STUDIA INFORMATICA 2009
Volume 30 Number 2A (83)

Krzysztof CZAJKOWSKI
Politechnika Krakowska, Instytut Teleinformatyki
Mieczysław DRABOWSKI
Politechnika Krakowska, Katedra Informatyki Technicznej

WYBRANE ZAGADNIENIA INTEGRACJI ZBIORÓW PRZYBLIŻONYCH I BAZ DANYCH

Streszczenie. Artykuł zawiera wybrane propozycje integracji pojęć i metod charakterystycznych dla zbiorów przybliżonych z technikami stosowanymi w relacyjnych bazach danych. Integracja ta ma na celu zwiększenie wydajności w realizacji złożonych obliczeniowo operacji. W pracy zaprezentowano implementacje w środowisku bazodanowym algorytmu selekcji atrybutów nieusuwalnych oraz metody do znajdowania reduktów. Przedstawiono oryginalne przykłady zastosowania takich podejść integracyjnych.

Słowa kluczowe: zbiory przybliżone, redukcja atrybutów, atrybuty nieusuwalne, redukty, kwerendy

SELECTED ISSUES OF ROUGH SETS AND DATABASE INTEGRATION

Summary. This paper includes selected propositions of concepts and methods integration that are characteristic for rough sets with techniques applied in relational databases. This integration has the aim to increase efficiency in realization of complex computational operations. In this work there have been presented implementations of the algorithm of core attributes selection and method for reducts finding in database environment. There have been presented original examples of such integration approaches application.

Keywords: rough sets, attributes reduction, core attributes, reducts, queries

1. Wstęp

W przypadku tradycyjnego podejścia do zbiorów przybliżonych algorytmy wyznaczające atrybuty nieusuwalne (rdzeń) i redukty oraz identyfikujące atrybuty zbędne mają zbyt dużą złożoność obliczeniową i są nieefektywne. Obliczenia te, w wielu systemach wykorzystujących zbiory przybliżone, są wykonywane z wykorzystaniem zwykłych plików „płaskich", nie korzystając ze zdecydowanie większej wydajności dostępnej w relacyjnych systemach baz danych.

Pojawiła się propozycja zwiększenia efektywności wyszukiwania atrybutów nieusuwalnych dzięki integracji z relacyjnym systemem baz danych, takie jak Rough Set Data Miner [3]. Stosuje ona wbudowane zapytania SQL w celu wykorzystania zalet technologii bazodanowych. Szczególnie interesujące wydaje się podejście zaproponowane w [4], oparte na zastosowaniu algebry relacyjnej do wyznaczania atrybutów nieusuwalnych i reduktów. Opierając się na operacjach bazodanowych, działających na zbiorach, takich jak zliczanie oraz projekcja, zwiększa ono szybkość wyznaczania atrybutów kluczowych. Dzięki temu możliwe staje się wykorzystanie efektywnych rozwiązań, takich jak indeksowanie i sortowanie. Wydajne implementacje języka SQL pozwalają zredukować koszt dostępu do pamięci dyskowej i dobrze sobie radzą w przypadku wielkich ilości danych.

W tym artykule omówiono metody redukcji atrybutów w teorii zbiorów przybliżonych, z wykorzystaniem bazy danych i algebry relacyjnej, zaprezentowanych w [4] i przedstawiono implementacje tych metod, wskazujące zalety takiego podejścia. Zaprezentowano przykłady zastosowania algorytmów ilustrujących te metody i pokazujące ich poprawność i efektywność.

2. Zbiory przybliżone w ujęciu bazodanowym

W teorii zbiorów przybliżonych dane mogą być przedstawione w postaci tablicy decyzyjnej, w której wiersze odpowiadają obiektom, a kolumny atrybutom tych obiektów. Niektóre z tych atrybutów (najczęściej jeden) tworzą zbiór atrybutów decyzyjnych (oznaczany przez *D*), podczas gdy pozostałe tworzą zbiór atrybutów warunkowych (oznaczany przez *C*). Formalnie tablicą decyzyjną nazywamy uporządkowaną piątkę [10]:

$$\mathrm{DT}=(U,C,D,V,f) \tag{1}$$

gdzie: $C, D \subset A; C \neq \varnothing, D \neq \varnothing; C \bigcup D = A; C \bigcap D = \varnothing$. U jest niepustym skończonym zbiorem zwanym uniwersum tablicy decyzyjnej, elementy tego zbiory nazywamy obiektami. *f* nazywamy funkcją decyzyjną. $V = \bigcup_{a \in A} V_a$, przy czym V_a nazywamy dziedziną atrybutu $a \in A$.

Zakładając, że $B \subseteq C$, zbiór atrybutów Q ($\subseteq B$) nazywamy reduktem zbioru atrybutów B względem atrybutu decyzyjnego d (przy założeniu że zbiór atrybutów decyzyjnych jest jednoelementowy), gdy zbiór atrybutów Q jest niezależny oraz IND(B,d)=IND(Q,d). IND(B,d) to relacja nierozróżnialności względem decyzji d, generowana przez zbiór atrybutów B. Redukt jest najmniejszym zbiorem atrybutów, przy którym zostaje zachowana dotychczasowa klasyfikacja (rozróżnialność) obiektów. W tabeli decyzyjnej może występować więcej niż jeden redukt.

Rdzeniem, oznaczanym jako CORE(B,d), nazywamy zbiór wszystkich atrybutów niezbędnych (nieusuwalnych) w zbiorze B, ze względu na atrybut decyzyjny d. Rdzeń stanowi część wspólna wszystkich reduktów.

Istotność atrybutu jest miarą skutków, jakie może wywołać usunięcie danego atrybutu warunkowego. Wartość tego współczynnika reprezentuje wkład wnoszony przez atrybut do zależności pomiędzy atrybutami warunkowymi a atrybutami decyzyjnymi.

W prezentowanym podejściu podstawowe pojęcia z teorii zbiorów przybliżonych zostały zdefiniowane z wykorzystaniem pojęć z teorii baz danych. Celem takiego rozwiązania jest zastosowanie mechanizmów istniejących w bazach danych dla uzyskania większej wydajności w wyznaczaniu atrybutów nieusuwalnych oraz reduktów względnych.

