1
	Genetyczny binarny	5,0	2
	Genetyczny permutacyjny	5,0	99
	Genetyczny binarny z dekompozycją	5,0	2

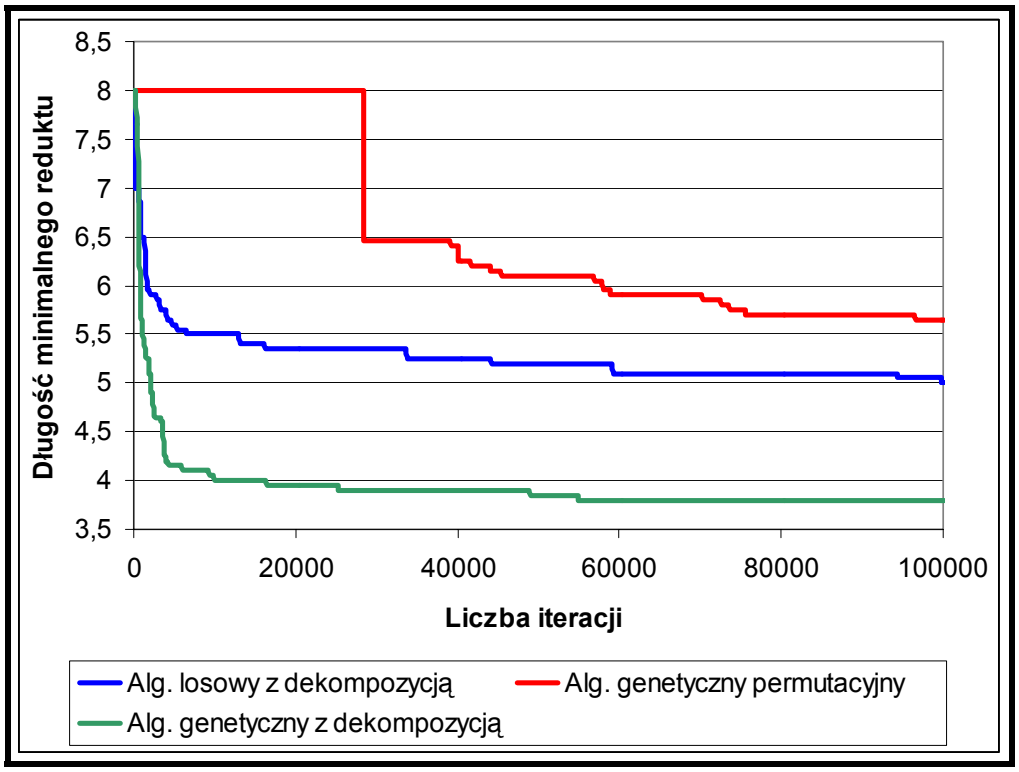
PRINCETON	Wyczerpujący	*	*
	Johnsona	10,0	1
	Johnsona rozszerzony	5,0	1
	Zachłanny dodający	8,0	1
	Zachłanny odejmujący	10,0	1
	Losowy z dekompozycją	5,2	2
	Genetyczny binarny	0,0	8
	Genetyczny permutacyjny	4,4	142
	Genetyczny binarny z dekompozycją	3,8	7
UNI_1	Wyczerpujący	*	*
	Johnsona	22,0	160
	Johnsona rozszerzony	21,0	68400
	Zachłanny dodający	23,0	27
	Zachłanny odejmujący	22,0	33
	Losowy z dekompozycją	22,0	14404
	Genetyczny binarny	0,0	12010
	Genetyczny permutacyjny	21,8	12523
	Genetyczny binarny z dekompozycją	19,7	13212
UNI_2	Wyczerpujący	*	*
	Johnsona	15,0	17,0
	Johnsona rozszerzony	12,0	157
	Zachłanny dodający	14,0	4
	Zachłanny odejmujący	15,0	12
	Losowy z dekompozycją	12,8	558
	Genetyczny binarny	0,0	400
	Genetyczny permutacyjny	13,3	512
	Genetyczny binarny z dekompozycją	11,2	545

SONAR	Wyczerpujący	*	*
	Johnsona	12,0	1
	Johnsona rozszerzony	10,0	12
	Zachłanny dodający	14,0	1
	Zachłanny odejmujący	12,0	1
	Losowy z dekompozycją	10,5	12
	Genetyczny binarny	11,5	120
	Genetyczny permutacyjny	9,5	800
	Genetyczny binarny z dekompozycją	8,0	25

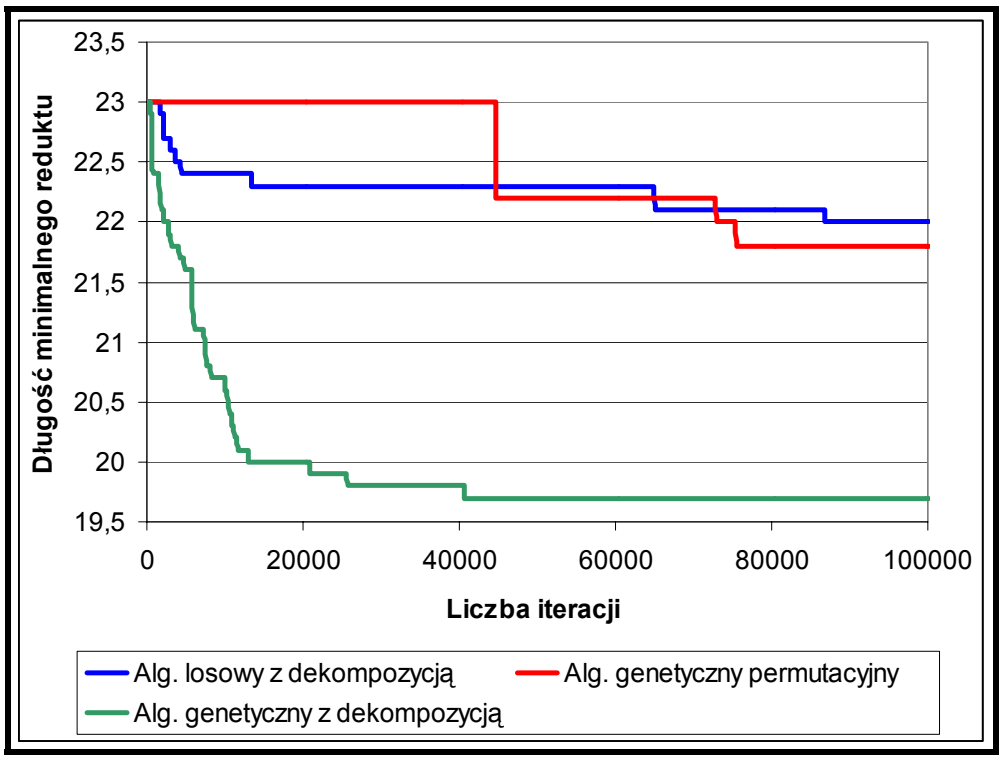
Tabela 4.5. Wyniki testów algorytmów wyznaczających minimalny redukt względny tablicy decyzyjnej.

Uzyskane rezultaty pokazują przewagę algorytmu wykorzystującego dekompozycję przestrzeni w połączeniu z algorytmem genetycznym z kodowaniem binarnym. W każdym przypadku wynik osiągnięty przez ten algorytm jest nie gorszy (a w czterech przypadkach jest lepszy) niż uzyskany przez inne algorytmy. Uwidacznia się także mała efektywność prostych algorytmów zachłannych (dodającego atrybuty oraz odejmującego) jak również standardowej postaci algorytmu Johnsona. Okazuje się, że zaproponowane usprawnienie tego ostatniego sprawdza się i w każdym przypadku rozszerzony algorytm Johnsona daje lepsze wyniki niż jego oryginalna wersja.

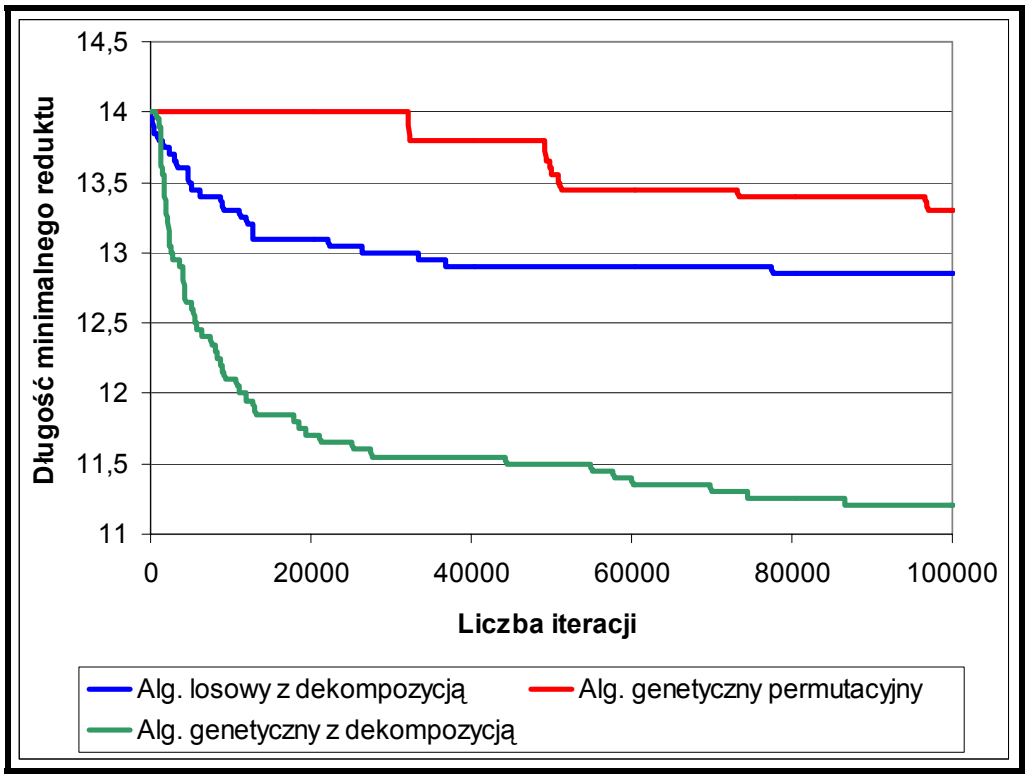
Na kolejnych wykresach została przedstawiona szybkość zbieżności wybranych algorytmów w funkcji iteracji. Warto zaznaczyć, że w tym przypadku pod pojęciem iteracji rozumiemy obliczenie parametru $|CS()|$ dla pewnego zbioru atrybutów, czyli jednokrotne przejrzenie całej tablicy nierozróżnialności. Takie sformułowanie iteracji wprowadza pewną elementarną operację, która jest dobrą wielkością odniesienia dla poszczególnych algorytmów. Wprowadzenie takiej miary jest dodatkowo spowodowane faktem, iż jedna „klasyczna" iteracja każdego z trzech badanych tu algorytmów różni się znacznie pod względem złożoności czasowej. I tak jedna „klasyczna" iteracja algorytmu losowego to tylko jednokrotne obliczenie wartości $|CS()|$, podczas gdy w przypadku algorytmów genetycznych w każdej „klasycznej" iteracji wykonywane jest wielokrotnie obliczenie wartości $|CS()|$ (w zależności od wielkości populacji, zastosowanego kodowania oraz funkcji przystosowania).



Wykres 4.1. Szybkość zbiegania algorytmów wyznaczających minimalny redukt względny dla zbioru danych PRINCETON.



Wykres 4.2. Szybkość zbiegania algorytmów wyznaczających minimalny redukt względny dla zbioru danych UNI_1.



Wykres 4.3. Szybkość zbiegania algorytmów wyznaczających minimalny redukt względny dla zbioru danych UNI_2.

Z zaprezentowanych wykresów wynika jednoznacznie, że algorytm genetyczny z dekompozycją przestrzeni zbiega znacznie szybciej od pozostałych. Skutkuje to faktem, iż algorytm ten pozwala uzyskać zadowalające rozwiązanie wcześniej (szybciej) niż pozostałe badane metody heurystyczne.

4.8. Praktyczne wykorzystanie reduktów.

Redukty służą redukcji systemów informacyjnych i tablic decyzyjnych. Ich cechą charakterystyczną jest fakt, iż zachowują one przybliżenia dolne i górne klas decyzyjnych. Wygenerowane reguły decyzyjne ze zredukowanej tablicy decyzyjnej, zgodnie ze schematem jeden obiekt jest jedną reguła, mają takie samo wsparcie i zaufanie co reguły wygenerowane z oryginalnej tablicy. Redukty umożliwiają więc usunięcie pewnej grupy atrybutów bez utraty

istotnych informacji. Oszczędzają więc miejsce przy jednoczesnym zachowaniu zdolności poprawnej klasyfikacji.

Redukty są wykorzystywane w wielu dziedzinach nauki. Potencjalne zastosowanie np. w medycynie może dotyczyć rozwiązania problemu poprawnego diagnozowanie chorób przy minimalizacji koniecznych do wykonania badań. Minimalizacja może mieć różny charakter np. minimalizacja kosztów badania czy też ich szkodliwości na organizm człowieka. Lekarz mając do dyspozycji redukty, czyli grupy badań, których wykonanie umożliwi mu zdiagnozowanie choroby może wybrać taki zestaw, który będzie najlepszy w danych okolicznościach dla danego pacjenta.

5. Metody wyznaczania współczynników wiarygodności dla obiektów tablicy decyzyjnej.

5.1. Wstęp.

W danych, a w szczególności takich, które nie są pozyskiwane w sposób automatyczny, tylko są wprowadzane do systemu komputerowego przez człowieka, mogą pojawić się różnego rodzaju błędy. Źródłem takich błędów mogą być powody techniczne (np. „literówka" przy wpisywaniu danych do formularza) jak również merytoryczne (np. wprowadzanie błędnych informacji wynikających z niewiedzy lub celowego działania). Przekłamane informacje, które będziemy nazywać także szumem informacyjnym, mogą mieć bardzo zły wpływ na wyniki uzyskiwane przy pomocy różnych metod odkrywania wiedzy. Szum informacyjny jest szczególnie niepożądany w takich dziedzinach wiedzy jak np. medycyna, gdzie każda pomyłka może mieć tragiczne skutki dla zdrowia i życia pacjenta. Dlatego wykorzystuje się szereg różnych metod, których celem jest minimalizacja ilości błędów w procesie akwizycji danych, a tym samym podwyższaniu ich jakości oraz wiarygodności. Ogólnie metody te są nazywane metodami służącymi do walidacji (sprawdzania poprawności) danych

Metody walidacji danych możemy podzielić na różne grupy, biorąc pod uwagę następujące kryteria:

- miejsce walidacji: walidacja po stronie klienta[4] oraz serwera[5]

  Do walidacji po stronie klienta zaliczymy te wszystkie metody, które mogą być wykorzystane do sprawdzania poprawności danych w momencie ich wprowadzania, czyli już na poziomie formularza z aplikacji klienckiej. Przykłady: sprawdzanie poprawności typów danych oraz zakresu zmienności wartości.

  Walidacja po stronie serwera polega na tym, że czynności walidacyjne są wykonywane w innym miejscu niż na poziomie aplikacji klienckiej. Przykłady: sprawdzanie więzów integralności przez serwer baz danych.

[4] przez klienta będziemy rozumieć aplikację służącą do wprowadzania i prezentowania danych
[5] przez serwer będziemy rozumieć aplikację przechowującą i przetwarzającą dane

- rodzaj walidacji: walidacja prosta oraz złożona

  Do prostych metod walidacji zaliczymy te metody, które w procesie sprawdzania poprawności danego obiektu korzystają tylko i wyłącznie z badanego obiektu. Przykłady te same co w przypadku walidacji po stronie klienta.

  Złożone metody walidacji wykorzystują zaś także inne informację/wiedzę niż proste metody np. wartości atrybutów innych obiektów z tablicy decyzyjnej.
- tryb walidacji: walidacja boolowska oraz rozmyta

  Metody walidacji typu boolowskiego jako wynik zwracają jedną z dwóch wartości: „obiekt poprawny" lub „obiekt niepoprawny".

