WYBRANE ZAGADNIENIA INFORMATYKI TECHNICZNEJ

pod redakcją naukową Piotra Hońki

O niektórych rozwiązaniach w dziedzinie eksploracji danych inspirowanych teorią zbiorów przybliżonych

Politechnika Białostocka

Białystok 2020

WYBRANE ZAGADNIENIA INFORMATYKI TECHNICZNEJ

O niektórych rozwiązaniach w dziedzinie eksploracji danych inspirowanych teorią zbiorów przybliżonych

Pod redakcją naukową Piotra Hońki

Politechnika Białostocka

OFICYNA WYDAWNICZA POLITECHNIKI BIAŁOSTOCKIEJ
BIAŁYSTOK 2020

Recenzenci:
dr hab. Jan G. Bazan, prof. UR
dr hab. inż. Szymon Łukasik
dr Małgorzata Lucińska
dr inż. Teresa Mroczek
dr inż. Tomasz Głowacki

Redaktor naukowy dyscypliny informatyka techniczna i telekomunikacja:
prof. dr hab. Jarosław Stepaniuk

Redaktor wydawnictwa:
mgr Janina Demianowicz

Skład, opracowanie graficzne:
Piotr Hońko

Okładka:
Marcin Dominów
Zdjęcie na okładce: Tony-Media
https://pixabay.com/pl/photos/streszczenie-sztuka-p%C4%99cherzyk%C3%B3w-5038753/

©Copyright by Politechnika Białostocka, Białystok 2020

ISBN 978-83-66391-58-1 (eBook)
DOI: 10.24427/978-83-66391-58-1

Publikacja jest udostępniona na licencji
Creative Commons Uznanie autorstwa-Użycie niekomercyjne-Bez utworów zależnych 4.0
(CC BY-NC-ND 4.0).
Pełną treść licencji udostępniono na stronie
creativecommons.org/licenses/by-nc-nd/4.0/legalcode.pl.
Publikacja jest dostępna w Internecie na stronie Oficyny Wydawniczej PB.

Oficyna Wydawnicza Politechniki Białostockiej
ul. Wiejska 45C, 15-351 Białystok
e-mail: oficyna.wydawnicza@pb.edu.pl
www.pb.edu.pl

Spis treści

Wstęp

Teoria zbiorów przybliżonych, zainicjowana w lata 80. XX wieku przez polskiego uczonego profesora Zdzisława Pawlaka, wciąż cieszy się dużym zainteresowaniem wśród naukowców, pracujących nad przetwarzaniem danych niepewnych. Jest ona stosowana i rozwijana w obszarach takich, jak m.in. sztuczna inteligencja, eksploracja danych, uczenie maszynowe, rozpoznawanie wzorców, systemy wspomagania decyzji, systemy wieloagentowe, systemy zarządzania ryzykiem, analiza konfliktów czy nauki kognitywne. Skuteczność rozwiązywania realnych problemów przy wykorzystaniu podejścia zbiorów przybliżonych jest udokumentowana publikacjami w takich dziedzinach, jak m.in. medycyna, farmakologia, bankowość, analiza rynkowa, media społeczne.

Przyczyn sukcesu zbiorów przybliżonych można upatrywać w tym, że odzwierciedlają one w pewnym stopniu sposób analizy zjawisk bliski ludzkiemu spostrzeganiu rzeczywistości. Ostra logika dwuwartościowa jest tu zastąpiona podejściem, które dopuszcza występowanie sytuacji niepewnych, niedających się jednoznacznie sklasyfikować. Odstępuje się od binarnej oceny, przykładowo dany klient banku *jest* albo *nie jest* wiarygodnym kredytobiorcą, na rzecz oceny przybliżonej, przykładowo dany klient banku *jest na pewno*, *jest prawdopodobnie*, *nie jest na pewno* albo *nie jest prawdopodobnie* wiarygodnym kredytobiorcą.

Dzięki teorii zbiorów przybliżonych dziedzina eksploracji danych, której częstym przedmiotem badań są bazy danych, opisujące przypadki niepewne, została wyposażona w dodatkowe narzędzia do radzenia sobie z danymi nieprecyzyjnymi, niekompletnymi czy zaszumionymi. Dzięki nim można skutecznie przeprowadzić na danych niepewnych operacje takie, jak selekcja cech, dyskretyzacja zbiorów wartości atrybutów, uzupełnianie danych niekompletnych, równoważenie klas niezbalansowanych, generowanie przybliżonych opisów czy samo klasyfikowanie danych.

W niniejszej monografii zamieszczono prace, w których zaprezentowano skuteczność wybranych narzędzi zbiorów przybliżonych, zastosowanych na etapie transformacji oraz samej eksploracji danych.

Rozdział Macieja Kopczyńskiego SPRZĘTOWO WSPOMAGANE WYZNACZANIE RDZENIA W STRUKTURACH FPGA jest poświęcona badaniom nad sprzętowym przyśpieszeniem wyznaczania zbioru atrybutów niezbywalnych – określanego mianem rdzenia – w procesie redukcji zbiorów danych. Badany sposób obliczania rdzenia bazuje na podejściu z zakresu teorii zbiorów przybliżonych, gdzie niezbywalne atrybuty są odnajdywane w macierzy, która dla każdej pary obiektów z bazy danych pokazuje zbiór odróżniających je atrybutów. Autor przedstawia wybrane metody obliczania rdzenia, bazujące na układach programowalnych FPGA,

które są zweryfikowane na dwóch referencyjnych zbiorach danych. Wyniki eksperymentów pokazują, że zastosowane rozwiązania sprzętowe znacznie przyśpieszają przetwarzanie danych na potrzeby wyznaczania rdzenia.

W rozdziale EFEKTYWNE METODY WYZNACZANIA REDUKTÓW OPARTE NA UKŁADACH FPGA Mateusz Chormański prezentuje metody, które w relatywnie szybkim czasie są w stanie dokonać istotnej selekcji cech zbioru danych. Wyznaczają one minimalne redukty, definiowane w teorii zbiorów przybliżonych jako najmniejsze zbiory atrybutów spośród wszystkich, które składają się jedynie z atrybutów wystarczających do prawidłowego opisu danych. W celu uzyskania znacznego przyśpieszenia obliczeń, metody wyznaczania minimalnych reduktów oparte na strategii przeszukiwania wszerz przestrzeni wszystkich potencjalnych reduktów zostały zaadaptowane sprzętowo przy wykorzystaniu układów z rodziny FPGA. Autor pokazuje zarówno uzyskane przyśpieszenia względem implementacji programowej na komputerze klasy PC, jak również różnice w czasach wykonywania obliczeń w zależności od zastosowania jednego z dwóch popularnych układów FPGA.

W rozdziale WPŁYW DYSKRETYZACJI DANYCH NA SKUTECZNOŚĆ DZIAŁANIA ALGORYTMÓW SELEKCJI NEGATYWNEJ, autorstwa Izabeli Kartowicz-Stolarskiej, dokonano przeglądu i porównania wybranych klasyfikatorów, bazujących na jednym z podejść sztucznych systemów immunologicznych, jakim jest selekcja negatywna. Zweryfikowano także skuteczność tych algorytmów w przypadku, gdy oryginalne dane opisane za pomocą atrybutów liczbowych są poddane dyskretyzacji dedykowanej zadaniu klasyfikacji. Między innymi został zbadany wpływ na wynik klasyfikacji metody dyskretyzacji zdefiniowanej w teorii zbiorów przybliżonych. Wyniki badań pokazują, że zastosowanie negatywnej selekcji w połączeniu z dyskretyzacją danych może skutecznie zwiększyć jakość klasyfikacji.

Dariusz Jankowski w rozdziale WPŁYW TECHNIK WSTĘPNEGO PRZYGOTOWANIA DANYCH NA SKUTECZNOŚĆ KLASYFIKACJI OBIEKTÓW BAZY DERMATOLOGY ZA POMOCĄ ALGORYTMU LEM2 bada problem klasyfikacji danych medycznych przy użyciu algorytmu indukcji reguł decyzyjnych, zdefiniowanego w teorii zbiorów przybliżonych. Autor weryfikuje skuteczność klasyfikacji za pomocą algorytmu LEM2, gdy oryginalne dane są podane wstępnej transformacji przy wykorzystaniu narzędzi selekcji cech i dyskretyzacji zbioru wartości atrybutów. Przeprowadzone eksperymenty umożliwiły określenie scenariuszy wstępnego przygotowania danych, przy których algorytm LEM2 jest w stanie zwiększyć trafność predykcji w porównaniu z tą otrzymaną na danych oryginalnych.

Białystok, grudzień 2020 Piotr Hońko

Rozdział 1

SPRZĘTOWO WSPOMAGANE WYZNACZANIE RDZENIA W STRUKTURACH FPGA

Maciej Kopczyński*

Streszczenie W rozdziale zaprezentowano kilka wybranych sprzętowych metod obliczania rdzenia bazujących na układach programowalnych FPGA wspieranych jednostkami obliczeniowymi CPU typu softcore. Stworzone zostały moduły sprzętowe, które należą do układów kombinacyjnych i sekwencyjnych. Zaproponowane architektury zostały przetestowane w rzeczywistym układzie FPGA na dwóch różnych zbiorach danych o różnych licznościach. Zostały również wykonane badania porównawcze w tożsamej implementacji programowej uruchamianej na komputerze PC. Otrzymane wyniki w sposób jednoznaczny wskazują na znaczące zmniejszenie czasu realizacji obliczeń dla modułów sprzętowych w porównaniu z funkcjonalnie taką samą implementacją programową. W rozdziale tym przytoczone są niezbędne definicje z zakresu zbiorów przybliżonych, pokazane są główne założenia architektury poszczególnych rozwiązań sprzętowych wraz z przykładem ich działania, jak również jest opisana charakterystyka zbiorów danych przed, jak i po przekształceniach niezbędnych do realizacji obliczeń przez moduły wykonawcze w strukturach FPGA.

Słowa kluczowe: zbiory przybliżone, rdzeń, FPGA, implementacja sprzętowa

Wprowadzenie

Jednym z największych problemów związanych z tworzeniem systemów wspomagania decyzji jest czas przetwarzania danych wejściowych, ze szczególnym uwzględnieniem dużych zbiorów danych (ang. *Big data*). *Big data* to termin odnoszący się do dużych, różnorodnych i zmiennych zbiorów danych, których przetwarzanie i analiza są trudne, ale jednocześnie cenne, ponieważ mogą prowadzić do zdobywania nowej wiedzy. W praktycznych zastosowaniach pojęcie dużego zbioru danych jest względne i oznacza sytuację, w której zbiór nie może być przetworzony przy użyciu prostych i powszechnie dostępnych metod (Marz i Warren, 2015). W zależności

* Wydział Informatyki, Politechnika Białostocka, Wiejska 45A, 15-351 Białystok, m.kopczynski@pb.edu.pl

DOI 10.24427/978-83-66391-58-1_1

od celu wybranej metody i złożoności algorytmu może to oznaczać rozmiar danych liczony w megabajtach, gigabajtach lub terabajtach.

Jednym z przykładów tak ogromnych zbiorów danych są dane odczytywane z czujników, np. z linii produkcyjnej. W każdej części takiej linii znajduje się bardzo wiele takich czujników, które cały czas dostarczają dane, co tworzy ogromny strumień danych, które muszą zostać przetworzone, np. w celu predykcji przyszłych awarii, a tym samym unikania przestojów linii z możliwie dużym wyprzedzeniem.

Powstało wiele metod przetwarzania takich zbiorów danych i wydobywania z nich wiedzy, jak również minimalizowania ilości danych, które muszą być przetwarzane. Jedną z nich jest teoria zbiorów przybliżonych (ang. *Rough sets*), pozwalająca między innymi na stosunkowo łatwe ograniczenie liczby atrybutów (kolumn) w wejściowych zbiorach danych. Główne założenia dotyczące metod zbiorów przybliżonych opisane są przez Pawlak i Skowron (2007). Pod kątem ograniczania liczby atrybutów, w metodach zbiorów przybliżonych zdefiniowane są dwa ważne pojęcia: redukt (ang. *Reduct*) oraz rdzeń (ang. *Core*). Redukty to zbiory atrybutów, które zawierają kluczowe informacje niezbędne do prawidłowej klasyfikacji danych. Rdzenie to zbiory atrybutów, których nie można w żadnych okolicznościach usunąć z początkowego zbioru danych. Innymi słowy, atrybuty istniejące w rdzeniu muszą również istnieć we wszystkich reduktach.

Teoria zbiorów przybliżonych, opracowana w latach osiemdziesiątych XX wieku przez prof. Z. Pawlaka, jest użytecznym narzędziem do analizy danych. Dlatego w naukowych i komercyjnych narzędziach przetwarzania danych zaimplementowano wiele algorytmów zbiorów przybliżonych. Jednak wraz ze wzrostem rozmiaru danych pojawia się problem z wydajnością i rosnącym czasem ich przetwarzania.

Jednym z rozwiązań jest opracowanie algorytmów o mniejszej złożoności obliczeniowej. Kolejnym jest przetwarzanie równoległe dużych zbiorów danych. Algorytmy do obliczania rdzenia i reduktów oparte na modelu programowania rozproszonego MapReduce można znaleźć w pracy Czołombitko i Stepaniuk (2015).

Innym podejściem do przetwarzania dużych zbiorów danych są sprzętowe implementacje istniejących algorytmów. Tego typu podejście można zrealizować za pomocą układów FPGA (ang. *Field Programmable Gate Arrays*). Jest to grupa cyfrowych układów scalonych, których funkcjonalność może być zmieniona w każdym momencie i jest oparta głównie na definiowanych przez inżyniera funkcjach boolowskich. Dlatego też mogą być one używane do wspomagania obliczeń w zakresie zbiorów przybliżonych, ze szczególnym uwzględnieniem równoległego przetwarzania danych. Należy jednak zaznaczyć, że wykorzystanie układów FPGA w zakresie implementacji algorytmów programowych stwarza trudność w aspekcie dostosowania tych algorytmów do charakteru przetwarzanych danych oraz ich analizy w celu identyfikacji tych elementów, które mogą być realizowane równolegle w strukturach programowalnych. Innymi słowy, implementacja sprzętowa algorytmu programowego nie zapewnia elastyczności tożsamej z funkcjonalnym odpowiednikiem rozwiązania zrealizowanego w języku zarówno wysokiego, jak i niskiego poziomu.

Obecnie istnieje kilka implementacji sprzętowych wybranych metod zbiorów przybliżonych. Idea przykładowego procesora generującego reguły decyzyjne z tablic decyzyjnych została opisana w pracy Pawlak (2004). Lewis, Perkowski i Jozwiak (1999) przedstawili architekturę teoretycznego procesora bazującego na sieciach komórkowych opisanych przez Muraszkiewicz i Rybinski (1993). Kanasugi i Yokoyama (2001) opracowali koncepcję urządzenia sprzętowego zdolnego do minimalizacji dużych funkcji logicznych utworzonych na podstawie macierzy rozróżnialności. Bardziej szczegółowe podsumowanie istniejących sprzętowych implementacji metod zbioru przybliżonego można znaleźć w opracowaniu Kopczynski i Stepaniuk (2013) i w pracy Tiwari i Kothari (2014). Wyniki poprzednich badań autora dotyczą idei procesora dla metod zbiorów przybliżonych (Stepaniuk, Kopczynski i Grzes, 2013), obliczania reduktu wspomaganego sprzętowo (Kopczynski, Grzes i Stepaniuk, 2014a), obliczania rdzenia przy użyciu rozwiązania opartego na FPGA (Kopczynski, Grzes i Stepaniuk, 2014b), jednostki sprzętowej do przetwarzania dużych zbiorów danych (Kopczynski, Grzes i Stepaniuk, 2015) czy też dwuetapowego algorytmu znajdowania reduktów (Grzes i Kopczynski, 2019).

W podrozdziale 1.1 przedstawiono najważniejsze definicje związane z operacjami obliczania rdzenia, jak również omówiono wykorzystywane algorytmy. Ponadto, w podrozdziale zawarto przykład działania algorytmu oraz opisano dane wykorzystywane w badaniach eksperymentalnych. W podrozdziale 1.2 skupiono się na opisie architektury sprzętowej opracowanych modułów wspomagających obliczanie rdzenia, jak również zawarto przykład działania takiego modułu zaprezentowany na poziomie danych binarnych. W podrozdziale 1.3 opisano rzeczywiste środowisko testowe oraz przedstawiono wyniki badań eksperymentalnych. W *Podsumowaniu* zaprezentowano najważniejsze wnioski oraz określono przyszłe możliwości optymalizacji modułów sprzętowych w kontekście badań naukowych.

1.1 Podstawowe definicje

1.1.1 Rdzeń w ujęciu zbiorów przybliżonych

Niech $DT = (U, A \cup \{d\})$ będzie tablicą decyzyjną, gdzie U jest zbiorem obiektów, A jest zbiorem atrybutów warunkowych, a d jest atrybutem decyzyjnym. W tablicy decyzyjnej niektóre atrybuty warunkowe ze zbioru A mogą być usunięte (innymi słowy: są zbędne), jednak ich usunięcie nie może pogorszyć jakości klasyfikacji. Zbiór $C \subseteq A$ wszystkich niezbędnych atrybutów warunkowych nazywany jest rdzeniem. Żaden z jego elementów nie może zostać usunięty bez wpływu na moc klasyfikacji wszystkich atrybutów warunkowych. Aby obliczyć rdzeń, można użyć macierzy rozróżnialności $[DM(x,y)]_{x,y \in U}$, gdzie $DM(x,y) = \{a \in A : a(x) \neq a(y) \text{ and } d(x) \neq d(y)\}$.

Rdzeń to zbiór wszystkich jednoelementowych komórek macierzy rozróżnialności, tj. $CORE = \bigcup_{x,y \in U, cardinality(DM(x,y))=1} DM(x,y)$. Znacznie bardziej szczegółowy opis definicji rdzenia można znaleźć w artykule (Pawlak i Skowron, 2007) lub w książce (Stepaniuk, 2008).

1.1.2 Podstawowy algorytm obliczania rdzenia

Jeden z najpopularniejszych algorytmów obliczania rdzenia wykorzystujący macierz rozróżnialności jest przedstawiony w postaci pseudokodu w tym podrozdziale. Największym jednak ograniczeniem tego algorytmu dla dużych zbiorów danych jest potrzeba przechowywania macierzy rozróżnialności *DM* jako dwuwymiarowej tablicy o rozmiarze $|U|x|U|$. Poszczególne wersje sprzętowe algorytmów obliczania rdzenia wykorzystują elementy algorytmu podstawowego, jednakże algorytmy te zostały w znacznym stopniu zmodyfikowane i dostosowane do rozwiązania sprzętowego.

Algorytm 1.1 Algorytm obliczania rdzenia oparty na definicji

INPUT: macierz rozróżnialności *DM*
OUTPUT: rdzeń *C*

```
1: C ← ∅
2: for x ∈ U do
3:     for y ∈ U do
4:         if |DM(x,y)| = 1 then
5:             C ← C ∪ DM(x,y)
6:         end if
7:     end for
8: end for
```

Dane wejściowe do algorytmu opartego na definicji stanowi macierz rozróżnialności *DM*, a wyjście to rdzeń *C*. Rdzeń jest inicjalizowany jako pusty zbiór w linii 1. Dwie pętle w liniach 2 i 3 iterują po wszystkich obiektach (oznaczonych jako U) w macierzy dostrzegalności. Instrukcja warunkowa w linii 4 sprawdza, czy komórka macierzy zawiera tylko jeden atrybut. Jeśli tak, to ten atrybut jest dodawany do rdzenia *C*.

1.1.3 Sprzętowy algorytm obliczania rdzenia dla dużych zbiorów danych

Główna koncepcja algorytmu CORE-HIDM (***CORE H**ardware **I**ndirect **D**iscernibility **M**atrix*) jest oparta na właściwości singletonu, czyli komórki z macierzy rozróżnialności składającej się tylko z jedynego atrybutu, co jest bezpośrednio związane z definicją przytoczoną na początku tego podrozdziału. Jak zostało wcześniej wspomniane, rozwiązania bazującego na definicji nie można uruchomić na sprzęcie dla większych zbiorów danych z powodu ograniczeń zasobów układu FPGA. Nie jest możliwe przechowywanie dużego zestawu danych w module sprzętowym. Dlatego też zaproponowany został algorytm dedykowany do implementacji sprzętowej - CORE-HIDM. Jest on również bazą do opracowania kilku jego wariantów dedykowanych dla różnych optymalizacji w zakresie przetwarzania równoległego. Jednym z takich wariantów sprzętowych zaprezentowanych w tym rozdziale jest implementacja opisana jako CORE-PHIDM (***CORE P**arallel **H**ardware **I**ndirect **D**iscernibility **M**atrix*). Szczegóły dotyczące tego rozwiązania podane zostaną w kolejnych podrozdziałach poświęconych omówieniu architektury modułów wykonawczych. Główną ideą algorytmu CORE-HIDM jest podzielenie całego zbioru danych na równe części przechowywane w dwóch niezależnych jednostkach pamięci RAM układu FPGA. Części te są następnie przetwarzane przez jednostkę obliczeniową (algorytm CORE-HIDM).

Dane wejściowe do algorytmu CORE-HIDM stanowi tablica decyzyjna DT, a wyjście rdzeń C. W pierwszym kroku rdzeń C jest inicjalizowany jako pusty zbiór. Dwie pętle w liniach 2 i 4 są odpowiedzialne za wybór fragmentu wejściowej tablicy decyzyjnej. Tabela decyzyjna DT jest podzielona na m części, z których każda ma rozmiar n obiektów. Linie 3 i 5 odpowiadają za załadowanie wybranych fragmentów zbioru danych do pamięci RAM zaimplementowanych w układzie FPGA. Dwie pętle w liniach 6 i 7 pobierają do porównania kolejne obiekty z fragmentu tablicy decyzyjnej. Linia 8 porównuje wartość atrybutu decyzyjnego dwóch obiektów x i y. Jeśli obiekty należą do różnych klas decyzyjnych, to wykonywana jest pozostała część algorytmu. Zmienna *count* odpowiedzialna za przechowywanie liczby różnic wartości atrybutów warunkowych pomiędzy obiektami x i y jest ustawiana na wartość 0 w linii 9. Pętla w linii 10 iteruje po zbiorze atrybutów warunkowych A. Wartości atrybutu warunkowego a są porównywane między obiektami x i y w linii 11. W przypadku różnicy wartości, zmienna *count* jest zwiększana, a atrybut a jest zapisywany w zmiennej *candidate*. Po zakończeniu pętli, atrybut w zmiennej *candidate* jest dodawany do rdzenia C, jeśli zmienna *count* jest równa 1 i atrybut a nie znajduje się w rdzeniu (wiersze od 16 do 18).

Algorytm 1.2 Algorytm CORE-HIDM

INPUT: tablica decyzyjna $DT = (U, A \cup \{d\})$, dwie liczby naturalne $n, m > 0$
OUTPUT: rdzeń C

```
1:  C ← ∅
2:  for cnt_1 ← 0 to m − 1 do
3:      RAM1 ← {x ∈ U : x_{cnt_1·n} to x_{(cnt_1+1)·n−1}}
4:      for cnt_2 ← cnt_1 to m − 1 do
5:          RAM2 ← {x ∈ U : x_{cnt_2·n} to x_{(cnt_2+1)·n−1}}
6:          for x ∈ RAM1 do
7:              for y ∈ RAM2 do
8:                  if d(x) ≠ d(y) then
9:                      count ← 0
10:                     for a ∈ A do
11:                         if a(x) ≠ a(y) then
12:                             count ← count + 1
13:                             candidate ← a
14:                         end if
15:                     end for
16:                     if count = 1 and candidate ∉ C then
17:                         C ← C ∪ {candidate}
18:                     end if
19:                 end if
20:             end for
21:         end for
22:     end for
23: end for
```

1.1.4 Przykład obliczania rdzenia na bazie definicji

W tabeli 1.1 przedstawiono przykładową tablicę decyzyjną składającą się z 4 atrybutów warunkowych i jednego atrybutu decyzyjnego. Zbiór danych opisuje decyzje związane z aktywnością fizyczną (ang. *activity*) na podstawie 4 opisowych cech pogody: warunki (ang. *outlook*), temperatura (ang. *temp*), wilgotność (ang. *humidity*) i wiatr (ang. *windy*).

Przed obliczeniem rdzenia należy najpierw stworzyć macierz rozróżnialności. Używając definicji przedstawionych w podrozdziale 1.1.1 została obliczona macierz rozróżnialności *DM* dla tablicy decyzyjnej pokazanej w tabeli 1.1. Macierz jest symetryczna względem swojej przekątnej, więc obliczany jest tylko dolny trójkąt. Dodatkowo, aby ograniczyć rozmiar tabeli, obliczono macierz rozróżnialności *DM* tylko dla 6 pierwszych obiektów ze zbioru danych. Macierz wynikowa pokazana jest w tabeli 1.2.

Tabela 1.1: Przykładowy zbiór danych składający się z 4 atrybutów warunkowych i 1 atrybutu decyzyjnego

ID	outlook	temp	humidity	windy	activity
1	sunnny	high	high	no	no
2	sunnny	high	high	yes	no
3	cloudy	high	high	no	yes
4	sunnny	low	high	no	yes
5	sunnny	high	normal	no	yes
6	sunnny	high	normal	yes	no
7	cloudy	high	normal	yes	yes
8	sunnny	low	high	no	no
9	sunnny	high	normal	no	yes
10	sunnny	low	normal	no	yes
11	sunnny	low	normal	yes	yes
12	cloudy	low	high	yes	yes

Zgodnie z algorytmem obliczania rdzenia opisanym w podrozdziale 1.1.2, poniżej przedstawiono przykład wyznaczania rdzenia na podstawie macierzy rozróżnialności *DM* pokazanej w tabeli 1.2. Na początku rdzeń C jest pusty. Dwie główne pętle wykonują pierwszą iterację przechodząc przez wszystkie komórki macierzy rozróżnialności. Dla każdej z nich sprawdzany jest warunek dodania atrybutu do rdzenia. Pierwszy taki przypadek występuje dla komórki na przecięciu obiektów 1 i 3. Komórka ta zawiera tylko jeden atrybut $\{o\}$. Ten atrybut nie istnieje w rdzeniu, więc jest do niego dodawany, czyli rdzeń $C = \{o\}$. Po wykonaniu pozostałych kroków algorytmu, końcowy rdzeń to $C = \{o,t,h,w\}$, co odpowiada $C = \{outlook, temp, humidity, windy\}$.

Tabela 1.2: Macierz rozróżnialności dla pierwszych 6 obiektów ze zbioru przykładowego

ID	**1**	**2**	**3**	**4**	**5**	**6**
1	$\emptyset$					
2	$\emptyset$	$\emptyset$				
3	$\{o\}$	$\{o,w\}$	$\emptyset$			
4	$\{t\}$	$\{t,w\}$	$\emptyset$	$\emptyset$		
5	$\{h\}$	$\{h,w\}$	$\emptyset$	$\emptyset$	$\emptyset$	
6	$\emptyset$	$\emptyset$	$\{o,h,w\}$	$\{t,h,w\}$	$\{w\}$	$\emptyset$

1.1.5 Dane do badań eksperymentalnych

Przeprowadzone eksperymenty wykorzystywały dwa zbiory danych: Poker Hand (autorstwa Roberta Cattrala i Franza Oppachera) oraz zbiór danych o dzieciach z cukrzycą insulinozależną typu 1 (autorstwa Jarosława Stepaniuka).

Pierwszy zestaw danych pozyskano z repozytorium UCI Machine Learning (Lichman, 2013). Każdy z 1 000 000 rekordów jest przykładem zestawu składającego się z pięciu kart do gry dobranych ze standardowej talii 52 kart. Każda karta jest opisana za pomocą dwóch atrybutów (kolor i ranga), co daje łącznie 10 atrybutów warunkowych. Atrybut decyzyjny opisuje "układ pokerowy".

Cukrzyca insulinozależna jest przewlekłą chorobą charakteryzującą się niezdolnością do wytwarzania wystarczającej ilości insuliny do wydajnego przetwarzania węglowodanów, tłuszczu i białek. Leczenie wymaga wstrzyknięć insuliny. Dwanaście atrybutów warunkowych obejmujących wyniki badań fizycznych i laboratoryjnych oraz jeden atrybut decyzyjny (mikroalbuminuria) opisuje bazę danych wykorzystywaną w eksperymentach. Zbiór danych składa się ze 107 przypadków. Baza danych jest zawarta jest w artykule Stepaniuk (2000).

Baza Poker Hand została wykorzystana do stworzenia mniejszych zbiorów danych przez wybranie określonej liczby wierszy oryginalnego zbioru danych z zachowaniem rozkładu klas decyzyjnych. Baza danych dotycząca cukrzycy została wykorzystana do wygenerowania większych zbiorów przez powielenie wierszy z oryginalnego zbioru danych.