Wszystkie atrybuty nieusuwalne są niezbędnymi elementami każdego reduktu [1], tak więc kluczowym problemem jest możliwość efektywnego wyszukiwania takich atrybutów. W podejściu tradycyjnym popularną metodą jest konstruowanie macierzy decyzyjnej, a następnie przeszukiwanie wszystkich pozycji w takiej macierzy celem znalezienia takich, które mają tylko jeden atrybut [6]. Metoda ta cechuje się niesatysfakcjonującą wydajnością, szczególnie w przypadku systemów gromadzących bardzo duże ilości danych. Z kolei inne metody, niewykorzystujące macierz decyzyjną, np. [8], mają złożoność obliczeniową O(mnlog(n)) (gdzie n to liczba wierszy, a m jest liczbą atrybutów).

Prezentowane podejście ma złożoność O(mn) [4] i jest realizowane bez potrzeby wyznaczania dolnego i górnego przybliżenia.

Oznaczając operację relacyjną zliczania (Count) przez Card, operację projekcji (Projection) jako Π, a zbiór atrybutów decyzyjnych jako D, możemy zapisać, że $C_j \in C$ jest atrybutem nieusuwalnym, jeżeli zachodzi (2).

$$Card\left(\Pi\left(C - C_j + D\right)\right) \neq Card\left(\Pi\left(C - C_j\right)\right) \tag{2}$$

Jeżeli liczba wierszy, które można rozróżnić za pomocą aktualnych wartości w atrybutach warunkowych (z pominięciem danego atrybutu) oraz atrybutu decyzyjnego, jest różna od liczby rozróżnialnych wierszy za pomocą tego samego podzbioru atrybutów warunkowych, ale bez udziału atrybutu decyzyjnego, oznacza to, że usunięcie spośród atrybutów warunko-

wych tego konkretnego atrybutu powoduje częściową utratę możliwości identyfikowania obiektów – atrybut ten jest nieusuwalny.

Wynika z tego, że można ustalić, czy atrybut jest atrybutem nieusuwalnym za pomocą podstawowych operacji języka SQL. Niezbędne jest wykonanie tylko dwóch projekcji: jednej na atrybutach $C–C_j+D$, a drugiej na atrybutach $C–C_j$. Jeżeli liczba wierszy w obu projekcjach jest różna, to dany atrybut jest atrybutem nieusuwalnym, w przeciwnym przypadku jest on zbędny (nie ma utraty informacji, po usunięciu atrybutu – każdy obiekt może być sklasyfikowany w ten sam sposób, bez względu na to, czy atrybut jest obecny, czy też nie).

Z drugiej strony, jeżeli liczba wierszy w obu projekcjach się nie zmienia – jak w (3) – to rozpatrywany atrybut jest atrybutem usuwalnym (zbędnym).

$$Card\left(\Pi\left(C - C_j + D\right)\right) = Card\left(\Pi\left(C - C_j\right)\right) \tag{3}$$

W celu wyznaczenia reduktów zbioru atrybutów warunkowych, można zdefiniować stopień zależności pomiędzy reduktem a zbiorem atrybutów decyzyjnych – jak w (4).

$$K(REDU, D) = \frac{Card\left(\Pi\left(REDU + D\right)\right)}{Card\left(\Pi\left(C + D\right)\right)} \tag{4}$$

Podzbiór zbioru atrybutów warunkowych $REDU(\subseteq C)$ jest reduktem zbioru atrybutów warunkowych C ze względu na zbiór atrybutów decyzyjnych D, jeżeli jest minimalnym zbiorem atrybutów, które mają taką samą zdolność klasyfikowania jak cały zbiór atrybutów warunkowych (5) (6).

$$K(REDU, D) = K(C, D) \tag{5}$$

oraz

$$K(REDU, D) \neq K(R', D), \forall R' \subset REDU \tag{6}$$

Jeżeli współczynnik $K(REDU,D)$ jest równy 1, to zbiór atrybutów decyzyjnych D całkowicie zależy od zbioru atrybutów warunkowych C. Przy $K(REDU,D) < 1$, mówimy o zależności częściowej (o stopniu $K(REDU,D)$) [2].

Idea redukcji atrybutów może być uogólniona przez wprowadzenie pojęcia istotności atrybutu. Nie jest stosowana dwuwartościowa skala: znaczący i nieznaczący. Zamiast tego konkretnym atrybutom przypisane są wartości z przedziału [0,1], wyrażające istotność tych atrybutów dla tablicy decyzyjnej. Istotność atrybutu może zostać określona przez efekt wywołany jego usunięciem z tablicy decyzyjnej [2].

Współczynnik istotności dla danego atrybutu C_j ze zbioru atrybutów C definiowano jako:

$$Merit(C_j, C, D) = 1 - \frac{Card\left(\Pi\left(C - C_j + D\right)\right)}{Card\left(\Pi\left(C + D\right)\right)} \tag{7}$$

Wartość tego współczynnika reprezentuje wkład wnoszony przez atrybut C_j do zależności pomiędzy C (atrybutami warunkowymi) a D (atrybutami decyzyjnymi). Większa wartość tego współczynnika oznacza, że jest on istotniejszy w rozpatrywanej tablicy decyzyjnej.

3. Wyszukiwanie atrybutów nieusuwalnych

W prezentowanym podejściu zakładamy, że w tabeli decyzyjnej nie występują krotki sprzeczne (niespójne), to jest takie, które, posiadając identyczne wartości atrybutów warunkowych, mają inne wartości atrybutów decyzyjnych (należą do innej klasy). Podejście prezentowane poniżej nie będzie klasyfikowało takich krotek, dlatego należy je uprzednio wyeliminować z tabeli, przed rozpoczęciem procesu przetwarzania. Przedstawiany algorytm [4] opiera się na operacjach istniejących w systemie bazodanowym, bez konieczności obliczania dolnego i górnego przybliżenia i cechuje się większą wydajnością od podejścia tradycyjnego.

```
Algorytm 1: Core Attribute Algorithm
Wejście: tablica decyzyjna T (C,D)
Wyjście: Core – zbiór atrybutów nieusuwalnych (rdzeń) dla tabeli T

   1. Ustaw Core = 0
   2. Dla każdego atrybutu A ∈ C {
      If Card(Π(C – A + D)) ≠ Card(Π(C – A))
      Then Core Core ∪ A
   }
```

W pierwszym kroku algorytmu następuje inicjalizacja zbioru atrybutów nieusuwalnych (rdzenia – Core) na wartość pustą. W kolejnym kroku następuje sprawdzanie wszystkich atrybutów warunkowych, czy są atrybutami nieusuwalnymi – w takim przypadku są dołączane do zbioru atrybutów Core.