  Metody walidacji typu rozmytego jako wynik zwracają pewną znormalizowaną wartość np. należącą do przedziału: $<0\%, 100\%>$. Wartość ta określa stopień poprawności/niepoprawności obiektu. W przeciwieństwie do metod boolowskich nie dostajemy więc jednoznacznego rezultatu walidacji tylko pewną informację, która może być następnie wykorzystana przez eksperta.

W rozdziale tym zostały zaproponowane trzy przykładowe algorytmy służące do wyznaczania współczynników wiarygodności dla poszczególnych obiektów tablicy decyzyjnej. Metody te wykorzystują zarówno teorię zbiorów przybliżonych jak i elementy statystyki. Są one przykładem algorytmów służących do walidacji danych. Biorąc pod uwagę ich cechy możemy je zaliczyć do walidacji typu:

- zaawansowanej

  Przy wyznaczaniu współczynnika wiarygodności dla danego obiektu brany jest pod uwagę charakter wszystkich obiektów należących do danej tablicy decyzyjnej (np. uwzględniane są pewne własności statystyczno-częstotliwościowe jak również aproksymacje klas decyzyjnych).
- zwracającej wartości rozmyte

  Wyznaczane współczynniki wiarygodności dla poszczególnych obiektów są znormalizowane i należą do przedziału $<0,1>$, gdzie wartość „0" – oznacza obiekt o małej wiarygodności, zaś wartość „1" – obiekt o dużej wiarygodności.

- wykonywanej po stronie serwera.

  Walidacja musi być wykonywana po stronie serwera na względu na to, iż w procesie wykorzystuje się wszystkie dane z tablicy decyzyjnej.

5.2. Metoda statystyczno-częstotliwościowa.

Działanie tego algorytmu opiera się na analizie częstotliwościowej wartości poszczególnych atrybutów obiektów tablicy decyzyjnej. Każdy z obiektów jest porównywany z obiektami, które należą do tej samej klasy decyzyjnej. Współczynnik wiarygodności jest tym większy im więcej jest obiektów z tymi samymi wartościami poszczególnych atrybutów co aktualnie rozpatrywany obiekt.

Algorytm 5.1: Algorytm statystyczno-częstotliwościowy.

Wejście: tablica decyzyjna: $TD = (U, C, \{d\}, V, f)$

Wyjście: wektor współczynników wiarygodności dla obiektów TD: $W_{TD}(d)[]$

Start procedury:

[1] **Dla każdego** obiektu $u \in U$ **powtarzaj**

[2] $pomWsp := 0$

[3] **Dla każdego** atrybutu $a \in C$ **powtarzaj**

[4] $K := \{x \in U : f(x,d) = f(u,d)\}$

[5] $W := \{x \in U : f(x,d) = f(u,d) \wedge f(x,a) = f(u,a)\}$

[6] $pomWsp := pomWsp + (card(W)/card(K))$

[7] $W_{TD}(d)[u] := pomWsp/card(C)$

Koniec procedury

Złożoność obliczeniowa: $O(card(U)^2 * card(C))$

Formalnie można zapisać, że współczynnik wiarygodności $W_{TD}(u,d)$, liczony przy pomocy metody statystyczno-częstotliwościowej, dla obiektu $u \in U$ tablicy decyzyjnej $TD = (U,C,\{d\},V,f)$ wyraża się wzorem:

$$W_{TD}(u,d) = \frac{\sum_{a \in C} \frac{card(W_{u,d,a})}{card(K_{u,d})}}{card(C)}$$

$$gdzie:$$

$$K_{u,d} = \{y \in U : f(y,d) = f(u,d)\}$$

$$W_{u,d,a} = \{y \in U : f(y,d) = f(u,d) \wedge f(y,a) = f(u,a)\}$$

Niewątpliwą zaletą tej metody jest jej prostota i mała złożoność obliczeniowa. Ponadto metoda ta w ten sam sposób traktuje wszystkie obiekty tablicy decyzyjnej tzn. na współczynnik wiarygodności danego obiektu w tym samym stopniu wpływają wszystkie obiekty tablicy.
Algorytm ten można poddać pewnej modyfikacji. W kroku [3] można brać pod uwagę redukt danej tablicy decyzyjnej a nie cały zbiór atrybutów. Możliwe jest również wyznaczenie dla każdego obiektu serii współczynników przy użyciu pewnego zbioru reduktów. Wynikowy współczynnik dla danego obiektu może być średnią arytmetyczną z tej serii.

Przykład 5.1.

Tabela 5.1 przedstawia tablicę decyzyjną z tabeli 2.1 wraz z wyznaczonymi współczynnikami wiarygodności metodą statystyczno-częstotliwościową dla poszczególnych obiektów tablicy.

Pacjent	Ból głowy (g)	Ból mięsni (m)	Temperatura (t)	Grypa (c)	W_{TD}
1	nie	tak	wysoka	tak	0,58
2	tak	nie	wysoka	tak	0,42
3	tak	tak	bardzo wysoka	tak	0,58
4	nie	tak	bardzo wysoka	tak	0,58
5	tak	nie	wysoka	nie	0,50
6	nie	tak	normalna	nie	0,50

Tabela 5.1. Tablica decyzyjna z wyznaczonymi współczynnikami wiarygodności dla obiektów metodą statystyczno częstotliwościową..

5.3. Metoda oparta na przybliżeniach klas decyzyjnych.

Metoda ta jest bardziej zaawansowana od poprzedniej. Wykorzystuje ona pojęcia aproksymacji zbiorów pochodzące z teorii zbiorów przybliżonych. Wprowadźmy definicję dodatkowych pojęć wykorzystywanych w tym algorytmie. Powiemy, że obiekt u tablicy decyzyjnej $TD = (U,C,\{d\},V,f)$ jest:

- ***obiektem niekonfliktowym względem klasy decyzyjnej s***, jeżeli:
  $u \in POS_C(X) \cup NEG_C(X)$, gdzie $X := \{y \in U : f(y,d) = s\}$
- ***obiektem konfliktowym względem klasy decyzyjnej s***, jeżeli:
  $u \in BN_C(X)$, gdzie $X := \{y \in U : f(y,d) = s\}$

Ogólna idea tego algorytmu opiera się na przekonaniu, że obiekty niekonfliktowe powinny mieć większe współczynniki wiarygodności niż obiekty konfliktowe. W przypadku konfliktu algorytm robi dodatkową analizę częstotliwościową wartości poszczególnych atrybutów uwzględniając przy tym obszary pozytywne i negatywne danej klasy decyzyjnej.

Algorytm ten analizuje po kolei wszystkie obiekty. Dla każdego z obiektów uniwersum rozpatruje wszystkie klasy decyzyjne. Każda klasa decyzyjna s względem której obiekt jest niekonfliktowy podwyższa współczynnik wiarygodności obiektu o wartość równą 1. Jeżeli zaś okaże się, że badany obiekt u jest konfliktowy względem klasy decyzyjnej s to wiarygodność takiego obiektu powiększa się o wartość pomocniczego współczynnika wiarygodności $pomWsp$ wyznaczanego zgodnie z następującymi zasadami:

- Jeżeli badany obiekt należy do klasy decyzyjnej s, to tworzymy zbiór K zawierający wszystkie obiekty należące do obszaru pozytywnego klasy decyzyjnej s.
- Jeżeli badany obiekt nie należy do klasy decyzyjnej s, to tworzymy zbiór K zawierający wszystkie obiekty należące do obszaru negatywnego klasy decyzyjnej s.
- Następnie dla każdego atrybutu warunkowego wyznaczamy zbiór W zawierający te obiekty, które należą do zbioru K oraz mają tę samą wartość atrybutu a co obiekt u. W każdym kroku do $pomWsp$ dodajemy wartość równą wyrażeniu: $card(K)/card(W)$. Na końcu normalizujemy pomocniczy współczynnik wiarygodności dzieląc go przez wartość $card(C)$ tzn. $(pomWsp=pomWsp/card(C))$.

Kolejnym krokiem jest znormalizowanie współczynników wiarygodności obiektów do przedziału <0,1>. Każdy współczynnik jest dzielony przez liczbę wartości atrybutu decyzyjnego dla danej tablicy decyzyjnej $(card(V_d))$.

Należy zwrócić uwagę, że algorytm ten będzie dawał bardzo złe rezultaty, jeżeli większość obiektów będzie należeć do granicy poszczególnych klas decyzyjnych.

Algorytm 5.2: Algorytm oparty na przybliżeniach klas decyzyjnych.

Wejście: tablica decyzyjna: $TD = (U, C, \{d\}, V, f)$

Wyjście: wektor współczynników wiarygodności dla obiektów TD: $W_{TD}(d)[]$

Start procedury:

[1] $S := \{v \in V_d\}$

[2] **Dla każdego** obiektu $u \in U$ **powtarzaj**

[3] $wsp:=0$

[4] **Dla każdej** klasy decyzyjnej $s \in S$ **powtarzaj**

[5] $X := \{y \in U : f(y,d) = s\}$

[6] **Jeżeli** $u \in POS_C(X)$ **lub** $u \in NEG_C(X)$ **to**

[7] $wsp:=wsp+1$

[8] **W przeciwnym przypadku**

[9] $pomWsp:=0$

[10] **Jeżeli** $f(u,d) = s$ **to**

[11] $K := \{y \in U : y \in POS_C(X)\}$

[12] **Dla każdego atrybutu** $a \in C$ **powtarzaj**

[13] $W := \{y \in U : y \in POS_C(X) \wedge f(y,a) = f(u,a)\}$

[14] $pomWsp := pomWsp + (card(W)/card(K))$

[15] **W przeciwnym przypadku**

[16] $K := \{y \in U : y \in NEG_C(X)\}$

[17] **Dla każdego atrybutu** $a \in C$ **powtarzaj**

[18] $W := \{y \in U : y \in NEG_C(X) \wedge f(y,a) = f(u,a)\}$

[19] $pomWsp := pomWsp + (card(W)/card(K))$

[20] $wsp:=wsp+(pomWsp/card(C))$

[21] $W_{TD}(d)[u]:=wsp/card(S)$

Koniec procedury

Złożoność obliczeniowa: $O(card(U)^2 * card(V_d) * card(C))$

Współczynnik wiarygodności $W_{TD}(u,d)$ dla obiektu $u \in U$ tablicy decyzyjnej $TD = (U,C,\{d\},V,f)$, liczony metodą opartą na przybliżeniach klas decyzyjnych, wyraża się wzorem:

$$W_{TD}(u,d) = \frac{\sum_{s \in S} def(u,d,s)}{card(S)}$$

gdzie :

$$S = \{v \in V_d\}$$

$$X_{d,s} = \{u \in U : f(u,d) = s\}$$

$$Kpos = \{u \in U : u \in POS_C(X)\}$$

$$Kneg = \{u \in U : u \in NEG_C(X)\}$$

$$Wpos_{u,a} = \{y \in U : y \in POS_C(X) \wedge f(y,a) = f(u,a)\}$$

$$Wneg_{u,a} = \{y \in U : y \in NEG_C(X) \wedge f(y,a) = f(u,a)\}$$

$$def(u,d,s) = \begin{cases} 1 & dla \quad u \in \{POS_C(X) \cup NEG_C(X)\} \\ \frac{\sum_{a \in A} pom(u,d,s,a)}{card(C)} & dla \quad u \in BN_C(X) \end{cases}$$

$$pom(u,d,s,a) = \begin{cases} \frac{card(Wpos_{u,a})}{card(Kpos)} & dla \quad f(u,d) = s \\ \frac{card(Wneg_{u,a})}{card(Kneg)} & dla \quad f(u,d) \neq s \end{cases}$$

Podobnie jak w poprzednim przypadku, również w tym algorytmie w krokach [12] i [17] można zamiast całego zbioru atrybutów brać pod uwagę jego redukt względny.

Przykład 5.2.

Tabela 5.2 przedstawia tablicę decyzyjną z tabeli 2.1 wraz z wyznaczonymi współczynnikami wiarygodności metodą opartą na przybliżeniach klas decyzyjnych dla poszczególnych obiektów tablicy.

Pacjent	Ból głowy (g)	Ból mięsni (m)	Temperatura (t)	Grypa (c)	W_{TD}
1	nie	tak	wysoka	tak	1,00
2	tak	nie	wysoka	tak	0,22
3	tak	tak	bardzo wysoka	tak	1,00
4	nie	tak	bardzo wysoka	tak	1,00
5	tak	nie	wysoka	nie	0,00
6	nie	tak	normalna	nie	1,00

Tabela 5.2. Tablica decyzyjna z wyznaczonymi współczynnikami wiarygodności dla obiektów metodą opartą na przybliżeniach klas decyzyjnych.

5.4. Metoda hybrydowa.

Algorytm hybrydowy łączy w sobie cechy dwóch poprzednich algorytmów. Jego działanie jest bardzo podobne do działania poprzedniego algorytmu. Jedyna różnicą jest sposób rozpatrywania przypadków, gdy obiekt jest obiektem konfliktowym względem klasy decyzyjnej. W takim przypadku wiarygodność jest obliczana podobnie jak w przypadku metody statystyczno-częstotliwościowej i zależy od ilości wystąpień wartości danego atrybutu w obiektach uniwersum należących do tej samej klasy decyzyjnej (nie uwzględniana natomiast przynależności obiektu do obszaru pozytywnego bądź negatywnego klasy decyzyjnej).

Algorytm ten jest wolny od wady poprzedniego algorytmu polegającej na uzyskiwaniu niskich współczynników wiarygodności (dość często równych zeru) dla przypadków gdy stosunkowo duża ilość obiektów należy do granicy poszczególnych klas decyzyjnych.

Algorytm 5.3: Algorytm hybrydowy.

Wejście: tablica decyzyjna: $TD = (U, C, \{d\}, V, f)$

Wyjście: wektor współczynników wiarygodności dla obiektów TD: $W_{TD}(d)[]$

Start procedury:

[1] $S := \{v \in V_d\}$

[2] **Dla każdego** obiektu $u \in U$ **powtarzaj**

[3] $wsp:=0$

[4] **Dla każdej** klasy decyzyjnej $s \in S$ **powtarzaj**

[5] $X := \{y \in U : f(y,d) = s\}$

[6] **Jeżeli** $u \in POS_C(X)$ **lub** $u \in NEG_C(X)$ **to**

[7] $wsp:=wsp+1$

[8] **W przeciwnym przypadku**

[9] $pomWsp:=0$

[10] **Jeżeli** $f(u,d)=s$ **to**

[11] **Dla każdego atrybutu** $a \in C$ **powtarzaj**

[12] $K := \{x \in U : f(x,d) = f(u,d)\}$

[13] $W := \{x \in U : f(x,d) = f(u,d) \wedge f(x,a) = f(u,a)\}$

[14] $pomWsp := pomWsp + (card(W)/card(K))$

[15] $wsp:=wsp+(pomWsp/card(C))$

[16] $W_{TD}(d)[u]:=wsp/card(S)$

Koniec procedury

Złożoność obliczeniowa: $O(card(U)^2 * card(V_d) * card(C))$

Współczynnik wiarygodności $W_{TD}(u,d)$ dla obiektu $u \in U$ tablicy decyzyjnej $TD = (U, C, \{d\}, V, f)$, liczony metodą hybrydową, wyraża się wzorem:

$$W_{TD}(u,d) = \frac{\sum_{s \in S} def(u,d,s)}{card(S)}, \text{ gdzie :}$$

$$S = \{v \in V_d\}$$

$$X_{d,s} = \{u \in U : f(u,d) = s\}$$

$$Kpos = \{u \in U : u \in POS_C(X)\}$$

$$Kneg = \{u \in U : u \in NEG_C(X)\}$$

$$Wpos_{u,a} = \{y \in U : y \in POS_C(X) \wedge f(y,a) = f(u,a)\}$$

$$Wneg_{u,a} = \{y \in U : y \in NEG_C(X) \wedge f(y,a) = f(u,a)\}$$

$$def(u,d,s) = \begin{cases} 1 & dla \quad u \in \{POS_C(X) \cup NEG_C(X)\} \\ \dfrac{\sum_{a \in C} \dfrac{card(W_{u,d,a})}{card(K_{u,d})}}{card(C)} & dla \qquad u \in BN_C(X) \end{cases}$$

$$K_{u,d} = \{y \in U : f(y,d) = f(u,d)\}$$

$$W_{u,d,a} = \{y \in U : f(y,d) = f(u,d) \wedge f(y,a) = f(u,a)\}$$

Modyfikacja przedstawiona dla poprzednich algorytmów może być także zastosowana w tym przypadku. W kroku [11] algorytmu możemy rozpatrywać redukt względny a nie cały zbiór atrybutów warunkowych tablicy decyzyjnej. .