Utworzone zbiory danych musiały zostać przekształcone do wersji binarnej. Wartości liczbowe zostały zdyskretyzowane, a wartość każdego atrybutu zakodowana przy użyciu czterech bitów dla obu zbiorów danych. Każdy obiekt został opisany na 44 bitach w przypadku zbioru Poker Hand i 52 bitach w przypadku bazy cukrzyków. Opisy obiektów musiały zostać rozszerzone do 64-bitowych słów, wypełniając nieużywane atrybuty binarnym "0" celem uproszczenia architektury przetwarzania danych po stronie układu FPGA.

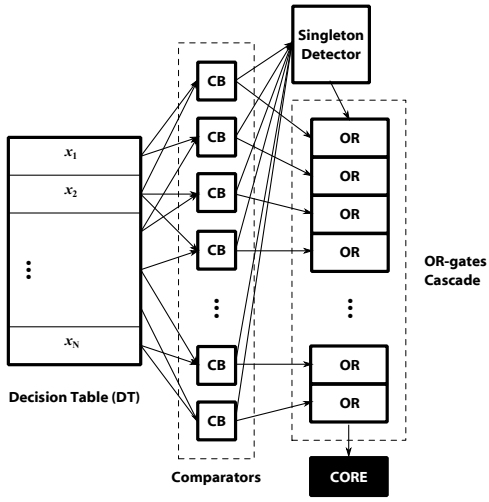
1.2 Implementacje sprzętowe

1.2.1 Obliczanie rdzenia jako układ kombinacyjny - "OR-cascade"

Pierwszym przedstawionym układem implementacji sprzętowej algorytmu obliczania rdzenia jest układ kombinacyjny bazujący bezpośrednio na definicji i metodzie obliczania rdzenia przedstawionej w podrozdziale 1.1.2.

Architekturę pierwszego rozwiązania pokazano na rysunku 1.1. Wejściem do tego bloku jest pełna tablica decyzyjna. Moduł obliczeniowy składa się z trzech bloków funkcjonalnych:

1. **Comparators CB** - blok komparatorów, które obliczają wpisy macierzy rozróżnialności. Każdy komparator ma dwa wejścia, które są podłączone do dwóch obiektów tablicy decyzyjnej.
2. **OR-gates Cascade** - blok bramek OR połączonych kaskadowo. Każda bramka wyznacza logiczną operację OR na dwóch wartościach: jednej z poprzedniej bramki w kaskadzie i drugiej z komparatora. Wynik przesyłany jest do następnej bramki OR i ostatecznie do rejestru **CORE**, który przechowuje wynik obliczeń. Wybrana bramka OR może zostać zablokowana przez wyjście z **Singleton Detector**.
3. **Singleton Detector** - blok sprawdzający, czy wpis w macierzy rozróżnialności jest singletonem, tj. składa się tylko z jednej logicznej "1". Wyjścia tego bloku są połączone z kaskadą bramek OR.

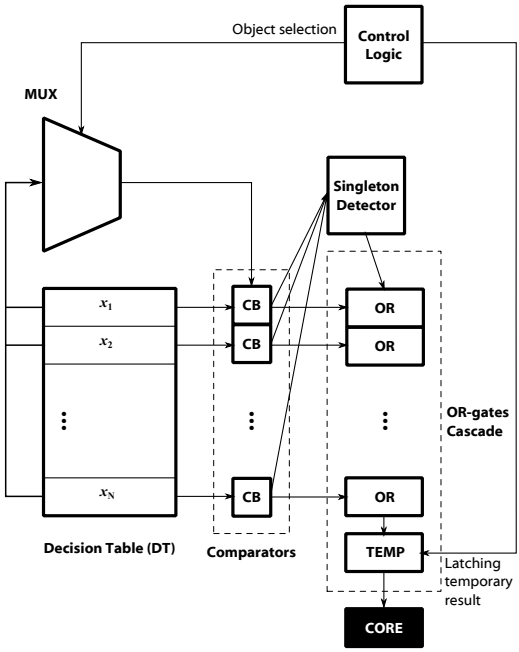


Rysunek 1.1: Diagram blokowy modułu obliczania rdzenia "OR-cascade" w wariancie kombinacyjnym

Komórki macierzy rozróżnialności są obliczane przez komparatory bardzo szybko, głównie ze względu na prostotę architektury każdego komparatora z bloku CB, jak również kombinacyjny charakter układu wykonawczego. Następnie wszystkie wejścia trafiają do kaskady bramek OR. Czas niezbędny do obliczenia wyniku zależy od wielkości tablicy decyzyjnej. Ostatnia bramka w kaskadzie przechowuje wynik obliczeń w rejestrze **CORE**.

1.2.2 Obliczanie rdzenia jako układ sekwencyjny - "Mixed"

Głównym ograniczeniem rozwiązania opartego na układzie kombinacyjnym jest mały rozmiar tablicy decyzyjnej, która może być przetwarzana. Im większy rozmiar danych, tym więcej zasobów układu FPGA jest wykorzystywane. Efektywnie, tego rodzaju rozwiązanie może przetwarzać zbiory o licznościach setek obiektów. Ze względu na ograniczenie zasobów struktury FPGA, zaproponowany został wariant modułu sprzętowego obliczania rdzenia oparty o układ sekwencyjny, który jest znacznie bardziej optymalny pod względem zużycia zasobów. Architekturę tego rozwiązania pokazano na rysunku 1.2.



Rysunek 1.2: Diagram blokowy modułu obliczania rdzenia "Mixed" w wariancie sekwencyjnym

Główne różnice w stosunku do poprzedniego rozwiązania to:

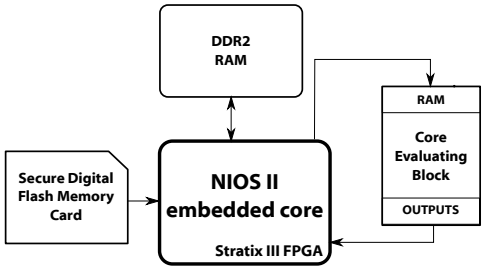
1. Liczba komparatorów **CB** jest równa liczbie obiektów w tablicy decyzyjnej, a nie liczbie elementów w macierzy rozróżnialności.
2. Liczba bramek OR w kaskadzie jest równa liczbie obiektów w tablicy decyzyjnej, a nie liczbie elementów w macierzy rozróżnialności.
3. Multiplekser **MUX** w każdym cyklu zegara taktującego wybiera kolejny obiekt z tablicy decyzyjnej i umieszcza go w komparatorach.

4. Blok **Control Blok** zlicza obiekty w tablicy decyzyjnej i zatrzaskuje wartości w rejestrze tymczasowym **TEMP**.

Ten wariant sprzętowy jest znacznie mniejszy pod względem wykorzystania zasobów niż poprzedni, ale wymaga więcej czasu na obliczenie końcowego wyniku. Liczba cykli potrzebnych do zakończenia obliczeń jest równa liczbie obiektów w tabeli decyzyjnej.

1.2.3 Obliczanie rdzenia dla dużych zbiorów danych - "CORE-HIDM"

Pomimo zmniejszenia liczby wykorzystywanych zasobów układu FPGA, układ "Mixed" nie może zostać w prosty sposób zwielokrotniony, co pozwoliłoby na przetwarzanie większej liczby obiektów. Głównym problemem są ograniczenia w wielkości bloków pamięciowych w strukturze FPGA. Z tego względu opracowana została implementacja sprzętowa wspomagana procesorem typu softcore oraz zewnętrzną pamięcią RAM. Ogólną architekturę tego rozwiązania pokazano na rysunku 1.3.



Rysunek 1.3: Ogólny diagram blokowy modułu obliczania rdzenia dla dużych zbiorów danych

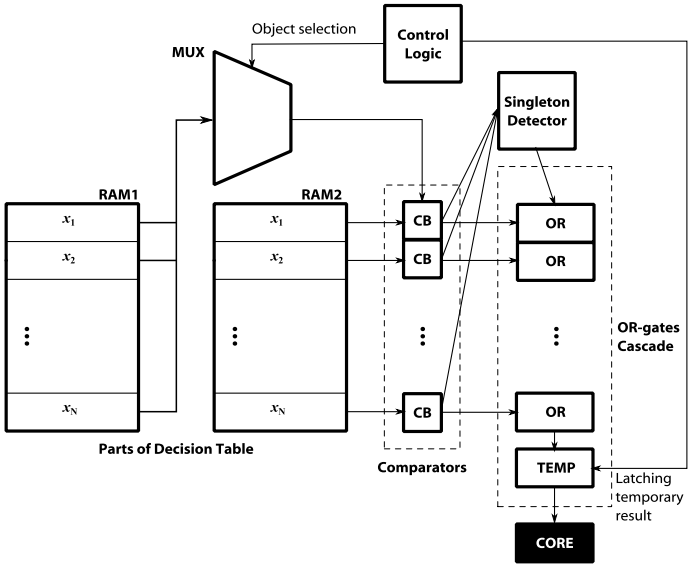
Procesor typu softcore NIOS II odpowiada za następujące operacje:

1. Kontroluje proces dzielenia na częśco dużej wejściowej tablicy decyzyjnej.
2. Nadzoruje podstawowe bloki funkcjonalne modułów sprzętowego obliczania rdzenia.
3. Przeładowuje dane pomiędzy wewnętrzną i zewnętrzną pamięcią RAM.
4. Przetwarza wyniki zwrócone przez sprzętowe moduły obliczania rdzenia.
5. Wykonuje podstawowe operacje na zbiorach zapisanych w rejestrach.

Wybrany procesor to NIOS II. Jest to jednostka typu softcore dostarczona przez firmę Altera dla jej układów typu FPGA. Jest to w pełni funkcjonalny, 32-bitowy procesor typu RISC ze wsparciem dla rozwiązań zwiększających moc obliczeniową

(np. potoki wielostopniowe, dynamiczne przewidywanie rozgałęzień, oddzielne pamięci podręczne dla instrukcji i dla danych, jednostki MMU, MPU i inne). Pamięć DDR2 przechowuje dużą tablicę decyzyjną. Karta SD to tymczasowe rozwiązanie do przenoszenia danych z komputera PC do rozwiązania opartego na FPGA. Dane z karty SD są w całości kopiowane do pamięci DDR2 na początku procesu obliczeniowego. Każda kolejna część danych gotowych do przetworzenia jest przechowywana we wbudowanych pamięciach układu FPGA (MLAB, M9k i M144k). Bloki MLAB to synchroniczne, dwuportowe pamięci z konfigurowalną organizacją 32x20 lub 64x10. Dwuportowe pamięci mogą być odczytywane i zapisywane jednocześnie, co przyśpiesza wykonywane operacje. Bloki M9k i M144k są również synchronicznymi, dwuportowymi blokami pamięci z wieloma możliwymi konfigurowalnymi organizacjami. Cechą charakterystyczną tych bloków jest możliwość przygotowania pamięci zdolnej do przechowywania niemal każdego rodzaju obiektów (opisanych jako słowa bitowe).

Szczegółowa architektura modułu sprzętowego obliczania rdzenia "CORE-HIDM" pokazana jest na rysunku 1.4.



Rysunek 1.4: Szczegółowy diagram blokowy modułu obliczania rdzenia "CORE-HIDM"

Należy zaznaczyć, że główne założenia modułu sprzętowego "CORE-HIDM" są rozwinięciem układu "Mixed", przez co wcześniej omówione elementy składowe występujące w obydwu rozwiązaniach nie będą ponownie omawiane. Wejścia modułu to:

- **RAM1** i **RAM2** - pamięci przechowujące fragmenty tablicy decyzyjnej,

- **Attribute Mask Register (AMR)** - rejestr przechowujący wartość odpowiadającą atrybutom warunkowymdo przetworzenia,
- **clock** - sygnał zegarowy układu,
- **reset** - sygnał resetujący do ustawienia początkowych wartości rejestrów wewnętrznych w module sprzętowym.

Wyjścia bloku CORE-HIDM to wartość obliczonego rdzenia w rejestrze *CORE* i sygnał *ready* informujący o zakończeniu obliczeń wykonanych przez blok.

Celem przetwarzania dużych zbiorów danych do rozwiązania dodano dwa bloki szybkich statycznych pamięci RAM, które zostały utworzone jako instancje dedykowanych bloków FPGA (MLAB, M9k i M144k). Obie pamięci danych używane w module (RAM1 i RAM2) przechowują fragmenty wejściowej tablicy decyzyjnej. Na początku algorytmu zawierają tę samą część tabeli decyzyjnej. Gdy obiekty z pamięci RAM2 zostaną porównane ze wszystkimi obiektami z pamięci RAM1, to pamięć RAM2 jest przeładowywana następną częścią tablicy decyzyjnej, aż do momentu, gdy w tablicy decyzyjnej nie będzie żadnych nieporównywanych elementów. Następnie do pamięci RAM1 i RAM2 ładowany jest drugi fragment zbioru danych i cały proces jest kontynuowany do momentu przetworzenia całego zbioru wejściowego.

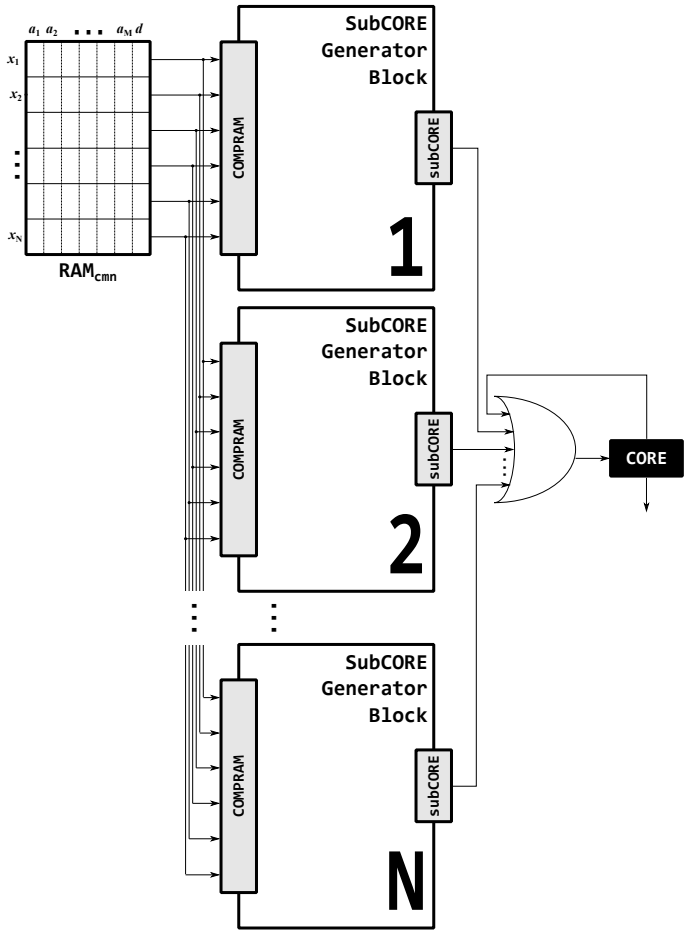
1.2.4 Obliczanie rdzenia dla dużych zbiorów danych z przetwarzaniem równoległym - "CORE-PHIDM"

Kolejnym rozwinięciem rozwiązania wspomagającego w sposób sprzętowy obliczanie rdzenia i dedykowanym dla dużych zbiorów danych jest układ "CORE-PHIDM". W tym przypadku zwielokrotniony został blok sprzętowy obliczania rdzenia, który przestawiony został na rysunku 1.4. Kluczowe jednak było wprowadzenie zmian związanych z organizacją pamięci na dane oraz synchronizacją pracy poszczególnych modułów sprzętowych. Architektura modułu obliczania rdzenia "CORE-PHIDM" pokazana jest na rysunku 1.5

System składa się z następujących bloków:

1. RAM_{cmn} - fragment tablicy decyzyjnej, która jest porównywana z częściami tablicy decyzyjnej przechowywanymi w powielonych blokach *SubCORE*.
2. *SubCORE* - blok, który dokonuje porównania dwóch części tablicy decyzyjnej i oblicza fragment rdzenia. Jego działanie jest tożsame do bloku CORE-HIDM.

Pamięć RAM_{cmn} jest wypełniana fragmentami tablicy decyzyjnej. Każdy blok *SubCORE* zawiera również kolejne fragmenty tablicy decyzyjnej. *SubCORE* oblicza częściową wartość rdzenia, która ostatecznie jest łączona w wynikowy rdzeń *C*. Powielenie bloków *SubCORE* umożliwia zrównoleglenie prowadzonych obliczeń. Całość procesu obliczeniowego jest nadzorowana przez procesor NIOS II.



Rysunek 1.5: Diagram blokowy modułu obliczania rdzenia “CORE-PHIDM”

1.2.5 Przykład działania dla modułu ”CORE-HIDM”

Przykład działania wyjaśnia sposób przetwarzania danych przez układ CORE-HIDM będący główną sprzętową jednostką obliczeniową dla generowania rdzenia. Dla przejrzystości pokazano tylko najważniejsze operacje związane z przepływem danych. Jednostka sprzętowa wymaga przekształcenia zbioru danych do wersji binarnej. Tablica decyzyjna z tabeli 1.1 po transformacji binarnej jest zaprezentowana w tabeli 1.3.

Wartości na wejściach modułu CORE-HIDM to:

- RAM_1 = 00111 01111 10110 10101 1001101011 11010 00101 10011 10001 11001 11100,

Tabela 1.3: Przykładowy zbinaryzowany zbiór danych składający się z 4 atrybutów warunkowych i 1 atrybutu decyzyjnego

ID	outlook	temp	humidity	windy	activity
1	1	1	1	0	0
2	1	1	1	1	0
3	0	1	1	0	1
4	1	0	1	0	1
5	1	1	0	0	1
6	1	1	0	1	0
7	0	1	0	1	1
8	1	0	1	0	0
9	1	1	0	0	1
10	1	0	0	0	1
11	1	0	0	1	1
12	0	0	1	1	1

- RAM_2 = 00111 01111 10110 10101 1001101011 11010 00101 10011 10001 11001 11100,

co odpowiada danym w przykładowej tablicy decyzyjnej po binaryzacji. Najbardziej znaczący bit (MSB) opisuje atrybut decyzyjny *activity*, zaś najmniej znaczący bit (LSB) odpowiada za atrybut warunkowy *outlook*.

Muliplexer MUX w pierwszym cyklu zegara wybiera z pamięci RAM_1 obiekt ID_1, który reprezentowany jest przez słowo 00111. Obiekt ten jest porównywany ze wszystkimi pozostałymi obiektami w pamięci RAM_2 przez komparatory CB. Obliczona wartość odpowiada pierwszej kolumnie macierzy rozróżnialności. Słowo binarne utworzone przez komparatory to 0000 0000 0001 0010 0100 0000 1101 0000 0100 0110 1110 1011. Każda z 4-bitowych podsekwencji jest przekazywana do bramek OR. Jednocześnie całe słowo utworzone przez komparatory jest kierowane do bloku SD, który wykrywa 4-bitowe podciągi zawierające tylko jedną logiczną "1". Wynik utworzony przez SD to 001110001000. Każdy bit utworzonej wartości steruje pojedynczą bramką OR w kaskadzie. MSB jest podłączony do pierwszej bramki OR. Wartości wejść i wyjść każdej z bramek to:

- $IN_{1CB} = 0000$; $IN_{1PREV} = 0000$; $IN_{1SD} = 0$; $OUT_1 = 0000$,
- $IN_{2CB} = 0000$; $IN_{2PREV} = 0000$; $IN_{2SD} = 0$; $OUT_2 = 0000$,
- $IN_{3CB} = 0001$; $IN_{3PREV} = 0000$; $IN_{3SD} = 1$; $OUT_3 = 0001$,
- $IN_{4CB} = 0010$; $IN_{4PREV} = 0001$; $IN_{4SD} = 1$; $OUT_4 = 0011$,
- $IN_{5CB} = 0100$; $IN_{5PREV} = 0011$; $IN_{5SD} = 1$; $OUT_5 = 0111$,
- $IN_{6CB} = 0000$; $IN_{6PREV} = 0111$; $IN_{6SD} = 0$; $OUT_6 = 0111$,
- $IN_{7CB} = 1101$; $IN_{7PREV} = 0111$; $IN_{7SD} = 0$; $OUT_7 = 0111$,
- $IN_{8CB} = 0000$; $IN_{8PREV} = 0111$; $IN_{8SD} = 0$; $OUT_8 = 0111$,
- $IN_{9CB} = 0100$; $IN_{9PREV} = 0111$; $IN_{9SD} = 1$; $OUT_9 = 0111$,

- $IN_{10CB} = 0110$; $IN_{10PREV} = 0111$; $IN_{10SD} = 0$; $OUT_{10} = 0111$,
- $IN_{11CB} = 1110$; $IN_{11PREV} = 0111$; $IN_{11SD} = 0$; $OUT_{11} = 0111$,
- $IN_{12CB} = 1011$; $IN_{12PREV} = 0111$; $IN_{12SD} = 0$; $OUT_{12} = 0111$.

Wejście IN_{CB} odpowiada wartości obliczonej przez dany komparator z bloku *CB*. Wejście IN_{PREV} jest połączone z wyjściem OUT poprzedniej bramki OR. Wyjście ostatniej bramki jest przekazywane do rejestru pośredniego *TEMP*.

W kolejnych cyklach zegarowych moduł CORE-HIDM przetwarza pozostałe obiekty w RAM_1 porównując je z pozostałą częścią zbioru danych w RAM_2. Ostatecznie utworzony rdzeń to $C = 1111$, co odpowiada rdzeniowi $C = \{outlook, temp, humidity, windy\}$.

1.3 Wyniki eksperymentalne

1.3.1 Środowisko testowe

Dla wszystkich opisanych w tym rozdziale rozwiązań sprzętowych zostały opracowane tożsame funkcjonalnie implementacje programowe w języku C. Wyniki dotyczące czasu działania wersji programowej uzyskano przy użyciu komputera PC wyposażonego w 8 GB pamięci RAM i 4-rdzeniowy procesor Intel Core i7 3632QM o maksymalnej częstotliwości taktowania 3,2 GHz w trybie Turbo w systemie operacyjnym Windows 10. Kod źródłowy aplikacji został skompilowany przy użyciu kompilatora GNU GCC w wersji 9.2.

Do projektowania, kompilacji, syntezy i weryfikacji symulacyjnej implementacji sprzętowych w języku VHDL wykorzystano środowisko Quartus II 13.1. Zsyntetyzowane bloki sprzętowe zostały uruchomione na płycie deweloperskiej TeraSIC DE- 3 wyposażonej w układ FPGA typu Stratix III EP3SL150F1152C2N. W układzie tym dostępnych jest 113 600 bloków LE (ang. *Logical Elements*). Źródłem zegara układu FPGA działającego z częstotliwością 50 MHz był oscylator kwarcowy na płycie rozwojowej.

Procesor typu softcore NIOS II, jak również większość elementów systemu wbudowanego, zostały utworzone za pomocą narzędzia Qsys 13.1. Oprogramowanie dla NIOS II zostało zaimplementowane w języku C przy użyciu NIOS II Software Build Tools for Eclipse IDE.

Pomiary czasowe dla krótkich odcinków czasu uzyskano za pomocą oscyloskopu LeCroy waveSurfer 104MXs-B (pasmo 1 GHz, próbkowanie 10 GS/s). Do pomiarów dłuższych czasów działania używano sprzętowych jednostek pomiaru czasu tworzonych wewnątrz układu FPGA.

Należy zauważyć, że zegar komputera PC jest $\frac{clk_{PC}}{clk_{FPGA}} = 64$ razy szybszy niż źródło zegara płyty deweloperskiej.

Wszystkie obliczenia zostały przeprowadzone przy użyciu zestawów danych opisanych w podrozdziale 1.1.5. Dane zostały wstępnie przetworzone na komputerze PC pod kątem transformacji binarnej i ewentualnej dyskretyzacji.

1.3.2 Wyniki dla małych zbiorów danych

W poniższym podrozdziale przedstawiono wyniki czasowe oraz zajętość zasobów FPGA dla układów w wariancie "OR-cascade" oraz "Mixed". Ze względu na ograniczenia zasobów układu FPGA, rozwiązania te nie mogły przetworzyć zbiorów danych większych niż 110 obiektów, dlatego też ograniczone są do jednego zbioru danych, czyli bazy cukrzyków.

Tabela 1.4 przedstawia wyniki czasowe otrzymane dla implementacji sprzętowej w wersji "OR-cascade" (t_H) oraz tożsamej implementacji programowej (t_S).

Tabela 1.4: Porównanie czasu obliczania rdzenia (moduł "OR-cascade")

Obiekty —	**Program -** t_S [μs]	**Sprzęt -** t_H [μs]	$\frac{t_S}{t_H}$ —
15	112.28	0.0084	13 366
30	420.22	0.0104	40 405
45	983.42	0.0216	45 528
60	1 736.67	0.0310	56 021
90	4 074.80	0.0584	69 773
107	5 990.00	0.0683	87 701

Tabela 1.5 przedstawia wyniki czasowe otrzymane dla implementacji sprzętowej w wersji "Mixed" (t_H) oraz tożsamej implementacji programowej (t_S).

Tabela 1.5: Porównanie czasu obliczania rdzenia (moduł "Mixed")

Obiekty —	**Program -** t_S [μs]	**Sprzęt -** t_H [μs]	$\frac{t_S}{t_H}$ —
15	112.28	0.57	196
30	420.22	1.17	359
45	983.42	1.77	555
60	1 736.67	2.40	723
90	4 074.80	3.58	1 138
107	5 990.00	4.26	1 406

Tabela 1.6 przedstawia wykorzystanie zasobów układu FPGA wyrażone jako liczba zajętych bloków LE uzyskana w procesie syntezy.

Tabela 1.6: Zajętość struktury układu FPGA wyrażona w LE

Obiekty	"Or-cascade"	"Mixed"
15	1 374	1 252
30	5 675	1 979
45	12 666	3 418
60	22 863	3 541
90	52 339	5 335
107	74 916	7 171

Przedstawione wyniki dla układów w wariancie "OR-cascade" oraz "Mixed" wskazują na duży wzrost szybkości przetwarzania danych. Czas wykonania dla modułu sprzętowego w porównaniu z implementacją programową jest co najmniej o 1 rząd wielkości krótszy dla jednostki sprzętowej w wariancie sekwencyjnym "Mixed", co pokazano w tabeli 1.5 w kolumnach $\frac{t_S}{t_H}$. Należy podkreślić, że przyśpieszenie rośnie wraz ze wzrostem rozmiaru przetwarzanych danych. Dla kombinacyjnych wersji jednostek sprzętowych czasy te są o 2 do 3 rzędów wielkości krótsze, jednak zużywają one znacznie więcej zasobów niż rozwiązania sekwencyjne. W przypadku praktycznych rozwiązań preferowane są jednostki sekwencyjne, głównie ze względu na fakt, że zajmują stosunkowo niewielkie zasoby układu FPGA. Należy podkreślić, że nawet implementacja sekwencyjna z 64-krotnie wolniejszym zegarem taktującym układ FPGA jest znacznie szybsza, niż jej programowy odpowiednik uruchamiany na komputerze PC.

1.3.3 Wyniki dla dużych zbiorów danych

W poniższym podrozdziale przedstawiono wyniki czasowe oraz zajętość zasobów FPGA dla układów w wariancie "CORE-HIDM" i "CORE-PHIDM". Ze względu na ich sekwencyjny charakter oraz wykorzystanie zewnętrznych zasobów pamięciowych, rozwiązania te mogą przetwarzać zbiory danych liczące miliony obiektów.

Tabela 1.7 przedstawia czas obliczenia rdzenia dla rozwiązania sprzętowego (t_H) i programowego (t_S) dla wariantu "CORE-HIDM", który również jest zgodny z modułem "CORE-PHIDM" w przypadku pojedynczej instancji bloku *SubCORE*. Badania przeprowadzone zostały dla dwóch zbiorów danych o różnych licznościach. Wykorzystane skróty w liczbie obiektów to: $k = 10^3$, $M = 10^6$.

Tabela 1.7: Porównanie czasu obliczania rdzenia (moduł “CORE-HIDM” oraz “CORE-PHIDM” z 1 instancją *SubCORE*)

Obiekty —	**Sprzęt -** t_H [s]	**Program -** t_S [s]	$\frac{t_S}{t_H}$ —
Zbiór Poker Hand			
1k	0.003	0.033	10.875
2.5k	0.013	0.143	11.119
5k	0.055	0.603	10.951
10k	0.207	2.410	11.623
25k	1.225	14.721	12.015
50k	4.710	58.726	12.469
100k	21.737	237.942	10.946
250k	130.947	1 515.449	11.573
500k	506.225	6 092.916	12.036
1M	1 850.523	24 313.094	13.138
Zbiór cukrzyków			
1k	0.003	0.018	5.911
2.5k	0.013	0.078	6.044
5k	0.055	0.328	5.953
10k	0.207	1.31	6.318
25k	1.225	8.002	6.531
50k	4.710	34.216	7.265
100k	21.737	135.309	6.225
250k	130.947	861.781	6.581
500k	506.225	3 464.821	6.844
1M	1 850.523	13 825.976	7.471

Tabele 1.8 oraz 1.9 przedstawiają czas obliczenia rdzenia dla rozwiązania sprzętowego (t_H) i programowego (t_S) dla wariantu “CORE-PHIDM” odpowiednio w przypadku podwójnej oraz poczwórnej instancji bloku *SubCORE*.

