Akademia Górniczo-Hutnicza im. Stanisława Staszica
Wydział Elektrotechniki, Automatyki, Informatyki
i Elektroniki

Rozprawa doktorska

Pozyskiwanie wiedzy z dużych zbiorów danych z zastosowaniem adaptacyjnych procedur generowania zapytań

Bartosz Jędrzejec

Promotor: prof. dr hab. inż. Edward Nawarecki

Kraków, 2008

Składam szczególne podziękowania Panu Prof. Edwardowi Nawareckiemu za pomoc i wsparcie, dzięki którym ta praca mogła powstać.

Chciałbym również podziękować Panu Dr hab. Marianowi Wysockiemu oraz Pani Dr hab. Barbarze Dębskiej za przychylność i cenne uwagi, które pomogły w przygotowaniu tej pracy.

Praca powstała dzięki wsparciu finansowemu Ministerstwa Nauki i Szkolnictwa Wyższego w postaci projektu badawczego promotorskiego nr N516 026 31/2545.

Spis treści

Wstęp

Wraz z rozwojem systemów informatycznych następuje stały wzrost ilości generowanych, przechowywanych i przetwarzanych danych. Duża objętość zasobów przechowywanych danych, często rzędu wielu tysięcy, a nawet milionów rekordów, powoduje trudności w wydobywaniu z tych danych informacji użytecznych z punktu widzenia użytkowników (lekarzy, menadżerów, ekonomistów, technologów). Z tego względu, w ostatnich latach wzrosło zapotrzebowanie na metody i narzędzia do efektywnego pozyskiwania informacji ukrytych w bazach i hurtowniach danych.

Jednym z najważniejszych problemów, jaki należy w tym zakresie rozwiązać, jest opracowanie odpowiednich modeli drążenia danych oraz dobór wartości parametrów tych modeli. Model zbyt skomplikowany powoduje utrudnienia w odkrywaniu wiedzy istotnej z punktu widzenia analityka. Natomiast model uproszczony jest wprawdzie łatwy w analizie, lecz pewne istotne dla analityka dane i informacje mogą zostać w nim nie uwzględnione. Z tego powodu poszukuje się nowych metod umożliwiających generowanie modeli uproszczonych, lecz zorientowanych na opis tylko wybranych zależności interesujących analityka dziedzinowego (użytkownika systemu). Uzyskanie uproszczenia modelu jest możliwe poprzez:

1. nałożenie ograniczeń przed stworzeniem modelu, lub
2. dokonanie uproszczeń w modelu już wygenerowanym.

Celem niniejszej pracy jest stworzenie metodyki i procedur analizy dużych, wygenerowanych w oparciu o bazę danych, modeli asocjacyjnych. Modele te, zapisane w wywodzącym się z XML standardzie PMML, mogą być w łatwy sposób analizowane przy pomocy zapytań w języku XQuery. Prawidłowe sformułowanie zapytania w języku zbliżonym do naturalnego powinno generować właściwy podzbiór reguł, który analityk może wykorzystać w procesie wnioskowania na poparcie postawionej przez siebie hipotezy.

W pracy zaproponowano metodę, która w oparciu o algorytm programowania genetycznego poszukuje coraz to lepszych zapytań do modelu reguł asocjacyjnych. Dzięki temu podzbiory reguł zwracane przez zapytania są ograniczane do tych, które są interesujące z punktu widzenia analizy. Dodatkowo liczba zwracanych reguł jest na tyle mała, że są one łatwe do szybkiego przeanalizowania.

Bardziej precyzyjne określenie zamierzeń i zawartości pracy oraz sformułowanie przyjętej tezy przedstawione zostaną po dokonaniu przeglądu podstawowych pojęć i scharakteryzowaniu aktualnego stanu odnośnej dziedziny wiedzy.

Rozdział 1

Przegląd pojęć i metod

1.1 Dane w systemach komputerowych

Pojęcie zbioru danych nieodłącznie związane jest z rozwojem informatyki. Dane dotyczące programów oraz wyniki obliczeń wymagają odpowiedniej formy przechowywania w tzw. pamięci nieulotnej (masowej) oraz funkcji umożliwiających do nich dostęp i zarządzanie w sposób szybki i prosty.

W początkowym okresie rozwoju systemów komputerowych jako pamięci masowe wykorzystywane były karty perforowane, a następnie pamięci taśmowe. Na pojedynczy zbiór danych składało się często kilka tysięcy kart, a zarządzanie zbiorami danych w tym czasie polegało na odpowiednim ich składowaniu i katalogowaniu. Dostęp do przechowywanych danych odbywał się w sposób sekwencyjny.