Przykład 5.3.

Tabela 5.3 przedstawia tablicę decyzyjną z tabeli 2.1 wraz z wyznaczonymi współczynnikami wiarygodności metodą hybrydową dla poszczególnych obiektów tablicy.

Pacjent	Ból głowy (g)	Ból mięsni (m)	Temperatura (t)	Grypa (c)	W_{TD}
1	nie	tak	wysoka	tak	1,00
2	tak	nie	wysoka	tak	0,21
3	tak	tak	bardzo wysoka	tak	1,00
4	nie	tak	bardzo wysoka	tak	1,00
5	tak	nie	wysoka	nie	0,25
6	nie	tak	normalna	nie	1,00

Tabela 5.3. Tablica decyzyjna z wyznaczonymi współczynnikami wiarygodności dla obiektów metodą hybrydową.

5.5. Badanie oraz porównanie zaproponowanych metod.

5.5.1. Ogólne właściwości metod.

Tabela 5.4 przedstawia porównanie wartości współczynników wiarygodności dla obiektów z tablicy decyzyjnej z tabeli 2.1 wyznaczonych za pomocą wcześniej zaprezentowanych metod. Tabela ta zawiera również średni współczynnik wiarygodność dla każdej z metod (średnia arytmetyczna współczynników dla wszystkich obiektów wyznaczonych przy pomocy danej metody – **Średnia dla metody**) oraz średni współczynnik wiarygodności dla każdego z obiektów (średnia arytmetyczna współczynników wyznaczonych przy pomocy wszystkich metody dla danego obiektu – **Średnia dla obiektu**).

Pacjent	Metoda statystyczno-częstotliwościowa	Metoda oparta na przybliżeniach klas decyzyjnych	Metoda hybrydowa	Średnia dla obiektu
1	0,58	1,00	1,00	0,86
2	0,42	0,22	0,21	0,28
3	0,58	1,00	1,00	0,86
4	0,58	1,00	1,00	0,86
5	0,50	0,00	0,25	0,25
6	0,50	1,00	1,00	0,83
Średnia dla metody	0,53	0,70	0,74	0,66

Tabela 5.4. Porównanie wartości współczynników wiarygodności dla obiektów tablicy decyzyjnej z tabeli 2.1.

Metoda druga oraz trzecia wyznaczyła obiektom o numerach 1,4,5 i 6 najwyższe możliwe współczynniki wiarygodności (równe wartości *1*). Obiekty o numerach 2 i 5 uzyskały stosunkowo niskie wartości współczynnika. Jest to spowodowane faktem, że to właśnie one powodują, że tablica ta jest niedeterministyczna. Tabela 5.5 przedstawia współczynniki wiarygodności obliczone dla tej samej tablicy decyzyjnej po usunięciu z niej obiektu o numerze 5. Po tym zabiegu tablica decyzyjna staje się deterministyczna co powoduje, że metoda oparta na

przybliżeniach klas decyzyjnych oraz hybrydowa dają wszystkim obiektom współczynniki o wartości 1.

Pacjent	Metoda statystyczno-częstotliwościowa	Metoda oparta na przybliżeniach klas decyzyjnych	Metoda hybrydowa	Średnia dla obiektu
1	0,58	1,00	1,00	0,86
2	0,42	1,00	1,00	0,81
3	0,58	1,00	1,00	0,86
4	0,58	1,00	1,00	0,86
6	1,00	1,00	1,00	1,00
Średnia dla metody	0,63	1,00	1,00	0,88

Tabela 5.5. Porównanie wartości współczynników wiarygodności dla obiektów tablicy decyzyjnej z tabeli 2.1 po usunięciu obiektu o numerze 5.

Rozpatrzmy teraz tablicę decyzyjną z tabeli 5.6, pochodzącą z [1.25], [1.28]. Zawiera ona m obiektów, trzy atrybuty warunkowe oraz jeden atrybut decyzyjny przyjmujący dwie wartości. Obiekty są więc podzielone pomiędzy dwie klasy decyzyjne. Pierwsze n obiektów posiada takie same wartości atrybutów warunkowych przy czym $n-1$ obiektów należy do klasy f zaś jeden obiekt do klasy g. Podobnie pozostałe $m-n$ obiektów ma takie same wartości atrybutów warunkowych przy czym $m-n-1$ obiektów należy do klasy decyzyjnej g, zaś jeden obiekt do klasy decyzyjnej f. Nie ma więc wątpliwości, że tak zdefiniowana tablica decyzyjna jest tablicą niedeterministyczną. Co więcej przybliżenia dolne poszczególnych klas decyzyjnych są równe zbiorowi pustemu, zaś górne całemu uniwersum.

Obiekt	Atrybut 1	Atrybut 2	Atrybut 3	Decyzja	Grupa
1	a	a	a	f	GRUPA 1
2	a	a	a	f	
...	a	a	a	f	
n-1	a	a	a	f	
n	a	a	a	g	
n+1	b	b	b	f	

n+2	b	b	b	g	GRUPA 2
…	b	b	b	g	
m-1	b	b	b	g	
m	b	b	b	g	

Tabela 5.6. Tablica decyzyjna.

W celu przetestowania algorytmów na tym zbiorze danych przeanalizujemy dwa warianty (W): pierwszy (W_1) dla $n=20$ i $m=40$ oraz drugi (W_2) dla $n=10$ i $m=25$. Wyznaczone współczynniki wiarygodności dla poszczególnych obiektów za pomocą różnych metod znajdują się w tabeli 5.7.

Obiekt	**Algorytm**					
	częstotliwościowy		**oparty na przybliżeniach**		**hybrydowy**	
	W_1	**W_2**	**W_1**	**W_2**	**W_1**	**W_2**
1..n-1	0,947	0,889	0,000	0,000	0,475	0,450
n	0,000	0,000	0,000	0,000	0,025	0,033
n+1	0,000	0,000	0,000	0,000	0,025	0,050
n+2..m	0,947	0,929	0,000	0,000	0,475	0,467
Średnia dla metody	0,900	0,840	0,000	0,000	0,453	0,427

Tabela 5.7. Współczynniki wiarygodności dla obiektów z tablicy decyzyjnej z tabeli 5.5.

Algorytm numer 2 przyznał wszystkim obiektom wiarygodność równą 0,000. Jest to spowodowane faktem, że obszary pozytywne oraz negatywne poszczególnych klas decyzyjnych z tablic decyzyjnej z tabeli 5.5 są równe $\emptyset$ (zbiorowi pustemu). Jak już wspomniałem wcześniej algorytm oparty na przybliżeniach nie nadaje się do tego typu tablic decyzyjnych.

Pozostałe dwa algorytmy przyznały słusznie niski współczynnik dla obiektów o numerach n i $n+1$. Są to obiekty, które potencjalnie mają wprowadzoną błędną klasę decyzyjną (taki wniosek można wysnuć analizując tabelę 5.5). Porównując dwa warianty W_1 oraz W_2 warto zauważyć, że oba algorytmy przyznały mniejsze wartości współczynników wiarygodności dla obiektów z grupy 1 ($1…n-1$) oraz z grupy 2 ($n+2…m$) w wariancie W_2 niż w W_1.

Zachowanie to jest jak najbardziej poprawne, gdyż w wariancie *W_2* zmniejszone zostały wartości parametrów *n* oraz *m*, a co za tym idzie tablica decyzyjna zawierała mniejszą liczbę tych samych przypadków z danej grupy obiektów. Algorytm hybrydowy uwzględnił ten fakt także w ocenie obiektów o numerach *n* i *n+1* w wariancie *W_2* zwiększając ich wartości współczynników wiarygodności w stosunku do wariantu *W_1*. Kolejną istotną obserwacją jest fakt, iż wiarygodności obiektów grupy 1 oraz grupy 2 były równe w wariancie *W_1* natomiast są różne w wariancie *W_2*. Dzieje się tak dlatego, że w wariancie *W_1* obie grupy zawierały taką samą liczbę obiektów, natomiast w wariancie *W_2* grupa 2 zawierała więcej obiektów niż grupa 1, więc automatycznie ma większą wiarygodność.

Można także zaobserwować, że średnia wiarygodność dla metody jest większa w wariancie *W_2* niż w wariancie *W_1*. Jest to spowodowane faktem, iż w wariancie *W_2* zostały usunięte obiekty z grupy 1 oraz 2, co spowodowało wzrost wagi obiektów o numerach *n* i *n+1* będących źródłem niedeterminizmu w tablicy decyzyjnej z 0,05 na 0,08.

Oczywistym jest fakt, że po usunięciu z tablicy decyzyjnej obiektów o numerach *n* i *n+1* wszystkie algorytmy wyznaczą dla wszystkich obiektów współczynniki o wartościach równych *1* (uzyskana w ten sposób tablica będzie w pełni deterministyczna).

5.5.2. Detekcja szumu informacyjnego.

5.5.2.1. Wprowadzenie.

Kolejny eksperyment będzie polegał na wstawieniu do tablicy decyzyjnej nowych elementów i określeniu ich wpływu na współczynniki wiarygodności wszystkich obiektów. Jako źródło danych zostanie użyta tablica decyzyjna, która była już wykorzystywana w rozdziałach 4.7.1 i 4.7.2, o nazwie ZOO. Dla przypomnienia warto wspomnieć, że tablica ta zawiera 101 obiektów oraz 16 atrybutów warunkowych i 1 atrybut decyzyjny. W swojej pierwotnej postaci (bez żadnych modyfikacji) jest ona tablicą deterministyczną.

Na początku zostaną wyznaczone współczynniki wiarygodności dla oryginalnej tablicy decyzyjnej ZOO. Tabela 5.8 przedstawia minimalne, średnie oraz maksymalne wartości współczynników wiarygodności dla obiektów z tej tablicy wyznaczonych za pomocą badanych algorytmów. Ponieważ tablica decyzyjna jest deterministyczna to metoda oparta na przybliżeniach klas decyzyjnych oraz metoda hybrydowa wyznaczyły dla wszystkich obiektów wartość współczynnika równą *1*, stąd też minimalna wartość jest równa średniej a zarazem maksymalnej wartości dla tych dwóch algorytmów.

Współczynnik wiarygodności dla obiektów	**Metoda statystyczno-częstotliwościowa**	**Metoda oparta na przybliżeniach klas decyzyjnych**	**Metoda hybrydowa**	**Średnia**
Minimalny	0,69	1,00	1,00	0,90
Średni	0,87	1,00	1,00	0,96
Maksymalny	0,95	1,00	1,00	0,98

Tabela 5.8. Porównanie wartości współczynników wiarygodności dla tablicy decyzyjnej o nazwie ZOO.

Tabela 5.9 przedstawia liczbę obiektów, tablicy decyzyjnej ZOO, posiadających wartość współczynnika wiarygodności, wyznaczoną różnymi metodami, należącą do określonych przedziałów.

Wartość współczynnika wiarygodności	**Procent wszystkich obiektów [%] (liczba obiektów)**			
	Metoda statystyczno-częstotliwościowa	**Metoda oparta na przybliżeniach klas decyzyjnych**	**Metoda hybrydowa**	**Średnia**
<0.0 ; 0,4)	0,0 (0)	0,0 (0)	0,0 (0)	0,0 (0)
<0.4 ; 0,6)	0,0 (0)	0,0 (0)	0,0 (0)	0,0 (0)
<0,6 ; 0.8)	13,9 (14)	0,0 (0)	0,0 (0)	4,7 (4,7)
<0,8 ; 0.9)	62,4 (63)	0,0 (0)	0,0 (0)	20,8 (21,0)
<0,9 ; 1.0>	23,8 (24)	100,0 (101)	100,0 (101)	74,6 (75,3)

Tabela 5.9. Porównanie liczby obiektów z określoną wartością współczynnika wiarygodności dla tablicy decyzyjnej o nazwie ZOO.

Rezultaty tych eksperymentów zostaną wykorzystane w kolejnym podrozdziale.

5.5.2.2. Dodawanie obiektów z przekłamanymi wartościami atrybutów decyzyjnych (klas decyzyjnych).

Celem poniższego eksperymentu jest sprawdzenie czy zaprezentowane metody mogą być wykorzystane do podstawowej detekcji przekłamań w klasach decyzyjnych. Podstawowe założenia eksperymentu:

- wykorzystanie tablicy decyzyjnej ZOO; obiekty do niej należące będą traktowane jako poprawne (o dużej wiarygodności)
- wprowadzenie niedeterminizmu do tablicy poprzez wstawienie nowych obiektów, będących kopią obiektów z uniwersum lecz ze zmienioną klasą decyzyjną; obiekty wstawiane będą traktowane jako obiekty błędne (o małej wiarygodności)
- wyznaczenie współczynników wiarygodności dla nowej tablicy decyzyjnej zaproponowanymi algorytmami

Na potrzeby eksperymentu rozpatrywana tablica decyzyjna (ZOO) zostanie teraz poddana pewnym modyfikacją. Z uniwersum zostanie wybranych w sposób losowy n obiektów, które ze zmienioną klasą decyzyjną (nowa wartość atrybutu decyzyjnego będzie wylosowana ze zbioru wartości stanowiących dziedzinę tego atrybutu) zostaną wstawione, jako nowe obiekty, do tablicy ZOO. Powstanie w ten sposób nowa, niedeterministyczna tablica decyzyjna ZOO_n, która będzie posiadać $101+n$ obiektów oraz tyle samo atrybutów co oryginalna tablica.
Ponieważ eksperyment zakłada losowość pewnych zdarzeń, więc dla każdego n zostanie on powtórzony 10 krotnie, oczywiście z różnymi ziarnami generatora liczb losowych, a przedstawione poniżej wyniki będą średnią arytmetyczną obliczoną ze wszystkich pojedynczych iteracji. Wszystkie losowana będą wykonywane z rozkładem równomiernym.
Eksperyment został przeprowadzony dla parametru n równego kolejno: 0, 5, 10, 15 oraz 20, czyli wynikowa tablica decyzyjna zawierała odpowiednio 101, 106, 111, 116 oraz 121 obiektów.

Tabela 5.10 przedstawia porównanie jakości oraz dokładności aproksymacji rodziny klas decyzyjnych (rodzin zbiorów złożonych z obiektów należących do tej samej klasy decyzyjnej) zmodyfikowanych tablic ZOO . Można zaobserwować, że wraz ze wzrostem liczby wstawianych do tablicy obiektów maleje zarówno dokładność jak i jakość aproksymacji. Jest to spowodowane

faktem, iż każdy nowy obiekt powoduje zmniejszenie liczby obiektów należących do przybliżenia dolnego a zwiększenie liczby obiektów należących do granicy przybliżenia danej klasy decyzyjnej.

Rodzaj miary	*n*				
	0	**5**	**10**	**15**	**20**
Dokładność aproksymacji	1,00	0,70	0,53	0,36	0,30
Jakość aproksymacji	1,00	0,82	0,70	0,56	0,50

Tabela 5.10. Porównanie jakości oraz dokładności aproksymacji klas decyzyjnych dla rodziny zmodyfikowanych tablic decyzyjnych ZOO.

W tabeli 5.11 zostały porównane dokładności aproksymacji pojedynczych klas decyzyjnych dla różnych wartości *n*. Podobnie jak w poprzednim przypadku wraz ze wzrostem liczby wstawianych obiektów maleje dokładność aproksymacji klas decyzyjnych.

Numer klasy decyzyjnej	**Dokładność aproksymacji**				
	n				
	0	**5**	**10**	**15**	**20**
1	1,00	0,68	0,52	0,36	0,30
2	1,00	0,76	0,50	0,38	0,28
3	1,00	0,88	0,67	0,48	0,46
4	1,00	0,63	0,47	0,37	0,25
5	1,00	0,92	0,69	0,50	0,44
6	1,00	0,86	0,61	0,40	0,44
7	1,00	0,69	0,63	0,50	0,38

Tabela 5.11. Porównanie jakości oraz dokładności aproksymacji klas decyzyjnych dla rodziny zmodyfikowanych tablic decyzyjnych ZOO.