Wykorzystanie zasobów FPGA jest stałe dla danej konfiguracji modułu sprzętowego i jest niezależne od rozmiaru danych wejściowych, ponieważ zbiory danych są podzielone na równe fragmenty, które są przetwarzane przez moduł w danej konfiguracji. Wykorzystanie zasobów układu FPGA w zależności od modułu:

- 21 562 LE dla “CORE-HIDM” oraz “CORE-PHIDM” z 1 instancją modułu *SubCORE*,
- 33 234 LE dla “CORE-PHIDM” z 2 instancjami modułu *SubCORE*,
- 45 668 LE dla “CORE-PHIDM” z 4 instancjami modułu *SubCORE*.

Powyższe liczby obejmują również zasoby zużywane przez procesor NIOS II.

Przedstawione wyniki czasowe wskazują na duży wzrost szybkości przetwarzania danych dla wszystkich prezentowanych rozwiązań. Czas obliczenia rdzenia dla

Tabela 1.8: Porównanie czasu obliczania rdzenia (moduł "CORE-PHIDM" z 2 instancjami *SubCORE*)

Obiekty —	**Sprzęt -** t_H [s]	**Program -** t_S [s]	$\frac{t_S}{t_H}$ —
Zbiór Poker Hand			
1k	0.002	0.033	17.770
2.5k	0.008	0.143	18.169
5k	0.0340	0.603	17.894
10k	0.127	2.410	18.991
25k	0.750	14.721	19.632
50k	2.882	58.726	20.375
100k	13.303	237.942	17.886
250k	80.139	1 515.449	18.910
500k	309.807	6 092.916	19.667
1M	1 132.511	24 313.094	21.468
Zbiór cukrzyków			
1k	0.002	0.018	9.659
2.5k	0.008	0.078	9.876
5k	0.034	0.328	9.727
10k	0.127	1.31	10.323
25k	0.750	8.002	10.672
50k	2.882	34.216	11.871
100k	13.303	135.309	10.171
250k	80.139	861.781	10.754
500k	309.807	3 464.821	11.184
1M	1 132.511	13 825.976	12.208

modułu sprzętowego w porównaniu z funkcjonalnie odpowiadającą implementacją programową wynosi od 5 (1 instancja modułu *SubCORE*) do 37 (4 instancje modułu *SubCORE*) razy szybciej. Jeśli uwzględni się różnicę częstotliwości zegara między komputerem PC a układem FPGA, wyniki te są jeszcze lepsze i średni współczynnik przyśpieszenia wynosi od 378 (1 instancja) do 2 376 (4 instancje). Współczynnik przyśpieszenia jest praktycznie stały dla danej konfiguracji modułu *SubCORE* i jest podobny dla wszystkich rozmiarów przetwarzanych zbiorów danych.

Czasy przetwarzania wariantu sprzętowego dla danego algorytmu w odniesieniu do obu zbiorów danych są takie same, ponieważ szerokość bitowa pojedynczego obiektu ze zbioru danych nie ma znaczenia dla czasu obliczeń przy założeniu, że słowo bitowe mieści się określonych granicach pamięci. Przetwarzanie porcji danych zajmuje taki sam czas, ponieważ moduł sprzętowy zawsze wykonuje ten sam rodzaj operacji. Dotyczy to wszystkich konfiguracji przedstawionych w tym podrozdziale modułów sprzętowego obliczania rdzenia.

Tabela 1.9: Porównanie czasu obliczania rdzenia (moduł "CORE-PHIDM" z 4 instancjami *SubCORE*)

Obiekty —	**Sprzęt - t_H** [s]	**Program - t_S** [s]	$\frac{t_S}{t_H}$ —
Zbiór Poker Hand			
1k	0.001	0.033	30.741
2.5k	0.005	0.143	31.432
5k	0.019	0.603	30.957
10k	0.073	2.410	32.855
25k	0.433	14.721	33.963
50k	1.666	58.726	35.249
100k	7.690	237.942	30.944
250k	46.323	1 515.449	32.715
500k	179.079	6 092.916	34.024
1M	654.631	24 313.094	37.140
Zbiór cukrzyków			
1k	0.001	0.018	16.710
2.5k	0.005	0.078	17.086
5k	0.019	0.328	16.828
10k	0.073	1.31	17.859
25k	0.433	8.002	18.462
50k	1.666	34.216	20.537
100k	7.690	135.309	17.596
250k	46.323	861.781	18.604
500k	179.079	3 464.821	19.348
1M	654.631	13 825.976	21.120

Należy zauważyć, że średni współczynnik przyśpieszenia związany z liczbą instancji modułu *SubCORE* dla "CORE-PHIDM" nie jest liniowy i wynosi:

- 1.634 w przypadku 2 instancji modułu *SubCORE*,
- 2.827 w przypadku 4 instancji modułu *SubCORE*.

Zmniejszający się współczynnik przyśpieszenia jest związany z narzutem obliczeniowym procesora NIOS II niezbędnym do przenoszenia danych binarnych z głównej pamięci RAM do pamięci RAM_n każdego z modułów *SubCORE*.

Podsumowanie

Implementacje sprzętowe algorytmów obliczania rdzenia dają duże przyśpieszenie w porównaniu z rozwiązaniami programowymi. Tego rodzaju podejście może być

jednym z kluczowych kierunków tworzenia skalowalnych systemów decyzyjnych w rozwiązaniach wymagających działania w czasie rzeczywistym.

Sprzętowe jednostki obliczające rdzenie nie zostały zoptymalizowane pod kątem wydajności. Czas przetwarzania można znacznie skrócić zwiększając częstotliwość zegara taktującego układu FPGA i dodatkowo reagując na dwa zbocza sygnału zegarowego. Przedstawione rozwiązania sprzętowe, w szczególności w wariancie sekwencyjnym, są łatwo skalowalne. Powielenie bloków obliczeniowych znacznie poprawiło szybkość przetwarzania. Należy zauważyć, że 4 instancje modułów w ostatnim przedstawionym wariancie architektury zajmują tylko około 50% średniej wielkości układu FPGA. Współcześnie dostępne największe na rynku układy posiadają ponad 10-krotnie większą liczbę bloków LE.

Dalsze badania będą koncentrować się na weryfikacji różnych rozmiarów modułów obliczeniowych oraz ich optymalizacji pod kątem wydajności. Prace badawcze będą się również skupiały na optymalizacji transferu danych między tablicą decyzyjną, a jednostkami obliczeniowymi. W chwili obecnej, czas niezbędny na transfer danych jest pomijany w wynikach empirycznych (kopiowanie danych z karty SD do pamięci RAM), jednak w przypadkach rzeczywistych nie można go pominąć. Istnieją jednak interfejsy transmisji danych oferujące przepływności na poziomie dziesiątek Gb/s (np. standard JESD204B zapewniający transmisje na poziomie 12,5 Gb/s, czy też USB 3.2 Gen 2x2 pozwalający na transfer danych z prędkością 20 Gb/s), które są znacznie szybsze, niż czas niezbędny do analizy danych przez opisane w tym rozdziale algorytmy sprzętowe.

W kolejnych badaniach należy również wziąć pod uwagę rodzaj przetwarzanych danych. Przedstawione obecnie rozwiązania są odpowiednie dla spójnych zbiorów danych. Zaprojektowane moduły nie obsługują poprawnie zbiorów danych z brakującymi wartościami.

Bibliografia

Czołombitko, M., i Stepaniuk, J. (2015). Generating core based on discernibility measure and MapReduce. W: M. Kryszkiewicz, S. Bandyopadhyay, H. Rybinski i S. K. Pal (red.), *Pattern recognition and machine intelligence - proceedings of 6th international conference,* 9124, Springer, 367-376.

Grzes, T., i Kopczynski, M. (2019). Hardware implementation on field programmable gate array of two-stage algorithm for rough set reduct generation. *Lecture Notes in Computer Science*, *11499*, Springer, 495-506.

Kanasugi, A., i Yokoyama, A. (2001). A basic design for rough set processor. W: *Proceedings of the 15th annual conference of japanese society for artificial intelligence,* 406–412.

Kopczynski, M., Grzes, T. i Stepaniuk, J. (2014a). FPGA in rough-granular compu-

ting: Reduct generation. W: *Proceedings of the 2014 IEEE/WCI/ACM International Joint Conferences on Web Intelligence,* 2, IEEE Computer Society, 364-370.

Kopczynski, M., Grzes, T. i Stepaniuk, J. (2014b). Generating core in rough set theory: Design and implementation on FPGA. *Lecture Notes in Computer Science*, *8537*, Springer, 209-216.

Kopczynski, M., Grzes, T. i Stepaniuk, J. (2015). Computation of cores in big datasets: An fpga approach. *Lecture Notes in Computer Science*, *9436*, Springer, 153-163.

Kopczynski, M., i Stepaniuk, J. (2013). Hardware implementations of rough set methods in programmable logic devices. W: A. Skowron i Z. Suraj (red.), *Rough sets and intelligent systems – professor zdzislaw pawlak in memoriam, intelligent systems reference library* 43, Springer, 309-321.

Lewis, T., Perkowski, M. i Jozwiak, L. (1999). Learning in hardware: Architecture and implementation of an fpga-based rough set machine. W: *Proceedings of the euromicro, 25th euromicro conference,* 1, 1326-1334.

Lichman, M. (2013). *Uci machine learning repository*. http://archive.ics.uci.edu/ml: Irvine, CA: University of California, School of Information and Computer Science.

Marz, N., i Warren, J. (2015). *Big data: Principles and best practices of scalable realtime data systems*. Greenwich: Manning Publications Co.

Muraszkiewicz, M., i Rybinski, H. (1993). Towards a parallel rough sets computer. W: W. Ziarko (red.), *Rough sets, fuzzy sets and knowledge discovery, proceedings of the international workshop on rough sets and knowledge discovery,* Springer, 434-443.

Pawlak, Z. (2004). Elementary rough set granules: Toward a rough set processor. W: S. K. Pal, L. Polkowski i A. Skowron (red.), *Rough-neurocomputing: Techniques for computing with words,* Springer-Verlag, 5-14.

Pawlak, Z., i Skowron, A. (2007). Rudiments of rough sets. *Information Sciences*, *177*, IOS Press, 3-27.

Stepaniuk, J. (2000). Knowledge discovery by application of rough set models. W: *Rough set methods and applications. new developments in knowledge discovery,information systems,* 56, Springer, 137-233.

Stepaniuk, J. (2008). *Rough-granular computing in knowledge discovery and data mining*. Berlin: Springer.

Stepaniuk, J., Kopczynski, M. i Grzes, T. (2013). The first step toward processor for rough set methods. *Fundamenta Informaticae*, *127*, IOS Press, 429-443.

Tiwari, K. S., i Kothari, A. G. (2014). Design and implementation of rough set algorithms on FPGA: A survey. *International Journal of Advanced Research in Artificial Intelligence*, *3*, The Science and Information Organization, 14-23.

Rozdział 2

EFEKTYWNE METODY WYZNACZANIA REDUKTÓW OPARTE NA UKŁADACH FPGA

Mateusz Choromański*

Streszczenie Redukcja atrybutów jest istotnym zagadnieniem w dziedzinie eksploracji danych, którego istotność wzrasta wraz z powiększającymi się danymi, które muszą być poddane analizie. Wyznaczanie minimalnych reduktów, czyli minimalnego zestawu atrybutów, znacznie przyśpiesza analizę danych, jednak wraz z ich wzrostem: ilościowym i objętościowym, pojawiła się także potrzeba maksymalnego przyśpieszenia wyliczania wspomnianych reduktów. Zagadnienie akceleracji obliczeń służących zredukowaniu liczby istotnych atrybutów stało się obiektem licznych badań i eksperymentów, ze względu na ogromną liczbę obliczeń oraz innych operacji, którym poddawane są dane. W ich wyniku powstawały coraz to wydajniejsze algorytmy. Oprócz opracowywania nowych rozwiązań, podejmowane są próby uzyskania dodatkowych przyśpieszeń z użyciem przetwarzania wielowątkowego i obliczeń równoległych. Znaczne przyśpieszenia są uzyskiwane także dzięki wykorzystaniu nowoczesnych rozwiązań sprzętowych. Do najpowszechniejszych należą współczesne cyfrowe układy programowalne z rodziny FPGA. Niniejszy rozdział jest przeglądem najefektywniejszych metod przyśpieszających wyliczanie minimalnego zestawu atrybutów koniecznych do prawidłowej analizy danych. Zostały w nim przedstawione autorskie podejścia sprzętowe wykorzystujące programowalny układ cyfrowy FPGA.

Słowa kluczowe: zbiory przybliżone, redukcja atrybutów, implementacja sprzętowa, układy FPGA

Wprowadzenie

Redukcja atrybutów (Pawlak, 1991) jest bardziej istotnym zadaniem wstępnego przetwarzania danych niż kiedykolwiek wcześniej w historii. W erze Big Data redukcja

* Wydział Informatyki, Politechnika Białostocka, Wiejska 45A, 15-351 Białystok, m.choromanski@pb.edu.pl
DOI 10.24427/978-83-66391-58-1_2

zbioru danych, nawet o jedną cechę/atrybut, może znacząco wpłynąć na rozmiar danych, a co za tym idzie na czas potrzebny do ich przetworzenia. Pomimo faktu, że redukcja jak największej liczby atrybutów dużego zbioru danych może być sama w sobie kosztowna, korzyści płynące z dalszego przetwarzania znacznie mniejszego zbioru mogą być nieporównywalnie większe.

Zadanie redukcji atrybutów było szeroko badane z teoretycznego i praktycznego punktu widzenia (np. Chen, Zhao, Zhang, Yang i Zhang (2012); Degang, Changzhong i Qinghua (2007); X. Hu i Cercone (1995); Kryszkiewicz (1998, 2001); Skowron i Rauszer (1992); Stepaniuk (2008); R. Swiniarski (2001); R. W. Swiniarski i Skowron (2003); Thi i Giang (2013); X. Zhang, Mei, Chen i Li (2013)) w teorii zbiorów przybliżonych (Pawlak, 1991; Pawlak i Skowron, 2007), gdzie jest postrzegane jako narzędzie matematyczne do radzenia sobie z niespójnymi danymi. Pomimo istnienia bogatej literatury, redukcja atrybutów jest nadal rozwojowym tematem; nieustannie odkrywane są nowe obszary jej zastosowania oraz nowe metody jej doskonalenia (np. Czolombitko i Stepaniuk (2016); Dong, Sun i Yang (2016); Y. Jiang i Yu (2016); Jing, Yunliang i Yong (2017); Li i Yang (2016); Liu, Hua i Chen (2017); Teng i in. (2016)).

Poniższe podrozdziały zawierają przegląd podejść znajdowania reduktów opartych na rozwiązaniach sprzętowych.

Programowe podejścia do wyznaczania reduktów

W teorii zbiorów przybliżonych opracowano różnorodne algorytmy służące do znajdowania reduktów systemu informacyjnego lub tablicy decyzyjnej. Można je podzielić na następujące ogólne grupy: podejście oparte na macierzy rozróżnialności (np. Degang i in. (2007); X. Hu i Cercone (1995); Kryszkiewicz (1998); Skowron i Rauszer (1992); C. Wang, He, Chen i Hu (2014); Ye i Chen (2002); W.-X. Zhang, Mi i Wu (2003)), podejście oparte na obszarze pozytywnym (np. Grzymala-Busse (1991); Q. Hu, Yu, Liu i Wu (2008); Jia, Liao, Tang i Shang (2013); Pawlak (1991); Qian, Liang, Pedrycz i Dang (2010); Xie, Shen, Liu i Xu (2013); Yao i Zhao (2008)) oraz podejście oparte na entropii informacyjnej (np. Q. Hu, Yu, Xie i Liu (2006); Liang, Mi, Wei i Wang (2013); Liang i Xu (2002); Ślęzak (2002); Wei, Liang, Qian, Wang i Dang (2010); Wei, Liang, Wang i Qian (2013)).

Algorytmy pierwszej grupy wykorzystują strukturę pomocniczą, zwaną macierzą rozróżnialności, która jest konstruowana na podstawie tabeli danych. Każda komórka macierzy pokazuje atrybuty, które są różne dla danej pary obiektów. Każdy minimalny podzbiór atrybutów zawierający co najmniej jeden atrybut z każdej komórki jest reduktem.

Druga grupa algorytmów działa na pozytywnym obszarze tablicy decyzyjnej. Obszar ten zawiera obiekty spójne dla danego podzbioru atrybutów. Minimalnym podzbiorem atrybutów, który zachowuje spójność danych jest redukt.

Ostatnia grupa algorytmów ocenia podzbiór atrybutów za pomocą miary entropii informacyjnej. Entropia informacyjna pokazuje stopień rozróżnialności danego podzbioru atrybutów. Ta o najwyższej jakości w ramach danego kryterium jest wybierana jako redukt.

Ważną gałęzią redukcji atrybutów jest znajdowanie minimalnych reduktów. Znalezienie takiego reduktu pozwala na skonstruowanie najmniejszej reprezentacji danych (pod względem liczby atrybutów), zachowując pierwotny poziom rozpoznawalności obiektów. Znalezienie minimalnego reduktu jest jednak bardziej złożonym zadaniem niż znalezienie jakiegokolwiek reduktu, a mianowicie zostało pokazane, że jest to problem NP-trudny (Skowron i Rauszer, 1992).

Jednym z ogólnych podejść jest ograniczenie problemu znalezienia wszystkich reduktów do tych, których wielkość jest najmniejsza. Rozwiązanie jest raczej proste, ponieważ może polegać na filtrowaniu wszystkich reduktów lub kontrolowaniu procesu ich wytwarzania, tak aby przynajmniej niektóre z nich, które zostały wcześniej rozpoznane jako nieminimalne nie były generowane. Wadą tego podejścia jest jego złożoność, która jest zbieżna z podejściem znajdowania wszystkich reduktów.

Inne, bardziej ogólne podejście opiera się na dobrze znanej strategii wyszukiwania wszerz. Pusty zestaw atrybutów, alternatywnie zbiór składający się z podstawowych atrybutów (rdzeń reduktów), jest za każdym razem iteracyjnie zwiększany o jeden atrybut. Jeśli dany podzbiór nie jest reduktem, to poprzednio dodany atrybut jest usuwany, a do podzbioru dodawany jest inny, dotychczas nieużywany. Takie podejście gwarantuje sprawdzenie tylko podzbiorów o liczności nie wyższej niż liczba minimalnych reduktów. Jednak rozwiązanie może być czasochłonne, jeśli liczba wszystkich atrybutów opisujących dane jest stosunkowo duża.

Modyfikacja powyższego podejścia polega na obliczeniu częstotliwości występowania każdego z atrybutów, która określa kolejność, w jakiej atrybuty mają być dodawane do podzbioru. Mianowicie, atrybut najczęściej występujący w macierzy rozróżnialności jest uznawany za pierwszy.

Można również znaleźć bardziej konkretne podejścia do znalezienia jednego lub wszystkich minimalnych reduktów.

Algorytm genetyczny został zaadaptowany w pracy Wroblewski (1995) w celu znalezienia minimalnego reduktu. Każdy podzbiór zestawu atrybutów jest indywidualny i reprezentowany przez ciąg bitów, gdzie „1” („0”) oznacza, że atrybut występuje (nie występuje) w podzbiorze. Funkcja dopasowania jest definiowana na podstawie liczności podzbioru i liczby wierszy w macierzy rozróżnialności, które są objęte podzbiorem atrybutów. Najlepszy osobnik z ostatniego pokolenia jest zwracany jako minimalny redukt. Podejście to może być szybkie tylko wtedy, gdy kryterium zatrzymania jest łatwe do osiągnięcia (np. mała liczba pokoleń) i nie gwarantuje, że znaleziony redukt będzie zawsze minimalny.

Podobne, ale prostsze rozwiązanie do tego z Wroblewski (1995) zostało umieszczona w X. Wang, Yang, Peng i Teng (2005). Autorzy postawili problem znalezienia minimalnych reduktów w ramach optymalizacji roju cząstek. Dzięki temu nie są potrzebne złożone operatory, takie jak krzyżowanie czy mutacje, a potrzebne są tylko prymitywne i proste operatory matematyczne. Zostało eksperymentalnie zweryfikowane, że propozycja jest mniej kosztowna obliczeniowo pod względem czasu wykonywania i użytej pamięci w porównaniu z podejściem używającym algorytmów genetycznych.

Aby znaleźć minimalny redukt, zbiór danych w pracy Bakar, Sulaiman, Othman i Selamat (2002) przekształca się w model programowania binarnych liczb całkowitych (BIP), a redukcję atrybutów uważa się za problem spełnialności. Aby rozwiązać problem za pomocą modelu BIP, zastosowano algorytm rozgałęzienia i powiązania. Weryfikacja eksperymentalna wykazała, że podejście to znacznie zmniejsza liczbę reguł, które można wygenerować na podstawie uzyskanych reduktów.

W pracy Jensen, Shen i Tuson (2005) problem znalezienia minimalnych reduktów zbioru przybliżonego jest przeformułowany w ramach propozycji spełnialności (SAT). Klauzule cech w koniunkcyjnej postaci normalnej (CNF) są generowane ze zbioru danych. Klauzule są spełnione, jeśli po przypisaniu wartości prawdziwych do ich wszystkich zmiennych formuła jest prawdziwa w zbiorze danych. Zadanie polega na znalezieniu najmniejszej liczby takich cech, aby spełnić formułę CNF. W tym podejściu problem SAT rozwiązano za pomocą algorytmu Davisa-Logemanna-Lovelanda (DPLL). Podejście zostało porównane eksperymentalnie z podejściem opartym na ocenie redukcji obszaru pozytywnego. Proponowane rozwiązanie jest bardziej czasochłonne, ale w przeciwieństwie do odniesienia zawsze zwraca minimalne redukty.

Technika optymalizacji roju cząstek (PSO) została zaadaptowana w badaniach X. Wang, Yang, Teng, Xia i Jensen (2007) w celu znalezienia minimalnych reduktów zbioru przybliżonego. Cząstki z PSO odpowiadają atrybutom. Pozycja cząstki to binarny ciąg bitów o długości równej całkowitej wielkości podzbioru atrybutów. Każdy bit ciągu określa, czy atrybut został wybrany (1), czy nie (0). Każda pozycja odpowiada podzbiorowi atrybutów. Znajdowanie minimalnych reduktów odbywa się zgodnie ze standardowym algorytmem PSO. Podejście jest stosunkowo szybkie i zawsze gwarantuje znalezienie minimalnych reduktów.

Do znalezienia minimalnych reduktów Xu, Liu i Zhou (2008) używają modyfikację metody redukcji atrybutów opartą na obszarze pozytywnym. Proces redukcji atrybutów jest kontrolowany przez próg, czyli minimalną liczbę atrybutów potrzebnych do rozróżnienia wszystkich obiektów w zbiorze danych.

W pracy Su i Guo (2017) użyto kombinacji teorii zbiorów przybliżonych i algorytmu stada, aby znaleźć minimalny redukt. Rdzeń reduktów obliczony na podstawie macierzy rozróżnialności jest iteracyjnie zwiększany o jeden atrybut, aż do znalezienia minimalnego reduktu. Zestawy atrybutów innych niż podstawowe, lecz tej samej liczności są kodowane jako liczby całkowite będące indywiduami w algorytmie

stada. Zależność atrybutów jest stosowana do obliczania wartości dopasowania podzbiorów atrybutów.

Sprzętowe podejście do wyznaczania reduktów

Wsparcie sprzętowe przetwarzania danych oparte na zbiorach przybliżonych ma długą historię i jest ściśle związane z początkami teorii zbiorów przybliżonych. Pomysł przykładowego procesora, który generuje reguły klasyfikacji z tablic decyzyjnych opisał Pawlak (2004). W proponowanym rozwiązaniu decyzję podejmuje się po sekwencji obliczeń czynników opisujących jakość decyzji (siła, pewność, pokrycie). Konstrukcja procesora pozwala na użycie arytmometru cyfrowego lub analogowego. Inne wczesne badania obejmują konstrukcję procesora opartego na sieciach komórkowych, przedstawioną przez Lewis, Perkowski i Jozwiak (1999), a także koncepcję urządzenia zdolnego do zminimalizowania dużych funkcji logicznych tworzonych na podstawie macierzy rozróżnialności, opracowanego przez Kanasugi i Yokoyama (2001).

Jeden z pierwszych rzeczywistych systemów wspierających przetwarzanie danych ze zbiorów przybliżonych został opisany przez Kanasugi i Matsumoto (2007). Autorzy przedstawili projekt i implementację wstępnie ustawionego procesora i pokazali ogromne przyśpieszenie w obliczeniach: proponowany procesor był dziesięciokrotnie szybszy od komputera PC, mimo że częstotliwość taktowania była mniejsza o około 70 razy. Sun, Qi i Zhang (2011) przedstawili implementację metod ze zbiorów przybliżonych dla badań technologii diagnostyki usterek w oparciu o układ FPGA. Ci sami autorzy w (Sun, Wang, Lu i He, 2013) pokazali sprzętową implementację dyskretyzacji atrybutów ciągłych. W badaniach D.-L. Jiang, Zhao, Wang, Li i Yang (2013) zbiory przybliżone zostały użyte do zgrupowania danych do eksploracji. Wszystkie te rozwiązania wykazały duże przyśpieszenie w porównaniu z implementacjami programowymi i udowodniły, że implementacje oparte na FPGA są obecnie jednym z najważniejszych problemów badawczych.

Najbardziej zaawansowane i najnowsze badania związane z problemem generowania reduktów opisane są w pracy K. Tiwari i Kothari (2015, 2016); K. S. Tiwari i Kothari (2014). Badania na temat sprzętowych implementacji metod zbiorów przybliżonych przedstawiono w pracy Grześ, Kopczyński i Stepaniuk (2013). Urządzenie generujące superreduktы zostało opisane w pracy Kopczynski, Grzes i Stepaniuk (2014). Żadne z opisanych rozwiązań nie pozwala na obliczenie wszystkich reduktów.

Celem tego rozdziału jest zaprezentowanie efektywnych metod wyznaczania minimalnego reduktu ze wsparciem sprzętowym w postaci układu z rodziny FPGA.

Na wstępie, na podstawie analizy istniejących metod i algorytmów wybrano rozwiązania wykorzystujące metodę znajdowania minimalnych reduktów za pomocą

wyszukiwania wszerz. W porównaniu z innymi nie są one skomplikowane, ponieważ wykonywane są tylko podstawowe operacje takie, jak generowanie kandydatów na redukt (czyli kombinacji atrybutów), obliczenie macierzy rozróżnialności i sprawdzenie kandydata na redukt w macierzy rozróżnialności.

Następnie zostały zdefiniowane algorytmy przeszukiwania wszerz oparte na poszukiwaniu „ślepym", oraz z wykorzystaniem częstotliwości występowania atrybutów (podrozdział 2.1.1). Przedstawione zostały architektury FPGA wybranych podejść (podrozdział 2.1.3), implementacja oraz jej efekty są weryfikowane pod kątem podejścia sprzętowego (podrozdział 2.2).

Na koniec omówiono ważne różnice między tymi dwiema strategiami.

2.1 Znajdowanie minimalnych reduktów metodą przeszukiwania wszerz

W niniejszym rozdziale zostały przedstawione algorytmiczne podejścia do znajdowania minimalnych reduktów przy użyciu strategii „ślepego" wyszukiwania wszerz oraz w oparciu o częstotliwości występowania atrybutów, jak również możliwości sprzętowej implementacji prowadzącej do zmniejszenia czasu wykonania algorytmu.

Definicje podstawowych pojęć, używanych w pracy, tj. tablica decyzyjna, macierz rozróżnialności, redukt, Czytelnik może odnaleźć np. w Choromański, Grześ i Hońko (2020).

2.1.1 Proponowane algorytmy wyznaczania reduktów

Wersja „ślepa" strategii opartej na wyszukiwaniu wszerz (patrz algorytm 2.1) (Choromański i in., 2020) rozpoczyna się od obliczenia macierzy rozróżnialności oraz rdzenia, tj. podzbioru atrybutów, który jest zawarty w każdym redukcie. Następnie wszystkie zestawienia o najmniejszej liczności (każde zestawienie składa się z podstawowych atrybutów i jednego dodatkowego atrybutu) są sprawdzane pod kątem bycia reduktem. Przeszukiwanie jest przerywane po znalezieniu wymaganej liczby reduktów lub po sprawdzeniu wszystkich zestawień i znalezieniu co najmniej jednego reduktu. Jeśli nie zostanie znaleziony żaden redukt, wszystkie zestawienia o liczności większej o jeden (każde zestawienie jest konstruowane na podstawie zestawienia z poprzedniego poziomu liczności przez dodanie kolejnego atrybutu) są sprawdzane w ten sam sposób. Zestawienia konstruowane są w kolejności alfanumerycznej, dzięki czemu każde nowe zestawienie nie jest powtórzeniem żadnego wygenerowanego wcześniej.