Można zapisać Algorytm 1 jako operację: $Card(\Pi(X))$, gdzie X może być C, C–A, C–A+D. Wykorzystując język SQL, operacja ta może zostać wyrażona za pomocą polecenia SELECT:

```
SELECT COUNT(*) FROM (SELECT DISTINCT X FROM T);
```

Jak udowodniono w [4], algorytm 1 może być zaimplementowany przy złożoności obliczeniowej O(mn), gdzie m to liczba atrybutów, natomiast n jest liczbą wierszy (zakładając wykorzystanie indeksów).

W powyższych rozważaniach zakładamy brak występowania niespójności danych. W rzeczywistości takie niespójności (zaszumienia) pojawiają się i powinny zostać wyeliminowane. W celu wykrycia, czy w zbiorze danych istnieją niespójności, wystarczy sprawdzić, czy zachodzi równość $Card(\Pi(C)) = Card(\Pi(C + D))$. Jeżeli równość nie jest spełniona, oznacza to, że istnieją dane niespójne.

W celu ustalenia, które wiersze są niespójne, wystarczy wykonać poniższe zapytanie:

```
SELECT * FROM T U WHERE EXISTS (SELECT * FROM T V WHERE (U.C=V.C) AND
(U.D<>V.D));
```

W ten sposób niespójności w danych mogą być wyeliminowane w czasie O(n), przy n równym ilości wierszy w tabeli.

4. Znajdowanie reduktów względnych

W tabeli decyzyjnej występować mogą dwa rodzaje atrybutów zbędnych z punktu widzenia ich znaczenia dla opisu obiektów. Pierwszym z nich są atrybuty nieistotne, takie jak różnego typu identyfikatory. Drugi rodzaj stanowią atrybuty nadmiarowe w stosunku do pozostałych atrybutów. Niektóre atrybuty są niezbędne dla poprawnej klasyfikacji, ale niekoniecznie wszystkie jednocześnie. W celu zredukowania liczby niepotrzebnych atrybutów do minimum można przeprowadzić selekcję atrybutów. Proces ten ma na celu wybranie podzbioru zbioru atrybutów, który składać się będzie tylko z tych atrybutów, które są istotne, przy jednoczesnym wyeliminowaniu atrybutów nieznaczących. Pozwala to zminimalizować czas w procesie dalszego przetwarzania informacji.

W tabeli decyzyjnej może występować więcej niż jeden redukt. Znalezienie wszystkich reduktów w tabeli decyzyjnej jest zadaniem NP-trudnym [9]. Istnieją jednak zastosowania, w których nie występuje konieczność znalezienia wszystkich reduktów, wystarczy znalezienie jednego z nich. Nasuwa się pytanie, który z reduktów jest najlepszy, jeżeli istnieje więcej niż jeden. Wybór uzależniony jest od przyjętego kryterium. Jeżeli z atrybutem związana jest funkcja kosztu, wówczas wybór będzie opierał się na minimalizowaniu kosztu całkowitego atrybutów. W przypadku niewystępowania funkcji kosztu jedynym źródłem informacji jest zawartość tabeli. W prezentowanym algorytmie [4] jako kryterium wyboru reduktu przyjęto, że w przypadku występowania większej ich liczby najlepszy jest ten, który ma najmniej atrybutów. W przypadku, gdyby kilka reduktów miało taką samą liczbę atrybutów, należy wybrać taki, który ma mniejszą liczbę kombinacji swoich atrybutów.

Częścią wspólną wszystkich reduktów jest zbiór atrybutów nieusuwalnych (rdzeń).

$$CORE(C) = \bigcap REDU(C) \tag{8}$$

Każdy element rdzenia stanowi część każdego redukatu, dlatego też zbiór atrybutów nieusuwalnych jest najważniejszym podzbiorem zbioru atrybutów warunkowych [1]. Żaden element z tego zbioru nie może zostać usunięty bez skutków w postaci mniejszych zdolności klasyfikacyjnych zbioru atrybutów warunkowych.

Poniżej zamieszczony jest algorytm (zachłanny) wyznaczania reduktów w procesie selekcji i eliminacji.

```
Algorytm 2: Wyznaczanie minimalnego podzbioru atrybutów (reduktu)
Wejście: Tabela decyzyjna T (C,D)
Wyjście: Minimalny podzbiór zbioru atrybutów (REDU)
```

- Uruchom Algorytm 1 w celu wyznaczenia atrybutów kluczowych CORE
- REDU = CORE
- AR = C – CORE
  {Forward selection}
- WHILE K(REDU,D) ≠ K(C,D)
  - Oblicz współczynnik istotności (merit) dla każdego atrybutu ze zbioru AR
  - Posortuj atrybuty w AR w oparciu o wartość merit w porządku malejącym
  - Wybierz atrybut C_j posiadający największą wartość merit (w sytuacji, gdy kilka atrybutów ma taką samą wartość merit, wybierz ten atrybut, który ma mniejszą liczbę kombinacji z atrybutami w REDU)
  - REDU = REDU ∪ C_j, AR = AR – C_j
- ENDWILE
  {Backward elimination}
- N = ||REDU||
- FOR j=0 to N-1 DO
  - IF a_j nie jest elementem CORE THEN oblicz K(REDU-a_j,D)
  - IF K(REDU-a_j,D) = K(REDU,D) THEN usuń a_j z REDU
- ENDFOR

Zgodnie z (7), pierwszym etapem działania Algorytmu 2 jest wykonanie operacji Algorytmu 1. Ustawienie warunków początkowych polega na przepisaniu atrybutów rdzenia (CORE) do zbioru atrybutów REDU oraz usunięciu atrybutów CORE ze zbioru AR, który będzie stanowił podstawę dalszego przetwarzania, w celu ustalenia pozostałych elementów zbioru REDU. Ustalenie zbioru CORE i usunięcie jego elementów z dalszego przetwarzania jest korzystne z tego względu, iż wyznaczanie jego elementów jest operacją mniej złożoną niż operacje służące ustaleniu atrybutów REDU, co zostało zaprezentowane w kolejnym rozdziale.