Tabele od 5.12 do 5.14 przedstawiają porównanie wartości współczynników wiarygodności wyznaczonych przy pomocy zaproponowanych algorytmów dla rodziny zmodyfikowanych tablic decyzyjnych ZOO. Dla każdego algorytmu został wyznaczony minimalna, średnia oraz maksymalna wartość współczynnika wiarygodności dla trzech grup obiektów: wszystkich obiektów, tylko obiektów poprawnych (czyli tych, które występowały w oryginalnej tablicy decyzyjnej) oraz obiektów wstawionych (czyli tych dodanych do oryginalnej tablicy, ze zmienioną klasą decyzyjną).

Metoda statystyczno-częstotliwościowa					
Wartość współczynnika dla obiektów	*n*				
	0	**5**	**10**	**15**	**20**
Minimalna dla wszystkich	0,69	0,48	0,52	0,52	0,49
Minimalna dla poprawnych	0,69	0,68	0,67	0,66	0,66
Minimalna dla wstawionych	---	0,48	0,52	0,52	0,49
Średnia dla wszystkich	0,87	0,84	0,81	0,79	0,77
Średnia dla poprawnych	0,87	0,85	0,84	0,82	0,81
Średnia dla wstawionych	---	0,54	0,57	0,56	0,55
Maksymalna dla wszystkich	0,95	0,95	0,91	0,90	0,93
Maksymalna dla poprawnych	0,95	0,95	0,91	0,90	0,93
Maksymalna dla wstawionych	---	0,57	0,61	0,62	0,58

Tabela 5.12. Porównanie wartości współczynnika wiarygodności wyznaczonego metodą statystyczno-częstotliwościową dla rodziny zmodyfikowanych tablic decyzyjnych ZOO.

Metoda oparta na przybliżeniach klas decyzyjnych					
Wartość współczynnika dla obiektów	*n*				
	0	**5**	**10**	**15**	**20**
Minimalna dla wszystkich	1,00	0,85	0,83	0,82	0,69
Minimalna dla poprawnych	1,00	0,96	0,93	0,93	0,73
Minimalna dla wstawionych	---	0,85	0,83	0,82	0,69

Średnia dla wszystkich	1,00	0,98	0,97	0,96	0,94
Średnia dla poprawnych	1,00	0,99	0,98	0,97	0,96
Średnia dla wstawionych	---	0,86	0,85	0,84	0,83
Maksymalna dla wszystkich	1,00	1,00	1,00	1,00	1,00
Maksymalna dla poprawnych	1,00	1,00	1,00	1,00	1,00
Maksymalna dla wstawionych	---	0,86	0,87	0,86	0,79

Tabela 5.13. Porównanie wartości współczynnika wiarygodności wyznaczonego metodą opartą na przybliżeniach klas decyzyjnych dla rodziny zmodyfikowanych tablic decyzyjnych ZOO.

Metoda hybrydowa					
Wartość współczynnika dla obiektów	*n*				
	0	**5**	**10**	**15**	**20**
Minimalna dla wszystkich	1,00	0,78	0,73	0,70	0,69
Minimalna dla poprawnych	1,00	0,91	0,83	0,85	0,73
Minimalna dla wstawionych	---	0,78	0,73	0,70	0,69
Średnia dla wszystkich	1,00	0,97	0,94	0,91	0,88
Średnia dla poprawnych	1,00	0,98	0,95	0,94	0,91
Średnia dla wstawionych	---	0,79	0,77	0,75	0,73
Maksymalna dla wszystkich	1,00	1,00	1,00	1,00	1,00
Maksymalna dla poprawnych	1,00	1,00	1,00	1,00	1,00
Maksymalna dla wstawionych	---	0,79	0,79	0,78	0,78

Tabela 5.14. Porównanie wartości współczynnika wiarygodności wyznaczonego metodą hybrydową dla rodziny zmodyfikowanych tablic decyzyjnych ZOO.

Tabele od 5.15 do 5.17 przedstawiają porównanie liczby obiektów, zmodyfikowanych tablic decyzyjnych ZOO, posiadających wartość współczynnika wiarygodności należącą do określonych przedziałów. Są one uzupełnieniem informacji, które zostały zawarte w poprzednich tabelach.

Metoda statystyczno-częstotliwościowa					
Wartość współczynnika wiarygodności	**Procent wszystkich obiektów [%] (liczba obiektów)**				
	n				
	0	**5**	**10**	**15**	**20**
<0.0 ; 0,4)	0,0 (0)	0,0 (0)	0,0 (0)	0,0 (0)	0,0 (0)
<0.4 ; 0,6)	0,0 (0)	4,7 (5)	7,2 (8)	9,5 (11)	20,9 (20)
<0,6 ; 0.8)	13,9 (14)	14,2 (15)	18,0 (20)	24,1 (28)	37,6 (36)
<0,8 ; 0.9)	62,4 (63)	67,0 (71)	70,3 (78)	66,4 (77)	65,7 (63)
<0,9 ; 1.0>	23,8 (24)	14,2 (15)	4,5 (5)	0,0 (0)	2,1 (2)

Tabela 5.15. Porównanie liczby obiektów z określoną wartością współczynnika wiarygodności wyznaczonego metodą statystyczno-częstotliwościową dla rodziny zmodyfikowanych tablic decyzyjnych ZOO.

Metoda oparta na przybliżeniach klas decyzyjnych					
Wartość współczynnika wiarygodności	**Procent wszystkich obiektów [%] (liczba obiektów)**				
	n				
	0	**5**	**10**	**15**	**20**
<0.0 ; 0,4)	0,0 (0)	0,0 (0)	0,0 (0)	0,0 (0)	0,0 (0)
<0.4 ; 0,6)	0,0 (0)	0,0 (0)	0,0 (0)	0,0 (0)	0,0 (0)
<0,6 ; 0.8)	0,0 (0)	0,0 (0)	0,0 (0)	0,0 (0)	0,0 (0)
<0,8 ; 0.9)	0,0 (0)	4,7 (5)	9,0 (10)	12,9 (15)	31,3 (30)
<0,9 ; 1.0>	100,0 (101)	95,3 (101)	91,0 (101)	87,1 (101)	94,9 (91)

Tabela 5.16. Porównanie liczby obiektów z określoną wartością współczynnika wiarygodności wyznaczonego metodą opartą na przybliżeniach klas decyzyjnych dla rodziny zmodyfikowanych tablic decyzyjnych ZOO.

Metoda hybrydowa					
Wartość współczynnika wiarygodności	**Procent wszystkich obiektów [%] (liczba obiektów)**				
	n				
	0	**5**	**10**	**15**	**20**
<0.0 ; 0,4)	0,0 (0)	0,0 (0)	0,0 (0)	0,0 (0)	0,0 (0)
<0.4 ; 0,6)	0,0 (0)	0,0 (0)	0,0 (0)	0,0 (0)	0,0 (0)
<0,6 ; 0.8)	0,0 (0)	4,7 (5)	9,0 (10)	12,9 (15)	31,3 (30)
<0,8 ; 0.9)	0,0 (0)	0,0 (0)	9,0 (10)	19,8 (23)	22,9 (22)
<0,9 ; 1.0>	100,0 (101)	95,3 (101)	82,0 (91)	67,2 (78)	72,0 (69)

Tabela 5.17. Porównanie liczby obiektów z określoną wartością współczynnika wiarygodności wyznaczonego metodą hybrydową dla rodziny zmodyfikowanych tablic decyzyjnych ZOO.

Analizując otrzymane wyniki można zauważyć, iż w przypadku wszystkich trzech algorytmów średnie współczynniki wiarygodności dla wszystkich obiektów maleją wraz ze wzrostem liczby dodawanych obiektów. Każdy nowy obiekt, który jest de facto błędnym obiektem, obniża wiarygodność pozostałych. Fakt ten potwierdza również malejąca wartość średniego współczynnika dla obiektów poprawnych.

Najistotniejszym jednak wnioskiem wynikającym z przeprowadzenia eksperymentu jest fakt, iż algorytmy poprawnie rozpoznały błędne obiekty (czyli te nowo wstawione). Zauważmy, iż w przypadku każdego algorytmu maksymalna wartość współczynnika dla obiektów wstawionych jest prawie zawsze mniejsza od minimalnej wartości współczynnika dla obiektów poprawnych. Wszystkie wstawione obiekty miały więc prawie zawsze mniejszy współczynnik wiarygodności niż jakikolwiek obiekt zaliczany do obiektów poprawnych. Wyjątek od tej reguły stanowi wariant dla $n=20$ dla algorytmów opartego na przybliżeniach oraz hybrydowego. Należy jednak pamiętać, że w tym przypadku liczba niepoprawnych obiektów jest dość duża i stanowi prawie 20% elementów całego uniwersum, co w znacznym stopniu utrudnia ich wykrycie.

Powyższy przykład pokazuje, że zaproponowane algorytmy wyznaczania współczynników wiarygodności dla obiektów mogą być bardzo pomocne w usuwaniu elementów, które są źródłem niedeterminizmu i mogą potencjalnie zawierać błędy (być przekłamane) w wartościach klas decyzyjnych. Tego rodzaju przekłamania powinny być wyeliminowane przed dalszą analizą danych, gdyż mogą mieć szczególnie niekorzystny wpływ na proces generacji reguł decyzyjnych wykonywany zgodnie z algorytmami przedstawionymi m.in. w [1.13], [1.14], [1.33], [1.36].

5.6. Praktyczne wykorzystanie zaprezentowanych metod.

Zaprezentowane metody mogą zostać wykorzystane jako element systemu do analizy danych. Pomagają one w wykryciu potencjalnego szumu informacyjnego w tablicach decyzyjnych. Mogą być także wykorzystane do znajdowania obiektów nietypowych, w jakiś sposób wyróżniających się (odbiegających) od pozostałych obiektów tablicy decyzyjnej. Przedstawione algorytmy mogą dokonywać walidacji danych w sposób automatyczny (w tym przypadku użytkownik definiuje graniczny poziom wiarygodności, poniżej którego obiekty są usuwane) lub też interaktywny z użytkownikiem (w tym przypadku system na bieżąco przelicza współczynniki wiarygodności dla obiektów po usunięciu poszczególnych pozycji przez użytkownika)

6. ARES Rough Set Exploration System.

6.1. Wstęp.

ARES Rough Set Exploration System jest aplikacją umożliwiającą analizę danych z wykorzystaniem teorii zbiorów przybliżonych. W programie tym zostało zaimplementowanych większość algorytmów przedstawionych w poprzednich rozdziałach.
Podstawowa funkcjonalność aplikacji umożliwia:

- wczytanie tablicy decyzyjnej
- przeprowadzenie dyskretyzacji tablicy decyzyjnej różnymi metodami
- wyznaczenie obszaru pozytywnego, negatywnego oraz granicy poszczególnych klas decyzyjnych tablicy oraz wyznaczenie liczbowych współczynników określających jakość aproksymacji
- wyznaczenie najkrótszego/wszystkich względnych reduktów tablicy decyzyjnej różnymi metodami
- wyznaczenie względnych reduktów dynamicznych (z użyciem różnych metod lokalnych) tablicy decyzyjnej
- wyznaczenie współczynników wiarygodności dla obiektów tablicy decyzyjnej przy użyciu różnych metod

Aplikacja jest wyposażona w przejrzysty, funkcjonalny oraz łatwy w użyciu interfejs użytkownika.

6.2. Architektura aplikacji.

Aplikacja została napisana w języku JAVA v 1.4.0.2 ([2.3]). Posiada ona graficzny interfejs użytkownika zrealizowany przy użyciu standardowej biblioteki wchodzącej w skład tego języka (SWING). Interfejs został wyposażony w elementy architektury wielodokumentowej (ang. MDI Multiple Document Interface) tzn. umożliwia użytkownikowi równoległą pracą nad kilkoma projektami, użytkownik ma do dyspozycji niezależne przestrzenie robocze.

W ramach oprogramowania została także wykonana biblioteka algorytmów genetycznych, która wykorzystuje pewne pomysły zaprezentowane w istniejących już bibliotekach JGAP ([3.2]) oraz gabi ([3.1]). Podstawy teoretyczne zostały zaś zasięgnięte z [1.1] oraz [1.19]. Dostarcza ona ogólnego schematu działania algorytmu genetycznego zapewniając duże możliwości konfiguracyjne. Wszelkie operatory oraz funkcje składowe algorytmu mają zdefiniowane odpowiednie interfejsy lub też klasy abstrakcyjne. Powoduje to, że implementacja własnych operatorów nie stanowi żadnego problemu. Programista dodaje nowe klasy (implementując interfejsy bądź też rozszerza klasy abstrakcyjne) bez potrzeby ingerencji w istniejący kod biblioteki. W skład biblioteki wchodzą implementacje operatorów genetycznych, które okazały się niezbędne do realizacji algorytmów w programie ARES Rough Set Exploration System.

Podobnie jak w przypadku biblioteki algorytmów genetycznych także inne elementy systemu zostały zrobione w sposób elastyczny (klasy abstrakcyjne oraz interfejsy), tak aby umożliwić łatwą rozbudowę aplikacji np. poprzez dodanie nowych algorytmów znajdujących redukty czy też współczynniki wiarygodności.

6.3. Dane wejściowe.

Aktualna wersja systemu wczytuje tablice decyzyjne, w postaci pliku tekstowego, zgodne z następującym formatem:

```
n m k
w_1_1 w_1_2 … w_1_m
w_2_1 w_2_2 … w_2_m
…
w_n_1 w_n_2 … w_n_m
```

gdzie:

- n – liczba obiektów tablicy decyzyjnej
- m – liczba atrybutów tablicy decyzyjnej

- k – numer atrybutu decyzyjnego ($k \in \langle 1, m \rangle$)
- w_x_y – wartość atrybutu numer y dla obiektu o numerze x, wartości mogą być typu całkowitego i rzeczywistego

Przykład 6.1

Przykładowy plik z danymi zgodny z formatem akceptowanym przez aplikację:

```
3 4 4
0 0 1 0
1 2 2 1
0 1 0 1
```

Tablica decyzyjna reprezentowana przez ten plik składa się z 3 obiektów oraz 4 atrybutów, z których pierwsze trzy to atrybuty warunkowe zaś ostatni to atrybut decyzyjny.

6.4. Dyskretyzacja tablicy decyzyjnej.

Aplikacja umożliwia przeprowadzenie dyskretyzacji atrybutów ciągłych tablicy decyzyjnej dwiema podstawowymi metodami: metodą według równej częstości oraz według równej szerokości. Po wybraniu metody i ustawieniu podstawowych parametrów (np. ilości przedziałów) algorytm dokonuje wstępnej dyskretyzacji i prezentuje użytkownikowi otrzymane przedziały oraz ilość obiektów należących do każdego z nich. Następnie użytkownik może ręcznie poprawić przedziały dyskretyzacji. W ten sposób mamy zapewnioną funkcjonalność algorytmu dyskretyzacji wykorzystującej wiedzę eksperta. Po ostatecznym zatwierdzeniu przedziałów dyskretyzacji aplikacja generuje nową tablicę decyzyjną z dyskretnymi wartościami poszczególnych atrybutów.

6.5. Wyznaczanie reduktów tablicy decyzyjnej.

Aplikacja umożliwia wyznaczanie reduktów względnych w dwóch kategoriach: reduktu minimalnego oraz maksymalnej ilość reduktów. Zostały zaimplementowane wszystkie

opisywane w tej pracy metody. Algorytmy ewolucyjne mogą być parametryzowane przez użytkownika poprzez podanie wartości prawdopodobieństwa krzyżowania, mutacji, liczności populacji oraz ilości iteracji. Ten ostatni parametr musi być także określony dla algorytmów losowych.

6.6. Wyznaczanie współczynników wiarygodności dla obiektów.

System umożliwia wyznaczenie współczynników wiarygodności dla obiektów tablicy decyzyjnej trzema, zaprezentowanymi w tej pracy, metodami. Po wybraniu algorytmu generowane jest zestawienie zawierające współczynniki dla wszystkich obiektów jak również wartości średnie. Warto dodać, że współczynniki wiarygodności mogą być obliczane nie tylko na podstawie wszystkich atrybutów tablicy decyzyjnej ale także z wykorzystaniem reduktu/zbioru reduktów. Aplikacja pozwala użytkownikowi na wybranie pewnych obiektów (np. tych o najniższej wiarygodności) a następnie na wygenerowanie nowej tablicy decyzyjnej nie posiadającej już tych elementów.