Algorytm 2.1 wykorzystuje następujące funkcje:

Algorytm 2.1 *GenerateMinReducts*

INPUT: $DT = (U, A \cup \{d\})$ – tablica decyzyjna; k – liczba minimalnych reduktów do odnalezienia (0 – wszystkie redukty);
OUTPUT: MR – zestaw znalezionych pierwszych k minimalnych reduktów;

```
1:  MR := ∅;
2:  DM := computeDiscMatrix(DT);
3:  C := computeCore(DM);
4:  𝒮 := ∅;
5:  while MR = ∅ do
6:      if 𝒮 = ∅ then
7:          𝒮 = {C}
8:      else 𝒮 := computeAllCombs(A, 𝒮)
9:      end if
10:     for S ∈ 𝒮 do
11:         if isReduct(DM, S) then
12:             MR = MR ∪ {S};
13:             if |MR| = k then
14:                 break
15:             end if
16:         end if
17:     end for
18: end while
```

1. $computeDiscMatrix(DT)$ oblicza macierz rozróżnialności dla tabeli decyzyjnej DT,
2. $computeCore(DM)$ oblicza rdzeń dla macierzy rozróżnialności DM,
3. $computeAllCombs(A, \mathscr{S})$ oblicza dla każdego zestawienia atrybutów $S \in \mathscr{S}$ wszystkie zestawienia (przez dodanie jednego atrybutu), które nie zostały do tej pory sprawdzone,
4. $isReduct(DM, S)$ sprawdza, czy zestawienie atrybutów S jest reduktem w nawiązaniu do macierzy rozróżnialności DM.

Opis działania algorytmu 2.1 w układzie FPGA został przedstawiony w rozdziale 2.1.3.3.

Wersja strategii opartej na wyszukiwaniu wszerz w oparciu o częstotliwości występowania atrybutów (patrz algorytm 2.2), (Choromański i in., 2020) zaczyna się od tych samych obliczeń co w przypadku metody „ślepej", tj. od wyliczenia macierzy rozróżnialności wraz z rdzeniem. Następnie dla każdego atrybutu, który nie jest zawarty w rdzeniu, obliczana jest jego częstotliwość występowania w macierzy rozróżnialności. Zestawienie o najmniejszej liczności (czyli podstawowe atrybuty plus jeden atrybut) są konstruowane zaczynając od najczęstszego atrybutu i kończąc po użyciu wszystkich lub najczęstszych atrybutów l (l zdefiniowanych przez użytkownika). Zestawienia są weryfikowane jako redukty w taki sam sposób, jak w przy-

padku wersji „ślepej”. Zestawienia o wyższym poziomie liczności są konstruowane na podstawie tych z poprzedniego poziomu przez dodanie jednego atrybutu wybranego zgodnie z kolejnością częstości występowania. Należy podkreślić, że taki sposób konstruowania zestawień atrybutów nie gwarantuje, że powstają tylko unikalne zestawienia, dlatego każde nowe zestawienie jest najpierw porównywana z innymi o tym samym poziomie liczności, które zostały do tej pory wygenerowane. W porównaniu z algorytmem 2.1, dodatkową funkcją wykorzystywaną przez algorytm 2.2 jest $computeAttrFreq(DM,B)$), który oblicza dla każdego atrybutu z zbioru B częstotliwość jego występowania w macierzy rozróżnialności DM.

Algorytm 2.2 *GenerateMinReductsFreq*

INPUT: $DT = (U, A \cup \{d\})$ – tablica decyzyjna; k – liczba minimalnych reduktów do odnalezienia (0 – wszystkie redukty); l – numer pierwszego, najczęściej występującego atrybutu (0 – wszystkie redukty);
OUTPUT: MR – zestaw pierwszych k odnalezionych minimalnych reduktów;

```
1:  MR := ∅;
2:  DM := computeDiscMatrix(DT);
3:  C := computeCore(DM);
4:  𝒮 := C;
5:  while MR = ∅ do
6:      𝒮' = ∅;
7:      for S ∈ 𝒮 do
8:          order := sort(computeAttrFreq(DM, A \ S));
9:          if l = 0 then
10:             l' = |order|;
11:         else l' = l;
12:         end if
13:         i := 0;
14:         while i < l' do
15:             S' = S ∪ order[i];
16:             if S' ∈ 𝒮' then
17:                 continue;
18:             end if
19:             if isReduct(DM, S') then
20:                 MR = MR ∪ {S'};
21:                 if |MR| = k then
22:                     break;
23:                 end if
24:             end if
25:             i := i + 1;
26:             𝒮' := 𝒮' ∪ {S'};
27:         end while
28:     end for
29:     𝒮 := 𝒮';
30: end while
```

Należy zauważyć, że w związku z wprowadzeniem parametru l, algorytm 2.2 jest w rzeczywistości połączeniem strategii przeszukiwania wszerz i wgłąb. Mianowicie,

- $l = 0$ – strategia przeszukiwania wszerz,
- $l = 1$ – strategia poszukiwania wgłąb,
- $l > 1$ – ograniczona strategia wyszukiwania wszerz, rozszerzona strategia poszukiwania wgłąb.

Opis działania algorytmu 2.2 w układzie FPGA został przedstawiony w rozdziale 2.1.3.3.

Algorytmy przeszukujące wszerz zaimplementowane na zwykłym komputerze klasy PC posiadają złożoność obliczeniową $O(n^2)$, gdzie n – liczność zbioru obiektów A, natomiast pamięciowa wynosi $O(2^m)$, gdzie m – liczba atrybutów warunkowych. W przypadku implementacji z użyciem układów kombinacyjnych na układzie FPGA, operacje mogą być wykonywane od razu, dzięki czemu może być osiągnięta złożoność nawet równa $O(1)$. Rzeczywista złożoność obliczeniowa jest jednak trudna do przewidzenia, ponieważ wszystko zależy od czasu propagacji sygnału do poszczególnych bloków wewnątrz FPGA, natomiast osiągnięcie takiego wyniku może być niemożliwe ze względu na fizyczne ograniczenie wynikające ze skończonej liczby elementów logicznych zawartych w układzie FPGA. W tym wypadku liczba wykorzystywanych elementów logicznych jest proporcjonalna do n^2.

2.1.2 Układy programowalne z rodziny FPGA

2.1.2.1 Definicja

FPGA (ang. *Field Programmable Gate Array*) jest to rodzaj programowalnego układu logicznego (ang. *Programmable Logic Device* – PLD) charakteryzujący się architekturą złożona z matrycy komórek logicznych połączonych za pomocą linii poprowadzonych w kanałach połączeniowych (Skahill, 2001). Na obrzeżach matrycy komórek logicznych znajdują się elementy wejścia/wyjścia, mogące być skonfigurowane jednokierunkowo (jako wejście lub wyjście) oraz dwukierunkowo. Współczesne układy FPGA mogą zawierać ponad pół miliona bloków logicznych oraz być zintegrowane z procesorem ARM tworząc tym samym układ SoC (ang. *System-on-Chip*). Najmniejszą jednostką w układzie FPGA jest posiadający cztery wejścia oraz 3 wyjścia element logiczny. Dzięki zastosowaniu tablicy przeglądowej (LUT) możliwe jest zaimplementowanie w układzie praktycznie każdej funkcji logicznej.

2.1.2.2 Języki opisu sprzętu

Języki opisu sprzętu (HDL) jest to rodzina wyspecjalizowanych języków komputerowych wykorzystywana do opisu struktury i działania układów cyfrowych. Pierwsi przedstawiciele tej rodziny powstali w latach 60. XX wieku, kiedy pojawiło się zapotrzebowanie na uproszczenie procesu projektowania układów cyfrowych oraz zautomatyzowania niektórych jego etapów. Oprócz samej syntezy układów, HDL umożliwiają również optymalizację układu, minimalizację funkcji czy tworzenie platform testowych do symulacji i tym samym sprawdzenia zaprojektowanego układu. Projektowanie układów cyfrowych z użyciem HDL można wykonać na dwa sposoby:

- *behawioralny* – opisywane są zależności pomiędzy danymi z wejścia oraz wyjścia układu, sama realizacja układu zostaje zrealizowana przez kompilator,
- *funkcjonalny* – tworzone są bloki funkcjonalne (np. sumatory, bloki zawierające funkcje logiczne) oraz opisywane są zależności między nimi.

Przykładami języków opisu sprzętu są:

- VHDL (ang. *Very High Speed Integrated Circuits Hardware Description Language*) – jeden z najpowszechniejszych języków z rodziny HDL. Pierwotnie stworzony na potrzeby dokumentacji układów ASIC. Język rozwijany jest do chwili obecnej, aktualnie najnowszą wersją jest IEEE Std 1076-2008 zwana potocznie VHDL 2008,
- Verilog – drugi najpopularniejszy język opisu sprzętu, składniowo bardzo przypominający język C; od roku 1990 język ten jest otwartym standardem; język ten podobnie jak VHDL jest stale rozwijany, najnowsza aktualna wersja to IEEE Std 1364-2001; z Veriloga wyewoluował inny język z rodziny HDL – SystemVerilog,
- AHDL (ang. *Altera Hardware Description Language*) – język opisu sprzętu stworzony przez firmę Altera (aktualnie jest ona częścią firmy Intel); język ten przypomina Veriloga, jednak może być wykorzystywany tylko w przypadku programowania układów CPLD i FPGA produkcji Altery/Intela,
- SystemC – HDL pozwalający modelować układy z poziomu systemu; składniowo praktycznie nie różni się od języka C++.

2.1.3 Sprzętowa implementacja algorytmów wyznaczania reduktów

W poniższych podrozdziałach opisano używany sprzęt, zestaw danych i zaimplementowane bloki służące do wyliczania minimalnych reduktów z użyciem układu FPGA.

2.1.3.1 Wykorzystany sprzęt

Projekt został zrealizowany na dwóch różnych urządzeniach. Pierwsze z nich to Intel Arria V SoC (5ASTFD5K3F40I3), określany dalej jako „Arria", będącym 28 nm FPGA z 462 tysiącami elementów logicznych i zintegrowanym z dwurdzeniowym procesorem ARM Cortex-A9 o taktowaniu 1,05 GHz. W naszym rozwiązaniu jednak nie zastosowano zintegrowanego procesora ARM; zamiast niego zastosowano procesor typu softcore Nios II. Został on wybrany ze względu na łatwiejsze debugowanie na poziomach rejestrów i ALU. Nios II był taktowany zegarem o częstotliwości 50MHz, co dawało większą szansę zaobserwowania różnic między rozwiązaniami. Oprogramowanie użyte do kompilacji, syntezy i generowania plików konfiguracyjnych to Quartus Prime 17.1.0 Build 590 25.10.2017 SJ Standard Edition. Kod C został skompilowany za pomocą Eclipse Mars.2 Release (4.5.2) Build 20160218-0600 z kompilatorem GCC w wersji 4.8.3.
Drugim urządzeniem był Xilinx Zynq Ultrascale+ MPSoC (XCZU9EG-2FFVB1156) określany dalej jako „Zynq US+", który jest 16 nm FPGA z 600 tysiącami elementów logicznych i zintegrowanym z czterordzeniowym procesorem ARM Cortex-A53 (taktowany na 1,5 GHz), dwurdzeniowym procesorem czasu rzeczywistego Cortex-R5 (taktowany z częstotliwością 600 MHz) i procesorem graficznym Mali-400MP2 (taktowany z częstotliwością 667 MHz). Oprogramowanie używane dla Zynq US+ to Vivado Design Edition 2018.3 (64-bit) SW Build 2405991, IP Build: 2404404. Kod C został skompilowany za pomocą Xilinx SDK, który jest oparty na Eclipse Wersja: 4.6.1.v20160907-1200, identyfikator kompilacji: M20160907-1200 z kompilatorem GCC w wersji 7.3.1.

2.1.3.2 Użyte dane oraz ich struktura

W tym projekcie zostały wykorzystane dane dotyczące dzieci z cukrzycą insulinozależną typu 1 (Stepaniuk, 1999). Struktura danych determinuje strukturę niektórych bloków w implementacji sprzętowej. Jednakże ogólna koncepcja działania pozostaje niezależna od typu, rodzaju i struktury danych. Dane wykorzystywane w badaniach obejmują 107 obiektów. Każdy z nich ma 12 atrybutów, które w reprezentacji binarnej tworzą łącznie 16-bitowe słowo:

- płeć - 1 bit,
- wiek w którym zdiagnozowano chorobę - 2 bity,
- czas trwania choroby - 2 bity,
- występowanie cukrzycy w rodzinie - 1 bit,
- typ terapii insulinowej - 1 bit,
- infekcje układu oddechowego - 1 bit,
- remisja - 1 bit,
- HbA1c - 2 bity,

- nadciśnienie - 1 bit,
- masa ciała - 2 bity,
- hipercholesterolemia - 1 bit,
- hipertriglicerydemia - 1 bit.

Istnieje również jeden 1-bitowy atrybut decyzji:

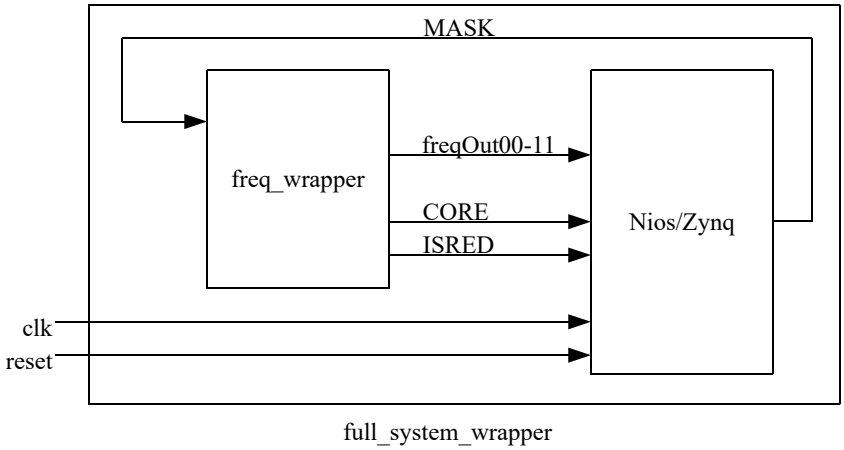
- mikroalbuminuria.

Dane są podzielone na dwa oddzielne zestawy:

- pozytywne - z 56 przypadkami, które mają mikroalbuminurię (wartość atrybutu decyzyjnego jest równa „1”),
- negatywna - z 51 przypadkami, które nie mają mikroalbuminurii (wartość atrybutu decyzyjnego jest równa „0”).

2.1.3.3 Zaimplementowane bloki sprzętowe

Architektura zaproponowanego rozwiązania została zaprezentowana na rysunku 2.1



Rysunek 2.1: Architektura rozwiązania na podstawie Choromański i in. (2019)

Full_system_wrapper jest plikiem najwyższego poziomu, który zawiera komponenty *freq_wrapper* oraz *nios/Zynq*, a także łączy je ze sobą. Blok ten można traktować jako cały system i posiada dwa porty wejściowe:

- **clk** – wejście zegarowe o częstotliwości 50 MHz dla procesora *nios* w Arrii (100 MHz w przypadku Zynq US+),
- **reset** – sygnał resetujący wszystkie komponenty.

Blok ten nie posiada portów wyjściowych. Cały system składa się z następujących elementów:

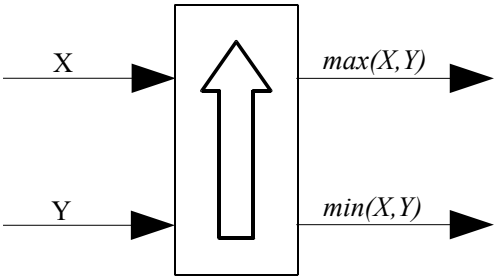
- **nios/Zynq** – instancja procesora Nios II/f (procesora Zynq w przypadku Zynq US+) używany do kontrolowania wykonania algorytmu. Głównym celem tego bloku jest znalezienie reduktu za pomocą aplikacji napisanej w C. Wykorzystuje pozostałe bloki systemu podczas wykonywania programu w celu zwiększenia szybkości obliczeń: odbiera rdzeń reduktów obliczony przez FPGA i zgodnie z nim wysyła możliwych kandydatów na redukt, dopóki sygnał ISRED zmieni wartość na „0”. Działanie tego bloku zostało przedstawione w liniach 1.6–1.15 w algorytmie 2.1 oraz w liniach 2.6–2.23 w algorytmie 2.2. Ten blok ma cztery porty wejściowe:
  - **clk** – wejście zegara o częstotliwości 50 MHz (100 MHz dla Zynq US+),
  - **reset** – sygnał do resetowania procesora,
  - **freqOut00…freqOut11** – 16-bitowe słowo z częstotliwością każdego atrybutu,
  - **CORE** – dostarcza rdzeń obliczony w komponencie *freq_wrapper*,
  - **ISRED** – sygnał wskazujący, że sygnał MASK jest reduktem.

  Ten blok ma jeden port wyjściowy:
  - **MASK** – wysyła do *freq_wrapper* kandydata na redukt na podstawie obliczonego rdzenia (*CORE*) podanego na wejściu.Ten sygnał jest również używany w innych komponentach.

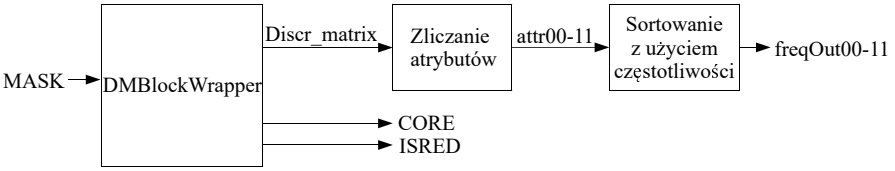
- **freq_wrapper** – komponent, w którym częstości atrybutów są obliczane i sortowane za pomocą algorytmu bitonicznego (przedstawionego na rysunku 2.2) od najczęściej do najrzadziej występujących. W algorytmie 2.2 działanie tego bloku jest przedstawione jako funkcja *sort()* znajdująca się w linii 2.8. Każdy wynik jest reprezentowany przez 16-bitowy wektor, gdzie pierwsze 4 bity zawierają numer atrybutu, a pozostałe 12 bitów to liczba wystąpień. *freq_wrapper* zawiera komponent *DMBlockWrapper* odpowiedzialny za obliczanie rdzenia i reduktu. Ten blok ma jeden port wejściowy:
  - **MASK** – kandydat na redukt dostarczony z komponentu *nios/Zynq*.

  Ten blok ma czternaście portów wyjściowych:
  - **freqOut00…freqOut11** – 16-bitowe słowo z częstotliwością każdego atrybutu,
  - **CORE** – obliczony rdzeń dostarczony z *DMBlockWrapper*,

- **ISRED** – pojedynczy bit informujący, że kandydat przekazany z komponentu *nios/Zynq* jest reduktem.



Rysunek 2.2: Schemat sortowania bitonicznego



Rysunek 2.3: Schemat bloku freq_wrapper na podstawie Choromański i in. (2020)

Architekturę bloku *freq_wrapper* przedstawiono na rysunku 2.3. Jak wspomniano wcześniej, ten komponent składa się z jednego bloku *DMBlockWrapper* opisanego poniżej:

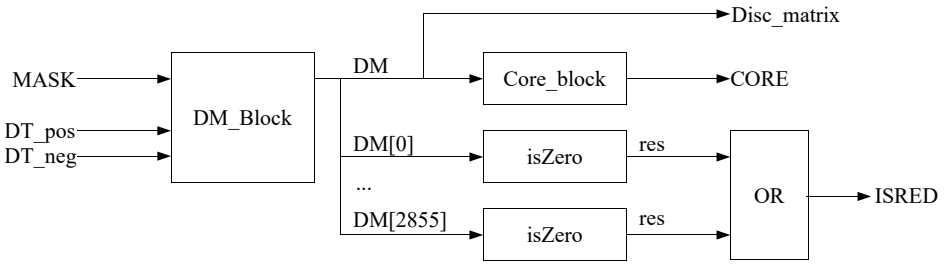
- **DMBlockWrapper** – wrapper dla komponentów używanych do obliczania rdzenia i reduktu. Składa się z bloków *DM_block*, *Core_block*, 2856 wygenerowanych komponentów *isZero* i łączy je ze sobą. Na podstawie wyliczonych danych poszukiwany jest redukt, co zostało przedstawione jako funkcja *isReduct()* linii 1.10 algorytmu 2.1 oraz linii 2.14 algorytmu 2.2. Gdy zostanie znaleziony redukt, wysyła on do *freq_wrapper* 1-bitowy sygnał '0'. Komponent *DMBlockWrapper* zawiera również pozytywne i negatywne zbiory danych opisane w 2.1.3.2, które są przekazywane do modułu *DM_block* z użyciem sygnałów *DT_pos* i *DT_neg*. Ten blok ma jeden port wejściowy:

  - **MASK** – kandydat na redukt dostarczony z komponentu *nios/Zynq*.

  Ten blok posiada trzy porty wyjściowe:

  - **Disc_matrix** - tablica zawierająca obliczoną macierz rozróżnialności,

- **CORE** – wysyła obliczony rdzeń do *freq_wrapper* gdzie następnie jest przekazywany do komponentu *nios/Zynq*,
- **ISRED** – wysyła 1-bitowy sygnał, który wskazuje, że wektor podany na wejściu MASK jest reduktem (lub nie) do bloku *freq_wrapper* gdzie następnie jest przekazywany do komponentu *nios/Zynq*.



Rysunek 2.4: Schemat bloku DMBlockWrapper na podstawie Choromański i in. (2020)

Połączenie między komponentami wewnątrz wrappera pokazano na rysunku 2.4. Bloki, które tworzą komponent *DMBlockWrapper*, są opisane poniżej:

- **DM_block** – komponent tworzący macierz rozróżnialności na podstawie danych z wygenerowanych 2856 modułów *DM_comp*. Działanie tego bloku jest przedstawione jako funkcja *computeDiscMatrix()*, która znajduje się w algorytmie 2.1 w linii 1.3 oraz w algorytmie 2.2 w linii 2.3. Ten blok ma trzy porty wejściowe:
  - **MASK** – kandydat na redukt dostarczony z komponentu *nios/Zynq*,
  - **DT_pos** – dostarcza 56-elementową tablicę zawierającą 16-bitowe wyrazy binarne złożone z atrybutów pochodzące z pozytywnego zestawu danych z komponentu *DMBlockWrapper*,
  - **DT_neg** – - dostarcza 51-elementową tablicę zawierającą 16-bitowe wyrazy binarne złożone z atrybutów pochodzące z negatywnego zestawu danych z komponentu *DMBlockWrapper*.

  Ten blok ma jeden port wyjściowy:

  - **DM** – wysyła tablicę złożoną z 2856 12-bitowych słów (wygenerowana macierz rozróżnialności) do komponentu *DMBlockWrapper*.

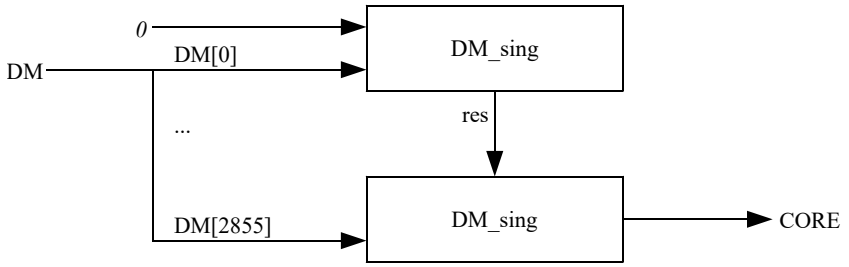
- **Core_block** – komponent, w którym obliczany jest rdzeń. Ten blok składa się z 2856 wygenerowanych komponentów *DM_sing*, które tworzą kaskadę elementów. Połączenia między blokami pokazano na rysunku 2.5. Działanie tego bloku jest przedstawione jako funkcja *computeCore()*, która znajduje się w algoryt-

mie 2.1 w linii 1.4 oraz w algorytmie 2.2 w linii 2.4. Ten blok ma jeden port wejściowy:

- **DM** – 2856-elementowa tablica 12-bitowych słów, która zawiera całą macierz rozróżnialności dostarczoną z *DMBlockWrapper*.

Ten blok ma jeden port wyjściowy:

- **CORE** – wysyła do *DMBlockWrapper* rdzeń obliczony na podstawie dostarczonej macierzy rozróżnialności.



Rysunek 2.5: Schemat połączeń wewnątrz Core_block na podstawie Choromański i in. (2020)

- **isZero** – komponent wyszukujący ’0’ w danym 12-bitowym słowie wykonując operację OR na każdym pojedynczym bicie z podanego wektora. wynik jest przekazywany do *DM_sing*, gdzie następuje sprawdzenie, czy nowy wpis macierzy rozróżnialności powinien być równy „0”, czy nie. Ten blok ma jeden port wejściowy:
  - **inval** - 12-bitowe słowo będące wynikiem porównania dwóch wartości z pozytywnego i negatywnego zbioru danych.

  Ten blok ma jeden port wyjściowy:
  - **res** - pojedynczy bit, który ma wartość ’0’ tylko wtedy, gdy sygnał podany na wejściu jest równy ’0’.

Jak wspomniano wcześniej, *DM_block* składa się z wielu generowanych komponentów *DM_comp* które opisano poniżej:

- **DM_comp** – komponent porównujący wartości danych z pozytywnego oraz negatywnego zestawu danych, określoną przez bit decyzyjny (dodatnie wartości są przypisywane osobom z mikroalbuminurią). Porównania dokonuje się przez wykonanie operacji XOR danych pozytywnych z negatywnymi, a następnie dokonywana jest koniunkcja wyniku z daną maską. Blok *DM_comp* używa komponentu *isZero* do wyszukiwania dowolnego „0” w wyrazie powstałym w wyniku

poprzednich operacji. Wynikiem tych operacji jest pojedynczy wpis do macierzy rozróżnialności. Jeśli blok *isZero* nie znajdzie „0”, wartość wprowadzona do macierzy rozróżnialności jest równa „0”. Ten blok ma trzy porty wejściowe:

- **MASK** - 12-bitowe słowo używane do maskowania atrybutów,
- **DT_pos** – wpis tablicy decyzyjnej ze zbioru danych klasy pozytywnej (słowo 16-bitowe) z komponentu *DMBlockWrapper*,
- **DT_neg** – wpis tablicy decyzyjnej ze zbioru danych klasy negatywnej (słowo 16-bitowe) ze składnika *DMBlockWrapper*.

Ten blok ma jeden port wyjściowy:

- **DM_entry** – wysyła obliczony wpis macierzy rozróżnialności (słowo 12-bitowe) do komponentu *DM_block*.

Jak pokazano na rysunku 2.5, *Core_block* składa się z wielu generowanych komponentów *DM_sing* opisanych poniżej:

- DM_sing –komponent, w którym każdy wpis macierzy rozróżnialności jest sprawdzany, czy jest singletonem (słowo binarne posiada tylko jedną wartość ‘1’), czy nie. Przyjmuje dwie wartości: poprzednią wartość z kaskady i wartość z macierzy rozróżnialności, a następnie sprawdza, czy podana wartość z macierzy rozróżnialności jest singletonem, czy nie. Jeśli wartość z macierzy nim jest, to wykonywana jest operacja OR z wartością kaskadową i wynik jest wypychany na wyjście. W przeciwnym razie wypychana na wyjście jest wartość kaskady. Ten blok ma dwa porty wejściowe:

  - **prev** – dostarcza 12-bitowe słowo z kaskady,
  - **DM** – zapewnia 12-bitowy wpis z macierzy rozróżnialności.

  Ten blok ma jeden port wyjściowy:

  - **res** – wysyła 12-bitowy wynik operacji OR lub wartość kaskadową do *Core_block*.

W systemie znajdują się również drobne bloki, które służą jako kontenery logiczne:

- **rMux** – moduł, w którym wszystkie atrybuty są liczone na potrzeby sortowania,
- **sort** – moduł używany do algorytmu sortowania bitonicznego i odpowiedzialny za porównanie dwóch podanych wartości, schemat tego bloku przedstawia rysunku 2.2,
- **isSingleton** – moduł sprawdzający czy wartość podana na wejściu jest singletonem czy nie.

2.1.3.4 Zużyte zasoby sprzętowe

Projekt dla układu Arria został zbudowany przy użyciu Quartus Prime firmy Altera. Raporty z kompilacji informuje o zużyciu następującej liczby zasobów FPGA:

- wykorzystanie elementów logicznych (w ALM): 15,492 / 176,160 (9 %),
- suma rejestrów: 2763,
- suma pinów: 1/876 (<1 %),
- zużyta pamięć: 8 452 928/23 367 680 (36 %),
- łączna liczba bloków DSP: 3/1090 (<1 %).