W kolejnych krokach algorytmu, w części Forward selection, wybierane są do zbioru atrybutów stanowiącego redukt tylko te atrybuty, które mają największy wkład w zależność pomiędzy zbiorem atrybutów warunkowych a atrybutem decyzyjnym (największą wartość współczynnika Merit). W ten sposób nie są brane pod uwagę atrybuty, które są nieistotne z punktu widzenia klasyfikacji.

Ponieważ jednak dodawanie kolejnych atrybutów odbywa się na podstawie współczynnika istotności, wyliczanego dla aktualnej listy atrybutów AR, możliwe jest, że do zbioru REDU zaliczone zostaną atrybuty, które na danym etapie pracy algorytmu wydają się istotne, ale po zakończeniu pracy algorytmu okazują się zbędne. Dlatego w dalszej części algorytmu, w sekcji Backward elimination, usuwane są ze zbioru REDU te atrybuty, których pominięcie nie ma wpływu na zdolność klasyfikacyjną zbioru REDU. Sprawdzane są wszystkie elementy aktualnie znajdujące się w REDU i usuwane są te spośród nich, które nie są niezbędne do pełnej klasyfikacji.

Prezentowany algorytm cechuje się większą wydajnością niż inne podejścia, ponieważ przetwarza zbiory dyskowe obsługiwane przez system bazodanowy i wszystkie działania służące ustalaniu niezbędności elementu w zbiorze atrybutów decyzyjnych (z punktu widzenia możliwości klasyfikacyjnych) oraz wyliczanie współczynników istotności dla poszczególnych atrybutów, odbywa się za pomocą operacji pod kontrolą systemu zarządzania relacyjnymi bazami danych.

5. Implementacje

Jako środowisko do zaimplementowania omawianych rozwiązań wybrany został system Oracle. Tablica decyzyjna reprezentowana jest w bazie danych jako tabela, której kolumny reprezentują poszczególne atrybuty warunkowe i decyzyjne, natomiast kolejne obiekty reprezentowane są w postaci rekordów. Operacje pozwalające określić istotność poszczególnych atrybutów warunkowych pod względem klasyfikowania obiektów realizowane są za pomocą wyrażeń języka SQL. Odpowiednie zapytania są generowane dynamicznie, stosownie do listy aktualnie rozpatrywanych, na danym etapie pracy algorytmu, atrybutów.

Algorytmy zostały zapisane jako zestaw procedur i funkcji w języku PL/SQL. Wszystkie procedury i funkcje zebrane zostały w ramach jednego pakietu. Poniżej przedstawiono specyfikację tego pakietu, stanowiącą listę jego publicznych elementów składowych.

```
CREATE OR REPLACE PACKAGE db_rs
IS
   PROCEDURE findCORE(dec_tab_in VARCHAR2);
   PROCEDURE findREDU(dec_tab_in VARCHAR2);
END db_rs;
```

Spoza pakietu dostępne są tylko dwie funkcje: *findCORE* oraz *findREDU*. Pierwsza z nich służy do określania zbioru atrybutów nieusuwalnych (rdzenia) ze zbioru atrybutów warunkowych. Druga ma za zadanie wyznaczenie reduktu. Każda z funkcji jako argument otrzymuje nazwę tabeli przechowującej dane tablicy decyzyjnej. Wszystkie pozostałe funkcje, mające charakter pomocniczy, zostały zadeklarowane w ciele pakietu i dostępne są tylko z poziomu jego wnętrza.

Na kolejnym listingu przedstawiono kod początkowego fragmentu ciała pakietu.

```
CREATE OR REPLACE PACKAGE BODY db_rs
IS
  TYPE attr_type IS TABLE OF VARCHAR2(30) INDEX BY BINARY_INTEGER;
  all_attr attr_type;
  core_attr attr_type;
  redu_attr attr_type;
...
```

Jako zmienne globalne zostały zadeklarowane trzy kolekcje: *all_attr, core_attr* oraz *redu_attr*. Każda z nich jest kolejką typu tablica asocjacyjna, której elementami są zmienne

typu varchar2. Kolekcje te służą do przechowywania nazw atrybutów, odpowiednio: *all_attr* – pełna lista atrybutów tablicy decyzyjnej, *core_attr* – lista atrybutów tworzących rdzeń, *redu_attr* – atrybuty stanowiące znaleziony redukt.

Poniżej zaprezentowana jest procedura znajdująca zbiór atrybutów nieusuwalnych – *findCORE*. W tej procedurze następuje przepisanie do kolekcji *all_attr* nazw wszystkich kolumn, z jakich składa się wejściowa tabela. Jest to o tyle istotne, że dzięki temu cały pakiet jest uniwersalny – działać może dla każdej tabeli, która zostanie podana na wejściu, bez zmiany kodu. Zarówno liczba, jak i nazwy konkretnych kolumn są ustalane w trakcie działania programu, na podstawie metadanych przechowywanych w bazie danych i dostępnych przez perspektywy systemowe. Po ustaleniu listy kolumn zliczana jest liczba wierszy w tabeli.

Zasadniczym fragmentem procedury *findCORE* jest pętla, która przechodzi po wszystkich atrybutach tablicy decyzyjnej (czyli po wszystkich elementach kolekcji *all_attr*). Ustalane jest (za pomocą poleceń SELECT, w funkcji *count_rows*), czy dla konkretnego elementu zachodzi nierówność opisana wzorem (2), a więc czy liczba wierszy rozróżnialnych za pomocą atrybutów warunkowych z pominięciem tego konkretnego atrybutu, ale z uwzględnieniem atrybutu decyzyjnego (*count_rows_C_A_D*) jest różna od liczby wierszy rozróżnialnych za pomocą tych samych atrybutów warunkowych, ale bez atrybutu decyzyjnego (*count_rows_C_A*). Jeżeli taka nierówność zachodzi, to atrybut jest atrybutem nieusuwalnych i jego nazwa jest dopisywana do kolekcji *core_attr*. Gdy natomiast nierówność jest nieprawdziwa, (czyli występuje równość opisana wzorem (3)), atrybut jest pomijany.

```
PROCEDURE findCORE(dec_tab_in VARCHAR2)
IS
  CURSOR col_list_cur IS SELECT column_name FROM user_tab_cols W
    HERE table_name   = dec_tab_in;
  attr_idx BINARY_INTEGER := 1;
  attr_idx_loc BINARY_INTEGER := 1;
  count_rows_C_A_D NUMBER(5);
  count_rows_C_A NUMBER(5);
BEGIN
  FOR col_list_rec IN col_list_cur
  LOOP
    all_attr(attr_idx) := col_list_rec.column_name;
    attr_idx:=attr_idx+1;
  END LOOP;
  SELECT COUNT(*) INTO attr_nr FROM user_tab_cols WHERE table_name = dec_tab;
  attr_idx := all_attr.FIRST;
  LOOP
    attr_idx := all_attr.NEXT(attr_idx);
    EXIT WHEN attr_idx IS NULL;
    count_rows_C_A_D := count_rows(attr_nr,attr_idx);
    count_rows_C_A := count_rows(attr_nr-1,attr_idx);
    IF count_rows_C_A_D <> count_rows_C_A THEN
      core_attr(attr_idx) := all_attr(attr_idx);
    END IF;
  END LOOP;
END findCORE;
```