Wraz z wprowadzeniem w 1951 roku taśm magnetycznych oraz, 5 lat później przez firmę IBM [82] pierwszego dysku twardego, zbiory danych zaczęły przybierać formę znanych dzisiaj plików. Pojedynczy plik przechowywał dane w postaci tabeli rekordów i był obsługiwany przez aplikację, która była jednocześnie odbiorcą, dostarczycielem i zarządcą danych. Rozwiązanie to niosło za sobą wiele ograniczeń i niedogodności, takich jak:

1. separacja i izolacja danych – dostęp do danych wymagał częstego przetwarzania i synchronizacji wielu plików, a zatem czas dostępu do danych znacząco się wydłużał,
2. redundancja danych – nadmiarowość danych związana z brakiem odpowiedniego systemu zarządzania, co mogło doprowadzić do utraty ich integralności oraz znacznego wzrostu objętości,
3. zależność danych od aplikacji – definicja struktury danych występowała w obsługującej je aplikacji, jakakolwiek jej zmiana wymuszała konieczność zmian w oprogramowaniu,
4. niekompatybilność formatów zapisu danych – brak możliwości przenoszenia danych pomiędzy aplikacjami różnych użytkowników [21].

Ograniczenia te były motywacją do podjęcia prac nad odpowiednio efektywnymi strukturami i metodami przechowywania i dostępu do danych.

Prace te zaowocowały, w 1961 roku, pierwszym systemem zarządzania bazą danych IDS (Integrated Data Store), stworzonym przez Charlesa Bachman'a. Umożliwiał on dostęp tylko do jednego pliku danych, a wszystkie operacje na danych wymagały ręcznego wprowadzania kodu.

W tym okresie powstały dwa typy modeli danych – zaproponowany przez firmę IBM, hierarchiczny model danych IMS (Information Management System) [54] oraz oparty na pomyśle Banchman'a, wprowadzony przez firmę CODASYL – sieciowy model danych [61].

Hierarchiczny model danych charakteryzuje się tym, iż struktura danych ma postać drzewa, a relacje pomiędzy danymi są typu nadrzędny-podrzędny. Zaletą modelu hierarchicznego jest jego czytelna struktura danych, natomiast znaczącą wadą, z punktu widzenia konceptualnego modelowania danych, jest brak możliwości przedstawienia w nim związków typu wiele-do-wielu. Model ten znalazł zastosowanie w organizacji systemów plików.

Sieciowy model danych stanowi rozszerzenie modelu hierarchicznego. W modelu tym związki pomiędzy danymi reprezentowane są poprzez powiązania wskaźnikowe, a struktura danych przyjmuję formę sieci (grafu).

Wadą modelu sieciowego jest duża uciążliwość tworzenia aplikacji w zaimplementowanych w tym modelu językach: opisu danych DDL i manipulacji danymi DML.

W 1970 roku Edgar F. Codd zaproponował oparty na podstawach matematycznych relacyjny model danych [20]. W modelu tym dane reprezentowane są w postaci zbioru krotek zwanego relacją, a dostęp do danych odbywa się przy pomocy operatorów relacyjnych, takich jak: selekcja, rzutowanie, złączenie i inne. Głównymi zaletami, które doprowadziły do sukcesu tego modelu danych, jako podstawy dla operacyjnych baz danych, są: niezależność danych od aplikacji oraz, dzięki formalizmowi matematycznemu stanowiącego jego podstawę, możliwość stosowania algorytmów optymalizujących zapytania do bazy danych. Dla potrzeb realizacji zapytań do bazy relacyjnej powstał w latach 70-tych język zapytań SQL (Structured Query Language), który jest językiem wysokiego poziomu, umożliwiającym definicję, manipulację, jak i kontrolę danych.

Wraz z rozwojem technologii gromadzenia danych, w szczególności wprowadzenia do powszechnego użycia kodów paskowych, powstało zapotrzebowanie na systemy, które nie tylko wspierałyby działania operacyjne przedsiębiorstw, ale również przechowywałyby dane historyczne dla potrzeb wspomagania procesów analizy i podejmowania decyzji. Gromadzenie i przechowywanie danych historycznych, często o wieloletnim zasięgu, zaowocowało zwiększeniem wielkości przechowujących je zbiorów danych oraz zapotrzebowaniem na struktury i narzędzia, które umożliwiałyby ich przetwarzanie i analizę. Rozwiązaniem, w tym zakresie, stały się hurtownie danych (ang. *data warehouse*) oraz składy danych (ang. *data marts*).

Jedna z najbardziej rozpowszechnionych definicji hurtowni danych pochodzi od W. H. Inmona [42]. Traktuje ona hurtownię danych jako zbiór zintegrowanych, zorientowanych tematycznie, niezmieniających się w czasie danych, dedykowany dla potrzeb wspierania procesów zarządzania.

Różnica pomiędzy hurtownią danych a składem danych polega na tym, że skład danych obejmuje pewien fragment danych przedsiębiorstwa i wykorzystywany jest przez dział lub filię organizacji, natomiast hurtownia danych przechowuje informacje przeznaczone do całościowej analizy danych przedsiębiorstwa [43].