6.7. Inne aplikacje wykorzystujące zbiory przybliżone do analizy danych.

6.7.1. Wstęp.

Istnieje szereg aplikacji zarówno komercyjnych jak i darmowych, które umożliwiają przeprowadzenie analizy danych z użyciem zbiorów przybliżonych. Podczas przygotowywania pracy zapoznałem się z kilkoma programami oferującymi właśnie taką funkcjonalność. Poniżej dokonam krótkiej charakterystyki wybranych (wartych uwagi) pozycji oraz porównania ich z aplikacją mojego autorstwa. Żaden z programów nie miał jednak możliwości generowania współczynników wiarygodności dla obiektów tablicy decyzyjnej.

6.7.2. ROSETTA (A Rough Set Toolkit for Analysis of Data version: 1.4.41).

ROSSETTA ([3.4]) jest jednym z najpopularniejszych systemów wykorzystujących zbiory przybliżone do analizy danych. Aplikacji ta jest powstała i jest rozwijana w dwóch ośrodkach akademickich: na uniwersytecie w Norwegii (Norwegian University of Science and Technology) oraz na Uniwersytecie Warszawskim (Wydział Matematyki).
ROSSETTA jest aplikacją typu wielodokumentowego działającą pod systemami zgodnymi z Microsoft Windows. Warto zaznaczyć, że implementacja wszystkich algorytmów pochodzi z biblioteki RSES-lib, wykorzystywanej również w następnej z prezentowanych aplikacji. ROSETTA jest bardzo rozbudowanym narzędziem umożliwiającym m.in.:

- wczytanie tabel decyzyjnych z różnych źródeł (pliku tekstowego, XML'owego, bazy danych, itd…)
- dyskretyzację wartości atrybutów ciągłych (obok metod przedstawionych w tej pracy również metody oparte na entropii)
- wyznaczanie reduktów (algorytm Johnsona, dokładny, genetyczny, …)
- klasyfikowanie obiektów różnymi metodami

W stosunku do mojej aplikacji ROSETTA posiada bardziej rozbudowany moduł wczytywania danych, moduł dyskretyzacji oraz klasyfikacji (którego moja aplikacja nie ma w ogóle). Na korzyść programu mojego autorstwa przemawia zaś moduł służący do wyznaczania reduktów, którego funkcjonalność (m.in. ilość dostępnych algorytmów) jest znacznie szersza niż w omawianym systemie ROSETTA.

6.7.3. RSES2 (Rough Set Exploration System version: 2.1).

RSES2 ([1.3], [3.3]) jest aplikacją, która powstała na Uniwersytecie Warszawskim w Instytucie Matematyki. Do obliczeń wykorzystuje ona wspomnianą już wcześniej bibliotekę RSES-lib, zaś interfejs użytkownika został w całości wykonany w języku JAVA przy użyciu biblioteki SWING. W porównaniu do ROSETTY ta aplikacja nie ma tak rozbudowanej funkcjonalności. Na funkcjonalność tej aplikacji składa się:

- importowanie danych z plików tekstowych w formatach różnych systemów (RSES, Weka, ROSETTA, XML)
- przeglądanie i wstępna obróbka danych (wypełnianie pustych miejsc, dyskretyzacja (tylko 2 metody), ekstrakcja nowych cech)
- wyznaczanie reduktów (algorytmy dokładny oraz genetyczny)
- konstruowanie i stosowanie różnych klasyfikatorów dla małych i dużych danych wraz z możliwością oceny jakości klasyfikatorów
- dwupoziomowa wizualizacja obiektów pojawiających się podczas pracy systemu (reduktów, reguł decyzyjnych, cięć, podziałów, sieci neuronowych, wyników klasyfikacji itd.).

Warto na sam koniec dodać, że program RSES2 (a dokładnie rzecz biorąc jądro obliczeniowe RSES-lib) współpracuje z pakietem DIXER 2.0, który również powstał na Uniwersytecie Warszawskim. DIXER umożliwia rozproszone wykonywanie algorytmów do przetwarzania danych, co w znacznym stopniu redukuje czas potrzebny na uzyskanie rozwiązania.

6.8. Perspektywy rozwoju.

Proponowane kierunki rozwoju programu ARES Rough Set Exploration System:

- rozbudowanie części odpowiedzialnej za wczytywanie danych tak, aby akceptowała wartości atrybutów typów wyliczeniowego oraz tekstowego
- dodanie zaawansowanych algorytmów dyskretyzacji atrybutów ciągłych
- wzbogacenie funkcjonalności aplikacji o moduł umożliwiający indukcję reguł decyzyjnych z wykorzystaniem różnych metod
- rozbudowanie interfejsu użytkownika mające na celu dodanie następującej funkcjonalności: dynamiczne definiowanie zbiorów atrybutów decyzyjnych oraz warunkowych tablicy decyzyjnej, dynamiczne zmienianie wartości atrybutów dla poszczególnych obiektów, wprowadzanie nowych obiektów itd.
- dodanie możliwości zapisywania rezultatów wykonania poszczególnych algorytmów do plików wyjściowych w zwykłym formacie tekstowym oraz formacie XML'owym

7. Przykład wykorzystanie ARES Rough Set Exploration System do analizy tablicy decyzyjnej.

7.1. Wstęp.

W rozdziale tym zostanie przedstawiony przykładowy, kompleksowy proces analizy prostego zbioru danych z wykorzystaniem napisanego przeze mnie oprogramowania. Na badanej tablicy decyzyjnej zostaną przeprowadzone kolejno następujące czynności:

- wczytanie zbioru do systemu
- dyskretyzacja atrybutów ciągłych
- wyznaczenie reduktów względnych
- wyznaczenie współczynników wiarygodności
- eliminacja obiektów

Przykład ten ma na celu pokazanie możliwości aplikacji ARES Rough Set Exploration System oraz sposobu z niej korzystania. Cały proces przetwarzania zbioru danych został dokładnie opisany (m.in. sposób wywoływania oraz parametryzacji dostępnych algorytmów, przeglądanie rezultatów ich wykonania, itd…) oraz zilustrowany licznymi, rzeczywistymi rysunkami pochodzącymi z programu.

7.2. Zbiór danych.

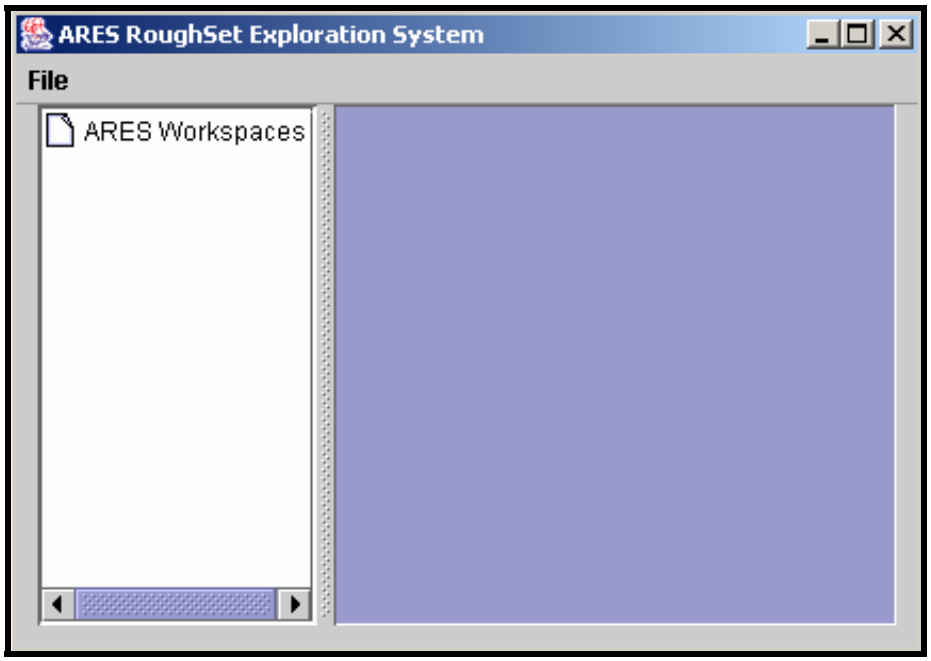
Zbiór danych, który zostanie poddany analizie to zmieniona tablica decyzyjna z tabeli 2.1. Została ona przekształcona do postaci akceptowanej przez program w taki sposób, aby pokazać możliwie w jak największym stopniu możliwości programu. Część atrybutów wyliczeniowych została zamieniona na atrybuty liczbowe o wartościach dyskretnych, zaś część na atrybuty liczbowe ciągłe. Ostateczna postać tablicy decyzyjnej wraz z odpowiadającym jej plikiem wejściowym do aplikacji została przedstawiona w tabeli 7.1.

Tablica decyzyjna				Plik wejściowy (mag_mod.sys)
Ból głowy	**Ból mięsni**	**Temperatura**	**Grypa**	
nie	tak	38.5	1	*6 4 4* *0 1 38.5 1* *1 0 38.1 1* *1 1 39.5 1* *0 1 40.0 1* *1 0 38.6 0* *0 1 36.6 0*
tak	nie	38.1	1	
tak	tak	39.5	1	
nie	tak	40.0	1	
tak	nie	38.6	0	
nie	tak	36.6	0	

Tabela 7.1. Przekształcona tablica decyzyjna z tabeli 2.1 oraz odpowiadający jej plik wejściowy do aplikacji ARES Rough Set Exploration System.

7.3. Uruchomienie aplikacji.

W celu wystartowania aplikacji należy uruchomić plik ***ares.bat***. Po wykonaniu tej czynności użytkownik otrzyma ekran podobny do przedstawionego na rysunku 7.1.



Rysunek 7.1. Aplikacja ARES Rough Set Exploration System po uruchomieniu.

W skład interfejsu użytkownika wchodzą dwa podstawowe widoki:

- widok drzewa – biały panel po lewej stronie

  Widok drzewa zawiera listę wszystkich aktualnie dostępnych dla użytkownika elementów np. przestrzeni roboczych, tablic decyzyjnych czy też wyników przeprowadzonych analiz
- widok przestrzeni roboczej – fioletowy panel zajmujący większość ekranu

  Widok przestrzeni roboczej jest miejscem, gdzie są otwierane odpowiednie okienka zgodnie z wybranymi elementami z widoku drzewa np. okno zawierające tablicę decyzyjną czy też znalezione redukty.

7.4. Wczytanie danych do programu.

Po uruchomieniu programu należy utworzyć nową przestrzeń roboczą, poprzez wciśnięcie prawego klawisza myszy na pozycji z widoku drzewa: ***ARES Workspaces*** i wybranie z menu kontekstowego opcji ***New Workspace***. Po wprowadzeniu nazwy i jej akceptacji w widoku drzewa pojawi się nowa przestrzeń robocza. Następnie należy wywołać menu kontekstowe skojarzone z nowo dodanym elementem do widoku drzewa, poprzez wciśnięcie na nim prawego klawisza myszy. Z menu należy wybrać opcję ***New InformationSystem*** oraz podać nazwę dla nowej tablicy decyzyjnej/systemu informacyjnego. W widoku drzewa w obrębie bieżącej przestrzeni roboczej pojawi się nowy element symbolizujący system informacyjny. Po wciśnięciu na nim prawego klawisza myszy uzyskamy menu kontekstowe, którego opcje zostały przedstawione oraz opisane w tabeli 7.2 (część z opcji może być aktywna lub nieaktywna w zależności od aktualnego stanu systemu informacyjnego).

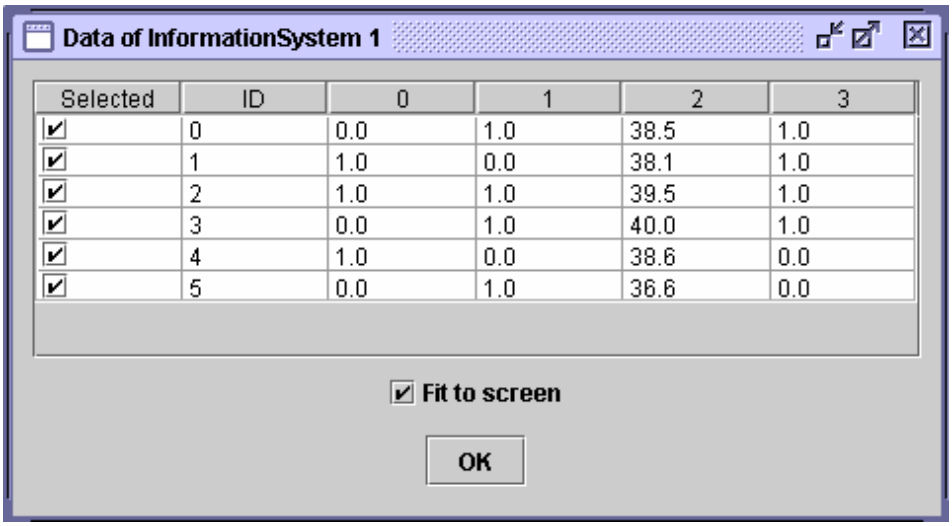
Opcja	Opis
Load InformationSystem Data	Wczytanie danych do systemu informacyjnego z zewnętrznego pliku.
Close InformationSystem	Zamknięcie systemu informacyjnego i usunięcie go z widoku drzewa.
Show Data	Wyświetlenie okienka z systemem informacyjnym.
Show Properties	Wyświetlenie okienka z właściwościami systemu informacyjnego.

Cut Information System	Wygenerowanie nowego systemu informacyjnego na podstawie istniejącego z wyłączeniem wybranych przez użytkownika obiektów.
Discretize Data	Dyskretyzacja systemu informacyjnego.
Calculate Decission Class Approximations	Wyznaczenie aproksymacji klas decyzyjnych dla systemu informacyjnego.
Calculate Discernibility Matrix	Wyznaczenie macierzy odróżnialności.
Find Global Reducts	Wyznaczenie reduktów względnych.
Find Credibility Coefficients	Wyznaczenie współczynników wiarygodności
Close Menu	Zamknięcie menu

Tabela 7.2. Lista opcji dostępnych w menu kontekstowym skojarzonym z systemem informacyjnym.

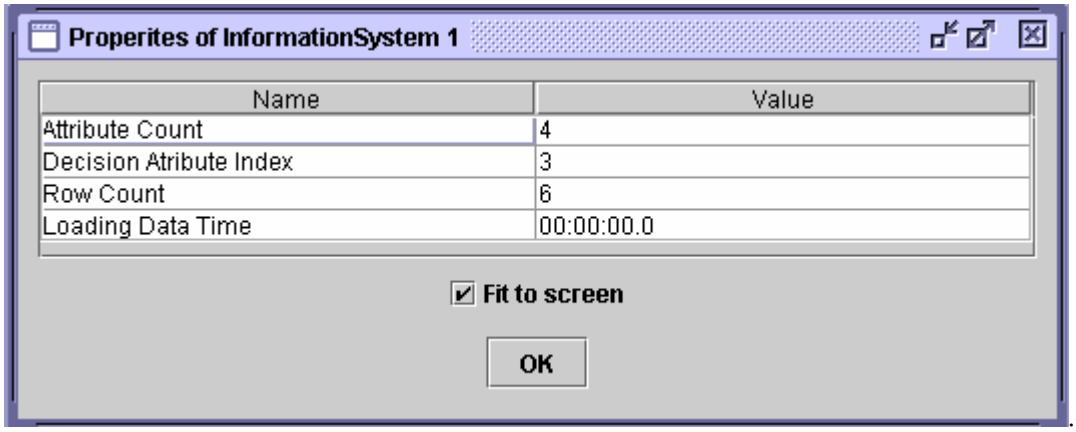
Aby wczytać dane z pliku zewnętrznego do systemu informacyjnego należy z menu kontekstowego tego elementu wybrać opcję ***Load InformationSystem Data*** a po otwarciu okienka wskazać plik z tablicą decyzyjną: ***mag_mod.sys***.

Po wczytaniu danych można np. obejrzeć zawartość systemu informacyjnego. Aby tego dokonać należy z menu kontekstowego wybrać opcję ***Show Data.*** W widoku przestrzeni roboczej pojawi się okno z systemem informacyjnym, takie jak na rysunku 7.2. Kolejne rzędy tabeli symbolizują obiekty, zaś kolumny atrybuty systemu informacyjnego.