Wykorzystanie zasobów FPGA jest niskie. Projekt wymagał mniej niż 10 % elementów logicznych (ALM). Należy zauważyć, że projekt wykorzystuje procesor softcore, który potrzebuje pewnych zasobów (około 3553 ALM zostało wykorzystanych do implementacji procesora Nios II), dlatego zużycie elementów logicznych można zmniejszyć po przejściu na zintegrowany procesor ARM.

Projekt dla układu Zynq US+ został zbudowany przy użyciu Vivado firmy Xilinx. Raporty kompilacyjne informują o wykorzystaniu następujących zasobów FPGA:

- wykorzystanie LUT: 26 930/274 080 (9,83 %),
- LUTRAM: 70/144 000 (0,05 %),
- liczba rejestrów: 3 161 / 548 160 (0,58 %),
- liczba buforów globalnych (BUFG): 1/404 (0,25 %).

Wykorzystanie zasobów FPGA jest bardzo niskie. Projekt przygotowany dla Zynq US+ wymagał mniej niż 10 % tablic przeglądowych (LUT) i mniej niż 1 % dostępnych rejestrów.

2.1.3.5 Implementacja oprogramowania sterującego

Wszystkie komponenty sprzętowe do obliczania macierzy rozróżnialności i rdzenia oraz do sprawdzania kandydatów pod kątem bycia reduktem, zostały napisane w języku VHDL.

Implementację sprzętową wspiera aplikacja napisana w języku C, która na podstawie danych dostarczonych przez sprzęt określa najbardziej prawdopodobnego kandydata do miana reduktu.

Macierz rozróżnialności jest obliczana w *DM_block* który składa się z wielu komparatorów, porównujących wartości dwóch obiektów z tablic decyzyjnych. Tablica decyzyjna następnie jest przekazywana z komponentu *DMBlockWrapper* w którym zadeklarowano zestaw danych zarówno negatywnych, jak i pozytywnych. Wyniki obliczeń są przekazywane z powrotem do *DMBlockWrapper*, gdzie są używane do obliczenia rdzenia dla podanych danych.

Rdzeń jest obliczany w *Core_block*, który składa się z kaskady bloków *isSingleton*. *Core_block* jest układem kombinacyjnym i nie potrzebuje sygnału zegarowego,

więc czas obliczeń zależy tylko od czasu propagacji sygnałów do poszczególnych bloków logicznych wewnątrz układu FPGA.

Obliczony rdzeń jest również używany jako pierwsza maska, która zeruje już użyte atrybuty w macierzy rozróżnialności. Po uprzednim wyzerowaniu, komponent *freq_wrapper* oblicza częstotliwość występowania każdego atrybutu, a następnie sortuje te dane od najczęściej do najrzadziej występujących. Posortowane dane są wysyłane z obliczonym rdzeniem i flagą ISRED (która informuje, czy aktualny zestaw atrybutów przesyłany jako maska jest reduktem, czy też nie) do instancji procesora Nios/Zynq.

W procesorze Nios/Zynq, aplikacja napisana w C na podstawie częstotliwości występowania atrybutów, wybiera najbardziej prawdopodobnego kandydata do bycia reduktem i odsyła go do komponentu *freq_wrapper* jako maskę. Maska, która została wysłana z aplikacji C jest sprawdzana pod kątem reduktu przez FPGA, po czym ponownie wykonywane jest zerowanie i obliczanie częstotliwości.

Nowe częstotliwości ze znacznikiem ISRED są odsyłane z powrotem do procesora Nios/Zynq, który wpisuje kolejnego kandydata. Proces jest powtarzany, aż zostaną znalezione wszystkie minimalne redukty.

2.2 Eksperymenty

W tym podrozdziale opisano badania eksperymentalne dotyczące znalezienia minimalnych reduktów za pomocą podejścia sprzętowego przy użyciu implementacji wybranych algorytmów wyszukiwania wszerz.

Na potrzeby badań eksperymentalnych w języku C napisano aplikację do obliczania rdzenia i minimalnych reduktów. Działa ona w dwóch trybach: samodzielnym i ze wsparciem systemu sprzętowego opisanego w podrozdziale 2.1.3.5. Główną różnicą między tymi dwoma trybami jest przeniesienie logiki odpowiedzialnej za obliczenia rdzenia i reduktów do FPGA (Choromański i in., 2020). W rzeczywistości tryb samodzielny można krótko opisać jako programową symulację algorytmu zaimplementowanego sprzętowo. Aby uzyskać jak najdokładniejszy wynik czasowy, każdy eksperyment wykonano 10, 100, 1000 i 10000 razy. Wszystkie przeprowadzono na zbiorze danych opisanym w podrozdziale 2.1.3.2.

Początkowo testy były wykonywane przy użyciu aplikacji C w trybie samodzielnym, uruchomionej na procesorze Nios II pracującym z zegarem o częstotliwości 50 MHz. Celem tych testów było zbadanie średniego czasu potrzebnego do obliczenia rdzenia i reduktów za pomocą „czystej" aplikacji C przed uruchomieniem jej ze wsparciem układu FPGA.

Celem pierwszego eksperymentu jest sprawdzenie, ile czasu zajmują obliczenia podczas korzystania ze strategii „ślepego" przeszukiwania wszerz. Tabela 2.1 przed-

stawia czas wielokrotnego wyliczania wszystkich minimalnych reduktów dla danego zbioru danych oraz czas potrzebny do pojedynczego wykonania obliczeń.

Tabela 2.1: Wyniki obliczania rdzenia i reduktu z użyciem oprogramowania w C, uruchomionego na procesorze Nios II - BBFS

Liczba wykonań	**Czas [s]**	**Czas [s] potrzebny na 1 wykonanie**
10	0.657	0.0657
100	6.572	0.06572
1000	65.728	0.065728
10000	657.272	0.0657272

Drugi eksperyment sprawdza czas potrzebny do obliczeń przy użyciu strategii wyszukiwania wszerz partej na częstotliwościach występowania atrybutów. Wyniki przedstawione w tabeli 2.2 w porównaniu z poprzednim testem pokazują, że strategia oparta na częstotliwościach jest około 1,38 razy szybsza, a tym samym lepsza.

Tabela 2.2: Wyniki obliczania rdzenia i reduktu z użyciem oprogramowania w C, uruchomionego na procesorze Nios II - FBFS

Liczba wykonań	**Czas [s]**	**Czas [s] potrzebny na 1 wykonanie**
10	0.732	0.0732
100	7.327	0.0732
1000	73.275	0.073275
10000	732.749	0.0732749

Po ustaleniu czasów odniesienia uruchomiono aplikację ze wsparciem stworzonego systemu sprzętowego. Kolejne eksperymenty zostały przeprowadzone dokładnie tak samo, jak dwa poprzednie, ale tym razem obliczenia zostały wykonane przez układ FPGA.

W trzecim eksperymencie zastosowano strategię „ślepego" przeszukiwania wszerz. Wyniki przedstawione w tabeli 2.3 pokazują czasy wielokrotnego i uśrednione czasy pojedynczego wykonania obliczeń.

Celem czwartego eksperymentu jest sprawdzenie, ile czasu zajmuje obliczenie rdzenia i reduktu przy zastosowaniu strategii wyszukiwania szerokości opartej na częstotliwościach. Wyniki przedstawione w tabeli 2.4 pokazują, że ta strategia daje prawie takie same wartości, jak strategia „ślepego" przeszukiwania wszerz.

Tabela 2.3: Wyniki obliczania rdzenia i reduktu przy użyciu FPGA - BBFS

Liczba wykonań	**Czas [s]**	**Czas [s] potrzebny na 1 wykonanie**
10	0.006	0.0006
100	0.059	0.00059
1000	0.590	0.000590
10000	5.898	0.0005898

Tabela 2.4: Wyniki obliczania rdzenia i reduktu przy użyciu FPGA - FBFS

Liczba wykonań	**Czas [s]**	**Czas [s] potrzebny na 1 wykonanie**
10	0.005	0.0005
100	0.048	0.00048
1000	0.482	0.000482
10000	4.817	0.0004817

Wyniki przedstawione w tabelach 2.3 i 2.4 pokazują, że jeśli do rozwiązania problemu zostanie użyty układ kombinacyjny, to nie ma znaczenia, jaki algorytm został użyty, ponieważ wszystko zależy głównie od czasu propagacji sygnału.

W celach testowych aplikacja C w trybie autonomicznym została uruchomiona na komputerze PC z procesorem Intel Core i7-770 @ 3,60 GHz i 32 GB RAM.

Tabela 2.5: Wyniki obliczania rdzenia i reduktu przy użyciu komputera PC - BBFS

Liczba wykonań	**Czas [s]**	**Czas [s] potrzebny na 1 wykonanie**
10	0	0
100	0.015622	0.00015622
1000	0.177181	0.000177181
10000	1.696734	0.0001696734

Jak widać w tabelach 2.5 i 2.6, wyniki są inne niż w przypadku FPGA oraz, że BBFS była nieco szybsza niż FBFS.

Uzyskane wyniki są również około 3,5 razy (w przypadku BBFS) i około 2,7 razy (w przypadku FBFS) lepsze niż wyniki z FPGA.

Tabela 2.6: Wyniki obliczania rdzenia i reduktu przy użyciu komputera PC - FBFS

Liczba wykonań	**Czas [s]**	**Czas [s] potrzebny na 1 wykonanie**
10	0	0
100	0.015652	0.00015652
1000	0.187467	0.000187467
10000	1.880553	0.0001880553

Należy jednak zauważyć, że zegar komputera PC jest $\frac{clk_{PC}}{clk_{FPGA}} = \frac{3600MHz}{50MHz} = 72$ razy szybszy niż źródło zegara FPGA, więc jeśli weźmiemy tę różnicę, wyniki są znacznie lepsze - współczynnik przyśpieszenia wynosi około 21 dla BBFS i około 28 dla FBFS.

Dla celów testowych, ta sama konfiguracja sprzętowa została uruchomiona również na drugim urządzeniu konkurencyjnej firmy - Zynq US+ (Choromański i in., 2019). Kolejne eksperymenty przebiegały identycznie do poprzednich.

Na początku zestawiono obliczenia z użyciem BBFS (tabela 2.7). Porównując wynik z czasami uzyskanymi z Arrii można stwierdzić, że czas realizacji jest znacznie krótszy (około 15,5 razy).

Następnie sprawdzono ile czasu zajmuje obliczenie rdzenia i reduktów przy użyciu FBFS. Wyniki przedstawione w tabeli 2.7 pokazują, że ta strategia daje prawie 1,75 razy wyższe wartości niż BBFS i około 7,2 razy lepsze wyniki niż w przypadku Arrii.

Tabela 2.7: Czas obliczania reduktu [ms] z użyciem Zynq US+

Liczba wykonań	**BBFS**	**FBFS**
10	0.38	0.67
100	3.80	6.69
1000	38.13	66.94
10000	380.10	665.80

Przedstawione wyniki pokazują, że użycie tego samego projektu sprzętu na dwóch różnych urządzeniach może prowadzić do poprawy wydajności. Opisane eksperymenty dowiodły, że możliwe jest osiągnięcie 15,5 razy lepszych czasów (w przypadku BBFS) i 7,2 razy lepszych (w przypadku FBFS). Należy zauważyć, że zegar używany w Zynq jest 2 razy szybszy niż zegar używany w Nios, więc jeśli weź-

miemy tę różnicę pod uwagę, to współczynnik przyśpieszenia wynosi około 7,75 dla BBFS i 3,6 dla FBFS.

Podsumowanie

Zdefiniowano i przeanalizowano dwa algorytmy realizujące strategie przeszukiwania wszerz, oparte na podejściu „ślepym" oraz na częstotliwościach występowania atrybutów.

Zaprezentowano dwie architektury FPGA służące do znajdowania minimalnych reduktów. Na podstawie badań eksperymentalnych przedstawionych w niniejszym rozdziale można sformułować wniosek, że stosowanie układów programowalnych do wspomagania obliczeń reduktów determinuje ogromne korzyści w postaci zredukowania czasu wykonania algorytmu.

W porównaniu do komputera PC, obliczenia robione na układzie FPGA są o wiele wydajniejsze. Największą akceleracje można jednak zauważyć w przypadku FBFS. Dzieje się tak, ponieważ większość bloków w implementacji sprzętowej jest kombinacyjnych, dzięki czemu wyniki uzyskiwane są niemal natychmiast (tylko czas propagacji sygnału opóźnia obliczenia).

Ogromną zaletą FPGA jest możliwość wykorzystania sprzętowego bloku sortowania, dzięki czemu FBFS jest znacznie bardziej wydajny niż BBFS.

Bibliografia

Bakar, A. A., Sulaiman, M. N., Othman, M. i Selamat, M. H. (2002). Propositional satisfiability algorithm to find minimal reducts for data mining. *International Journal of Computer Mathematics*, *79*, 379–389.

Chen, D., Zhao, S., Zhang, L., Yang, Y. i Zhang, X. (2012). Sample pair selection for attribute reduction with rough set. *IEEE Transactions on Knowledge and Data Engineering*, *24*, 2080-2093.

Choromański, M., Grześ, T. i Hońko, P. (2019). Two FPGA devices in the problem of finding minimal reducts. W: K. Saeed, R. Chaki i V. Janev (red.), *Computer information systems and industrial management,* Springer, 410–420.

Choromański, M., Grześ, T. i Hońko, P. (2020). Breadth search strategies for finding minimal reducts: towards hardware implementation. *Neural Computing and Applications*, *32*, 14801–14816.

Czolombitko, M., i Stepaniuk, J. (2016). Attribute Reduction Based on MapReduce Model and Discernibility Measure. W: K. Saeed i W. Homenda (red.),

Computer Information Systems and Industrial Management: 15th IFIP TC8 International Conference, 9842, Springer, 55–66.

Degang, C., Changzhong, W. i Qinghua, H. (2007). A new approach to attribute reduction of consistent and inconsistent covering decision systems with covering rough sets. *Information Sciences*, *177*, 3500–3518.

Dong, Z., Sun, M. i Yang, Y. (2016). Fast algorithms of attribute reduction for covering decision systems with minimal elements in discernibility matrix. *International Journal of Machine Learning and Cybernetics*, *7*, 297–310.

Grześ, T., Kopczyński, M. i Stepaniuk, J. (2013). FPGA in rough set based core and reduct computation. W: P. Lingras, M. Wolski, C. Cornelis, S. Mitra i P. Wasilewski (red.), *Rough sets and knowledge technology,* Springer, 263–270.

Grzymala-Busse, J. (1991). An algorithm for computing a single covering. W: J. Grzymala-Busse (red.), *Managing uncertainty in expert systems,* Kluwer Academic Publishers.

Hu, Q., Yu, D., Liu, J. i Wu, C. (2008). Neighborhood rough set based heterogeneous feature subset selection. *Information Sciences*, *178*, 3577-3594.

Hu, Q., Yu, D., Xie, Z. i Liu, J. (2006). Fuzzy probabilistic approximation spaces and their information measures. *Transactions on Fuzzy Systems*, *14*, 191-201.

Hu, X., i Cercone, N. (1995). Learning in relational databases: A rough set approach. *Computational Intelligence*, *11*, 323-338.

Jensen, R., Shen, Q. i Tuson, A. (2005). Finding rough set reducts with SAT. W: *Rough sets, fuzzy sets, data mining, and granular computing, proceedings of 10th international conference,* Springer, 194–203.

Jia, X., Liao, W., Tang, Z. i Shang, L. (2013). Minimum cost attribute reduction in decision-theoretic rough set models. *Information Sciences*, *219*, 151-167.

Jiang, D.-L., Zhao, B., Wang, Y., Li, H.-W. i Yang, G.-Q. (2013). FPGA low level data mining based on cluster and rough set. *Guangxue Jingmi Gongcheng/Optics and Precision Engineering*, *21*, 233-238.

Jiang, Y., i Yu, Y. (2016). Minimal attribute reduction with rough set based on compactness discernibility information tree. *Soft Computing*, *20*, 2233–2243.

Jing, F., Yunliang, J. i Yong, L. (2017). Quick attribute reduction with generalized indiscernibility models. *Information Sciences*, *397-398*, 15 - 36.

Kanasugi, A., i Matsumoto, M. (2007). Design and implementation of rough rules generation from logical rules on FPGA Board. W: M. Kryszkiewicz, J. F. Peters, H. Rybinski i A. Skowron (red.), *Rough sets and intelligent systems paradigms,* Springer, 594–602.

Kanasugi, A., i Yokoyama, A. (2001). A basic design for rough set processor. W: *Proceedings of the 15th annual conference of japanese society for artificial intelligence,* 406–412.

Kopczynski, M., Grzes, T. i Stepaniuk, J. (2014). FPGA in rough-granular computing: Reduct generation. W: *Proceedings of the 2014 IEEE/WIC/ACM In-*

ternational Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), IEEE, 364–370.

Kryszkiewicz, M. (1998). Rough set approach to incomplete information systems. *Inforamtion Sciences*, *112*, 39-49.

Kryszkiewicz, M. (2001). Comparative study of alternative type of knowledge reduction in inconsistent systems. *International Journal of Intelligent Systems*, *16*, 105-120.

Lewis, T., Perkowski, M. i Jozwiak, L. (1999). Learning in hardware: architecture and implementation of an FPGA-based rough set machine. W: *Informatics: Theory and practice for the new millennium: Proceedings of 25th euromicro conference,* 1, IEEE, 326-334.

Li, F., i Yang, J. (2016). A new approach to attribute reduction of decision information systems. W: Y. Qin, L. Jia, J. Feng, M. An i L. Diao (red.), *Proceedings of the 2015 international conference on electrical and information technologies for rail transportation: Transportation,* Springer, 557-564.

Liang, J., Mi, J., Wei, W. i Wang, F. (2013). An accelerator for attribute reduction based on perspective of objects and attributes. *Knowledge-Based Systems*, *44*, 90-100.

Liang, J., i Xu, Z. (2002). The algorithm on knowledge reduction in incomplete information systems. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *10*, 95-103.

Liu, G., Hua, Z. i Chen, Z. (2017). A general reduction algorithm for relation decision systems and its applications. *Knowledge-Based Systems*, *119*, 87-93.

Pawlak, Z. (1991). *Rough sets. theoretical aspects of reasoning about data*. Kluwer Academic, Dordrecht.

Pawlak, Z. (2004). Elementary rough set granules: Toward a rough set processor. W: S. K. Pal, L. Polkowski i A. Skowron (red.), *Rough-neural computing: Techniques for computing with words,* Springer, 5–14.

Pawlak, Z., i Skowron, A. (2007). Rudiments of rough sets. *Information Sciences*, *177*, 3–27.

Qian, Y., Liang, J., Pedrycz, W. i Dang, C. (2010). Positive approximation: An accelerator for attribute reduction in rough set theory. *Artificial Intelligence*, *174*, 597-618.

Skahill, K. (2001). *Język VHDL.* Warszawa: Wydawnictwo Naukowo-Techniczne.

Skowron, A., i Rauszer, C. (1992). The discernibility matrices and functions in information systems. W: *Intelligent decision support,* Springer, 331–362.

Ślęzak, D. (2002). Approximate entropy reducts. *Fundamamenta Informaticae*, *53*, 365–390.

Stepaniuk, J. (1999). Rough set data mining of diabetes mellitus data. *Lecture Notes in Computer Science*, *1906*, 457—465.

Stepaniuk, J. (2008). *Rough-granular computing in knowledge discovery and data mining*. Berlin Heidelberg: Springer.

Su, Y., i Guo, J. (2017). A novel strategy for minimum attribute reduction based on rough set theory and fish swarm algorithm. *Computational Intelligence and Neuroscience*, *2017*, 6573623:1–6573623:7.

Sun, G., Qi, X. i Zhang, Y. (2011). A FPGA-based implementation of Rough Set Theory. W: *Proceedings of the 2011 chinese control and decision conference,* 2561-2564.

Sun, G., Wang, H., Lu, J. i He, X. (2013). A FPGA-based discretization algorithm of continuous attributes in rough set. W: Y.-H. Kim i P. Yarlagadda (red.), 278-280, 1167-1173.

Swiniarski, R. (2001). Rough sets methods in feature reduction and classification. *International Journal of Applied Mathematics and Computer Science*, *11*, 565–582.

Swiniarski, R. W., i Skowron, A. (2003). Rough set methods in feature selection and recognition. *Pattern Recognition Letters*, *24*, 833-849.

Teng, S.-H., Lu, M., Yang, A.-F., Zhang, J., Nian, Y. i He, M. (2016). Efficient attribute reduction from the viewpoint of discernibility. *Information Sciences*, *326*, 297-314.

Thi, V. D., i Giang, N. L. (2013). A method for extracting knowledge from decision tables in terms of functional dependencies. *Cybernetics and Information Technologies*, *13*, 73-82.

Tiwari, K., i Kothari, A. (2015). Design and implementation of rough set coprocessor on FPGA. *International Journal of Innovative Computing, Information and Control*, *11*, 641-656.

Tiwari, K., i Kothari, A. (2016). Design of intelligent system for medical applications using rough set theory. *International Journal of Data Mining, Modelling and Management*, *8*, 279-301.

Tiwari, K. S., i Kothari, A. G. (2014). Design and implementation of rough set algorithms on FPGA: A survey. *International Journal of Advanced Research in Artificial Intelligence*, *3*, 14-23.

Wang, C., He, Q., Chen, D. i Hu, Q. (2014). A novel method for attribute reduction of covering decision systems. *Information Sciences*, *254*, 181-196.

Wang, X., Yang, J., Peng, N. i Teng, X. (2005). Finding minimal rough set reducts with particle swarm optimization. W: D. Ślęzak, G. Wang, M. Szczuka, I. Düntsch i Y. Yao (red.), *Rough sets, fuzzy sets, data mining, and granular computing,* Springer, 451–460.

Wang, X., Yang, J., Teng, X., Xia, W. i Jensen, R. (2007). Feature selection based on rough sets and particle swarm optimization. *Pattern Recognition Letters*, *28*, 459–471.

Wei, W., Liang, J., Qian, Y., Wang, F. i Dang, C. (2010). Comparative study of decision performance of decision tables induced by attribute reductions. *International Journal of General Systems*, *39*, 813-838.

Wei, W., Liang, J., Wang, J. i Qian, Y. (2013). Decision-relative discernibility

matrices in the sense of entropies. *International Journal of General Systems*, *42*, 721-738.

Wroblewski, J. (1995). Finding minimal reducts using genetic algorithms. W: *Proceedings of the second annual join conf. on information sciences,* 186—189.

Xie, J., Shen, X., Liu, H. i Xu, X. (2013). Research on an incremental attribute reduction based on relative positive region. *Journal of Computational Information Systems*, *9*, 6621-6628.

Xu, N., Liu, Y. i Zhou, R. (2008). A tentative approach to minimal reducts by combining several algorithms. W: *Advanced intelligent computing theories and applications,* Springer, 118–124.

Yao, Y., i Zhao, Y. (2008). Attribute reduction in decision-theoretic rough set models. *Information Sciences*, *178*, 3356-3373.

Ye, D., i Chen, Z. (2002). A new discernibility matrix and the computation of a core. *Acta Electronica Sinica*, *30*, 1086-1088.

Zhang, W.-X., Mi, J.-S. i Wu, W.-Z. (2003). Approaches to knowledge reductions in inconsistent systems. *International Journal of Intelligent Systems*, *18*, 989-1000.

Zhang, X., Mei, C., Chen, D. i Li, J. (2013). Multi-confidence rule acquisition oriented attribute reduction of covering decision systems via combinatorial optimization. *Knowledge-Based Systems*, *50*, 187-197.

Rozdział 3

WPŁYW DYSKRETYZACJI DANYCH NA SKUTECZNOŚĆ DZIAŁANIA ALGORYTMÓW SELEKCJI NEGATYWNEJ

Izabela Kartowicz-Stolarska*

Streszczenie Algorytmy selekcji negatywnej (ang. *negative selection algorithm*, NSA) są jednym z podejść sztucznych układów immunologicznych, które wzorują się na mechanizmach obronnych zachodzących w naturalnych organizmach. Przez ostatnie lata algorytmy te znalazły szerokie zastosowanie w ochronie danych oraz procesach związanych z bezpieczeństwem sieci komputerowych. Jak pokazują dotychczasowe badania, NSA z powodzeniem może być również używany jako klasyfikator, zarówno binarny, jak i dwuklasowy i wieloklasowy. Niniejsza praca jest krótkim wprowadzeniem do teorii i zastosowania sztucznych systemów immunologicznych w kontekście dyskretyzacji danych. W rozdziale przedstawiono przegląd i porównanie wybranych rozwiązań na przykładzie algorytmu selekcji negatywnej. Omówiono tu ogólne zasady konstruowania algorytmów selekcji negatywnej, jak również szczegóły implementacji podanych rozwiązań. Następnie zbadano wpływ dyskretyzacji atrybutów na działanie poszczególnych implementacji. Z uwagi na możliwość przedstawienia wyników eksperymentów z danymi przed i po dyskretyzacji, w przeglądzie uwzględniono algorytmy NSA, w których przeciwciała reprezentowane są jako liczby rzeczywiste. Badania zostały przeprowadzone na kilku zbiorach danych. Do eksperymentów wykorzystano różne algorytmy dyskretyzacji, w tym metody oparte na teorii zbiorów przybliżonych.

Słowa kluczowe: sztuczne systemy immunologiczne, selekcja negatywna, klasyfikacja, dyskretyzacja atrybutów

Wprowadzenie

Mechanizmy działania naturalnego systemu immunologicznego (ang. *natural immune system*, NIS) kręgowców, a w szczególności jego adaptacja do zmiennego otoczenia oraz możliwość rozpoznania patogenów (m.in. wirusów, bakterii czy pierwot-

* Wydział Informatyki, Politechnika Białostocka, Wiejska 45A, 15-351 Białystok, i.stolarska@pb.edu.pl

DOI 10.24427/978-83-66391-58-1_3

niaków), stały się inspiracją dla naukowców do opracowania i rozwoju sztucznych systemów immunologicznych.

Układ odpornościowy jest rozproszony i unikatowy dla każdego osobnika. Ma zdolność uczenia się i zapamiętywania patogenów, z którymi zetknął się w czasie swojego funkcjonowania, a do wykrycia ich wystarczy mu tylko znajomość struktur komórek własnych (Wierzchoń, 2001). Za rozpoznanie komórek obcych w organizmie i ich eliminację odpowiadają produkowane w szpiku kostnym krwinki białe (limfocyty). Limfocyty można podzielić na limfocyty typu T i typu B (Lydyard, Whelan i Fanger, 2001).

Limfocyty typu T dojrzewają w grasicy, gdzie przechodzą proces nauki rozpoznawania struktur komórek własnych. Dopiero tak wykształcone krwinki białe typu T trafiają do krwi obwodowej i narządów limfatycznych, gdzie są odpowiedzialne za rozpoznanie patogenów i podejmowanie akcji obronnej organizmu. Proces nauki rozpoznawania komórek własnych jest w immunologii nazywany selekcją negatywną. W dużym uproszczeniu polega on na eliminacji niedojrzałych limfocytów typu T, które rozpoznają komórki własne jako obce, stanowiące zagrożenie dla organizmu.

Limfocyty typu B odpowiadają za zapamiętanie i niszczenie patogenów. Znany organizmowi patogen nazywany jest antygenem. Na powierzchni limfocytów typu B znajduje się około 100 tysięcy receptorów. Receptory i antygeny posiadają fragmenty, które tworzą wiązanie: paratop w receptorze i epitop w antygenie. Pojedynczy limfocyt posiada receptory z paratopami umiejącymi wiązać tylko jeden rodzaj epitopów (unikatowa swoistość). Istniejące limfocyty ulegają licznym podziałom z uwzględnieniem mutacji. W procesach tych receptory komórek potomnych nieznacznie się zmieniają, umożliwiając tym samym skuteczniejsze od komórki rodzica wiązanie patogenów. Dzięki temu odpowiedź immunologiczna (reakcja organizmu na kontakt z antygenem) jest precyzyjna.

Limfocyty są aktywowane, gdy ich receptory rozpoznają antygen. W efekcie powstaje wiele klonów o identycznej z komórką oryginalną swoistości (Lasek, Lasek i Pęczkowski, 2013). Dzięki temu dochodzi do wytworzenia swoistej odporności organizmu na dany antygen. Opisany powyżej proces nazywany jest w immunologii selekcją klonalną (ang. *clonal selection*). W procesie tym intensywnemu klonowaniu ulegają limfocyty, których receptory są najlepiej dopasowane do antygenu. Klony z receptorami słabo wiążącymi antygen są usuwane z organizmu. Natomiast te, które mają silne powiązania paratop-epitop pozostają, przekształcając się w komórki plazmatyczne lub pamięciowe. W ten sposób zostaje zapewniona pamięć immunologiczna i szybsza reakcja obronna organizmu przy ponownym kontakcie z antygenem.

Wymienione wyżej mechanizmy, jak również inne teorie z dziedziny immunologii (np. teoria niebezpieczeństwa (Matzinger, 1994)) zainspirowały badaczy do przełożenia własności naturalnego układu immunologicznego na pole informatyki i inżynierii matematycznej.