Funkcja *count_rows* realizuje opisane wcześniej polecenia SELECT, generując dynamicznie ich treść i zapisując ją do zmiennej *sql_statement*. Następnie polecenia te są realizo-

wane jako dynamiczny SQL. Dzięki takiemu rozwiązaniu nie jest konieczna ich znajomość w trakcie tworzenia funkcji. W ten sposób prezentowany pakiet może działać dla dowolnej tablicy decyzyjnej, której struktura jest określana przez jednostki pakietu dopiero w chwili jej przetwarzania.

Operacje na atrybutach odbywają się z pominięciem pierwszego z nich, który służy jako identyfikator wierszy i nie jest istotny w punktu widzenia algorytmu. Do funkcji przekazywane są, jako parametry wejściowe, dwa atrybuty: *attr_nr* służący do określenia, czy w danym wywołaniu funkcja ma brać pod uwagę atrybut decyzyjny, czy też nie, oraz atrybut *attr_idx* oznaczający numer (w ramach kolekcji) tego atrybutu, który w danym wywołania ma być pominięty przy konstruowaniu zapytania.

```
FUNCTION count_rows(attr_nr NUMBER, attr_idx BINARY_INTEGER) RETURN NUMBER IS
  attr_idx_loc BINARY_INTEGER := 1;
  sql_statement VARCHAR2(500);
  sql_statement_cols VARCHAR2(500);
  count_rows NUMBER(5);
BEGIN
    attr_idx_loc := 2;
    sql_statement_cols := '';
    WHILE attr_idx_loc <= attr_nr LOOP
      IF attr_idx_loc <> attr_idx AND sql_statement_cols IS NULL THEN
        sql_statement_cols := all_attr(attr_idx_loc);
      ELSIF attr_idx_loc <> attr_idx THEN
        sql_statement_cols := sql_statement_cols || ','
          || all_attr(attr_idx_loc);
      END IF;
      attr_idx_loc := all_attr.NEXT(attr_idx_loc);
    END LOOP;
    sql_statement := 'SELECT COUNT(*) FROM (SELECT DISTINCT '
      || sql_statement_cols || ' FROM '||dec_tab||')';
    EXECUTE IMMEDIATE sql_statement INTO count_rows;
  RETURN count_rows;
END count_rows;
```

W podobny sposób do opisanego powyżej realizowane jest wyznaczanie reduktu. Utworzono do tego celu procedurę *findREDU*, której kluczowe fragmenty przedstawione są poniżej. Podstawą działania jest zmienna *ar_attr*, która jest kolekcją typu *attr_type_ar* złożonego ze zmiennych typu rekordowego *attr_rec*, który z kolei składa się z dwóch elementów: name (do przechowywania nazw atrybutów) i *value* (dla wartości współczynnika istotności, wyznaczanego dla każdego atrybutu).

Pierwszym krokiem w tej procedurze jest uruchomienie procedury *findCORE*. Znalezione atrybuty stanowiące rdzeń są następnie przepisywane do kolekcji *redu_attr*, natomiast pełna lista atrybutów jest przepisywana do kolekcji *ar_attr* (zgodnie z Algorytmem nr 2).

Następnie, po usunięciu z kolekcji ar_attr atrybutów rdzenia oraz atrybutu pierwszego (identyfikatora) i ostatniego (decyzyjnego), następuje w pętli, której liczba powtórzeń ograniczona jest wartością współczynnika *K*, określanie istotności danego atrybutu. Jednakże ponieważ we wzorze (6) zmienia się tylko licznik ułamka, to zamiast wyznaczać Merit i wybierać atrybut o największej jego wartości, wyliczany jest tylko licznik ułamka i wybierany atrybut o najmniejszej wartości tego licznika (co jest równoznaczne z największą

atrybut o najmniejszej wartości tego licznika (co jest równoznaczne z największą wartością Merit). Atrybut o najmniejszej wartości licznika jest dodawany do zbioru reduktów (kolekcja *redu_attr*) oraz usuwany z kolekcji *ar_attr*. Określenie wartości współczynnika K i ewentualne przerwanie wykonywania pętli kończy ten fragment procedury (Forward selection). Następnie wykonywane jest usuwanie atrybutów, które w wyznaczonym redukcie okazują się jednak zbędne (na zasadzie eliminacji wstecznej – fragment Backward elimination).

```
PROCEDURE findREDU(dec_tab_in VARCHAR2)
IS
  TYPE attr_rec IS RECORD (name VARCHAR2(20), value REAL);
  TYPE attr_type_ar IS TABLE OF attr_rec INDEX BY BINARY_INTEGER;
  ar_attr attr_type_ar;
.............................
BEGIN
  findCORE(dec_tab_in);
  redu_attr := core_attr;
  attr_idx := all_attr.FIRST;
  WHILE attr_idx IS NOT NULL LOOP
    ar_attr(attr_idx).name := all_attr(attr_idx);
    attr_idx := all_attr.NEXT(attr_idx);
  END LOOP;
.............................
  LOOP
    attr_idx := ar_attr.FIRST;
    WHILE attr_idx IS NOT NULL LOOP
      ar_attr(attr_idx).value := count_rows_redu(attr_nr,attr_idx);
      attr_idx := ar_attr.NEXT(attr_idx);
    END LOOP;
.............................
    redu_attr(min_dist_idx) := ar_attr(min_dist_idx).name;
    ar_attr.delete(min_dist_idx);
.............................
    EXIT WHEN k_value = 1;
  END LOOP;
.............................
END findREDU;
```