Narzędziem wykorzystywanym w analizie danych pochodzących z hurtowni danych jest technologia OLAP (On-Line Analytical Processing), która umożliwia dynamiczną syntezę, analizę i konsolidację dużych ilości wielowymiarowych danych. Dzięki tej technice możliwe jest tworzenie podsumowań dla danych oraz przeglądanie tych agregacji z różnych punktów widzenia – hierarchicznych wymiarów. Tworzone podsumowania dają ogólną wiedzę na temat danych i zależności pomiędzy nimi, natomiastbardziej szczegółowe zależności możliwe są do uzyskania w procesie drążenia danych (który jest częścią procesu pozyskiwania wiedzy (1.2)), dla którego hurtownia lub skład danych stanowi źródło danych.

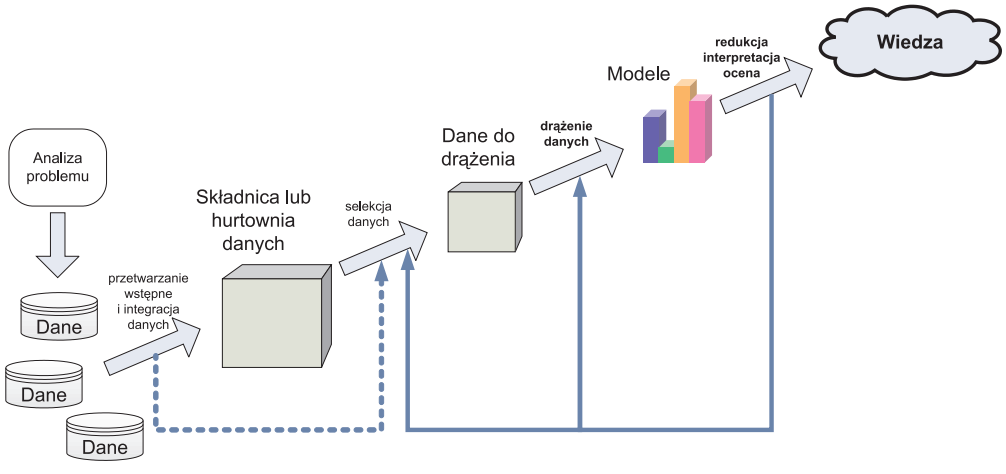
1.2 Pozyskiwanie wiedzy ze zbiorów danych

W okresie ostatnich kilkunastu lat możemy zaobserwować dynamiczny rozwój badań prowadzonych w dziedzinie gromadzenia i przetwarzania danych. Dziś zbiory danych osiągające rozmiary rzędu gigabajtów, a nawet terabajtów, nie są już niczym szczególnym. Tak duże ilości danych występują w wielu dziedzinach, w szczególności takich jak: astronomia, telekomunikacja, medycyna, marketing [4, 28, 74, 95]. Nie można w tym miejscu nie wspomnieć o największym zbiorze danych na świecie, jakim niewątpliwie są zasoby światowej sieci Internet.

W sytuacjach, gdy tradycyjne techniki wykorzystywane do przetwarzania danych, takie jak języki zapytań czy narzędzia statystyczne, przestały być wystarczające, koniecznością stało się opracowania nowych metod, odpowiadającym aktualnym rozmiarom zbiorów danych. Można powiedzieć, iż

ta „powódź" danych stanowi jedno z większych wyzwań informatycznych, jakie pojawiły się przed specjalistami w ostatnim okresie.

Jednym z problemów z jakim trzeba się zmierzyć jest znalezienie sposobu, w jaki te gromadzone dane mogą być wykorzystane w procesach wnioskowania i podejmowania decyzji. Innymi słowy, jak z danych pozyskać wiedzę niezbędną dla analityków i decydentów. Działania zmierzające do rozwiązania tego problemu noszą w literaturze nazwę pozyskiwania lub odkrywania wiedzy w zbiorach danych (ang. *KDD – Knowledge Discovery in Databases*) Można też spotkać inne określenia, takie jak ekstrakcja wiedzy (ang. *knowledge extraction*) czy, dość często spotykane, drążenie danych (ang. *data mining*). W tym ostatnim przypadku bywa ono mylnie interpretowane jako część procesu pozyskiwania wiedzy [26]. Głównymi prekursorami tej dziedziny badań są Usama Fayyad i Gregory Piatetsky-Shapiro, którzy zdefiniowali odnośne podejście [27] jako: *nietrywialny, zautomatyzowany proces odkrywania nowych, istotnych, potencjalnie użytecznych i zrozumiałych wzorców wśród danych.* Na proces odkrywania wiedzy składa się wiele dziedzin i dyscyplin naukowych, takich jak: statystyka, uczenie maszynowe, sztuczna inteligencja, optymalizacja, bazy danych czy też obliczenia równoległe.