*Rysunek 7.2. Okno z systemem informacyjnym (opcja **Show Data**)*

Rysunek 7.3. przedstawia charakterystykę systemu informacyjnego. Okno to jest dostępne po wybraniu z menu kontekstowego opcji ***Show Properties***.

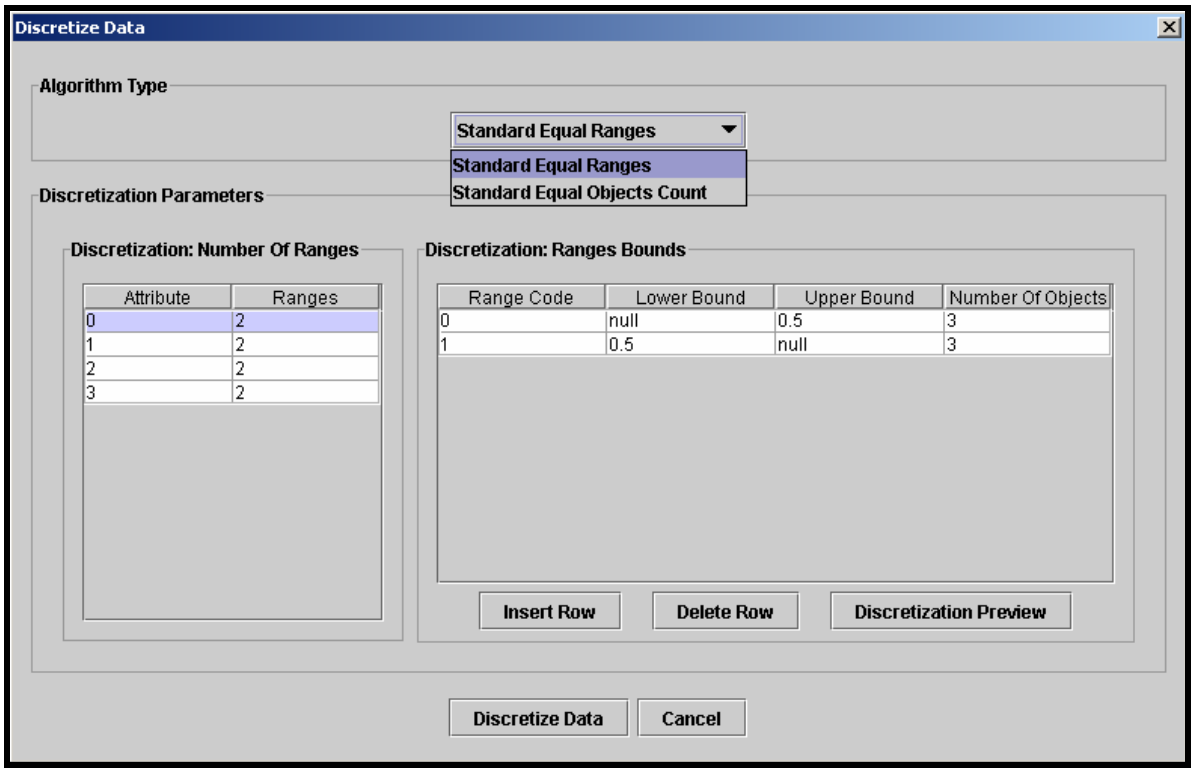


*Rysunek 7.3. Okno z systemem informacyjnym (opcja **Show Properties**)*

Tabela z rysunku 7.3. zawiera następujące informacje o systemie informacyjnym: ilość atrybutów, indeks atrybutu decyzyjnego, ilość obiektów oraz czas jaki był potrzebny na wczytanie danych do programu.

7.5. Dyskretyzacji atrybutów ciągłych.

W celu przeprowadzenia dyskretyzacji atrybutów ciągłych należy z menu kontekstowego dla wybranego systemu informacyjnego wybrać opcję ***Discretize Data***. Po wykonaniu tej czynności pojawi się okno, które umożliwia wybór algorytmu dyskretyzacji oraz jego parametryzację (patrz rysunek 7.4). W górnej części okna możemy wybrać jeden z podstawowych algorytmów do wstępnej dyskretyzacji atrybutów. Dolna część umożliwia zaś parametryzacją dyskretyzacji. W lewej tabeli ***Discretization: Number Of Ranges*** definiujemy ilość przedziałów dyskretyzacji dla każdego atrybutu (wartość *0* oznacza, że dany atrybut nie będzie dyskretyzowany). W prawej tabeli ***Discretization: Ranges Bounds*** definiujemy dolne i górne wartości dla każdego przedziału dyskretyzacji aktualnie wybranego atrybutu w tabeli po lewej stronie (wartość *null* oznacza nieskończoność). Po wciśnięciu przycisku ***Discretization Preview*** zostaną uaktualnione pola ***Number of Objects***, które określają ile obiektów trafi do jakiego przedziału przy aktualnych ustawieniach dyskretyzatora.



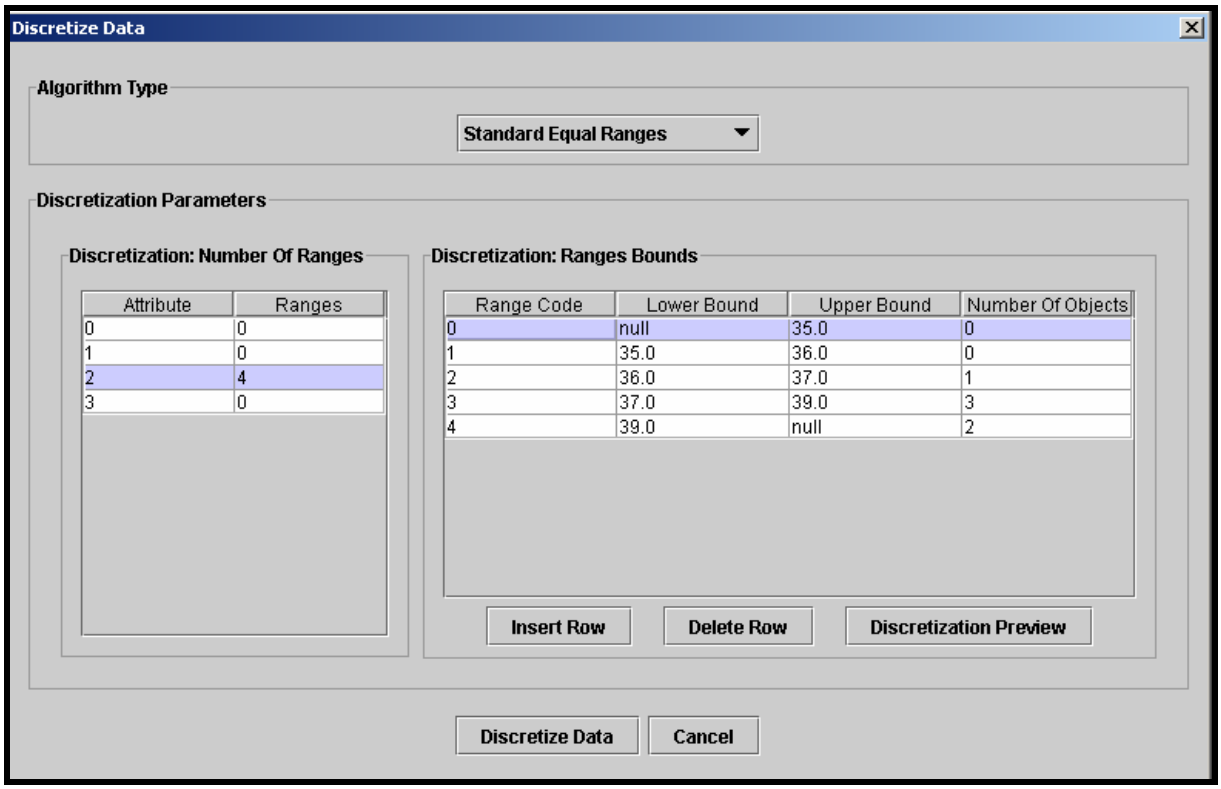
Rysunek 7.4. Okno dyskretyzacji atrybutów z wartościami domyślnymi.

W naszym przypadku tylko atrybut numer 2 (temperatura) jest atrybutem ciągłym i to tylko on powinien zostać poddany dyskretyzacji. Dodatkowo załóżmy, że dostaliśmy od eksperta informacje jak należy przekształcić ten atrybut na postać dyskretną. Odpowiednie wartości ciągłe oraz dyskretne zostały zawarte w tabeli 7.3.

Lp.	**Temperatura**		
	Wartości ciągłe	**Opinia eksperta**	**Wartość dyskretna**
1.	do 35.0	bardzo niska	0
2.	od 35.0 do 36.0	niska	1
3.	od 36.0 do 37.0	normalna	2
4.	od 37.0 do 38.0	wysoka	3
5.	od 39.0	bardzo wysoka	4

Tabela 7.3. Wartości ciągłe oraz dyskretne atrybutu temperatura.

Uwzględniając wymagania odpowiednia parametryzacja dyskretyzatora została przedstawiona na rysunku 7.5.



Rysunek 7.5. Okno dyskretyzacji atrybutów z wypełnionymi przez użytkownika parametrami.

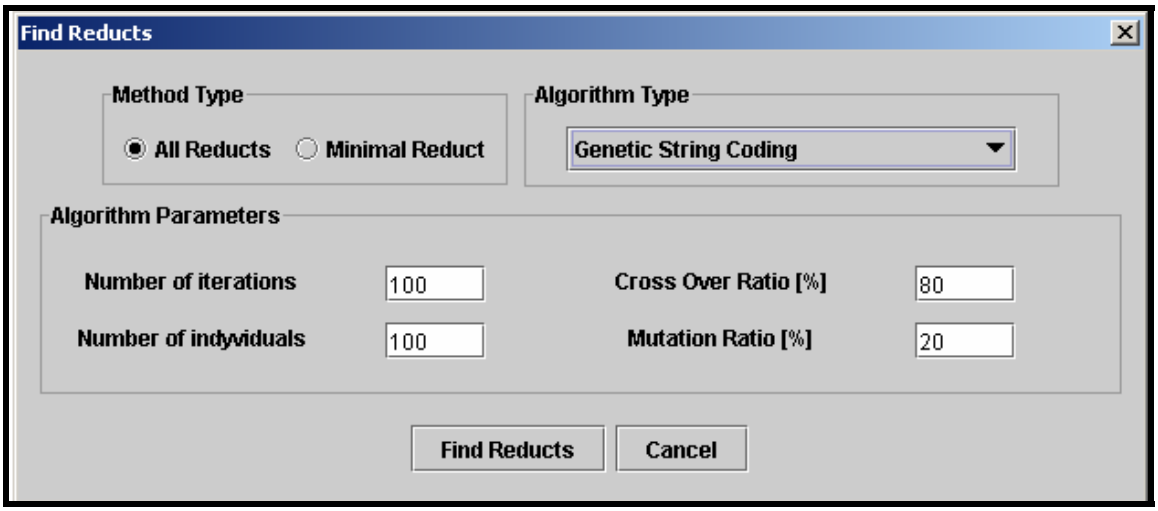
Po wciśnięciu przycisku ***Discretize Data*** w widoku drzewa pojawia się nowy system informacyjny wraz z obiektem symbolizującym podsumowanie procesu dyskretyzacji (można tam m.in. zobaczyć parametry dyskretyzacji, wybrany algorytm oraz czas potrzebny do jej przeprowadzenia).

7.6. Wyznaczenie reduktów względnych.

Kolejnym krokiem będzie wyznaczenie reduktów względnych dla dyskretyzowanego systemu informacyjnego. Po wybraniu z menu kontekstowego (dla systemu informacyjnego) opcji ***Find Global Reducts*** wyświetli się okno umożliwiające wyznaczenie reduktów względnych (patrz rysunek 7.6).

Metody wyznaczające redukty względne zostały podzielone na dwie grupy:

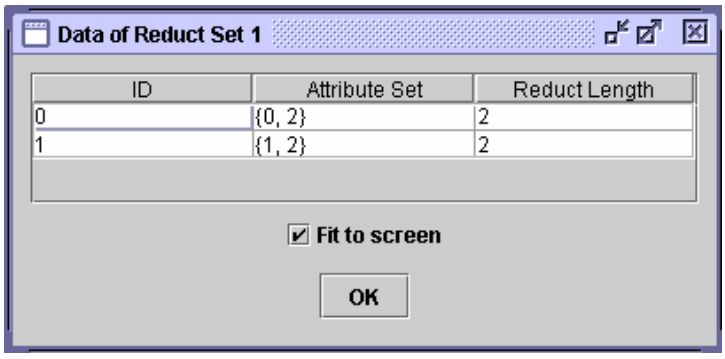
- All Reducts – metody wyznaczające wszystkie (maksymalną ilość) redukty względne
- Minimal Reduct – metody wyznaczające minimalny redukt względny



Rysunek 7.6. Okno służące do wyznaczania reduktów względnych.

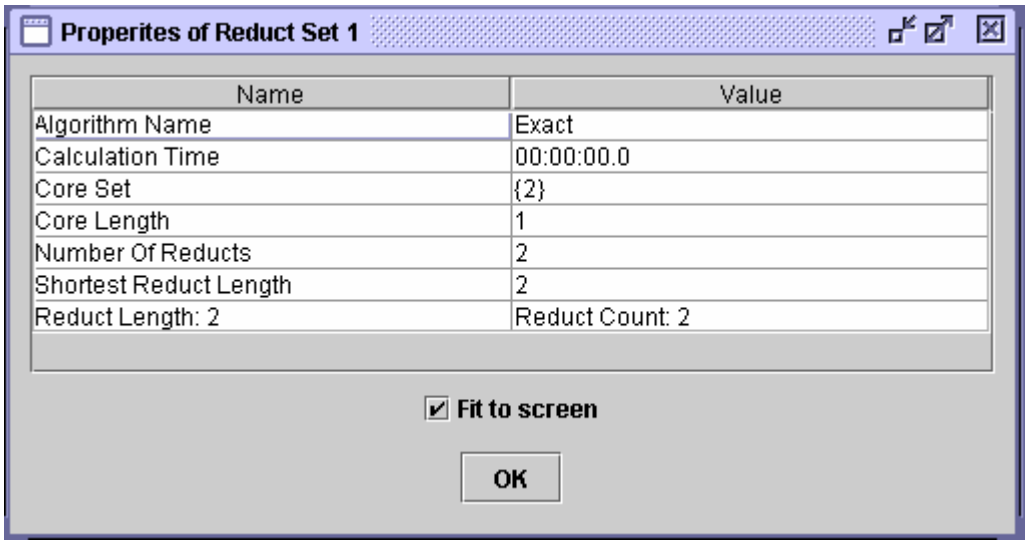
W ramce ***Algorithm Type*** pojawiają się dostępne algorytmy w wybranej przez użytkownika grupie metod. Dodatkowo użytkownik może dokonać parametryzacji wybranego algorytmu (jeżeli algorytm może być parametryzowany). Do wybraniu odpowiedniej metody i dokonaniu jej parametryzacji wciśniecie przycisku ***Find Reducts*** powoduje uruchomienie algorytmu wyznaczającego redukty. Efektem wykonania algorytmu jest pojawienie się nowego elementu w widoku drzewa zawierającego rezultat jego działania. Jeżeli wybrany algorytm wymagał np. obliczenia macierzy odróżnialności (np. dla algorytmu dokładnego), to również taki element pojawi się w strukturze drzewa, jeżeli oczywiście nie było go wcześniej.

Rysunki 7.7 oraz 7.8 przedstawiają rezultat wykonania algorytmu dokładnego (grupa metod: All Reducts, algorytm: Exact) dla rozpatrywanego systemu informacyjnego (okienka te pojawiają się po wybraniu z menu kontekstowego dla elementu ***Reduct Set*** odpowiednio opcji ***Show Data*** oraz ***Show Properties***).



*Rysunek 7.7. Rezultat wykonania algorytmu dokładnego znajdującego redukty względne (opcja **Show Data**).*

Tabela z rysunku 7.7 zawiera listę znalezionych reduktów względnych tablicy decyzyjnej wraz z ich długością.