Funkcja *count_rows_redu* przedstawiona jest poniżej. Zwraca ona liczbę wierszy, wyznaczaną przez zapytanie skorelowane, którego podzapytanie zwraca wartość tylko dla tych wierszy z zapytania nadrzędnego, dla których nie istnieją inne wiersze o takich samych wartościach wszystkich atrybutów warunkowych, a innej wartości atrybutu decyzyjnego. W ten sposób można ustalić, który z atrybutów jest najistotniejszy, z uwagi na to, iż jego usunięcie daje pozostałym atrybutom najmniejsze możliwości klasyfikowania.

```
FUNCTION count_rows_redu(attr_nr NUMBER, attr_idx BINARY_INTEGER) RETURN NUMBER
IS
  attr_idx_loc BINARY_INTEGER := 1;
  sql_statement VARCHAR2(1000);
  sql_statement_cols VARCHAR2(500);
  count_rows NUMBER(5);
BEGIN
  attr_idx_loc := all_attr.FIRST;
  attr_idx_loc := all_attr.NEXT(attr_idx_loc);
    sql_statement_cols := '';
    WHILE attr_idx_loc <= attr_nr - 1 LOOP
      IF attr_idx_loc <> attr_idx AND sql_statement_cols IS NULL THEN
        sql_statement_cols := 'a.' || all_attr(attr_idx_loc) || '=' ||
          all_attr(attr_idx_loc);
      ELSIF attr_idx_loc <> attr_idx THEN
```

```
        sql_statement_cols := sql_statement_cols || ' AND ' ||
          'a.' || all_attr(attr_idx_loc) || '=' || all_attr(attr_idx_loc);
      END IF;
      attr_idx_loc := all_attr.NEXT(attr_idx_loc);
    END LOOP;
    sql_statement := 'SELECT COUNT(*) FROM ' || dec_tab ||
      ' a WHERE NOT EXISTS ' || ' (SELECT 1 FROM ' || dec_tab ||
      ' WHERE ' ||  sql_statement_cols || ' AND a.'
      || all_attr(attr_idx_loc) || '<>' || all_attr(attr_idx_loc) || ')';
    EXECUTE IMMEDIATE sql_statement INTO count_rows;
  RETURN count_rows;
END count_rows_redu;
```

6. Przykład zastosowania

Do zaprezentowania omawianego podejścia w niniejszej pracy wykorzystano zbiór danych Wisconsin Breast Cancer z repozytorium ICS UCI [17]. Dane zostały zebrane metodą biopsji cienkoigłowej [18]. Jest to metoda stosunkowo tania i mało inwazyjna, jednak trudno za jej pomocą uzyskać dobrej jakości materiał i trudno go ocenić. Polega na pobraniu materiału z piersi za pomocą cienkiej igły, a następnie ocenie skupisk komórek. Materiał z łagodną odmianą nowotworu zawiera komórki małe, o takim samym kształcie i rozmiarze, układające się ciasno w pojedynczą warstwę. Nieliczne komórki oddalone od grupy są pozbawione cytoplazmy. W przypadku odmiany złośliwej komórki są większe, różnią się między sobą rozmiarem i kształtem. Komórki oddzielają się od grupy z nienaruszoną cytoplazmą. Barwa jąder komórek jest jednolita w przypadku odmiany łagodnej, w przeciwieństwie do złośliwej, gdzie kolory są niejednolite. Rozróżnienie pomiędzy formą łagodną a złośliwą nie zawsze jest jednoznaczne. System komputerowy dokonuje analizy komórek (kształtu, rozmiaru, regularności, koloru itp.) i zwraca zestaw wartości liczbowych opisujących badaną próbkę. Postać próbki przedstawiono w tabeli 1.

Zestaw danych zawiera 239 próbek sklasyfikowanych jako nowotwór złośliwy oraz 444 sklasyfikowane jako odmiana łagodna [16] (w sumie 683 próbki zawierające pełny zestaw danych). Dane są dostępne w postaci pliku tekstowego, dlatego należy zaimportować je do bazy danych. Do tego celu utworzona została tabela zewnętrzna podłączona do pliku tekstowego. Następnie dane zostały skopiowane do tabeli tradycyjnej (chociaż możliwa byłaby praca bezpośrednio na tabeli zewnętrznej, to nie jest to wydajne rozwiązanie).

Tabela 1

Opis próbek wykorzystanych w eksperymencie

Parametr (oryginał)	Parametr (polski)	Wartości
Clump Thickness	Gęstość grupy	1-10
Uniformity of Cell Size	Jednorodność rozmiaru komórki	1-10
Uniformity of Cell Shape	Jednorodność kształtu komórki	1-10
Marginal Adhesion	Przyleganie komórek	1-10
Single Epithelial Cell Size	Nabłonkowy rozmiar komórki	1-10

Bare Nuclei	Odsłonięte jądra	1-10
Bland Chromatin	Łagodna chromatyna	1-10
Normal Nucleoli	Jądra zwyczajne	1-10
Mitoses	Mitozy	1-10
Class	Diagnoza	2 – łagodny 4 – złośliwy

Poniżej zaprezentowane jest polecenie tworzące tabelę zewnętrzną, która umożliwia pobranie danych z pliku tekstowego (w tym przypadku z pliku bc_data.txt). Dane z tej tabeli są następnie kopiowane do bazy danych, do tabeli o takiej samej strukturze, ale której dane przechowywane są juz w plikach danych. W ten sposób możliwe jest wykorzystanie wszystkich mechanizmów wydajnościowych (indeksy, statystyki), które udostępnia system zarządzania relacyjną bazą danych.

```
CREATE TABLE bc_data_imp (
   id NUMBER(8),
   clump_th NUMBER(2),
   uni_cell_size NUMBER(2),
   uni_cell_shape NUMBER(2),
   marg_ad NUMBER(2),
   sing_epi_cell_size NUMBER(2),
   bare_nu NUMBER(2),
   blan_ch NUMBER(2),
   norm_nu NUMBER(2),
   mitoses NUMBER(2),
   class NUMBER(1)
)
ORGANIZATION EXTERNAL
(
   TYPE ORACLE_LOADER
   DEFAULT DIRECTORY data_imp
   ACCESS PARAMETERS
   (
      RECORDS DELIMITED BY NEWLINE
      FIELDS TERMINATED BY ','
   )
   LOCATION ('bc_data.txt')
)
REJECT LIMIT 100
;
```