Rysunek 1.1: Proces pozyskiwania wiedzy ze zbiorów danych (na podstawie [35])

Pozyskiwanie wiedzy przebiega w kilku etapach (rys. 1.1), realizowanych niejednokrotnie w sposób rekurencyjny. Są to:

1. **analiza problemu** – ma na celu poznanie rozważanej dziedziny w celu odpowiedniego doboru technik i rodzaju wyników procesu pozyskiwania wiedzy;

2. **przetwarzanie wstępne i integracja danych** – na tym etapie następuje pozyskanie danych z różnych źródeł (m.in. z baz operacyjnych przedsiębiorstwa, hurtowni danych i baz zewnętrznych, np. informacje giełdowe, itp.) oraz wstępne przetwarzanie danych, w skład którego wchodzą:

   - wybór danych, które będą przydatne w analizie,
   - czyszczenie danych z szumów i wartości daleko odbiegających od normy przyjętej dla obiektów danej klasy (ang. *outliers*),
   - normalizacja danych,
   - interpretacja danych niepełnych lub błędnych,
   - odwzorowanie danych, w zależności od przewidywanego typu modelu drążenia, np. z wartości numerycznych na skategoryzowane lub odwrotnie.

   Proces przygotowania danych do analizy jest często procesem czasochłonnym, może zajmować 50%-90% czasu całego procesu pozyskiwania wiedzy, jednak prawidłowa realizacja tego ważnego kroku daje możliwość budowy modeli drążenia danych, które będą spełniać stawiane im wymagania [65, 87]. Odpowiednio przetworzone dane są przechowywane w składnicach lub hurtowniach danych lub też są bezpośrednio składowane w roboczym zbiorze danych przeznaczonym dla procesu drążenia danych;

3. **eksploracja danych** – właściwy proces budowy modelu(i) drążenia – zbioru(ów) wzorców, który składa się z następujących etapów zależnych od rozwiązywanego problemu:
   - wybór danych (atrybutów) dla procesu drążenia,
   - dobór odpowiedniego typu wzorców i metody ich pozyskania,
   - dobór wartości progowych dla algorytmu drążenia danych.

   Najczęściej wykorzystywanymi typami wzorców (modeli) pozyskiwanych w procesie drążenia są: reguły asocjacyjne i drzewa decyzyjne. Model może powstać również w wyniku przeprowadzenia algorytmu grupowania (klasteryzacji – ang. *clustering*) lub przez dopasowanie funkcji regresji;

4. **przygotowanie wyników dla potrzeb analizy i oceny** – aby wzorce mogły być zrozumiałe dla analityka, muszą zostać przetworzone z formy bezpośrednio uzyskanej w procesie drążenia do takiej, która będzie łatwa w interpretacji i ocenie. W wielu przypadkach liczba wygenerowanych wzorców jest trudna do „ogarnięcia" przez użytkownika i wymaga redukcji.

Problem automatycznej redukcji modeli drążenia danych (etap 4 procesu pozyskiwania wiedzy), w szczególności modeli reguł asocjacyjnych, jest głównym zagadnieniem rozpatrywanym w niniejszej pracy i zostanie szczegółowo omówiony w dalszej jej części.

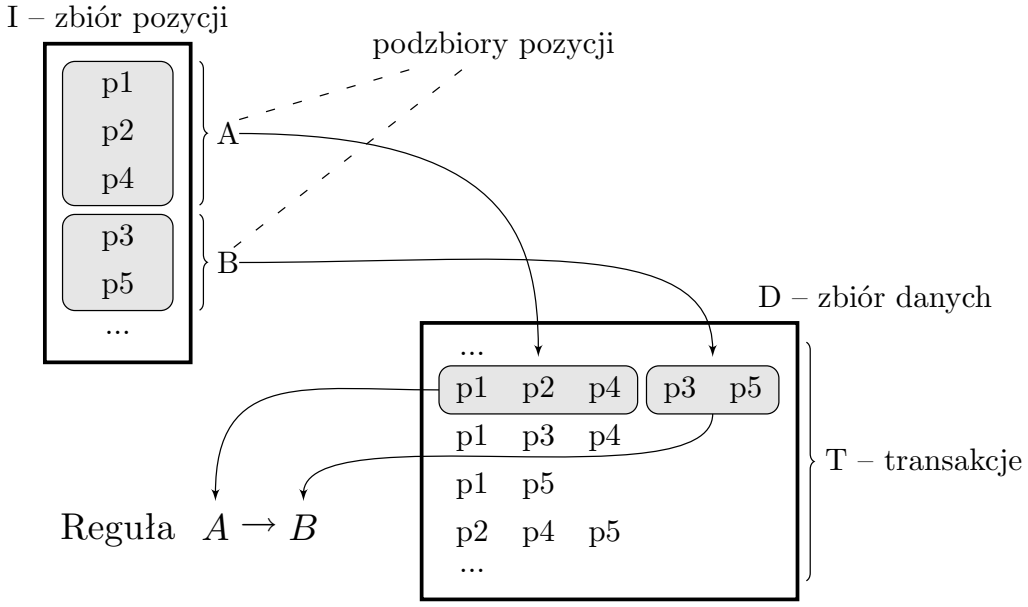
1.3 Modele drążenia danych

1.3.1 Model reguł asocjacyjnych

Model oparty na wykorzystaniu reguł asocjacyjnych jest jednym z ważniejszych typów modeli drążenia danych [2, 34].