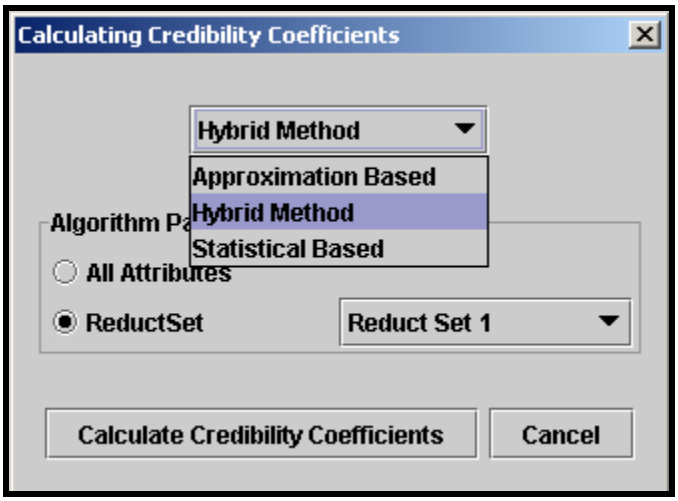


*Rysunek 7.8. Rezultat wykonania algorytmu dokładnego znajdującego redukty względne (opcja **Show Properties**).*

Tabela z rysunku 7.8. zawiera podstawowe informacje o przeprowadzonym procesie wyznaczania reduktów: nazwę użytego algorytmu, czas jego działania, rdzeń (wyznaczony na podstawie zbioru znalezionych reduktów), ilość znalezionych oraz długość najkrótszego reduktu, a także zestawienie przedstawiające liczbę reduktów o konkretnej długości.

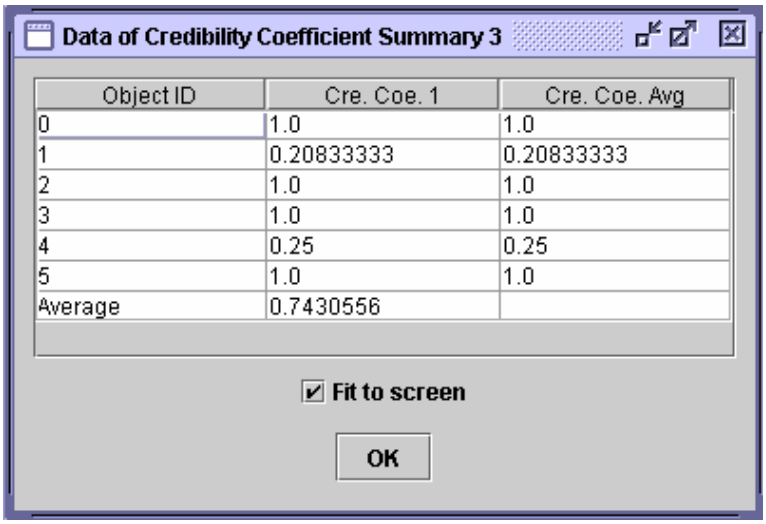
7.7. Wyznaczenie współczynników wiarygodności.

Kolejnym krokiem będzie wyznaczenie współczynników wiarygodności dla badanej tablicy decyzyjnej. Po wybraniu z menu kontekstowego (dla systemu informacyjnego) opcji ***Find Credibility Coefficients*** wyświetli się okno umożliwiające wyznaczenie współczynników wiarygodności dla obiektów (patrz rysunek 7.9). Użytkownik może wybrać jedną z trzech metod oraz jeden z dwóch sposobów wyliczenia współczynników (bazujący na wszystkich atrybutach lub bazujący na jednym ze zbiorów wcześniej wyznaczonych reduktów). Efektem wyznaczenia współczynników jest nowy element w widoku drzewa (domyślnie o nazwie ***Credibility Coefficient Summary***).



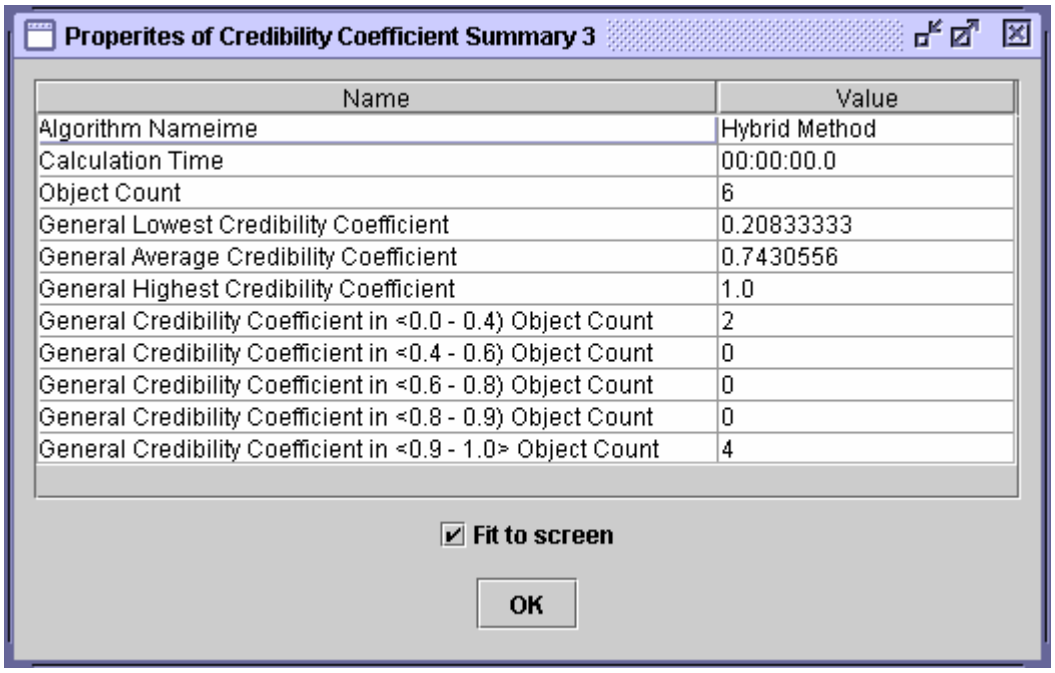
Rysunek 7.9. Okno służące do wyznaczania współczynników wirygodności dla obiektów.

Rysunki 7.10 oraz 7.11 przedstawiają rezultat wykonania algorytmu hybrydowego dla rozpatrywanego systemu informacyjnego przy wykorzystaniu wszystkich atrybutów (okienka te pojawiają się po wybraniu z menu kontekstowego dla elementu ***Credibility Coefficient Summary*** odpowiednio opcji ***Show Data*** oraz ***Show Properties***).



*Rysunek 7.10. Rezultat wykonania algorytmu hybrydowego (opcja **Show Data**).*

Tabela z rysunku 7.10 zawiera współczynniki wiarygodności dla każdego z obiektów systemu informacyjnego (***Cre. Coe. 1***) oraz wartość średnią dla wszystkich obiektów (***Average***). Kolumna ***Cre. Coe. Avg.*** zawiera wartość wyznaczoną jako średnia arytmetyczna współczynników dla danego obiektu. Ma ona znaczenie, gdy używamy zbioru reduktów do wyznaczenia wiarygodności obiektów, wtedy każdemu reduktowi odpowiada jeden współczynnik.



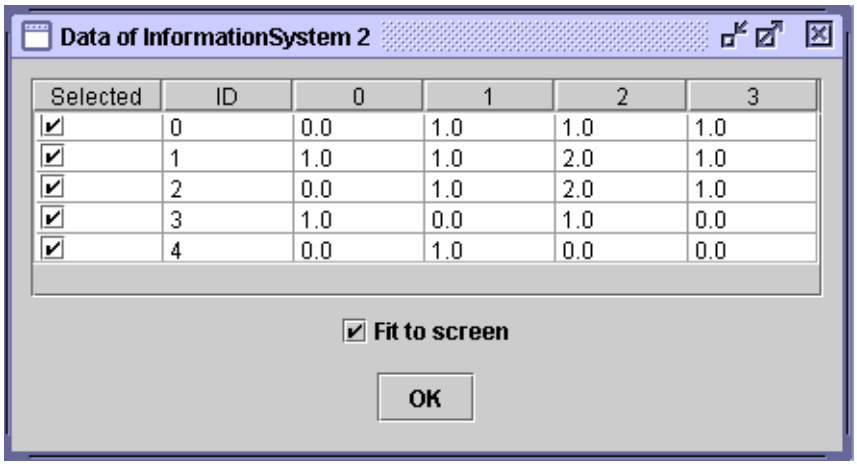
*Rysunek 7.11. Rezultat wykonania algorytmu hybrydowego (opcja **Show Properties**).*

Tabela z rysunku 7.11 zawiera krótką charakterystykę dotyczącą wyników wykonania algorytmu: nazwę algorytmu, czas działania, ilość obiektów w systemie, najniższy, średni oraz najwyższy współczynnik wiarygodności, a także zestawienie przedstawiające ilość obiektów mających współczynnik wiarygodności w określonym przedziale wartości.

7.8. Eliminacja obiektów.

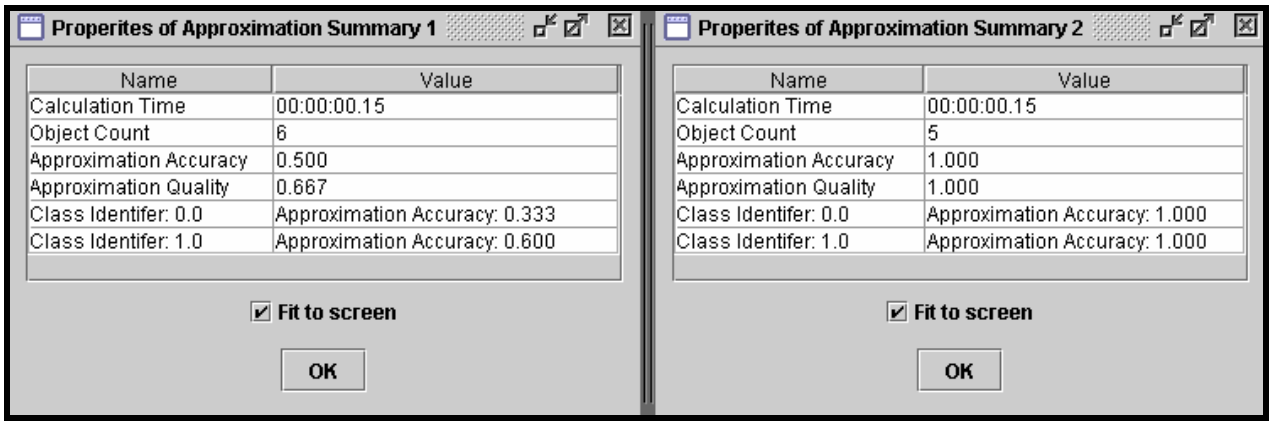
Po wyznaczeniu współczynników wiarygodności dla obiektów użytkownik może usunąć te obiekty, które są najmniej wiarygodne. Po otwarciu okna z systemem informacyjnym (opcja ***Show Data*** z menu kontekstowego systemu informacyjnego) użytkownik może wybrać obiekty, które jego zdaniem wprowadzają szum informacyjny. Aby tego dokonać należy odznaczyć pole wyboru w kolumnie ***Selected*** (patrz rysunek 7.2) dla konkretnego obiektu.

Nowy system informacyjny (bez wyeliminowanych obiektów) można wygenerować używając opcji z menu kontekstowego systemu informacyjnego ***Cut Information System***. Rysunek 7.12 przedstawia badany system informacyjny po usunięciu z niego obiektu o numerze *ID* równym *1* (ten obiekt miał najmniejszy współczynnik wiarygodności).



*Rysunek 7.12. Okno ze dyskretyzowanym systemem informacyjnym po usunięciu najmniej wiarygodnego obiektu (opcja **Show Data**).*

Z oryginalnego systemu został usunięty obiekt o najmniejszej wiarygodności. Teraz można wyznaczyć współczynniki wiarygodności dla nowego systemu i zobaczyć jaki wpływ na wyniki miał fakt wyeliminowania najmniej wiarygodnego elementu. Można również dokonać porównania liczbowych charakterystyk aproksymacji (jakości oraz dokładności aproksymacji) dla obu tych zbiorów danych. Odpowiednie okienka z danymi uzyskamy po wybraniu opcji ***Show Properties*** z menu kontekstowego obiektu Approximation Sumary. Porównanie tych wartości znajduje się na rysunku 7.13.

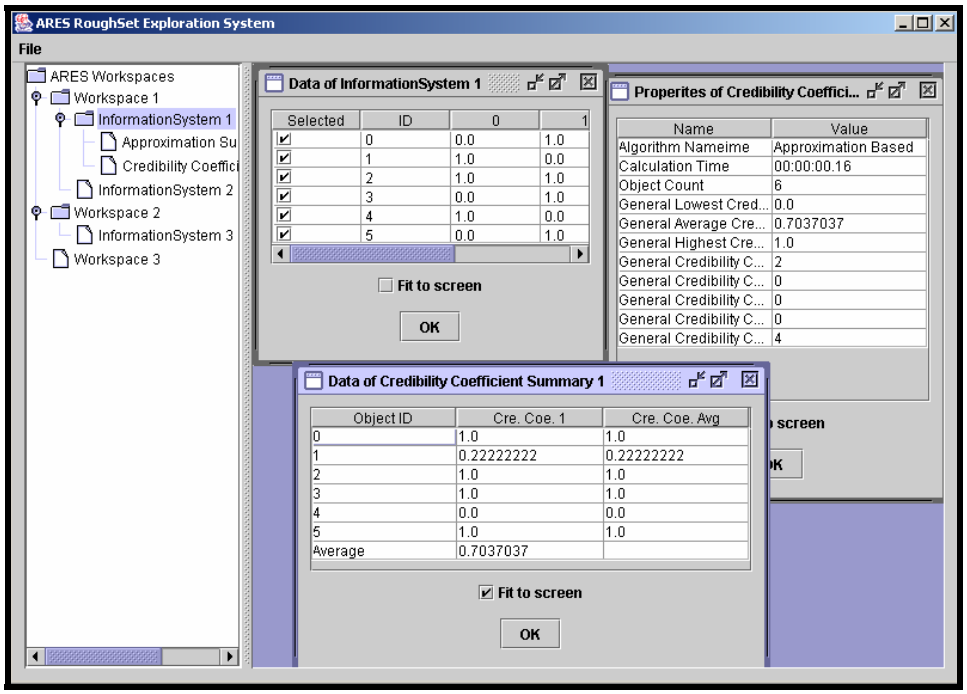


*Rysunek 7.13. Okna z informacjami dotyczącymi jakości oraz dokładności aproksymacji dla dwóch systemów informacyjnych: przed (lewa strona) i po (prawa strona) usunięciu najmniej wiarygodnego obiektu (opcja **Show Properties** obiektu Approximation Sumary).*

7.9. Podsumowanie.

W rozdziale tym został przedstawiony przykład wykorzystania programu ARES Rough Set Exploration System do analizy prostej tablicy decyzyjnej. Została przedstawiona przykładowa ścieżka, która posłużyła do wyeliminowania obiektu o najmniejszej wiarygodności. Możliwości aplikacji są znacznie większe niż funkcje, które zostały opisane w tym rozdziale.

Warto przypomnieć, że wykorzystywana aplikacja wykorzystuje architekturę MDI, co umożliwia prowadzenie niezależnych analiz różnych zbiorów danych w tym samym czasie. Po wybraniu z widoku drzewa elementu z innej przestrzeni roboczej, niż aktualnie wyświetlana w prawym panelu, program automatycznie zmienia kontekst i wyświetla widok skojarzony z nową przestrzenią roboczą (zapamiętując przy tym aktualny kontekst aplikacji). Ponadto każda przestrzeń robocza może zawierać dowolną ilość elementów w widoku drzewa oraz aktywnych (otwartych) okienek w widoku przestrzeni Bardziej złożony ekran pochodzący z aplikacji został przedstawiony na rysunku 7.14.



Rysunek 7.14. Okno główne aplikacji z otwartymi okienkami w widoku przestrzeni roboczej.