Jedną z pierwszych prac przekładających działanie mechanizmów NIS na działanie sieci komputerowych była teoria sieci idiotypowej zaproponowana przez Jerne

(1973). Artykuł ten stał się podstawą do powstania grupy rozwiązań nazwanych sztucznymi sieciami immunologicznymi. Przełomową pracą dotyczącą modelowania sieci idiotypowych był artykuł Farmera, Packarda i Perelsona z 1986 roku "The Immune System, Adaptation, and Machine Learning", w którym zaproponowano zastosowanie rozwiązań naśladujących działanie układów odpornościowych do ochrony sieci (Wierzchoń, 2001). W kolejnych latach pojawiły się liczne prace dotyczące implementacji algorytmów symulujących działanie naturalnych systemów immunologicznych.

Na podstawie literatury można wnioskować, że wśród badaczy największe zainteresowanie, oprócz sztucznych sieci immunologicznych, znalazły algorytmy selekcji negatywnej i selekcji klonalnej. W przypadku algorytmów selekcji klonalnej najważniejszą cechą jest dopasowanie komórki limfocytów B do antygenów. Ogólna idea ich działania polega na wybraniu najlepiej dopasowanych przeciwciał do antygenów i ich sklonowaniu. W algorytmie uwzględniono proces mutacji przeciwciał, w taki sposób, aby nowe przeciwciała odznaczały się lepszym dopasowaniem do antygenu, niż przed mutacją. Algorytmy te znalazły zastosowanie głównie w zagadnieniach optymalizacji i rozpoznawania wzorców. Z kolei algorytmy selekcji negatywnej znalazły szerokie zastosowanie przy ochronie danych i procesów realizowanych w sieci (Hofmeyr, 1999) oraz w wykrywaniu nieprawidłowości takich, jak wirusy komputerowe bądź zakłócenia w szeregach czasowych (Dasgupta i Forrest, 1995). Z powodzeniem mogą być również używane do zagadnień klasyfikacji (Esponda, Forrest i Helman, 2003). W ostatnich latach popularne są szczególnie rozwiązania hybrydowe, łączące różne dziedziny z zakresu informatyki, np. wspomagające wspomnianą wyżej klasyfikację obiektów (Hońko, 2018) lub optymalizację funkcji multimodalnych (Lucińska, 2010), a nawet symulujące układ odpornościowy okrętu (Praczyk, 2010).

Dyskretyzacja jest jednym z istotnych zadań wykonywanych na potrzeby eksploracji danych (García, Luengo, Sáez, López i Herrera, 2013). Co prawda, jest ona wykorzystywana przy wykrywaniu anomalii, ale w innych obszarach niż algorytmy immunologiczne. W pracy ”Continuous Features Discretization for Anomaly Intrusion Detectors Generation” zaproponowano zastosowanie dyskretyzacji w algorytmach genetycznych do generowania detektorów wykrywających nieprawidłowości w sieci. W związku z tym, że dane wykorzystywane w systemach do wykrywania włamań są wielowymiarowe, dyskretyzacja jest wskazana jako wstępny etap przetworzenia zbiorów (Aziz, Azar, Hassanien i Hanafy, 2014) i wyostrzenia pewnych różnic między obiektami.

Celem poniższej pracy jest wstępne zbadanie, w jaki sposób dyskretyzacja cech wpływa na klasyfikację algorytmów immunologicznych na przykładzie wybranych rozwiązań. Aby wyjaśnić tę kwestię, w dalszej części pracy szczegółowo omówiono trzy różne implementacje algorytmu symulującego działanie mechanizmu selekcji negatywnej. Następnie porównano wyniki dla danych zdyskretyzowanych za po-

mocą różnych algorytmów. Scenariusz eksperymentów i jego wyniki opisano w rozdziale 3.2, w ostatniej części zawarto podsumowanie.

3.1 Algorytm selekcji negatywnej

Algorytm selekcji negatywnej (ang. *negative selection algorithm*, NSA) wzoruje się na mechanizmie usuwania limfocytów typu T, które rozpoznały własne struktury jako patogenne. Po raz pierwszy został wdrożony przez Forrest, Perelson, Allen i Cherukuri (1994). Przez ostatnie 25 lat algorytm ten był wielokrotnie modyfikowany, ale główne cechy zaproponowane w oryginalnym rozwiązaniu pozostały bez zmian.

Algorytm selekcji negatywnej składa się z dwóch podstawowych kroków:

- generowanie detektorów (przeciwciał) i ich cenzurowanie,
- monitorowanie danych.

Na wejściu algorytm generowania detektorów otrzymuje zbiór z wzorcami własnych komórek. Następnie losowo generuje zbiór detektorów reprezentujących przeciwciała. Podczas cenzurowania usuwane są te detektory, które rozpoznają własne struktury. W efekcie na wyjściu otrzymuje się zbiór tylko tych detektorów, które nie rozpoznały własnych komórek jako obcych. W uproszczeniu algorytm można opisać za pomocą pseudokodu algorytm 3.1.

Algorytm 3.1 Generowanie i cenzurowanie detektorów

INPUT:
S - zbiór wzorców własnych komórek
OUTPUT:
D - zbiór detektorów komórek obcych

```
1: D ← losowo wygeneruj detektory
2: for d ∈ D do
3:     if d rozpoznaje komórkę własną z S then
4:         D ← D \ {d}
5:     end if
6: end for
7: return D
```

Kolejne kroki algorytmu selekcji negatywnej służą do rozpoznania komórek obcych w monitorowanym zbiorze danych. Proces monitorowania przedstawiony został za pomocą pseudokodu algorytm 3.2.

Algorytm 3.2 na wejściu otrzymuje wzorce wszystkich komórek: zarówno własnych, jak i obcych oraz zbiór wygenerowanych detektorów. Na podstawie tych da-

Algorytm 3.2 Monitorowanie danych

INPUT:
D - zbiór detektorów
OUTPUT:
S_{self} - zbiór komórek rozpoznanych jako własne
$S_{nonself}$ - zbiór komórek rozpoznanych jako obce

1: $S_{self} \leftarrow \emptyset$
2: $S_{nonself} \leftarrow \emptyset$
3: **for** $s \in S$ **do**
4: **if** $\exists_{d \in D} match(d, s)$ **then**
5: $S_{nonself} \leftarrow S_{nonself} \cup \{s\}$
6: **else**
7: $S_{self} \leftarrow S_{self} \cup \{s\}$
8: **end if**
9: **end for**

nych oraz reguły dopasowania (ang. *matching rule*) struktury komórki rozpoznawane są jako własne lub obce.

Reguła dopasowania jest wykorzystywana zarówno podczas generowania detektorów, jak i w fazie wykrywania anomalii. Można ją zdefiniować jako odległość między dwoma punktami: detektorem, a komórką danych. Co ciekawe dwa punkty nie muszą być dokładnie takie same, aby zostały uznane za zgodne. Jest to możliwe dzięki wprowadzeniu do algorytmu progu dopasowania (Ji i Dasgupta, 2007).

Wybór reguły dopasowania lub progu w regule dopasowania jest specyficzny dla aplikacji i zależny od reprezentacji (ang. *data representation*).

Reprezentacja danych jest jedną z podstawowych różnic w implementacjach algorytmu selekcji negatywnej. Wyróżnia się dwie główne reprezentacje: niskiego i wysokiego poziomu. Reprezentacja niskiego poziomu to ciągi znaków (ang. *string representation*), do których zaliczana jest również reprezentacja binarna. Z kolei do reprezentacji wysokiego poziomu zaliczają się wektory wartości liczb rzeczywistych (ang. *real-value representation*). Reprezentacja wektorów liczb rzeczywistych została wprowadzona, by przyśpieszyć proces generowania detektorów. Ponadto, dzięki wykorzystaniu takiej reprezentacji nie są tracone informacje o poszczególnych cechach zbioru (Gonzalez i Dasgupta, 2003).

Z uwagi na fakt przeprowadzenia eksperymentów w kontekście dyskretyzacji danych, w dalszej części pracy skoncentrowano się na omówieniu wybranych algorytmów selekcji negatywnej w reprezentacji liczb rzeczywistych. Porównane zostały algorytmy:

- Real-Valued Negative Selection (Gonzalez i Dasgupta, 2003),
- V-detector (Ji i Dasgupta, 2004),
- algorytm detektorów RST (Chmielewski, 2017).

3.1.1 Algorytm Real-Valued Negative Selection (RNS)

Algorytm RNS został po raz pierwszy zaproponowany w 2002 roku przez Gonzaleza. Głównym założeniem, jakim odróżnia się on od standardowego algorytmu selekcji negatywnej jest reprezentacja danych osadzona w przestrzeni liczb rzeczywistych. W algorytmie tym przestrzeń komórek własnych i obcych reprezentowana jest jako punkty w przestrzeni N-wymiarowej, których współrzędne są wartościami liczb rzeczywistych z zakresu [0, 1]. Detektor (przeciwciało) jest definiowany jako N-wymiarowa hipersfera, której środek określa N-wymiarowy wektor, a promień jest stały i wyraża się liczbą rzeczywistą.

W procesie generowania detektorów (algorytm 3.3) w pierwszym kroku losowanych jest T_{max} detektorów o zadanym promieniu l (funkcja $GenerateRandomDetector(l)$). Następnie dla każdego wylosowanego detektora wybieranych zostaje k- najbliższych sąsiadów spośród komórek własnych. Za ten krok w pseudokodzie odpowiada funkcja $GetKNearestCells(d,S)$, gdzie d to wylosowany detektor, a S to zbiór komórek własnych. Dalej pobierany jest środkowy element spośród posortowanych według odległości od detektora sąsiadów (funkcja $GetMedianNearestCell$). A następnie za pomocą funkcji odległości euklidesowej sprawdzana jest odległość między detektorem, a pobraną komórką. Jeśli w badanym obszarze znajdzie się więcej, niż połowa z najbliższych komórek, uznaje się, że detektor je rozpoznaje i w związku z tym detektor zostaje przesunięty na większą odległość. Istotnym elementem przy wyliczaniu wartości powyższej odległości jest funkcja $\mu_d(x)$, która określa stopień nachodzenia detektora na komórkę (własną lub obcą). Funkcja zdefiniowana jest jako:

$$\mu_d(x) = e^{\frac{-\|d-x\|^2}{2r^2}} \tag{3.1}$$

gdzie:

- $\|.\|$ - norma euklidesowa,
- r - promień detektora,
- d - współrzędne środka detektora,
- x - współrzędne komórki.

Proces przesuwania detektorów jest powtarzany, aż mniej niż połowa sąsiadów jest wykrywana lub przekroczono liczbę założonych iteracji algorytmu. Powyższe warunki są niewystarczające, aby algorytm był zbieżny. Początkowe tempo adaptacji oraz tempo zanikania adaptacji powodują przesuwanie detektora o coraz mniejsze odległości, zapewniając zbieżność algorytmu.

Może się zdarzyć, że wylosowany detektor będzie otoczony przez komórki własne w taki sposób, że przesunięcie go w dowolnym kierunku nie spowoduje zmniejszenia wykrywalności komórek własnych. Aby zapobiec takiej sytuacji, z każdą iteracją detektory starzeją się (zmienna age_d) i są usuwane, gdy osiągną założoną dojrzałość, a wciąż będę wykrywały zbyt wiele komórek własnych.

Algorytm 3.3 RNS - generowanie detektorów

INPUT:

l - promień komórki
S - zbiór komórek własnych
r - promień detektora
η_0 - początkowe tempo adaptacji
τ - tempo zanikania adaptacji
t - wiek, w którym detektor uznawany jest za dojrzały
T_{max} - liczba detektorów do wygenerowania
I_{max} - liczba iteracji

OUTPUT:

D - zbiór detektorów

1: $D \leftarrow \{GenerateRandomDetector_i(l) | i \in 1 \rightarrow T_{max}\}$
2: $i \leftarrow 0$
3: **while** $i < I_{max}$ **do**
4: $\quad \eta \leftarrow \eta_0 * e^{-i/\tau}$
5: $\quad$ **for** $d \in D$ **do**
6: $\quad\quad NearCells \leftarrow GetKNearestCells(d, S)$
7: $\quad\quad NearSelf \leftarrow GetMedianNearestCell(NearCells)$
8: $\quad\quad$ **if** $distance(d, NearestSelf) < r$ **then**
9: $\quad\quad\quad dir \leftarrow \frac{\sum_{c \in NearCells}(d-c)}{|NearCells|}$
10: $\quad\quad\quad$ **if** $age_d > t$ **then** $\quad \triangleright\ age_d$ – wiek detektora d
11: $\quad\quad\quad\quad d \leftarrow GenerateRandomDetector(l)$
12: $\quad\quad\quad$ **else**
13: $\quad\quad\quad\quad age_d \leftarrow age_d + 1$
14: $\quad\quad\quad\quad d \leftarrow d + \eta * dir$
15: $\quad\quad\quad$ **end if**
16: $\quad\quad$ **else**
17: $\quad\quad\quad age_d \leftarrow 0$
18: $\quad\quad\quad dir \leftarrow \frac{\sum_{d' \in D} \mu_d(d') * (d - d')}{\sum_{d' \in D} \mu_d(d')}$
19: $\quad\quad\quad d \leftarrow d + \eta * dir$
20: $\quad\quad$ **end if**
21: $\quad$ **end for**
22: $\quad i \leftarrow i + 1$
23: **end while**

Warunek stopu algorytmu 3.3 określany jest przez liczbę iteracji. Dzięki temu złożoność czasowa algorytmu wyraża się jako:

$$O(I_{max} * |D| * (|D| + |S|)) \tag{3.2}$$

W procesie monitorowania danych (algorytm 3.4) reguła dopasowania wyrażana jest za pomocą funkcji odległości euklidesowej pomiędzy detektorem, a komórką. Komórka, która znajdzie się w promieniu detektora, uznawana jest za komórkę obcą.

Algorytm 3.4 RNS - monitorowanie

INPUT:
D - zbiór detektorów
S - zbiór monitorowanych komórek
OUTPUT:
S_{self} - zbiór komórek rozpoznanych jako własne
$S_{nonself}$ - zbiór komórek rozpoznanych jako obce

1: $S_{self} \leftarrow \emptyset$
2: $S_{nonself} \leftarrow \emptyset$
3: **for** $s \in S$ **do**
4: **if** $\exists_{d \in D} match(d, s)$ **then**
5: $S_{nonself} \leftarrow S_{nonself} \cup \{s\}$
6: **else**
7: $S_{self} \leftarrow S_{self} \cup \{s\}$
8: **end if**
9: **end for**

3.1.2 Algorytm V-detector

Zaproponowany w 2004 algorytm jest modyfikacją algorytmu RNS. W odróżnieniu od powyższego rozwiązania, które bazuje na detektorach o stałej długości promienia, algorytm V-detector umożliwia modyfikację długości promienia danego detektora w procesie generowania detektorów. Komórki własne i obce nie są punktami, ale hipersferami w N-wymiarowej przestrzeni.

W pierwszej kolejności losowany jest jeden detektor. Następnie sprawdzane jest, czy wylosowany detektor znajduje się w przestrzeni monitorowanej już przez inne, istniejące detektory. Jeśli tak, to losowany jest nowy detektor. Promień detektora ustawiany jest jako odległość do najbliższej komórki własnej. Proces jest powtarzany, aż zostanie osiągnięta założona liczba detektorów (algorytm 3.5).

Algorytm zatrzyma się, jeżeli zostanie spełniony jeden z trzech warunków:

- wygenerowana zostanie założona liczba detektorów (T_{max}),
- osiągnięty zostanie założony poziom pokrycia przestrzeni przez detektory (c_0),
- stwierdzone zostanie zbyt duże pokrycie przestrzeni przez komórki własne (maximum self coverage).

Algorytm 3.5 V-detector - generowanie detektorów

INPUT:
l - promień komórki
S - zbiór komórek własnych
R_s - promień komórki własnej
c_0 - oczekiwane pokrycie detektorami
T_{max} - maksymalna liczba detektorów
OUTPUT:
D - zbiór detektorów

1: $D \leftarrow \emptyset$
2: $T \leftarrow 0$
3: **repeat**
4: $t \leftarrow 0$
5: **repeat**
6: $c \leftarrow GenerateRandomDetector(l)$
7: $FallsInsideOtherDetector \leftarrow false$
8: **for** $d \in D$ **do**
9: **if** $distance(d,c) < r(d)$ **then**
10: $t \leftarrow t+1$
11: **if** $t >= \frac{1}{1-c_0}$ **then**
12: **return** D
13: **end if**
14: $FallsInsideOtherDetector \leftarrow true$
15: $T \leftarrow 0$
16: **break**
17: **end if**
18: **end for**
19: **until** $\neg FallsInsideOtherDetector$
20: $r = min_{s \in S} distance(s,c) - R_s$
21: **if** $r > R_s$ **then**
22: $D \leftarrow D \cup \{\langle c, r \rangle\}$
23: **else**
24: $T \leftarrow T+1$
25: **end if**
26: **if** $T > \frac{1}{1-\text{maximum self coverage}}$ **then**
27: **exit**
28: **end if**
29: **until** $|D| = T_{max}$

Złożoność czasową algorytmu 3.5 można wyrazić jako:

$$O(|D| * |S|) \tag{3.3}$$

Proces monitorowania danych, podobnie jak w oryginalnym rozwiązaniu bazuje na funkcji odległości euklidesowej pomiędzy komórką a detektorem (algorytm 3.4).

Celem algorytmu V-detector jest przygotowanie takich detektorów, które maksymalnie pokryją przestrzeń nienależącą do komórek własnych. Wykorzystanie detektorów o zmiennych promieniach umożliwia pokrycie dużej powierzchni przy mniejszej liczbie detektorów. Dzięki parametrowi T_{max} możliwe jest sterowanie liczbą detektorów, co przekłada się na szybkość monitorowania. Natomiast parametr c_0 umożliwia dobór progu skuteczności wykrywania komórek obcych.

3.1.3 Algorytm detektorów RST

Algorytm jest rozszerzeniem algorytmu V-detector i jest inspirowany teorią zbiorów przybliżonych (Pawlak, 1991). Modyfikacja dotyczy rozbudowania procesu generowania detektorów i ich monitorowania o użycie dwóch zbiorów detektorów: D_{upp} i D_{low}, zamiast jednego.

W procesie generowania detektorów (algorytm 3.6) używane są dwa zbiory komórek własnych: pierwszy z oryginalnymi komórkami o promieniu $R_{s_{upp}}$ oraz drugi z komórkami z granicą tolerancji o promieniu $R_{s_{low}}$. Zbiór komórek z granicą tolerancji powstaje na podstawie oryginalnego zbioru komórek własnych, a promień ich wartości zwiększany jest o zadaną wartość progową. Na podstawie powyższych zbiorów tworzone są zbiory detektorów:

- zbiór D_{upp} jest generowany za pomocą algorytmu V-detector,
- zbiór D_{low} powstaje na podstawie zbioru D_{upp} przez skrócenie promienia każdego z detektorów o różnicę promieni między komórkami własnymi ze zbiorów $R_{s_{upp}}$ i $R_{s_{low}}$ (r_{diff}).

Algorytm 3.6 Algorytm detektorów RST - generowanie detektorów

INPUT:
D_{upp} - detektory wygenerowane algorytmem V-detector promieniem $R_{s_{upp}}$
$R_{s_{upp}}$ - promień komórki własnej
$R_{s_{low}}$ - promień komórki własnej z granicą tolerancji
$R_{s_{low}} > R_{s_{upp}}$
OUTPUT:
D_{upp} - zbiór detektorów
D_{low} - zbiór bardziej tolerancyjnych detektorów

1: $D_{low} \leftarrow \emptyset$
2: $R_{diff} \leftarrow R_{s_{low}} - R_{s_{upp}}$
3: **for** $\langle c, r \rangle \in D_{upp}$ **do**
4: $\quad D_{low} \leftarrow D_{low} \cup \{\langle c, r - r_{diff} \rangle\}$
5: **end for**

Złożoność czasową algorytmu 3.6 można wyrazić jako:

$$O(|D| * |S|) \tag{3.4}$$

W trakcie monitorowania danych (algorytm 3.7) komórki są weryfikowane za pomocą bardziej tolerancyjnego zbioru D_{low}. Jeśli detektory ze zbioru D_{low} rozpoznają komórkę, jest ona uznana za obcą. W przeciwnym wypadku jest ona weryfikowana przez detektory ze zbioru D_{upp}. Jeżeli komórka nie zostanie rozpoznana przez detektor ze zbioru D_{upp}, oznacza to, że jest to komórka własna. W przeciwnym razie komórka zostaje uznana za niepewną, co wiąże się z jej dalszą analizą (np. przez algorytm klasyfikujący).

Algorytm 3.7 Algorytm detektorów RST - monitorowanie

INPUT:
S - zbiór monitorowanych komórek
D_{upp} - zbiór detektorów
D_{low} - zbiór bardziej tolerancyjnych detektorów
OUTPUT:
S_{self} - zbiór komórek rozpoznanych jako własne
$S_{nonself}$ - zbiór komórek rozpoznanych jako obce
$S_{uncertain}$ - zbiór komórek do dalszej analizy

1: $S_{self} \leftarrow \emptyset; S_{nonself} \leftarrow \emptyset; S_{uncertain} \leftarrow \emptyset$
2: **for** $s \in S$ **do**
3: **if** $\exists_{d \in D_{low}} match(d,s)$ **then**
4: $S_{nonself} \leftarrow S_{nonself} \cup \{s\}$
5: **else**
6: $S_{uncertain} \leftarrow S_{uncertain} \cup \{s\}$
7: **end if**
8: **end for**
9: **for** $s \in S_{uncertain}$ **do**
10: **if** $\nexists_{d \in D_{upp}} match(d,s)$ **then**
11: $S_{uncertain} \leftarrow S_{uncertain} \setminus \{s\}$
12: $S_{self} \leftarrow S_{self} \cup \{s\}$
13: **end if**
14: **end for**

3.2 Eksperymenty

W poniższej części opisano eksperymenty, które badają wpływ dyskretyzacji atrybutów na skuteczność wybranych algorytmów selekcji negatywnej. Badania zostały

przeprowadzone na kilku zbiorach danych z wykorzystaniem różnych algorytmów dyskretyzacji.

Implementacje algorytmów zostały zrealizowane w języku Python 3.8 z wykorzystaniem modułów numpy i pandas oraz uruchomione na środowisku Arch Linux.

Tabela 3.1: Charakterystyka zbiorów

Nazwa	**Skrót**	**Licz. atrybutów**	**Licz. obiektów**	**Dystrybucja klas**
Wine	Wine	13	178	59:71:48
Mammographic Mass	Mamm	5	961	516:445
KDDCup'99 (10%)	Kdd	41	499020	97277:396743

Do badań użyto trzech zbiorów danych (tabela 3.1) dostępnych na repozytorium UCI Repository (`https://archive.isc.uci.edu/ml`): Wine, Mammographic Mass i KDDCup'99 (10%).

Zbiór Wine posiada trzynaście atrybutów i trzy klasy decyzyjne. W eksperymencie jako komórki własne uznano elementy należące do klasy o wartościach 1. Pozostałe elementy są traktowane jako komórki obce.

Zbiór Mamm ma pięć atrybutów i dwie klasy decyzyjne. Jako komórki obce traktowane są dane wskazujące na guz złośliwy. Mimo że, klasyfikacja algorytmami NSA powyższego zbioru nie daje najlepszych rezultatów, to został on umieszczony w przeglądzie ze względu na możliwość sprawdzenia, jak dyskretyzacja cech może wpłynąć na bardziej problematyczne dane.

Ostatni zbiór Kdd jest zbiorem największym ze względu na ilość danych oraz liczbę atrybutów. Za anomalię w przypadku tego zbioru przyjęto wszystkie próby ataków sieciowych. W tabeli 3.1 liczebność zbiorów komórek własnych została oznaczona za pomocą podkreślenia.

Każdy ze zbiorów został znormalizowany w taki sposób, aby wartości atrybutów reprezentowały liczby rzeczywiste ze zbioru [0,1]. Następnie podzielono każdy zbiór na część treningową i testową. Proporcje podziału zostały ustalone empirycznie i dostosowane do zbioru indywidualnie. Komórki określone jako własne z części treningowej zostały użyte do generowania detektorów, natomiast cała część testowa została użyta do monitorowania skuteczności wygenerowanych detektorów. Warto zaznaczyć, że taki podział danych jest istotny przy zagadnieniu wykrywania anomalii w ruchu sieciowym, gdy nie wszystkie struktury komórek własnych i obcych są znane. Na koniec pięciokrotnie uruchomiono algorytmy dla danych niezdyskretyzowanych i uśredniono wyniki.

W kolejnym kroku eksperymentów dokonano dyskretyzacji danych z wykorzystaniem narzędzi Weka (`https://www.cs.waikato.ac.nz/ml/weka/`) oraz RSES (`https://www.mimuw.edu.pl/~szczuka/rses/start.html`). Narzędzie RSES wykorzystuje zbiory przybliżone (Pawlak, 1991) do wykonania cięć, według których generowane są przedziały wartości. Dzięki temu zachowane zostają różnice pomiędzy obiektami z różnych klas zbiorów, a zestaw cięć jest ograniczony do minimum (Bazan, Nguyen, Nguyen, Synak i Wróblewski, 2000). W ramach eksperymentu zdyskretyzowano dane według metody lokalnej i/lub globalnej. Metoda globalna dyskretyzuje dane w odniesieniu do całego zbioru danych. Używa jednego zestawu interwałów w ramach jednego zadania klasyfikacyjnego. Z kolei metoda lokalna tworzy różne zestawy interwałów dla pojedynczego atrybutu. Warto zwrócić uwagę na liczba przedziałów wygenerowanych przez narzędzie RSES. Dla zbioru Mammographic Mass z ponad siedemdziesięciu unikalnych wartości na atrybucie Age udało się zejść odpowiednio do 38 przedziałów dla metody globalnej i 46 przedziałów dla metody lokalnej. Na podstawie zaproponowanych podziałów można wywnioskować, że atrybut Age nie jest dobrym rozróżnikiem klas decyzyjnych. Z kolei na zbiorze KDDCup'99 10% metoda lokalna zaproponowała cięcia na 26 atrybutach generując od 2 do 38 przedziałów (w zależności od atrybutu). W przypadku zbioru Wine przedziały wygenerowane przez metody lokalną i globalną są podobne i dotyczą trzech atrybutów: Alcohol (trzy przedziały), Magnesium (dwa przedziały) i Flavonoids (dwa przedziały).

W przypadku narzędzia Weka dokonano dyskretyzacji na wybranych atrybutach rozpatrywanych zbiorów z podziałem na 5 i 10 przedziałów wartości. Jako metodę dyskretyzacji wybrano metodę nienadzorowaną. Podczas dyskretyzacji uwzględniono również tworzenie przedziałów w sposób binary dla każdego atrybutu oraz bez tego podziału. W wyniku tworzenia przedziałów w sposób binarny, w zbiorach tworzone są nowe atrybuty dla każdego interwału określające w sposób binarny podział wartości. W zestawieniu wyników zastosowanie tej metody oznaczono za pomocą kolumny *atrybuty binarne*. Jeśli użycie metody dyskretyzacji utworzyło nowe atrybuty z przedziałami wartość w kolumnie *atrybuty binarne* wstawiono *tak*. W przeciwnym wypadku użyto *nie*.

Dla każdego algorytmu pięciokrotnie uruchomiono testy na zbiorach po dyskretyzacji, a następnie uśredniono wyniki. Użyto takich samych ustawień i podziałów na zbiory testowe i treningowe, jak w przypadku testów na danych przed dyskretyzacją. W obu przypadkach (przed i po dyskretyzacji) użyto takich samych metryk mierzących skuteczność algorytmu: metryki dokładności (ang. *accuracy*) dla algorytmów RNS i detektorów V-detector oraz współczynnika wyników fałszywie pozytywnych (ang. *false alarm*) dla algorytmu z detektorami RST. Wyniki zostały zaprezentowane w tabelach w poszczególnych podrozdziałach. Rezultaty dla danych niezdyskretyzowanych zostały oznaczone podkreśleniem. Brak wartości oznaczono znakiem –.

3.2.1 Algorytm RNS

W przypadku zbioru Wine zbadano skuteczność algorytmu RNS dla detektorów o promieniu od 1,4 do 3,5. Najwyższy wynik wynoszący 0,82 algorytm osiągnął dla promienia 1,5, a powyżej wartości 1,9 algorytm nie był w stanie dopasować detektorów do wylosowanych zestawów testowych. Po dokonaniu dyskretyzacji wartości trzech atrybutów uzyskano wyższą skuteczność dla promienia 1,5 i 1,9 w przypadku zastosowania metody lokalnej lub globalnej (tabela 3.2). Warto zauważyć, że po dokonaniu dyskretyzacji z podziałem na atrybuty binarne, skuteczność algorytmu dla podanych wyżej wartości promienia spada z uwagi na fakt utworzenia nowych atrybutów. Z drugiej strony zastosowanie tej metody umożliwia uruchomienie algorytmu dla promienia, który w przypadku oryginalnych danych (sprzed dyskretyzacji) nie jest możliwe i uzyskanie wyższej detekcji (np. $r = 3,5$).