Reguły asocjacyjne pozyskiwane są poprzez analizę zbioru danych D (rys. 1.2), który składa się z rekordów T, często nazywanym zbiorem

transakcji. Każda transakcja składa się z pozycji ze zbioru I. Dla każdej transakcji T i podzbioru pozycji A, mówimy, że T zawiera A, gdy $A \subseteq T$.



Rysunek 1.2: Pozyskiwanie reguł asocjacyjnych

Regułę asocjacyjną można przedstawić w postaci (1.1) (przy czym zapis ten należy traktować jako pewnego rodzaju powiązanie, a nie jako implikację matematyczną).

$$A \rightarrow B, \text{ przy } A \cap B = \oslash, \tag{1.1}$$

gdzie:

A, B – podzbiory pozycji, które pochodzą z pewnego zbioru pozycji I.

Dla określenia w jakim stopniu dana reguła $A \rightarrow B$ jest interesująca, najczęściej stosuje się miary tzw. wsparcia i zaufania reguły. Definiuje się je w sposób następujący:

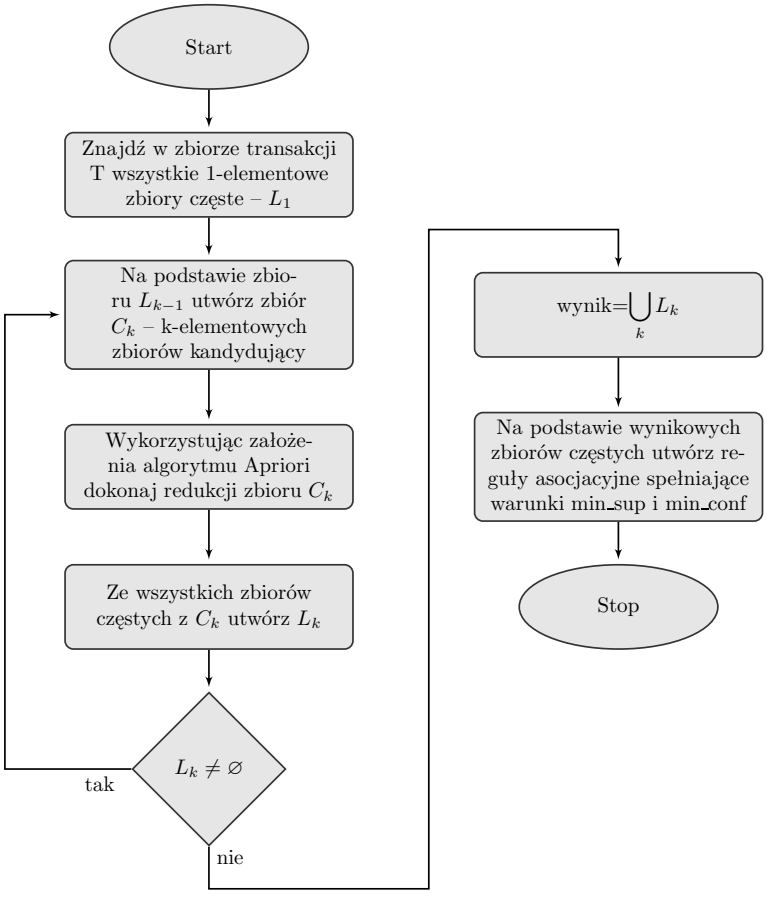
Definicja 1. *Reguła $A \rightarrow B$ w zbiorze transakcji D posiada wsparcie sup, gdzie sup określa procentowo liczbę transakcji w zbiorze D, które zawierają podzbiór pozycji A i podzbiór pozycji B*

Definicja 2. *Reguła $A \rightarrow B$ posiada zaufanie conf w zbiorze transakcji D, gdzie conf określa procentowo liczbę transakcji, które zawierają podzbiór pozycji A oraz podzbiór B.*

W skład reguły wchodzą dwa podzbiory pozycji, które muszą być tzw. zbiorami częstymi.

Definicja 3. *Zbiorem częstym nazywamy podzbiór pozycji, który spełnia warunek minimalnego wsparcia.*

Progowe wartości minimalnego wsparcia i zaufania są powszechnie wykorzystywane do kontroli procesu budowy modelu reguł asocjacyjnych (rys 1.3).