W wyniku działania procedury *findCORE* z pakietu *db_rs*, dla rozpatrywanej tablicy decyzyjnej został wyznaczony jeden atrybut nieusuwalny: *bare_nu*. Usunięcie tego elementu ze zbioru elementów warunkowych jest niemożliwe bez zmniejszenia możliwości klasyfikacyjnych. Polecenie SELECT DISTINCT (z usunięciem duplikatów) wygenerowane w pakiecie *db_rs*, bez atrybutu *bare_nu* oraz bez atrybutu decyzyjnego (*class*) zwraca mniejszą wartość (jest mniej rekordów spełniających kryterium) niż dla polecenia bez atrybutu *bare_nu*, ale z uwzględnieniem atrybutu decyzyjnego.

```
SELECT count(*) FROM (SELECT distinct clump_th, uni_cell_size, uni_cell_shape,
marg_ad, sing_epi_cell_size, blan_ch, norm_nu, mitoses FROM bc_data);

wynik: 432 wiersze spełniające kryterium

SELECT count(*) FROM (SELECT distinct clump_th, uni_cell_size, uni_cell_shape,
```

```
marg_ad, sing_epi_cell_size, blan_ch, norm_nu, mitoses, class FROM bc_data);

wynik: 433 wiersze spełniające kryterium
```

Z uwagi na fakt, iż elementy rdzenia są częścią składową wszystkich reduktów, w każdym z nich dla rozpatrywanej tabeli musi znajdować się atrybut *bare_nu*. Potwierdza to wykorzystanie dla tego przypadku środowiska Rosetta [20] – we wszystkich reduktach znajduje się ten konkretny atrybut.

Funkcja findREDU jako zbiór atrybutów stanowiących redukt, w pierwszym etapie swojego działania (sekcja Forward selection), proponuje zbiór pięcioelementowy: *clump_th, marg_ad, sing_epi_cell_size, bare_nu, blan_ch*. Taki zbiór atrybutów ma faktycznie takie same zdolności klasyfikacyjne jak pełny zbiór atrybutów warunkowych, spełniając tym samym założenie określone wzorem (5). Klasyfikowane są poprawnie 683 przypadki tak samo jak dla pełnego zbioru atrybutów warunkowych.

```
SELECT count(*) FROM bc_data a WHERE NOT EXISTS
(SELECT 1 FROM bc_data WHERE a.clump_th=clump_th AND a.marg_ad=marg_ad AND
a.sing_epi_cell_size=sing_epi_cell_size AND a.bare_nu= bare_nu AND a.blan_ch=
blan_ch AND a.class<>class);

wynik: 683 wiersze spełniające kryterium
```

Jednak ustalony w tym fragmencie (sekcja Forward selection) zbiór pięciu elementów nie spełnia założenia określonego wzorem (6), czyli nie jest zbiorem, w którym nie można byłoby wybrać podzbioru o takich samych możliwościach klasyfikacyjnych (nie jest minimalny). Dlatego też w dalszej części procedury (sekcja Backward elimination) wykonywane jest usuwanie nadmiarowych atrybutów. Ostatecznie, jako zbiór spełniający założenia (5) i (6), wyznaczany jest zbiór czteroelementowy, składający się z atrybutów: *clump_th, marg_ad, bare_nu, blan_ch* (wynik ten można potwierdzić w programie Rosetta).

```
SELECT count(*) FROM bc_data a WHERE NOT EXISTS
(SELECT 1 FROM bc_data WHERE a.clump_th=clump_th AND a.marg_ad=marg_ad AND
a.bare_nu= bare_nu AND a.blan_ch= blan_ch AND a.class<>class);

wynik: 683 wiersze spełniające kryterium
```

Za pomocą zbiorów przybliżonych możliwe jest określenie, czy w tablicy decyzyjnej nie występują niepotrzebnie, nadmiarowe informacje, które nie są niezbędne do klasyfikacji – można ustalić minimalny zbiór atrybutów niezbędnych do klasyfikacji. Informacje takie mogą być bardzo przydatne, szczególnie w systemach, w których gromadzenie danych jest kosztowne i czasochłonne. W zastosowaniach medycznych, dodatkową korzyścią z określenia atrybutów niezbędnych do poprawnej klasyfikacji może być również ograniczenie wykonywanych badań, co jest szczególnie istotne, jeżeli badania te cechują się wysoką inwazyjnością. Możliwe jest wówczas gromadzenie tylko informacji niezbędnych, bez poświęcania czasu i środków na wykonywanie dodatkowych badań, z których uzyskane dane nie okażą się przydatne.

7. Podsumowanie

Jednym z problemów występujących w praktycznych zastosowaniach teorii zbiorów przybliżonych jest złożoność obliczeń wykonywanych przy wyznaczaniu atrybutów kluczowych oraz reduktów. Celowe jest zintegrowanie zbiorów przybliżonych z bazami danych. Rozwiązanie takie daje możliwość wykorzystania wydajnych mechanizmów istniejących w systemach zarządzania bazami danych. Może to zwiększyć wydajność stosowania teorii zbiorów przybliżonych dla dużych zbiorów danych. Wykorzystanie języka SQL, powszechnie znanego i wbudowanego w każdy system bazodanowy, umożliwia stosunkowo łatwą implementację. Operacje takie, jak zliczanie, projekcja, selekcja mogą być sprawnie realizowane nawet dla wielkich ilości danych. Przedstawiona w pracy implementacja algorytmów, wykorzystująca wbudowany w środowisko bazodanowe firmy Oracle, efektywnie działający język PL/SQL, jest uniwersalna i może być stosowana dla dowolnych tabel decyzyjnych. Wydajność języków SQL oraz PL/SQL jest nieustannie zwiększana (również dzięki coraz lepszej ich wzajemnej integracji) przez producenta tego środowiska, dzięki czemu można oczekiwać jeszcze lepszych wyników w przyszłości. W pracy zaprezentowano zastosowanie zaimplementowanych algorytmów na przykładzie danych dotyczących choroby nowotworowej. Wyznaczony tą metodą rdzeń atrybutów warunkowych oraz redukt okazały się poprawne, zgodne z oczekiwaniami, co potwierdziło prawidłowość tego podejścia. Uzasadnione są dalsze prace w tym zakresie, ukierunkowane w szczególności na weryfikację wydajności metody oraz implementację innych aspektów teorii zbiorów przybliżonych w ramach środowiska bazodanowego.