Podobne regularności można zaobserwować przy zbiorze Mammographic Mass (tabela 3.3).

Algorytm uruchomiono dla promienia o wartościach z zakresu od 0,45 do 1,5. W tabeli 3.3 zostały umieszczone rezultaty o najwyższej mierze dokładności dla każdej z metod. Podobnie jak w przypadku zbioru Wine najwyższą dokładność uzyskano przy zastosowaniu metody lokalnej i globalnej. Co ciekawe, mimo że atrybut Age nie jest najlepszym rozróżnikiem klas, to na danych zdyskretyzowanych uzyskano nieznacznie wyższe wyniki niż na danych przed dyskretyzacją.

Z kolei na zbiorze KDDCup'99 10% dokonano dyskretyzacji siedmiu atrybutów metodą nienadzorowaną z podziałem na 5 i 10 przedziałów oraz metodą lokalną. Rezultaty z przeprowadzonych testów były nieznacznie niższe od wyników zwróconych przy uruchomieniu algorytmu na danych oryginalnych i spadły z 0,99 do 0,97. Przetestowano również cięcia na 26 atrybutach zaproponowanych przez metodę lokalną. Dyskretyzacja wartości tych atrybutów znacznie pogorszyła wyniki klasyfikacji algorytmu spadając z 0,99 do 0,28. Podobnie jak w przypadku mniejszych zbiorów, zaobserwowano spadek detekcji komórek obcych przy zastosowania atrybutów binarnych. Zwiększenie wartości promienia przy ponownych testach pozwoliło uzyskać zadowalające wyniki na poziomie 0,99.

Tabela 3.2: Działanie algorytmu RNS na zbiorze Wine po dyskretyzacji trzech atrybutów

Metoda	**Liczba przedziałów**	**Atrybuty binarne**	*r*	**Skuteczność**
-			1,5	0,82
-			1,9	-
bez nadzoru	5	tak	1,5	0
bez nadzoru	5	nie	1,5	0,75
bez nadzoru	10	nie	1,5	0,75
lokalna	2-3	nie	1,5	0,88
lokalna	2-3	nie	1,9	0,99
bez nadzoru	5	tak	1,9	0,84
bez nadzoru	10	tak	1,9	0
bez nadzoru	5	nie	1,9	0,55
bez nadzoru	10	nie	1,9	-
globalna	2-3	nie	1,5	0,99
globalna	2-3	nie	1,9	0,99
bez nadzoru	10	tak	3,5	0,97

r - promień detektora, Skuteczność liczona jest metryką dokładności

Tabela 3.3: Skuteczność algorytmu RNS na zbiorze Mammographic Mass po dyskretyzacji atrybutu Age

Metoda	**Liczba przedziałów**	**Atrybuty binarne**	*r*	**Skuteczność**
-			0,45	0,5
-			1	-
bez nadzoru	5	tak	1,5	0
bez nadzoru	5	nie	0,45	0,52
bez nadzoru	10	nie	0,45	0,53
lokalna	46	nie	0,45	0,55
globalna	38	nie	0,45	0,53
bez nadzoru	5	tak	0,45	0,01
bez nadzoru	10	tak	0,45	0,09
bez nadzoru	5	tak	1	0,47
bez nadzoru	10	tak	1	-
bez nadzoru	10	tak	1,5	0,54

r - promień detektora, Skuteczność liczona jest metryką dokładności

3.2.2 Algorytm V-detector

Algorytm V-detector przetestowano z następującymi ustawieniami:

- Wine - r od 0,01 do 0,1, T_{max}=20,
- Mammographic Mass - r=0,45, T_{max}=100,
- KDDCup'99 10% - r=1,5, T_{max}=100,

gdzie r to promień komórki własnej, T_{max} maksymalna liczba detektorów. Oczekiwane pokrycie c_0 wynosiło 0,9. Wyniki eksperymentu przedstawiono zbiorczo dla wszystkich zbiorów w tabeli 3.4.

Tabela 3.4: Skuteczność algorytmu V-detektor po dyskretyzacji danych

Metoda	**Liczba przedziałów**	**Atrybuty binarne**	**Skuteczność**
		Zbiór Wine	
-	-	-	0,84
bez nadzoru	5	tak	0,29
bez nadzoru	5	nie	0,78
bez nadzoru	10	tak	0,40
bez nadzoru	10	nie	0,79
lokalna	2-3	nie	0,59
globalna	2-3	nie	0,83
		Zbiór Mamm	
-	-	-	0,70
bez nadzoru	5	tak	0,60
bez nadzoru	5	nie	0,68
bez nadzoru	10	tak	0,47
bez nadzoru	10	nie	0,69
lokalna	2-3	nie	0,65
globalna	2-3	nie	0,74
		Zbiór Kdd	
-	-	-	0,98
bez nadzoru	5	tak	0,60
bez nadzoru	5	nie	0,98
bez nadzoru	10	tak	0,95
bez nadzoru	10	nie	0,98
lokalna (7)	2-26	nie	0,98
lokalna (26)	2-34	nie	0,27

Skuteczność liczona jest metryką dokładności

W przypadku algorytmu V-detector dla mniejszych zbiorów najlepsze rezultaty w klasyfikacji dała metoda globalna. Dyskretyzacja atrybutów tą metodą pozwala uzyskać wynik zbliżony do tego, który otrzymujemy na danych oryginalnych. W przypadku zbioru KDDCup'99 10% dyskretyzacja siedmiu atrybutów nie wpłynęła znacząco na działanie algorytmu. Zastosowanie cięć zbiorów na ponad połowie atrybutów spowodowało znaczne obniżenie jakości klasyfikacji.

3.2.3 Algorytm detektorów RST

Głównym celem algorytmu detektorów RST jest zmniejszenie liczby rozpoznań komórek własnych jako obcych. W związku z tym jako miarę skuteczności algorytmu wybrano współczynnik wyników fałszywie pozytywnych. Na podstawie otrzymanych rezultatów (tabela 3.5) można wnioskować, że dyskretyzacja danych dla małych zbiorów zminimalizowała błędną klasyfikację komórek własnych jako obcych, zwiększając tym samym liczbę komórek niepewnych, wymagających dalszej analizy. W przypadku zbioru KDDCup'99 10% wartość metryki utrzymało się na poziomie 0, niezależnie od zastosowanej metody dyskretyzacji.

Tabela 3.5: Skuteczność algorytmu detektorów RST po dyskretyzacji danych

Metoda	**Liczba przedziałów**	**Atrybuty binarne**	**Skuteczność**	**Komórki niepewne**
		Zbiór Wine		
-	-	-	0,56	0,02
bez nadzoru	5	tak	0,30	0,03
bez nadzoru	5	nie	0,55	0,02
bez nadzoru	10	tak	0,09	0,02
bez nadzoru	10	nie	0,55	0,02
lokalna	2-3	nie	0,14	0,05
globalna	2-3	nie	0,30	0,03
		Zbiór Mamm		
-	-	-	0,2	0,47
bez nadzoru	5	tak	0,07	0,41
bez nadzoru	5	nie	0,04	0,57
bez nadzoru	10	tak	0,09	0,30
bez nadzoru	10	nie	0,04	0,58
lokalna	2-3	nie	0,05	0,50
globalna	2-3	nie	0,01	0,67

Skuteczność jest określana miarą współczynnika wyników fałszywie pozytywnych

Podsumowanie

W powyższej pracy przedstawiono wstępne badania nad wpływem dyskretyzacji cech na działanie algorytmów selekcji negatywnej.

Dyskretyzacja cech jest procesem polegającym na zmianie atrybutów ciągłych na dyskretne. W związku z tym, może wpłynąć na utratę istotnych informacji opisujących obiekty z różnych klas. Jednak, jak pokazały przeprowadzone w tej pracy eksperymenty, jeżeli obiekty z różnych klas mają bliskie wartości na danym atrybucie, to po dokonaniu dyskretyzacji danego atrybutu, różnice między obiektami mogą być bardziej widoczne. Dzięki temu skuteczność algorytmów selekcji negatywnej może wzrosnąć. Ponadto, dyskretyzacja kilku atrybutów na małych zbiorach (maksymalnie kilkanaście atrybutów) może nieznacznie poprawić działanie algorytmów NSA, nawet jeśli dany atrybut nie jest najlepszym wyróżnikiem klas. Przy dużych zbiorach (kilkadziesiąt atrybutów) dyskretyzacja wartości nieistotnych cech, nie wpływa znacząco na obniżenie jakości klasyfikacji. Niemniej zamiana wartości atrybutów na przedziały dla zbyt dużej liczby (np. połowy) atrybutów może doprowadzić do zupełnego zatarcia różnic między obiektami. W przeprowadzonych badaniach najlepsze rezultaty otrzymano na podstawie cięć zbiorów dokonanych metodą globalną narzędzia RSES. Metoda ta zachowuje informację o klasach obiektów. Można więc wnioskować, że algorytmy NSA zwracają bardziej satysfakcjonujące wyniki, analizując dane zdyskretyzowane metodą z nadzorem i podziałami dokonanymi na podstawie całego zbioru danych.

Badania mogą być w przyszłości rozszerzone o kolejne implementacje algorytmów selekcji negatywnej (uwzględniające również rozwiązania hybrydowe) oraz użycie dodatkowych metod dyskretyzacji, np. z nadzorem na danych wielowymiarowych.

Bibliografia

Aziz, A. S. A., Azar, A. T., Hassanien, A. E. i Hanafy, S. E.-O. (2014). Continuous features discretization for anomaly intrusion detectors generation. W: V. Snášel, P. Krömer, M. Köppen i G. Schaefer (red.), *Soft computing in industrial applications,* Springer, 209–221.

Bazan, J. G., Nguyen, H. S., Nguyen, S. H., Synak, P. i Wróblewski, J. (2000). Rough set algorithms in classification problem. W: L. Polkowski, S. Tsumoto i T. Y. Lin (red.), *Rough set methods and applications: New developments in knowledge discovery in information systems,* Springer, 49–88.

Chmielewski, A. (2017). Application of rough sets to negative selection algorithms. W: T. K. Dang, R. Wagner, J. Küng, N. Thoai, M. Takizawa i E. J. Neuhold (red.), *Future data and security engineering,* Springer, 381–394.

Dasgupta, D., i Forrest, S. (1995). Novelty detection in time series data using ideas from immunology. W: *Proceedings of 8th international conference on intelligent systems,* 6.

Esponda, F., Forrest, S. i Helman, P. (2003). The crossover closure and partial match detection. W: J. Timmis, P. J. Bentley i E. Hart (red.), *Artificial immune systems,* Springer, 249–260.

Forrest, S., Perelson, A. S., Allen, L. i Cherukuri, R. (1994). Self-nonself discrimination in a computer. W: *Proceedings of 1994 ieee computer society symposium on research in security and privacy,* IEEE, 202-212.

García, S., Luengo, J., Sáez, J. A., López, V. i Herrera, F. (2013). A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, *25*, 734-750.

Gonzalez, F., i Dasgupta, D. (2003). Anomaly detection using real-valued negative selection. *Genetic Programming and Evolvable Machines*, *4*, 384–403.

Hofmeyr, S. A. (1999). *An immunological model of distributed detection and its application to computer security*. Unpublished doctoral dissertation, The University of New Mexico.

Hońko, P. (2018). Adaptive positive-negative selection approach. *Journal of Physics: Conference Series*, *1061*, 012-020.

Jerne, N. K. (1973). Towards a network theory of the immune system. *Annals of Immunology*, *125*, 373-389.

Ji, Z., i Dasgupta, D. (2004). Real-valued negative selection algorithm with variable-sized detectors. W: K. Deb (red.), *Genetic and evolutionary computation – gecco 2004,* Springer, 287–298.

Ji, Z., i Dasgupta, D. (2007). Revisiting negative selection algorithm. *Evolutionary Computation*, *15*, 223–251.

Lasek, M., Lasek, W. i Pęczkowski, M. (2013). Od immunologii do modelowania, przetwarzania i analiz danych. *Informatyka Ekonomiczna*, *4*, 196–225.

Lucińska, M. (2010). Hybrid immune algorithm for many optima. W: L. Rutkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh i J. M. Zurada (red.), *Artifical intelligence and soft computing* Springer, 540–547.

Lydyard, P. M., Whelan, A. i Fanger, M. W. (2001). *Immunologia. krótkie wykĺady.* Warszawa: Wydawnictwo Naukowe PWN.

Matzinger, P. (1994). Tolerance, danger, and the extended family. *Annual Review of Immunology*, *12*, 991-1045.

Pawlak, Z. (1991). *Rough sets. Theoretical aspects of reasoning about data*. Dordrecht: Kluwer Academic Publishers Group.

Praczyk, T. (2010). Using real valued detectors in ship immune system. *Computing and Informatics*, *29*, 975–987.

Wierzchoń, S. T. (2001). *Sztuczne systemy immunologiczne. teoria i zastosowania.* Warszawa: Akademicka Oficyna Wydawnicza EXIT.

Rozdział 4

WPŁYW TECHNIK WSTĘPNEGO PRZYGOTOWANIA DANYCH NA SKUTECZNOŚĆ KLASYFIKACJI OBIEKTÓW BAZY DERMATOLOGY ZA POMOCĄ ALGORYTMU LEM2

Dariusz Jankowski*

Streszczenie Choroby skóry mają wiele różnych odmian i są częstym obiektem badań w medycynie. Szczególnie niebezpieczne są wszelkie odmiany choroby prowadzące do powstania raka. Z dotychczasowych badań medycznych wynika, że wczesne wykrycie symptomów choroby nowotworowej pozwala na znaczne zmniejszenie prawdopodobieństwa powstania raka lub też jego rozwoju. Najnowsze doniesienia medyczne wskazują, że liczba pacjentów z chorobami skóry stale zwiększa się. Według prognoz do 2025 roku w Polsce liczba zachorowań na czerniaka skóry podwoi się. Trend jest taki sam również w innych krajach. Na całym świecie są prowadzone liczne badania nad poszukiwaniem najbardziej efektywnych metod budowy modeli danych opisujących choroby dermatologiczne skóry człowieka. Modele te budowane są na bazie niepełnych danych - na podstawie wybranej próby statystycznej, co wymusza zastosowanie metod, pozwalających na uogólnianie wyników uzyskanych z próby. Głównym problemem badawczym niniejszej pracy było poszukiwanie odpowiedzi na pytanie: jaki wpływ na jakość klasyfikacji chorób skóry modelu opartego na algorytmie LEM2 mają techniki wstępnego przygotowania danych (podziału tabeli, usuwania brakujących wartości, dyskretyzacji, selekcji atrybutów) w pozyskiwaniu wiedzy z bazy danych Dermatology, zawierającej historię osób cierpiących na choroby skóry. Podczas poszukiwania podobnych badań zauważono małą liczbę wyników z wykorzystaniem LEM2 i bazy Dermatology.

Słowa kluczowe: klasyfikacja chorób skóry, algorytm LEM2, zbiory przybliżone, selekcja atrybutów, dyskretyzacja

* Wydział Informatyki, Politechnika Białostocka, Wiejska 45A, 15-351 Białystok, d.jankowski@pb.edu.pl

DOI 10.24427/978-83-66391-58-1_4

Wprowadzenie

Choroby skóry mają wiele różnych odmian i są częstym obiektem badań w medycynie. Szczególnie niebezpieczne są wszelkie odmiany choroby prowadzące do powstania raka. Z dotychczasowych badań medycznych wynika, że wczesne wykrycie symptomów choroby nowotworowej pozwala na znaczne zmniejszenie prawdopodobieństwa powstania raka lub też jego rozwoju. Podkreśla to Amerykańska Fundacja Raka Skóry (Foundation, 2020), która definiuje raka skóry jako „niekontrolowany wzrost nieprawidłowych komórek w naskórku, najbardziej zewnętrznej warstwie skóry, spowodowany przez nienaprawialne, uszkodzone DNA, które wyzwala mutacje. Mutacje te prowadzą do szybkiego namnażania się komórek skóry i tworzenia złośliwych guzów. Główne typy raka skóry to rak podstawnokomórkowy (BCC), rak płaskonabłonkowy (SCC), czerniak i rak z komórek Merkla (MCC)". „Skin cancer is the out-of-control growth of abnormal cells in the epidermis, the outermost skin layer, caused by unrepaired DNA damage that triggers mutations. These mutations lead the skin cells to multiply rapidly and form malignant tumors. The main types of skin cancer are basal cell carcinoma (BCC), squamous cell carcinoma (SCC), melanoma and Merkel cell carcinoma (MCC)". Ich badania ustaliły, że przyczyną większości chorób raka skóry są: szkodliwe promienie ultrafioletowe (UV) słoneczne oraz korzystanie z solariów UV. „The two main causes of skin cancer are the sun's harmful ultraviolet (UV) rays and the use of UV tanning machines" (Foundation, 2020).

Najnowsze doniesienia medyczne wskazują, że liczba pacjentów z chorobami skóry stale zwiększa się (Didkowska, Wojciechowska, Czderny, Olasek i Ciuba, 2019). Według prognoz do 2025 roku w Polsce liczba zachorowań na czerniaka skóry podwoi się (Didkowska, Wojciechowska i Zatorski, 2009). Rosnący trend zachorowalności jest taki sam również w innych krajach. Zgodnie ze statystykami (Wojciechowska i Didkowska, 2020) w Polsce jest prawie o połowę mniejsza zachorowalność na czerniaka skóry niż w Unii Europejskiej, natomiast nieco większa niż przeciętna umieralność (o około 20%).

Na całym świecie prowadzone są liczne badania nad poszukiwaniem najbardziej efektywnych metod budowy modeli danych opisujących choroby dermatologiczne skóry człowieka. Modele te budowane są w oparciu o niepełne dane - na podstawie wybranej próby statystycznej, co wymusza zastosowanie metod, pozwalających na uogólnianie wyników.

Za budowę i przetwarzanie modeli danych medycznych obecnie odpowiedzialne są wyspecjalizowane programy komputerowe, wykorzystujące metody statystyczne i metody sztucznej inteligencji. Na ich podstawie proces wykrywania chorób oraz proces podejmowania decyzji o sposobie leczenia pacjentów staje się coraz bardziej efektywny.

Wśród baz danych ogólnodostępnych, które umożliwiają poszukiwanie nowych metod analitycznych, służących budowie modeli danych, dominują:

1. HAM1000.
2. Dermatology (Dua i Graff, 2017).
3. Melanoma Gene Database (MGDB).
4. Melanoma (Australia bioplatforms data portal).
5. MelanomaDB.

Ze względu na rodzaj informacji w nich zawartych o chorobach skóry, bazy te można podzielić na:

- Bazy obrazów skóry z etykietami (np. HAM10000).
- Bazy genów (np. MelanomaDB).
- Bazy danych opisowych (np. Dermatology).

Bazy danych opisowe reprezentowane są formalnie jako systemy informacyjne.

Definicja 4.1. Systemem informacyjnym S (Pawlak, 1980) (Pawlak, 1991) nazywamy układ:

$$SI =< U, A, V, f > \tag{4.1}$$

gdzie:

- U - niepusty, skończony zbiór obiektów zwany uniwersum,
- A - niepusty, skończony zbiór atrybutów opisujących obiekty uniwersum,
- $V = \cup_{a \in A} V_a$, gdzie V_a jest zbiorem wartości atrybutu a, zaś card(V_a)>1,
- $f : U \times A \rightarrow V$ - funkcja informacji, taka że: $\forall_{u \in U, a \in A}\ f(u,a) \in V_a$.

Baza danych Dermatology jest przykładem reprezentacji systemu informacyjnego.

Zgodnie z Pawlak (2005) oraz Stepaniuk (2008) podczas analizy danych w systemach informacyjnych podstawową kwestią jest poszukiwanie wzorców wśród danych, w celu odnalezienia zależności pomiędzy wybranymi zbiorami atrybutów.

Charakterystyczną cechą systemów informacyjnych, zawierających dane medyczne, jest problem występowania brakujących wartości oraz błędnie wprowadzonych danych do systemu informacyjnego lub błędnie zmierzonymi wartościami, o czym piszą: Little i in. (2012), Dziura, Post, Zhao, Fu i Peduzzi (2013), O'Neill i Temple (2012), Pezoulas i in. (2019), Cao, Stojkovic i Obradovic (2016), Khare i in. (2017), Tremblay, Hevner i Berndt (2012), Thabane i in. (2013), Kannan, Manoj i Arumugam (2015). Powyższe badania potwierdzają, że problem jest wciąż aktualny. Od jakości danych zależy jakość znajdowanych wzorców danych, które służą do budowy systemów decyzyjnych, a następnie trafności decyzji takich systemów.

Do innych, często spotykanych problemów w analizie danych medycznych są wykorzystywane systemy informacyjne zawierające historię o małej liczbie pacjentów, lecz wielu atrybutach (np. bazy z danymi genetycznymi) oraz systemy o bardzo dużej liczbie obiektów i wielu atrybutach. W zależności od metody analitycznej, przetworzenie wszystkich informacji może nie być możliwe i dlatego wymagany jest

dodatkowy etap preselekcji atrybutów, aby zmniejszyć wymiarowość przestrzeni poszukiwanych rozwiązań.

W przypadku wielu badań eksperymentalnych, można spotkać bazy zawierające małą liczbę obiektów i atrybutów, co związane może być np. z ograniczoną liczbą ochotników biorących udział w eksperymencie. W takim przypadku, wyniki obarczone są dodatkowym ryzykiem niedopasowania wzorców do całej populacji.

Należy zauważyć, że powyższe problemy z danymi dotyczą wszystkich systemów informacyjnych, a nie tylko medycznych.

Chcąc odpowiedzieć na bieżące problemy, autor niniejszej pracy przedstawił nowe możliwości wykorzystania algorytmu LEM2 (Grzymala-Busse, 1992) do pozyskiwania modeli danych z medycznych baz danych opisowych na podstawie ogólnodostępnej bazy Dermatology (Dua i Graff, 2017), zawierającej historię osób cierpiących na choroby skóry. Wyniki tych badań mają charakter uniwersalny i można je wykorzystać przy analizie innych systemów informacyjnych.

Algorytm LEM2 należy do zbioru metod poszukiwania minimalnego zbioru reguł w systemach informacyjnych za pomocą indukcji reguł. Umożliwia przetwarzanie tablic decyzyjnych zawierających sprzeczności. Różni się pod tym względem od pozostałych metod, jak np. drzewa decyzyjne, które wymagają usunięcia sprzeczności przed etapem budowy modelu danych. Algorytm ten nie zastępuje metod badania obrazów skóry chorób pacjentów czy też genów, a jedynie je uzupełnia.

Analiza porównawcza dostępnych badań naukowych nad bazą Dermatology oraz wykorzystania algorytmu LEM2 do badań medycznych nad nią, wykazała małe zainteresowanie wykorzystaniem tej metody do budowy klasyfikatorów wykrywających choroby skóry i symptomów raka. Większość opublikowanych badań wykorzystuje modele oparte o metody sieci neuronowych, drzewa decyzyjne i inne. Dotychczasowe badania z wykorzystaniem algorytmu LEM2 w stosunku do danych o chorobach skóry, skupiały się w głównej mierze na benchmarkingu metod. Opublikowane badania wykazują skuteczność predykcji klasyfikatorów opartych o algorytm LEM2 w zakresie 87-90%, a jednocześnie wysoką skuteczność takich metod jak: sieci neuronowe, drzewa decyzyjne, SVM - gdzie uzyskano skuteczność klasyfikacji na poziomie 95 - 100%. Zestawienie skuteczności klasyfikatorów dla różnych baz danych, w tym bazy Dermatology, przygotował Zhang, Liu, Zhang i Almpanidis (2017). Zgodnie z jego zestawieniem, najskuteczniejszą metodą klasyfikacji bazy Dermatology jest klasyfikator zbudowany na podstawie algorytmu SVM, dla którego współczynnik skuteczności predykcji wyniósł 100%.

Kusunoki i Inuiguchi (2006) na podstawie algorytmu LEM2 w stosunku do bazy Dermatology zbudowali klasyfikator o skuteczności predykcji 90.24%, a Borowik, Kraśniewski i Łuba (2015) uzyskali skuteczność 87,77% używając systemu RSES oraz 78% wykorzystując metodę autorską.

Srimani i Koti (2014) wykorzystali co prawda algorytm LEM2 do wygenerowania reguł, otrzymując współczynnik pokrycia równy 90%, jednak nie zbudowali klasyfikatora i testów jego skuteczności. Badania Koti (2014) również objęły ana-

lizę pokrycia reguł wygenerowanych z użyciem algorytmu LEM2 w systemie RSES (bez budowy klasyfikatora), a także badania zbioru PIMA (zawierającego przypadki pacjentów cierpiących na cukrzycę), dla którego skuteczność algorytmu LEM2 wyniosła 76%.

Metoda przyśpieszająca generowanie reduktów, zaprezentowana w Borowik (2019) pozwoliła na obliczenie wszystkich reduktów bazy danych Dermatology w 2 minuty. Autor zwrócił uwagę, że w systemie RSES obliczenie reduktów nie było możliwe z powodu dużego zużycia pamięci.

W dalszej części artykułu, autor pracy prezentuje wyniki i możliwości dalszego rozwoju prac nad wykorzystaniem algorytmu LEM2 w analizie danych medycznych.

4.1 Metodyka badań

4.1.1 Opis danych i stanowiska badawczego

Zgodnie z wprowadzeniem, do badania została wybrana baza danych Dermatology, opublikowana w Dua i Graff (2017). Od strony programistycznej zostały wykorzystane biblioteki języka R oraz Python, z wyszczególnieniem bibliotek: RoughSets, arules oraz scikit-learn. Poniżej przedstawiona jest charakterystyka bazy Dermatology.

- liczba przebadanych pacjentów: 366,
- liczba atrybutów opisujących: 34,
- liczba atrybutów decyzyjnych: 1 (6 klas decyzyjnych),
- liczba rekordów zawierających brakujące dane: 8 - wszystkie dotyczą atrybutu Age. Wartości brakujące zostały zastąpione znakiem '?'.

Oznaczenie klas decyzyjnych:

Kod klasy - Nazwa klasy - Liczba obiektów,
1 - psoriasis (łuszczyca) - 112,
2 - seboreic dermatitis (łojotokowe zapalenie skóry) - 61,
3 - lichen planus (liszaj płaski, liszaj czerwony, liszaj Wilsona) - 72,
4 - pityriasis rosea (łupież różowy Giberta) - 49,
5 - cronic dermatitis (przewlekłe zapalenie skóry) - 52,
6 - pityriasis rubra pilaris (łupież czerwony mieszkowy) - 20.

Podane choroby charakteryzuje jedna właściwość: trudno jest je rozpoznać za pomocą obserwacji i najczęściej potrzebne jest wykonanie biopsji, lecz niestety choroby te mają wiele cech histopatologicznych. Według autorów bazy, pacjenci byli w pierwszej kolejności badani klinicznie. Wyniki tych badań reprezentują cechy

o numerach: 1-11, 34. W drugiej kolejności badane były próbki skóry, a wyniki zapisane za pomocą cech o numerach: 12-33. Wszystkie nazwy cech dostępne są na stronie projektu Dua i Graff (2017). Należy zauważyć, że jedynie cecha Age zawiera rzeczywisty wiek pacjenta. Pozostałe cechy warunkowe są zakodowane. W przypadku cechy "family history"mamy dwie możliwe wartości: 1 - oznacza, że choroba wystąpiła wcześniej u innego członka rodziny pacjenta, a 0 - brak wystąpienia. Pozostałe cechy mogą przybierać wartości: 0, 1, 2, 3, gdzie: 0 - oznacza brak wystąpienia cechy, 3 - oznacza największą możliwą wartość (przedział), a 1 i 2 - odpowiednio wartości pośrednie.

4.1.2 Przygotowanie danych treningowych i testowych

Zgodnie z założeniami uczenia maszynowego, proces budowy rozwiązań dzieli się na etap trenowania (budowy modelu danych, klasyfikatora) oraz predykcji nowych obiektów (nieznanych na etapie trenowania) i zebraniu mierników jakościowych. Etap ten powtarza się wielokrotnie dla różnych kombinacji parametrów i różnych podziałów zbioru danych, a następnie wybiera się jeden z najlepszych.

Z uwagi na mały rozmiar danych (366 obiektów) oraz brak dostępu do nowych pacjentów, każdy nowy eksperyment rozpoczyna się podziałem bazy na 2 tabele: treningową i testową. Tabela treningowa używana jest wyłącznie na etapie trenowania, a tabela testowa wyłącznie w końcowej ocenie jakości klasyfikacji (zastępuje dane o nowych pacjentach). Przed dokonaniem podziału rekordy bazy Dermatology są losowo przestawiane.

Podział danych był dokonywany w 2 wariantach: 80:20 oraz 90:10.