Rysunek 1.3: Proces budowy modelu reguł asocjacyjnych

L_1 = {zbiory częste 1-elementowe};
for *(k=2;*$L_{k-1} \neq \varnothing$*;k++)* **do**
begin
C_k = apriori-gen(L_{k-1});
forall *transakcji* $t \in T$ **do**
begin
C_t = subset(C_k,t);
forall *zbiorów kandydujących* $c \in C_t$ **do** c.count++;
end
$L_k = \{c \in C \mid \text{c.count} \geqslant \text{minsup}\}$;
end
Wynik = $\bigcup_k L_k$

Algorytm 1.1: Algorytm Apriori

W procesie tym, do generowania zbiorów częstych wykorzystuje się najczęściej zaproponowany w [3] Algorytm *Apriori*. Algorytm ten (algorytm 1.1) opiera się na twierdzeniu mówiącym, że jeśli dany podzbiór pozycji nie jest częsty (ma wsparcie mniejsze od minimalnego) to każdy zbiór pozycji go zawierający również nie jest częsty. W algorytmie tym w pierwszej kolejności wyszukiwane są jednoelementowe zbiory częste, następnie zaś iteracyjnie tworzone są kolejne n+1 elementowe zbiory częste z n-elementowych. Z tak uzyskanych zbiorów tworzone są reguły asocjacyjne. Algorytm *Apriori* jest kompletny tzn. znajduje wszystkie reguły, których wsparcie jest większe od zadanych wartości progowych, co stanowi zarówno jego zaletę, jak i wadę, gdyż w wielu przypadkach generowana przez niego liczba reguł sięga kilku, a nierzadko kilkudziesięciu tysięcy.

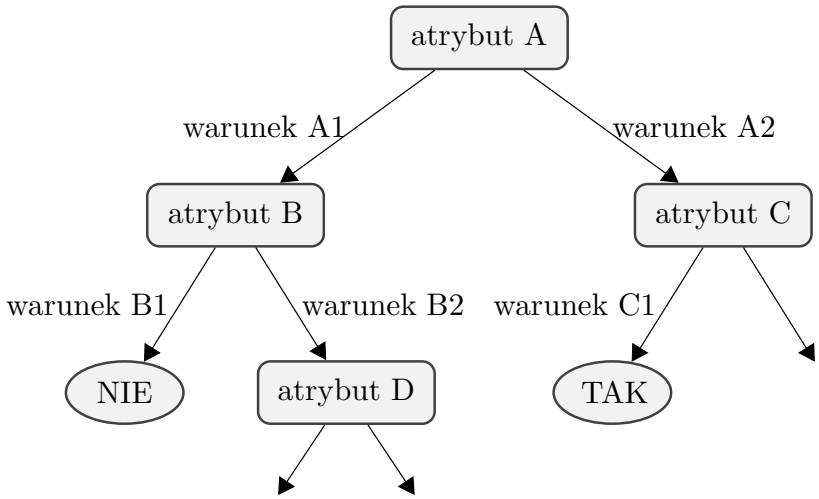
Przedstawiony algorytm odkrywania reguł asocjacyjnych uchodzi za standardowy, ale są znane również jego modyfikacje m.in. w [3, 13, 75, 62, 90]. Zmierzają one do poprawienia efektywności obliczeniowej lub uzyskania reguł asocjacyjnych nieco innego rodzaju (np. reguł klasyfikacyjnych).

Klasycznym przykładem zastosowania reguł asocjacyjnych jest analiza koszyka zakupów. Ma on również zastosowanie np. w dziedzinie ubezpieczeń, medycynie, telekomunikacji czy analizie odwiedzanych stron internetowych itp.

1.3.2 Model drzew decyzyjnych i klasyfikacyjnych

Kolejnym ważnym modelem drążenia danych jest model drzewa decyzyjnego. Powstaje on poprzez zastosowanie algorytmów, pochodzących z dziedziny uczenia maszynowego, które to dzielą populację danych wejściowych na podstawie testów pojedynczych atrybutów. W wyniku otrzymujemy drzewo, które składa się z następujących elementów (rys. 1.4):

- **węzeł wewnętrzny (decyzyjny)** – zawiera atrybut testowy,
- **krawędź drzewa** – wartość atrybutu testowego decydująca o wyborze danego poddrzewa,
- **węzeł końcowy (liść)** – wartość atrybutu decyzyjnego lub rozkład klas decyzyjnych.



Rysunek 1.4: Drzewo decyzyjne

Podstawą algorytmów budowy drzew decyzyjnych jest rekursja, w wyniku której tworzona jest struktura drzewa od korzenia do liści (wgłąb) (ang. *top-down*) poprzez podział na kolejne poddrzewa. Podział ten dokonywany jest poprzez analizę zbioru danych trenujących metodą dziel i zwyciężaj (ang. *divade and conquer*). Kolejną własnością algorytmów budowy drzew decyzyjnych jest to, że są one zachłanne (ang. *greedy*). Może być to powodem

problemu związanego z tym, że podczas budowy drzewa, w niektórych węzłach pomijane są testy istotne z punktu widzenia całej populacji, na rzecz nieistotnych testów lokalnych. Główną różnicą pomiędzy poszczególnymi algorytmami jest zastosowana funkcja podziału dla węzłów wewnętrznych:

- algorytmy uczenia maszynowego (ID3, C4.5, C5) [17, 68] opierają się na tzw. mierze zysku informacyjnego zbudowanej przy użyciu entropii informacyjnej,
- algorytmy statystyczne (CART) [10] wykorzystują miary częstości występowania rekordów (np. *Gini*),
- algorytmy rozpoznawania obrazu (CHAID) [46] wykorzystują miary korelacji między atrybutami (np. χ^2).