BIBLIOGRAFIA

1. Pawlak Z.: Some Issues on Rough Sets. In: Transactions on Rough Sets I, LNCS, Springer 2004, s. 1÷58.
2. Komorowski, J., Pawlak, Z., Polkowski, L., Skowron, A.: A Rough Set Perspective on Data and Knowledge. In: Rough Fuzzy Hybridization (S. K. Pal, A. Skowron, Eds.), Springer-Verlag, 1999, s. 107÷121.
3. Ferdinandez-Baizan A., Ruiz E., Sanchez J.: Integrating RDMS and Data Mining capabilities using rough sets. In Proc. IPMU'96, Granada, Spain, 1996, s. 1439÷1445.
4. Hu X., Lin T., Han J.: A New Rough Sets Model Based on Database Systems. Fundamenta Informatica. 59, IOS Press 2004, s. 135÷152.

5. Hu X.: Using rough sets theory and database operations to construct a good ensemble of classifiers for data mining applications. Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference, s. 233÷240.
6. Cercone N., Ziarko W., Hu X.: Rule discovery from databases: A Decision matrix approach. In Proc. ISMIS'96, Zakopane, 1996, s. 653÷662.
7. Hu X., Shun N., Cercone N., Ziarko W.: DBROUGH: A Rough Set Based Knowledge Discovery System. Lecture Notes in Computer Science, Springer-Verlag, 1994, s. 386÷395.
8. Nguyen, S. H., Nguyen, H. S.: Some efficient algorithms for rough set methods. Proceedings of the Sixth International Conference, Information Procesing and Management of Uncertainty in Knowledge-Based Systems (IPMU-96), 2, Granada, Spain, 1996, s. 1541÷1457.
9. Skowron, A., Rauszer C.: The Discernibility Matrices and Functions in Information Systems. Intelligent Decision Support – Handbook of Applications and Advances of the Rough Sets Theory, K. Slowinski (ed), Kluwer, Dordrecht, 1992, s. 331÷362.
10. Mrózek A., Płonka L.: Analiza danych metodą zbiorów przybliżonych – Zastosowania w ekonomii, medycynie i sterowaniu. Akad. Oficyna Wyd. PLJ, Warszawa 1999.
11. Zaluski J., Szoszkiewicz R., Krysinski J., Stefanowski J.: Rough Set Theory and Decision Rules in Data Analysis of Breast Cancer Patients. In: Transactions on Rough Sets I, LNCS, Springer 2004, s. 375÷391.
12. Thangavel K., Pethalakshmi A.: Feature selection for medical database using rough system. International Journal on Artificial Intelligence and Machine Learning. v6 i1 2005, s. 11÷17.
13. Thangavel K., Karnan M., Pethalakshmi A.: Performance Analysis of Rough Reduct Algorithms in Mammograf. ICGST International Journal on Graphics, Vision and Image Processing, v5 i8, 2005, s. 13÷21.
14. Wójcik B., Ziarko W.: Rough sets approach to analysis of databases of women with breast cancer treated in the U.S. military facilities. Proceedings of IEEE-SMC International Conference on Computational Engineering in Systems Applications 1996, s. 748÷752.
15. Li J., Cercone N.: Empirical Analysis on the Geriatric Care Data Set Using Rough Sets Theory. Technical Report, CS-2005-05, University of Waterloo, 2005.
16. Drabowski M., Czajkowski K.: Sieci neuronowe i Prolog w inteligentnej bazie danych. Próby analizy wybranych objawów choroby nowotworowej, w: R. Tadeusiewicz, A.Ligęza, M. Szymkat (red.) "Computer Methods and Systems", tom II, Oprogramowanie Naukowo-Techniczne, Kraków 2005, s. 421÷426.
17. Wolberg W.: Benign breast disease and breast cancer tutorial, http://www.wisc.edu-/wolberg/breast.html.

18. Szopiński K., Szopinska M.: Współczesne poglądy na diagnostykę obrazową i biopsję zmian ogniskowych sutka. PZWL, 2003.
19. Olson D., Delen D.: Advanced Data Mining Techniques. Springer, Berlin 2008.
20. A Rough Set Toolkit for Analysis of Data – http://www.lcb.uu.se/tools/rosetta/.
21. Garcia-Molina H., Ullman J., Widom J.: Systemy baz danych. Pełny wykład, WNT, 2006.
22. Urman S., Hardman R., McLaughlin M.: Oracle Database 10g. Programowanie w języku PL/SQL, Helion 2007.

Recenzent: Dr inż. Alina Momot

Wpłynęło do Redakcji 19 stycznia 2009 r.

Abstract

One of the problems that appears in practical application of rough sets theory is complexity of computations performed during computing of core attributes and reducts. It is purposeful to integrate rough sets with databases. Such solution gives possibilities to take advantages of efficient mechanisms existing in database management systems. It could improve the efficiency of application of rough sets theory for large data sets. The use of SQL language, generally known and embedded in each database system, gives the possibility of relatively easy implementation. Operations such as counting, projection, selection, could be efficiently realized even for large amount of data.

This paper includes selected propositions of concepts and methods integration that are characteristic for rough sets with techniques applied in relational databases. This integration has the aim to increase efficiency in realization of complex computational operations. In this work there have been presented implementations of the algorithm of core attributes selection and method for finding reducts in database environment.

Algorithms implementation presented in this paper, which makes use of embedded in Oracle database environment, working efficiently PL/SQL language, is universal and could be used for any decision tables.

In this paper application of implemented algorithms on the example of data concerning cancer has been presented. Core and reduct determined by this method have turned out to be correct, consistent with expectations, and that confirmed the correctness of this approach. Further research in this area is justified, particularly focused on verification of efficiency of

the method as well as implementation of other aspects of rough sets theory in the frame of database environment.

Adresy

Krzysztof CZAJKOWSKI: Politechnika Krakowska, Instytut Teleinformatyki,
ul. Warszawska 24, 31-155 Kraków, Polska, kc@pk.edu.pl.
Mieczysław DRABOWSKI: Politechnika Krakowska, Katedra Informatyki Technicznej,
ul. Warszawska 24, 31-155 Kraków, Polska, drabowski@pk.edu.pl.