8. Podsumowanie.

Niniejsza praca jest częścią większego projektu, dotyczącego wykorzystania metodologii zbiorów przybliżonych do analizy danych (a szczególnie danych medycznych), zaprezentowanego w [1.27]. Praca ta została podzielona na osiem powiązanych ze sobą rozdziałów.

Praca przedstawia w sposób usystematyzowany podstawowe definicje oraz twierdzenia związane z omawianą tematyką, w zakresie niezbędnym do pełnego zrozumienia omawianych we wszystkich rozdziałach problemów. Tradycyjne, dobrze ugruntowane w literaturze, definicje zostały uzupełnione o pewne nowe propozycje pojęć np. wektor odróżnialności czy też zbiór pokrywający obiekty systemu informacyjnego. Pojęcia te zostały wprowadzone na potrzeby algorytmów, które zostały opisane w kolejnych rozdziałach. Część teoretyczna została uzupełniona licznymi praktycznymi przykładami.

Praca opisuje także popularne reprezentacje oraz typy danych. Zostały przedstawione charakterystyczne sposoby postępowania mające na celu wstępne przygotowanie zbioru danych do procesu analizy z użyciem zbiorów przybliżonych. Szczególna uwaga została skierowana na dyskretyzację atrybutów o wartościach ciągłych, gdyż zdyskretyzowana postać tablicy decyzyjnej jest znacznie wygodniejsza z perspektywy analizy danych algorytmami zaprezentowanymi w kolejnych rozdziałach. Opisano cztery podstawowe algorytmy służące do przekształcania atrybutów ciągłych na postać dyskretną.

Istotnym problemem poruszanym w pracy jest problem wyznaczania reduktów względnych tablicy decyzyjnej. Zostały przedstawione oraz porównane najpopularniejsze algorytmy rozwiązujące ten problemy, jak również przedstawiono pewne modyfikacje poprawiające ich działanie. Zestawienie zostało przeprowadzone w dwóch kategoriach: algorytmów wyznaczających maksymalną liczbę reduktów oraz algorytmów wyznaczających minimalny redukt. W obu przypadkach nowy, zaproponowany algorytm, opierający się na idei dekompozycji przestrzeni w połączeniu z odpowiednio zaprojektowanym algorytmem ewolucyjnym, okazał się być znacznie lepszy od pozostałych, dotychczas wykorzystywanych metod. Jego przewaga jest widoczna zarówno pod względem jakości uzyskiwanych rozwiązań

oraz czasu działania algorytmu. Dowodzą tego przeprowadzone na kilku różnych zbiorach danych eksperymenty, których rezultaty zostały również zawarte w tej pracy.

Głównym elementem pracy jest opisanie propozycji metod służących do wyznaczania współczynnika wiarygodności dla obiektów tablicy decyzyjnej. Zaprezentowano trzy różne podejścia: statystyczno-częstotliwościowe, oparte na przybliżeniach klas decyzyjnych oraz podejście hybrydowe. Pierwsze z nich opiera się tylko i wyłącznie na analizie częstotliwościowej wartości poszczególnych atrybutów, drugie wykorzystuje pojęcia aproksymacji zbiorów (klas decyzyjnych), zaś trzecie jest pewnym połączeniem dwóch poprzednich. Opierając się na wynikach pochodzących z przeprowadzonych testów na przykładowych danych można stwierdzić, że współczynniki wyznaczone przy użyciu tych metod, spełniają podstawowy cel, jaki zostały przed nimi postawiony tzn. pozwalają na wskazywanie obiektów, które potencjalnie są najmniej wiarygodne, a tym samym mogą zawierać błędy. Zaproponowane algorytmy poprawnie wskazywały te obiekty, które zostały wstawione do tablicy decyzyjnej z losowo wybraną klasą decyzyjną, jako najmniej wiarygodne elementy. W każdym przypadku usunięcie najmniej wiarygodnych elementów skutkowało podwyższeniu parametrów liczbowej charakterystyki aproksymacji (dokładności oraz jakości aproksymacji) rodziny zbiorów złożonych z obiektów należących do tej samej klasy decyzyjnej. Rosnąca jakość oraz dokładność aproksymacji wskazuje na fakt, iż z tablicy decyzyjnej zostały wyeliminowane te obiekty, które wprowadzały do niej niedeterminizm. Należy przy tym pamiętać, że tylko z deterministycznych tablic decyzyjnych można wygenerować zbiór reguł decyzyjnych, który będzie w sposób poprawny klasyfikował wszystkie obiekty badanej tablicy.

Integralną częścią pracy jest również aplikacja ARES Rough Set Exploration System. Opisane zostały podstawowe cechy (funkcjonalność) oraz architektura tego systemu, jak również dokonano porównania z innymi programami służącymi do analizy danych metodami zbiorów przybliżonych. Z przeprowadzonego porównania wynika, że napisany program ma znacznie bardziej rozbudowany moduł służący do wyznaczania reduktów niż inne rozpatrywane aplikacje. Ponadto żadna z nich nie posiada możliwości detekcji oraz usunięcia z tablicy decyzyjnej obiektów potencjalnie błędnych. Taką funkcjonalność zapewnia zaś program mojego autorstwa.

Przedstawiony został także kompleksowy przykład analizy pewnego zbioru danych z użyciem ARES Rough Set Exploration System. W przystępnej formie zostały zaprezentowane następujące

kroki postępowania: dyskretyzacja atrybutów ciągłych, wyznaczanie reduktów, wyznaczanie współczynników wiarygodności oraz usuwania obiektów nie pasujących do pozostałych.

Reasumując, w pracy tej dokonałem pewnego usystematyzowania oraz wyjaśnienia pojęć związanych z teorią zbiorów przybliżonych. Przedstawiłem także algorytmy wykorzystywane do wyznaczania reduktów oraz zaproponowałem pewne sposoby ich modyfikacji. Zaproponowałem także nowe, efektywne metody rozwiązujące ten problem. Kolejnym zagadnieniem, które rozpatrywałem, jest problem spójności danych. Zaproponowałem trzy metody, które mogą być wykorzystane w celu eliminacji obiektów z tablicy decyzyjnej, które nie pasują do pozostałych, zawartych w niej elementów. Integralną częścią pracy jest także program, w którym zaimplementowałem wszystkie opisywane tutaj algorytmy. Aplikacja ta może być wykorzystywana jako system do analizy danych z wykorzystaniem metod zbiorów przybliżonych.

9. Bibliografia.

9.1. Publikacje.

[1.1] Jarosław Arabas „Wykłady z algorytmów ewolucyjnych”, Wydawnictwo Naukowo-Techniczne, Warszawa, 2001

[1.2] Jan G. Bazan „Metody wnioskowań aproksymacyjnych dla syntezy algorytmów decyzyjnych”, praca doktorska, Uniwersytet Warszawski – Wydział Matematyki, Informatyki i Mechaniki; Instytut Matematyki

[1.3] Jan G. Bazan „RSES 2.1 i DIXER 2.0. Narzędzia informatyczne do analizy tablic danych z zastosowaniem teorii zbiorów przybliżonych”, Seminarium KBN, Zakopane, 2003

[1.4] Anders Torvill Bjorvand, Jan Komorowski „Practical Applications of Genetic Algorithms for Efficient Reduct Computation”, Proceedings of 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics, Achim Sydow, Ed. Berlin, Aug 24 - 29 1997, Vol 4, pp. 601-606

[1.5] Maciej Artur Bodych „Interfejs do analizy danych z zastosowaniem metod zbiorów przybliżonych”, praca dyplomowa inżynierska, Politechnika Warszawska – Wydział Elektroniki i Technik Informacyjnych, Instytut Informatyki, 2002

[1.6] Fleurent Charles, Jacques A. Ferland „Genetic and hybrid algorithms for graph coloring”, Annals of Operations Research, 1996, Vol.63, pp. 437-461

[1.7] Paweł Cichosz „Systemy uczące się”, Wydawnictwo Naukowo-Techniczne, Warszawa, 2000

[1.8] Adam Cykier „Implikanty pierwsze funkcji boolowskich, metody wyznaczania i zastosowania”, praca dyplomowa magisterska, Uniwersytet Warszawski – Wydział Matematyki, Informatyki i Mechaniki, 1997

[1.9] David Goldberg „Algorytmy genetyczne i ich zastosowania”, Wydawnictwo Naukowo-Techniczne, Warszawa, 1998

[1.10] Lech Jóźwiak „Information Relationships and Measures An Analysis Apparatus for Efficient Information System Synthesis”, 23rd EUROMICRO Conference '97 New Frontiers of Information Technology, Budapest, Hungary, 1997, pp. 13-24

[1.11] Lech Jóźwiak „Information Relationships and Measures in Application to Logic Design", Twenty Ninth IEEE International Symposium on Multiple-Valued Logic, Freiburg im Breisgau, Germany, 1999, pp. 228-236

[1.12] Adam Jurkowski „Ocena wiarygodności danych z wykorzystaniem teorii zbiorów przybliżonych" praca dyplomowa magisterska, Politechnika Warszawska – Wydział Elektroniki i Technik Informacyjnych, Instytut Informatyki, 2003

[1.13] Marzena Kryszkiewicz, Henryk Rybiński „Knowledge Discovery From Large Databases Using Rough Sets", Proceedings of 6th European Congress on Intelligent Techniques and Soft Computing EUFIT '98, Aachen, Germany, 1998, Vol. 1, pp. 85-89

[1.14] Marzena Kryszkiewicz „Strong Rules in Large Databases", Proceedings of EUFIT '98, Aachen, Germany, 1998

[1.15] Ruhul A. Sarker (Edt) „Heuristic and Optimization for Knowledge Discovery", Idea Group Pub, 2002

[1.16] Lin Li, Jiang Yunfei „Computing minimal hitting sets with genetic algorithm", International Workshop on Principles of Diagnosis, Semmering, Austria, 2002

[1.17] Lin Li, Yunfei Jiang „The computation of hitting sets: Review and new algorithms", Information Processing Letters, 2003, Vol. 86, pp. 177-184

[1.18] Wiktor Marek, Janusz Onyszkiewicz „Elementy logiki I teori mnogości w zadanich", Wydawnictwo Naukowe PWN, Warszawa, 1999

[1.19] Zbigniew Michalewicz „Genetic Algorithms + Data Structures = Evolution Programs", Springer Verlag, Warszawa, 1992

[1.20] Adam Mrózek, Leszek Płonka „Analiza danych metodą zbiorów przybliżonych. Zastosowania w ekonomii, medycynie i sterowaniu" Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1999

[1.21] Zdzisław Pawlak „Rough Sets. Theoretical Aspects of Reasoning about Data", Wydawnictwo Politechniki Warszawskiej, Warszawa, 1990

[1.22] Zdzisław Pawlak „Systemy informacyjne – Podstawy teoretyczne", Wydawnictwo Naukowo-Techniczne, Warszawa, 1983

[1.23] Roman Podraza, Adam Jurkowski „Coefficient of Credibility in Rough Set System", the 22 nd. IASTED International Conference on Artificial Intelligence and Applications (AIA), Insbruck, Austria, 2004, CD-ROM

[1.24] Roman Podraza, Andrzej Dominik, Mariusz Walkiewicz „Decission suport system for medical applications", Proceedings of the IASTED International Conference on Applied Simulations and Modeling, Marbella, Spain, 2003, pp. 329-334

[1.25] Roman Podraza, Wojciech Podraza „Rough Set System with Data Elimination", Proceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS'2002), Vol. II, Las Vegas, Nevada, USA, 2002, pp. 493-499

[1.26] Roman Podraza, Tomasz Strąkowski „Tubing of Real Type Data Coding for Decision Tables", the 22 nd. IASTED International Conference on Artificial Intelligence and Applications (AIA), Insbruck, Austria, 2004, CD-ROM

[1.27] Roman Podraza „Zastosowanie zmodyfikowanej teorii zbiorów przybliżonych do implementacji systemu wspomagania diagnostyki medycznej", Seminarium KBN, Zakopane, 2003

[1.28] Wojciech Podraza „Modyfikacja zastosowania teorii zbiorów przybliżonych w medycynie w celu ograniczenia błędów przypadkowych", II Sympozjum Modelowanie i Pomiary w Medycynie, Krynica Górska, 2000

[1.29] Wojciech Podraza, Agnieszka Kordek, Roman Podraza, Hanna Domek „Rough Set Method in the Diagnosis of Neonatal Infection", 6th World Congress on Trauma, Shock, Inflamation and Sepsis - Pathophysiology, Immune Consequences and Therapy, LMU Munich, 2004

[1.30] Sylwester Przybyło, Andrzej Szlachtowski „Algebra i wielowymiarowa geometria analityczna w zadaniach", Wydawnictwo Naukowo-Techniczne, Warszawa, 1994

[1.31] Knut Magne Risvik „Discretization of Numerical Attributes. Preprocessing for Machine Learning", Norwegian University of Science and Technology – Department of Computer and Information Science, 1997

[1.32] Piotr Ryszkowski „Interfejs do analizy danych z zastosowaniem metod zbiorów przybliżonych", praca dyplomowa inżynierska, Politechnika Warszawska – Wydział Elektroniki i Technik Informacyjnych, Instytut Informatyki, 2002

[1.33] Jerzy Stefanowski „Algorytmy indukcji reguł decyzyjnych w odkrywaniu wiedzy", rozprawa habilitacyjna, Wydawnictwo Politechniki Poznańskiej, 2001

[1.34] Tomasz Strąkowski „Analiza danych medycznych z zastosowaniem metod zbiorów przybliżonych" , praca dyplomowa magisterska, Politechnika Warszawska – Wydział Elektroniki i Technik Informacyjnych, Instytut Informatyki, 2003

[1.35] Zbigniew Walczak, Andrzej Dominik, Paweł Terlecki „Space decomposition in the problem of finding minimal reducts", VII Krajowa Konferencja Algorytmy Ewolucyjne i Optymalizacja Globalna, Kazimierz Dolny, 2004, str.193-201

[1.36] Jakub Wróblewski „Covering with Reducts – a Fast Algorithm for Rule Generation", Rough Sets and Current Trends in Computing, 1998, pp. 402-407

[1.37] Jakub Wróblewski „Finding minimal reducts using genetic algorithm", Technical report, University of Warsaw – Institute of Mathematics, 1995

9.2. Zasoby internetowe.

[2.1] Włodzisław Duch „Sztuczna Inteligencja: Reprezentacja wiedzy i logika przybliżona", http://www.phys.uni.torun.pl/~duch/Wyklady/AI/AI4-4.ppt

[2.2] Dane pochodzące z UCI Repository of Machine Learning Databases and Domain Theories (http://www.ics.uci.edu)

[2.3] Dokumentacja języka JAVA (http://www.sun.com)

[2.4] Giuseppe Nicosia „Some Combinatorial Optimization Problems: K-Sat, Graph Coloring and Minimum Hitting Set", (http://www.dmi.unict.it/~nicosia/)

[2.5] „Noty biograficzne, Doktorzy Honoris Causa: Zdzisław Pawlak", http://www.put.poznan.pl/ludzie/Zdzislaw_Pawlak.html

[2.6] Jakub Wróblewski „Narzędzia sztucznej inteligencji – materiały do wykładów", (http://www.qed.pl/ai/nai2003/index.html)

[2.7] „Zbiory przybliżone", http://sound.eti.pg.gda.pl/rekonstrukcja/zbiory_przyblizone.html

[2.8] Życiorys Zdzisława Pawlaka, http://www.ii.pw.edu.pl/news/z_pawlak-zyciorys.doc

[2.9] „Wikipedia, wolna encyklopedia", http://pl.wikipedia.org/

9.3. Inne materiały.

[3.1] Biblioteka gabi, implementacja algorytmów ewolucyjnych w języku C, Jarosław Arabas (http://elektron.elka.pw.edu.pl/~jarabas/)

[3.2] Biblioteka JGAP (Java Genetic Algorithm Package) implementacja algorytmów ewolucyjnych w języku JAVA (http://jgap.sourceforge.net)

[3.3] Program RSES2 (Rough Set Exploration System version: 2.1) Warsaw University – Institute of Mathematics, (http://logic.mimuw.edu.pl/~rses)

[3.4] Program ROSETTA (A Rough Set Toolkit for Analysis of Data version: 1.4.41) Knowledge Systems Group, Department of Computer and Information Science, Norwegian University of Science and Technology (http://rosetta.lcb.uu.se/general/)