Podczas tworzenia modelu drzew decyzyjnych częstym problemem jest zbyt duży rozrost drzewa (przetrenowanie), w którym uwydatniają się anomalie występujące w zbiorze trenującym. Aby rozwiązać ten problem, stosuje się, w zależności od algorytmu, dwie różne strategie redukcji:

1. przycinanie wstępne (ang. *prepruning*) – warunkiem zakończenia procesu tworzenia drzewa jest osiągnięcie minimalnej, ustalonej wcześniej, wartości progowej funkcji podziału,

2. przycinanie końcowe (ang. *postpruning*) – generowane są wszystkie węzły drzewa, aż do osiągnięcia pojedynczego rekordu, a następnie obcinane są gałęzie zbyt szczegółowe.

W przypadku przycinania wstępnego problemem jest taki dobór wartości progowej funkcji podziału, aby wzrost drzewa nie został zatrzymany zbyt wcześnie. Dlatego w praktyce stosuje się zazwyczaj strategię przycinania końcowego, która choć bardziej złożona obliczeniowo, daje w rezultacie drzewa bardziej wiarygodne [35]. Istnieją dwie operacje przycinania końcowego: *Subtree replacement* i *Subtree raising*, które wykorzystywane są

w wielu algorytmach redukcji np. *cost complexity pruning* [10], *reduced error pruning* [66], *pessimistic pruning* [67], Minimum Description Length (MDL) [69].

Drzewa decyzyjne wykorzystywane są również do klasyfikowania nowych rekordów. W tym przypadku wyodrębnia się tzw. zbiór testowy, na podstawie którego tworzony jest model, który następnie wykorzystuje się do klasyfikacji nowych danych do określonych grup na podstawie wartości atrybutu decyzyjnego.

1.4 Problem redukcji modeli drążenia

Podczas etapu budowy modelu drążenia, aspekt doboru odpowiednich parametrów tego modelu jest bardzo istotny. Dla modelu asocjacyjnego są to wartości progowe wsparcia i zaufania dla reguł asocjacyjnych. Założenie wysokich wartości tych parametrów skutkuje wygenerowaniem modelu o małej ilości reguł, a co za tym idzie modelu przejrzystego – łatwego w analizie. W tym przypadku jednak istotne dane, które mogą interesować analityka, mogą zostać pominięte, a w konsekwencji uzyskana wiedza jest niepełna. Gdy wartości progowe parametrów są zbyt małe – generowana jest duża liczba reguł, co powoduje znaczną złożoność modelu oraz trudność w wydobywaniu najbardziej istotnej dla analityka wiedzy dziedzinowej. Tutaj nie następuje utrata interesujących informacji, lecz są one ukryte i trudno dostępne dla użytkownika. W takich przypadkach zastosowanie metod wydobywania istotnych dla danego zadania reguł staje się sprawą kluczową.

1.4.1 Redukcja poprzez ograniczenie danych źródłowych

W znanych autorowi pracy publikacjach (np. [34, 37, 40, 41, 55, 56, 58]) opisujących problematykę redukcji modelu z wykorzystaniem języków zapytań występują trzy istotnie różniące się podejścia do rozwiązania

tego problemu. Pierwsze z nich, to redukcja danych źródłowych poprzez nadanie ograniczeń na wartości atrybutów tworzonego modelu. W tym przypadku budowa modelu odbywa się jedynie dla wybranych rekordów bazy danych, spełniających zadane kryterium, co powinno skutkować zmniejszeniem liczby generowanych reguł. Przykładem takiego podejścia są: język DMQL [34], Operator MINE RULE [55, 56] oraz język MineSQL [58].

Język DMQL (Data Mining Query Language) jest językiem zapytań stworzonym dla potrzeb pozyskiwania wiedzy z relacyjnych baz danych. Umożliwia on budowę różnego typu modeli drążenia danych z ograniczeniami dla przetwarzanych danych.

find association rules
related to gpa, birth_place, address
from student
where major = "cs" and birth_place = "Canada"
with support threshold = 0.05
with confidence threshold = 0.7

Algorytm 1.2: Zapytanie w języku DMQL [35]

Przykład zapytania w języku DMQL pokazano w treści algorytmu 1.2. Zapytanie to, w pierwszej kolejności grupuje odpowiednie rekordy ze zbioru danych spełniające zadane warunki dla określonych atrybutów, a następnie buduje model reguł asocjacyjnych z odpowiednimi warunkami progowymi wsparcia i zaufania. Język ten został wykorzystany w systemie DBMiner [33] do kontroli procesu budowy modeli drążenia danych.