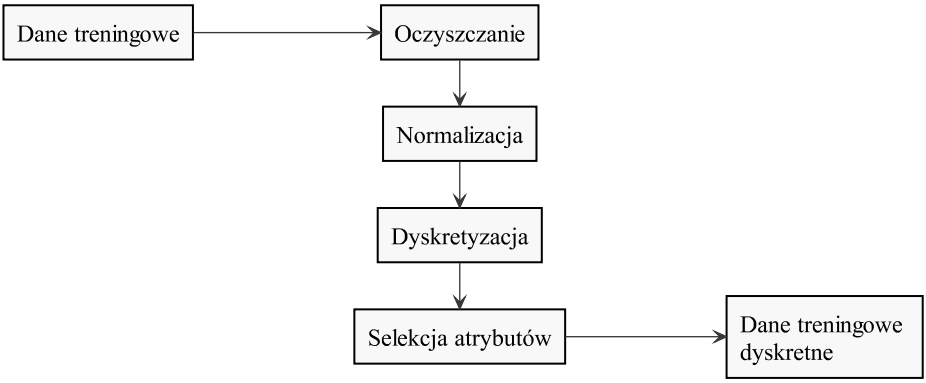
W badaniu starano się zachować równomierne proporcje klas w tabeli walidacyjnej, aby nie dopuścić do sytuacji, że większość przypadków należałaby do klasy 1 (najbardziej licznej).

Jak wspomniano wcześniej, baza Dermatology zawiera 8 rekordów z brakującymi wartościami cechy Age. W celu dostosowania bazy do dalszej analizy, obliczono dla każdej klasy decyzyjnej najczęściej występującą wartość i zgodnie z tym kryterium uzupełniono brakujące wartości w bazie danych.

Ponieważ baza danych została znormalizowana przez ich twórców, dlatego etap normalizacji został pominięty.

Z całej bazy, tylko 1 cecha (Age) wymagała przeprowadzania procesu dyskretyzacji. Do tego celu wykorzystywano zamiennie metody z pakietu RoughSets oraz arules.

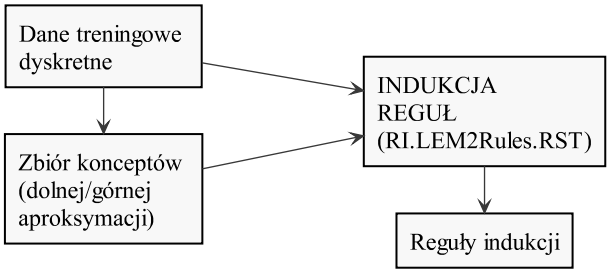
Do etapu selekcji atrybutów wybrano metody znajdowania reduktów z pakietu RoughSets.



Rysunek 4.1: Przygotowanie danych treningowych dla algorytmu LEM2

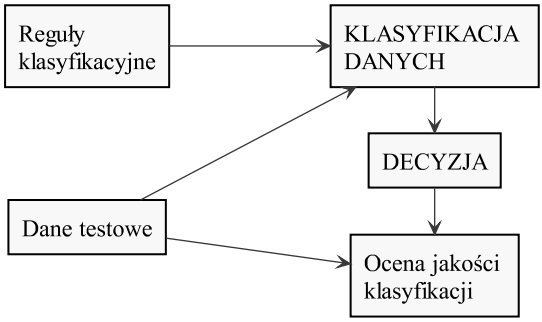
4.1.3 Indukcja reguł i ocena ich jakości

Indukcja reguł za pomocą algorytmu LEM2 wymaga określenia podzbioru danych treningowych, tworzących zbiór konceptów, a dokładnie ich dolnej lub górnej aproksymacji zgodnie z metodą zastosowaną w systemie LERS (Grzymala-Busse, 1997). W przypadku funkcji z biblioteki RoughSets (RI.LEM2Rules.RST), zbiorem konceptów jest zbiór obiektów należących do obszaru pozytywnego w rozumieniu teorii zbiorów przybliżonych.



Rysunek 4.2: Indukowanie reguł z użyciem algorytmu LEM2

Na podstawie reguł utworzonych na etapie indukcji przeprowadzono wstępną klasyfikację obiektów, korzystając ze zbioru testowego), utworzonego w I etapie przetwarzania. Podczas klasyfikacji, informacja o przyporządkowanej klasie była niedostępna dla klasyfikatora. Wyniki klasyfikacji zostały zebrane w formie tabelarycznej a następnie porównane z wartościami wcześniej zapisanymi w zbiorze testowym.



Rysunek 4.3: Wstępna weryfikacja reguł klasyfikacyjnych za pomocą danych testowych

Na podstawie wzorów dla wyznaczenia macierzy pomyłek i typowych mierników jakości przedstawionych przez Manliguez (2016) i Fawcett (2006) obliczono wskaźnik skuteczności klasyfikacji całego zbioru testowego (w celu porównania osiągniętych wyników z przedstawionymi we wprowadzeniu innymi badaniami) oraz dla każdej klasy obliczono takie wartości jak: czułość, specyficzność, PPV, NPV, skuteczność zrównoważoną oraz ich wartość ważoną wg wzoru:

$$WWM = \frac{\sum_{i=1}^{k} (WM_i) * (IL_i)}{\sum_{i=1}^{k} IL_i} \tag{4.2}$$

gdzie:

- WWM - wartość ważona miernika,
- WM_i - wartość miernika dla klasy i,
- IL_i - liczba obiektów klasy i,
- k - liczba klas.

Do obliczeń mierników wykorzystano funkcję confusionMatrix z biblioteki R - caret oraz funkcję F1_score z biblioteki MLmetrics.

4.1.4 Dziesięciokrotna walidacja krzyżowa z 20 powtórzeniami

W celu znalezienia najlepszego modelu danych, wykorzystano metodę dziesięciokrotnej walidacji krzyżowej z dwudziestoma powtórzeniami, zachowującą równomierne rozłożenie klas w każdej próbce danych. Do obliczeń użyto autorski estymator i funkcje RepeatedStratifiedKFold oraz GridSearchCV z biblioteki Python - scikit-learn, odpowiedzialne za wielokrotne budowanie modeli w języku R.

Podczas walidacji krzyżowej pominięto etap selekcji atrybutów i poszukiwano najlepsze modele danych, podstawiając na przemian następujące parametry dyskretyzacji z biblioteki arules:

- metoda dyskretyzacji: frequency, cluster, interval,
- liczba podziałów: 2, 3, 4, 5.

Do oceny jakości klasyfikacji użyta została funkcja classification_report z pakietu scikit-learn oraz statystyki, które zebrała metoda walidacji krzyżowej, tj. średnią wartość dokładności klasyfikacj zbioru testowego i odchylenie standardowe dla każdego zbioru parametrów wejściowych.

Podczas walidacji krzyżowej z powtórzeniami dla każdego zestawu parametrów, zostały wyznaczone najlepsze modele oraz zebrane statystyki, tj. średnia wartość dokładności klasyfikacj zbioru testowego i odchylenie standardowe.

Zestaw parametrów dla którego średnia wartość dokładności klasyfikacji zbioru testowego była największa, został wybrany jako najlepszy, a wyznaczony na jej podstawie najlepszy model, został wybrany jako końcowy model danych wyznaczony za pomocą metody poszukiwania hiperparametrów połączonej z walidacją krzyżową oraz poddano go ostatecznej ocenie jakości klasyfikacji na podstawie całego zbioru danych.

4.2 Rezultaty

W wyniku badań nad bazą Dermatology z użyciem algorytmu LEM2 i metod pomocniczych zauważono, że odpowiedni dobór metody dyskretyzacji oraz liczba punktów podziału poprawił znacząco skuteczność klasyfikacji obiektów w stosunku do przedstawionych podobnych badań nad bazą Dermatology z użyciem algorytmu LEM2, bez potrzeby używania etapu selekcji atrybutów.

Przy podziale wejściowego zbioru w proporcji 90:10, nie stosując walidacji krzyżowej, uzyskano skuteczność równie wysoką, jak najlepsze algorytmy z zestawienia, które przygotował Zhang i in., tj. 99% w przypadku najlepszego modelu danych. Ocenę jakości trzech najlepszych modeli prezentuje tabela 4.1.

Podział zbioru w proporcji 80:20 wykazał dokładność klasyfikacji na poziomie 95% za pomocą miernika jakości skuteczności zrównoważonej. Podział ten obarczony jest mniejszym błędem generalizacji niż w pierwszym przypadku. Różnica jakości klasyfikacji wynosi 4%. Wyniki prezentuje tabela 4.2.

Dla modelu danych o skuteczności zrównoważonej ważonej równej 99%, z badania o identyfikatorze W20, została wyznaczona macierz pomyłek (tabela 4.3) wraz ze szczegółowymi wskaźnikami klasowymi (tabela 4.4). Macierz pomyłek potwierdza, że tylko jeden przypadek testowy został błędnie zaklasyfikowany.

Tabela 4.1: Ocena klasyfikacji najlepszych modeli dla podziału 90:10

	Model 1	**Model 2**	**Model 3**
Identyfikator badania (ID)	W3	W16	W20
Metoda dyskretyzacji cechy Age	frequency	interval	cluster
Liczba punktów podziału cechy Age	3	3	3
Skuteczność	0,975	0,8919	0,973
F1-score	0,96	0,857	1
Czułość ważona	0,98	0,89	1
Specyficzność ważona	0,99	0,96	1
PPV ważona	0,98	0,91	0,97
NPV ważona	1	0,98	0,99
Skuteczność zrównoważona ważona	0,98	0,93	0,99

Tabela 4.2: Ocena klasyfikacji najlepszych modeli dla podziału 80:20

	Model 1	**Model 2**	**Model 3**
Identyfikator badania (ID)	W32	W40	W47
Metoda dyskretyzacji cechy Age	interval	frequency	cluster
Liczba punktów podziału cechy Age	3	3	3
Skuteczność	0,919	0,904	0,9189
F1-score	0,857	0,93	0,96
Czułość ważona	0,94	0,88	0,87
Specyficzność ważona	0,97	0,96	0,98
PPV ważona	0,92	0,9	0,93
NPV ważona	0,98	0,98	0,99
Skuteczność zrównoważona ważona	0,95	0,92	0,93

Tabela 4.3: Macierz pomyłek dla badania o identyfikatorze W20

	Reference					
Prediction	**1**	**2**	**3**	**4**	**5**	**6**
1	12	0	0	0	0	0
2	0	6	0	0	0	0
3	0	0	6	0	0	0
4	0	1	0	4	0	0
5	0	0	0	0	6	0
6	0	0	0	0	0	2

Tabela 4.4: Wskaźniki klasowe macierzy pomyłek dla badania o identyfikatorze W20

	Class					
	1	**2**	**3**	**4**	**5**	**6**
Sensitivity	1.0000	0.8571	1.0000	1.0000	1.0000	1.00000
Specificity	1.0000	1.0000	1.0000	0.9697	1.0000	1.00000
PPV	1.0000	1.0000	1.0000	0.8000	1.0000	1.00000
NPV	1.0000	0.9677	1.0000	1.0000	1.0000	1.00000
Prevalence	0.3243	0.1892	0.1622	0.1081	0.1622	0.05405
Detection Rate	0.3243	0.1622	0.1622	0.1081	0.1622	0.05405
Detection Prevalence	0.3243	0.1622	0.1622	0.1351	0.1622	0.05405
Balanced Accuracy	1.0000	0.9286	1.0000	0.9848	1.0000	1.00000

Końcowe poszukiwania najlepszego modelu danych zostały przeprowadzone przy pomocy metody poszukiwania najlepszych parametrów dyskretyzacji w połączeniu z metodą walidacji krzyżowej z powtórzeniami. Pozwoliły one wyznaczyć model danych o skuteczności klasyfikacji równej 100%, dla parametrów:

- metoda dyskretyzacji: interval,
- liczba podziałów: 4.

Najlepszy klasyfikator został wybrany spośród wygenerowanych 2 400 modeli danych.

Średnia skuteczność walidacji dla najlepszych parametrów wyniosła 87,5% z odchyleniem standardowym +/-0.100 (dokładność klasyfikacji najsłabszego modelu wyniosła 70,27% - 34 reguły, o maksymalnej długości równej 8).

W tabeli Tabela 4.5 zostały przedstawione wyniki oceny klasyfikacji wszystkich modeli opartych o zestawy parametrów podczas walidacji krzyżowej.

Zbiór testowy dla najlepszego modułu zawierał 37 przypadków, w którym klasy od 1 do 6 pokrywały odpowiednio: 11,7,7,5,5,2 przypadków. 100% skuteczność klasyfikacji modelu nie wyklucza mocnego dopasowania do zbioru danych, jednak obserwując wygenerowane reguły przy użyciu algorytmu LEM2 należy stwierdzić, że model został znacząco uogólniony w stosunku do zbioru wejściowego - zmniejszyła się liczba reguł z 366 do 32 przy równoczesnym ograniczeniu długości reguł. Przykładowo, długość reguły nr 11 o wsparciu 82 (22,4%) wyniosła 5. Poza tym, reguły nr 12 i 15, o długości równej 1, pokrywają dużą liczbę zbioru - 55 i 47 przypadków.

Najlepszy model danych został wyznaczony na podstawie następujących reguł, uzyskanych podczas indukcji algorytmem LEM2:

1 (disappearance of the granular layer,0) & (band-like infiltrate,0) & (koebner phenomenon,0) & (knee and elbow involvement,0) & (elongation of the rete ridges,0) & (hyperkeratosis,0) & (scaling,2) -> (class,2)

Tabela 4.5: Ocena jakości klasyfikacji hiperparametrów

Lp	Metoda dyskretyzacji	Liczba podziałów	Średnia skuteczność	Odchylenie standardowe
1	frequency	2	0.866	+/-0.099
2	frequency	3	0.870	+/-0.095
3	frequency	4	0.868	+/-0.097
4	frequency	5	0.869	+/-0.095
5	cluster	2	0.872	+/-0.104
6	cluster	3	0.870	+/-0.097
7	cluster	4	0.869	+/-0.101
8	cluster	5	0.871	+/-0.097
9	interval	2	0.872	+/-0.103
10	interval	3	0.862	+/-0.098
11	interval	4	0.875	+/-0.100
12	interval	5	0.869	+/-0.098

2 (fibrosis of the papillary dermis,0) & (disappearance of the granular layer,0) & (band-like infiltrate,0) & (koebner phenomenon,0) & (knee and elbow involvement,0) & (parakeratosis,0) -> (class,2)
3 (parakeratosis,2) & (acanthosis,2) & (erythema,2) & (spongiosis,3) -> (class,2)
4 (fibrosis of the papillary dermis,0) & (disappearance of the granular layer,0) & (band-like infiltrate,0) & (perifollicular parakeratosis,0) & (koebner phenomenon,0) & (scalp involvement,0) & (hyperkeratosis,0) & (PNL infiltrate,0) -> (class,2)
5 (koebner phenomenon,0) & (spongiosis,2) & (scaling,3) -> (class,2)
6 (disappearance of the granular layer,0) & (acanthosis,2) & (thinning of the suprapapillary epidermis,0) & (PNL infiltrate,1) -> (class,2)
7 (PNL infiltrate,2) & (knee and elbow involvement,0) & (age,(37.5,56.2]) & (parakeratosis,2) -> (class,2)
8 (fibrosis of the papillary dermis,0) & (koebner phenomenon,0) & (disappearance of the granular layer,0) & (band-like infiltrate,0) & (eosinophils in the infiltrate,1) -> (class,2)
9 (acanthosis,1) & (band-like infiltrate,2) -> (class,2)
10 (PNL infiltrate,1) & (knee and elbow involvement,1) & (focal hypergranulosis,0) & (elongation of the rete ridges,0) -> (class,2)
11 (spongiosis,0) & (fibrosis of the papillary dermis,0) & (eosinophils in the infiltrate,0) & (follicular papules,0) & (exocytosis,0) -> (class,1)
12 (thinning of the suprapapillary epidermis,2) -> (class,1)
13 (spongiosis,0) & (polygonal papules,0) & (oral mucosal involvement,0) & (fibrosis of the papillary dermis,0) & (perifollicular parakeratosis,0) & (definite borders,2) -> (class,1)

14 (definite borders,3) & (polygonal papules,0) & (fibrosis of the papillary dermis,0) -> (class,1)
15 (band-like infiltrate,3) -> (class,3)
16 (band-like infiltrate,2) & (scalp involvement,0) -> (class,3)
17 (band-like infiltrate,2) & (parakeratosis,2) -> (class,3)
18 (fibrosis of the papillary dermis,0) & (knee and elbow involvement,0) & (scalp involvement,0) & (PNL infiltrate,0) & (hyperkeratosis,0) & (itching,0) -> (class,4)
19 (fibrosis of the papillary dermis,0) & (family history,0) & (saw-tooth appearance of retes,0) & (PNL infiltrate,0) & (erythema,2) & (exocytosis,2) & (parakeratosis,1) -> (class,4)
20 (saw-tooth appearance of retes,0) & (PNL infiltrate,0) & (scaling,2) & (definite borders,2) & (exocytosis,3) -> (class,4)
21 (saw-tooth appearance of retes,0) & (eosinophils in the infiltrate,0) & (inflammatory monoluclear inflitrate,2) & (spongiosis,2) & (erythema,1) & (scaling,1) -> (class,4)
22 (inflammatory monoluclear inflitrate,2) & (age,(18.8,37.5]) & (disappearance of the granular layer,0) & (scaling,2) & (hyperkeratosis,2) -> (class,4)
23 (spongiform pustule,0) & (saw-tooth appearance of retes,0) & (hyperkeratosis,0) & (age,(18.8,37.5]) & (disappearance of the granular layer,0) & (koebner phenomenon,1) -> (class,4)
24 (eosinophils in the infiltrate,0) & (definite borders,0) & (spongiosis,3) & (parakeratosis,2) -> (class,4)
25 (thinning of the suprapapillary epidermis,0) & (saw-tooth appearance of retes,0) & (disappearance of the granular layer,1) -> (class,4)
26 (itching,0) & (erythema,1) & (koebner phenomenon,2) -> (class,4)
27 (PNL infiltrate,0) & (koebner phenomenon,0) & (band-like infiltrate,0) & (knee and elbow involvement,0) & (clubbing of the rete ridges,0) & (spongiosis,0) -> (class,5)
28 (PNL infiltrate,0) & (koebner phenomenon,0) & (disappearance of the granular layer,0) & (band-like infiltrate,0) & (follicular horn plug,0) & (scaling,1) -> (class,5)
29 (fibrosis of the papillary dermis,1) -> (class,5)
30 (fibrosis of the papillary dermis,0) & (koebner phenomenon,0) & (munro microabcess,0) & (age,(0,18.8]) & (PNL infiltrate,0) -> (class,6)
31 (perifollicular parakeratosis,2) -> (class,6)
32 (knee and elbow involvement,3) & (follicular papules,2) -> (class,6)

Tabela 4.6: Wskaźniki oceny reguł

Nr reguły	Dł. reguły	Pokrycie	Laplace	RI	Confidence
1	7	23	6.28	0.827586	1
2	6	19	5.19	0.8	1
3	4	3	0.82	0.444444	1
4	8	9	2.46	0.666667	1
5	3	7	1.91	0.615385	1
6	4	13	3.55	0.736842	1
7	4	3	0.82	0.444444	1
8	5	15	4.1	0.761905	1
9	2	1	0.27	0.285714	1
10	4	2	0.55	0.375	1
11	5	82	22.4	0.943182	1
12	1	55	15.03	0.918033	1
13	6	66	18.03	0.930556	1
14	3	24	6.56	0.833333	1
15	1	47	12.84	0.90566	1
16	2	16	4.37	0.772727	1
17	2	7	1.91	0.615385	1
18	6	20	5.46	0.807692	1
19	7	11	3.01	0.705882	1
20	5	4	1.09	0.5	1
21	6	3	0.82	0.444444	1
22	5	1	0.27	0.285714	1
23	6	6	1.64	0.583333	1
24	4	2	0.55	0.375	1
25	3	15	4.1	0.761905	1
26	3	2	0.55	0.375	1
27	6	33	9.02	0.871795	1
28	6	31	8.47	0.864865	1
29	1	7	1.91	0.615385	1
30	5	14	3.83	0.75	1
31	1	11	3.01	0.705882	1
32	2	1	0.27	0.285714	1

Podsumowanie

Celem niniejszej pracy było lepsze poznanie możliwości i wpływu metod, przetwarzających dane przed rozpoczęciem procesu indukcji reguł. Badania przeprowadzono na podstawie medycznej bazy danych Dermatology, przechowującej informacje o pacjentach chorych na 6 różnych chorób skóry. Przed rozpoczęciem badań zapoznano się również z innymi opracowaniami na ten temat stwierdzając, iż ich liczba jest niewielka w stosunku do badań nad innymi metodami uczenia maszynowego.
W niniejszej pracy poprawiono znacząco uzyskane do tej pory wyniki, uzyskując rozwiązanie o 100% skuteczności klasyfikacji. Wyniki potwierdziły, że algorytm

LEM2 może być bardzo skuteczny, ale trzeba zachować szczególną uwagę, w jaki sposób przygotowuje się dla niego dane. Z przedstawionych rezultatów wynika, że mierniki jakościowe ważone dokładniej opisują skuteczność klasyfikacji reguł opartych o algorytm LEM2, a zatem również wpływ metod odpowiedzialnych za przygotowanie danych dla algorytmu LEM2. Badania potwierdziły dokładność znajdowania modeli danych o dużej skuteczności klasyfikacji za pomocą przeszukiwania przestrzeni parametrów przy równoczesnym zastosowaniu walidacji krzyżowej z powtórzeniami.

Wprowadzenie do oceny jakości klasyfikacji mierników ważonych pozwoliło spojrzeć na bazę Dermatology i możliwości algorytmu LEM2 w nowy sposób, niespotykany dotąd w literaturze. Z uwagi na dysproporcje w liczbie obiektów w każdej z klas, mierniki ważone, a szczególnie ważony miernik skuteczności zrównoważonej (ang. balanced accuracy), wydają się bardziej adekwatne do oceny jakości tego typu modeli danych. W przypadku modelu o 100% skuteczności, nie miały jednak większego znaczenia.

Używając wyników tej pracy, autor zamierza kontynuować badania na większych zbiorach danych i doskonalić metodę oceny jakości klasyfikacji za pomocą wskaźników ważonych. Baza Dermatology jest znormalizowana przez jej autorów, więc dostęp do pierwotnego zbioru danych mógłby umożliwić wyznaczenie lepszych wartości dyskretyzacji.

Bibliografia

Borowik, G. (2019). *Methods and algorithms of logic synthesis in data analysis and data mining*. Unpublished doctoral dissertation. Źródło: `https://wcy.wat.edu.pl/sites/default/files/gb_autoreferat_en.pdf`, dostęp: 04.12.2020.

Borowik, G., Kraśniewski, A. i Łuba, T. (2015). Rule Induction Based on Logic Synthesis Methods. *Advances in Intelligent Systems and Computing*, *1089*, 813–816.

Cao, X. H., Stojkovic, I. i Obradovic, Z. (2016). A robust data scaling algorithm to improve classification accuracies in biomedical data. *BMC Bioinformatics*, *17*, 359.

Didkowska, J., Wojciechowska, U., Czderny, K., Olasek, P. i Ciuba, A. (2019). *Nowotwory złośliwe w Polsce w 2017 roku.* Źródło: `http://onkologia.org.pl/wp-content/uploads/nowotwory_2017.pdf`, dostęp: 04.12.2020.

Didkowska, J., Wojciechowska, U. i Zatorski, W. (2009). *Prognozy zachorowalności i umieralności na nowotwory złośliwe w Polsce do 2025 roku.*

Dua, D., i Graff, C. (2017). *UCI machine learning repository.* Źródło:

https://archive.ics.uci.edu/ml/datasets/Dermatology, dostęp: 04.12.2020.

Dziura, J. D., Post, L. A., Zhao, Q., Fu, Z. i Peduzzi, P. (2013). Strategies for dealing with missing data in clinical trials: from design to analysis. *The Yale journal of biology and medicine*, *86*, 343–58.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*, 861–874.

Foundation, S. C. (2020). *Skin cancer 101.* Źródło: https://www.skincancer.org/skin-cancer-information/, dostęp: 04.12.2020.

Grzymala-Busse, J. W. (1992). LERS-A System for Learning from Examples Based on Rough Sets. W: R. Slowinski (red.), *Intelligent decision support. handbook of applications and advances of the rough sets theory,* Springer, 3–18.

Grzymala-Busse, J. W. (1997). A New Version of the Rule Induction System LERS. *Fundamenta Informaticae*, *31*, 27–39.

Kannan, K. S., Manoj, K. i Arumugam, S. (2015). Labeling Methods for Identifying Outliers. *International Journal of Statistics and Systems(IJSS)*, *10*, 231–238.

Khare, R., Utidjian, L., Ruth, B. J., Kahn, M. G., Burrows, E., Marsolo, K., Patibandla, N., Razzaghi, H., Colvin, R., Ranade, D., Kitzmiller, M., Eckrich, D. i Bailey, L. C. (2017). A longitudinal analysis of data quality in a large pediatric data research network. *Journal of the American Medical Informatics Association*, *24*, 1072–1079.

Koti, M. S. (2014). *RST Approach for the Prediction of Rules and Cost Effective Feature Selection in Medical Data*. Unpublished doctoral dissertation, Bharathiar University. Źródło: http://hdl.handle.net/10603/97869, dostęp: 04.12.2020.

Kusunoki, Y., i Inuiguchi, M. (2006). Rule Induction Via Clustering Decision Classes. W: S. Greco i in. (red.), *Rough sets and current trends in computing,* Springer, 928–938.

Little, R. J., D'Agostino, R., Cohen, M. L., Dickersin, K., Emerson, S. S., Farrar, J. T., Frangakis, C., Hogan, J. W., Molenberghs, G., Murphy, S. A., Neaton, J. D., Rotnitzky, A., Scharfstein, D., Shih, W. J., Siegel, J. P. i Stern, H. (2012). The Prevention and Treatment of Missing Data in Clinical Trials. *New England Journal of Medicine*, *367*, 1355–1360.

Manliguez, C. (2016). Generalized Confusion Matrix for Multiple Classes. *Machine Learning*.

O'Neill, R. T., i Temple, R. (2012). The Prevention and Treatment of Missing Data in Clinical Trials: An FDA Perspective on the Importance of Dealing With It. *Clinical Pharmacology & Therapeutics*, *91*, 550–554.

Pawlak, Z. (1980). Toward the Theory of Information Systems. W: *CS PAS Reports 419/80,* 1–35.

Pawlak, Z. (1991). *Rough Sets Theoretical Aspects of Reasoning about Data*. Springer, Dordrecht. Źródło: https://bcpw.bg.pw.edu.pl/Content/

1845/download/, dostęp: 04.12.2020.

Pawlak, Z. (2005). A Treatise on Rough Sets. W: *Transactions on Rough Sets IV,* Springer, 1–17.

Pezoulas, V. C., Kourou, K. D., Kalatzis, F., Exarchos, T. P., Venetsanopoulou, A., Zampeli, E., Gandolfo, S., Skopouli, F., De Vita, S., Tzioufas, A. G. i Fotiadis, D. I. (2019). Medical data quality assessment: On the development of an automated framework for medical data curation. *Computers in Biology and Medicine*, *107*, 270–283.

Srimani, P. K., i Koti, M. S. (2014). Knowledge discovery in medical data by using rough set rule induction algorithms. *Indian Journal of Science and Technology*, *7*, 905–915.

Stepaniuk, J. (2008). *Rough – Granular Computing in Knowledge Discovery and Data Mining* (152). Berlin, Heidelberg: Springer Berlin Heidelberg.

Thabane, L., Mbuagbaw, L., Zhang, S., Samaan, Z., Marcucci, M., Ye, C., Thabane, M., Giangregorio, L., Dennis, B., Kosa, D., Debono, V. B., Dillenburg, R., Fruci, V., Bawor, M., Lee, J., Wells, G. i Goldsmith, C. H. (2013). A tutorial on sensitivity analyses in clinical trials: The what, why, when and how. *BMC Medical Research Methodology*, *13*, 92.

Tremblay, M. C., Hevner, A. R. i Berndt, D. J. (2012). Design of an information volatility measure for health care decision making. *Decision Support Systems*, *52*, 331–341.

Wojciechowska, U., i Didkowska, J. (2020). Zachorowania i zgony na nowotwory złośliwe w Polsce - Czerniak skóry (C43). Lata 1965-2010. *Krajowy Rejestr Nowotworów, Narodowy Instytut Onkologii im. Marii Skłodowskiej-Curie – Państwowy Instytut Badawczy*. Źródło: http://onkologia.org.pl/czerniak-skory-c43/, dostęp: 04.12.2020.

Zhang, C., Liu, C., Zhang, X. i Almpanidis, G. (2017, oct). An up-to-date comparison of state-of-the-art classification algorithms. *Expert Systems with Applications*, *82*, 128–150.

Politechnika Białostocka