Podobnym rozwiązaniem do języka DMQL jest zaproponowany przez R. Meo w [55] Operator MINE RULE. Stanowi on rozszerzenie języka SQL o dodatkowe elementy, dzięki którym możliwe jest budowanie modeli reguł asocjacyjnych bezpośrednio dla danych przechowywanych w relacyjnej bazie danych. Ograniczenia dla danych źródłowych nadawane są przy pomocy standardowego zapytania SQL, natomiast proces pozyskiwania reguł asocjacyjnych wykonywany jest przy pomocy klauzuli MINE RULE [56] (algorytm 1.3).

MINE RULE SimpleAssociations **AS**
SELECT DISTINCT l..n item **AS** BODY, l..l item **AS** HEAD,
SUPPORT, CONFIDENCE
FROM Purchase **WHERE** price = 150
GROUP BY transaction
EXTRACTING RULES WITH SUPPORT: 0.1,CONFIDENCE: 0.2

Algorytm 1.3: Operator MINE RULE [55]

W wyniku działania Operatora MINE RULE otrzymujemy nieznormalizowaną tabelę reguł asocjacyjnych o czterech atrybutach: BODY (poprzednik reguły), HEAD (następnik reguły), SUPPORT (wsparcie), CONFIDENCE (zaufanie). Ten sposób zapisu, w szczególności przechowywanie wszystkich pozycji zbioru częstego jako jednej komórki tabeli, utrudnia dalsze przetwarzanie i analizę modelu, np. wyszukiwanie reguł, w skład których wchodzi zadana pozycja itp.

Kolejnym językiem budowy modeli reguł asocjacyjnych, zdefiniowanym, podobnie jak Operator MINE RULE, jako rozszerzenie języku SQL, jest MineSQL [58]. Charakteryzuje się on tym, że dla potrzeb przechowywania utworzonych przy jego pomocy reguł asocjacyjnych został zdefiniowany specjalny typ danych RULE oraz określono w nim funkcje umożliwiające dostęp do poszczególnych elementów reguły. Język ten daje również możliwość zapamiętania reguł w tabelach bazy danych, jak również w plikach tekstowych.

MINE rule, support(rule) s., confidence(rule) c.
FOR product
FROM shoppings
WHERE ‘product=“crescent”’ **IN** body(rule)
AND bodylen(rule) = 2
AND support(rule) > 0.2
GROUP BY trans id

Algorytm 1.4: Wyrażenie języka MineSQL [58]

Przykład wyrażenia w języku MineSQL pokazano w algorytmie 1.4. W wyniku wykonania zapytania wygenerowane zostaną tylko reguły asocjacyjne dla danych z tabeli *shoppings* ograniczonej do atrybutu *product* przy pomocy klauzuli *FOR*.

Podsumowując, można stwierdzić, że ograniczenie danych wejściowych dla budowanego modelu nie gwarantuje, że otrzymany podzbiór reguł będzie wystarczająco mały, aby być prostym w analizie. Dodatkowo, nie zawsze istnieje możliwość redukcji danych źródłowych, ponieważ jest to uzależnione od rozwiązywanych w konkretnym przypadku, często złożonych, problemów analitycznych i poszukiwanych zależności.

1.4.2 Redukcja poprzez ograniczenia dla generowanych reguł

Kolejnym sposobem redukcji modelu przy użyciu języków zapytań jest zadanie ograniczeń dla generowanych reguł. Ograniczenia te zmniejszają liczbę reguł w modelu tylko do takich, które spełniają zadane kryteria. Nadawanie ograniczeń odbywa się podczas procesu budowy modelu. Możliwościami takimi dysponuje, przedstawiony w podrozdziale 1.4.1 język MineSQL oraz język MSQL [40, 41].

Główną zaletą języka MineSQL, oprócz omówionej w podrozdziale 1.4.1 redukcji danych źródłowych, jest możliwość uzyskania tylko tych reguł, które spełniają określone uwarunkowania. W algorytmie 1.4 zapytanie zadane w języku MSQL nie tylko ogranicza dane wejściowe dla modelu, ale wskazuje również, że reguły muszą dodatkowo spełniać następujący warunek: zbiór częsty poprzednika reguły (*body*) ma mieć dokładnie dwie wartości i jedną z nich ma być *crescent*. Dodatkowo wsparcie utworzonych reguł ma być większe niż 0.2.

Drugim językiem zapytań redukującym liczbę reguł w trakcie procesu budowy modelu jest język MSQL [40, 41]. W języku tym reguła asocjacyjna ma postać **Body** → **Consequent** [support, confidence] (Body – poprzednik

reguły, Consequent – następnik reguły), natomiast jego składnia zbliżona jest do języka SQL. Algorytm 1.5 przedstawia przykład zapytania w języku MSQL, które przy pomocy polecenia *GetRules* generuje i zapisuje do zbioru reguł R tylko te reguły asocjacyjne z tabeli *Emp*, które spełniają następujący warunek: poprzednik reguły jest wartością atrybutu *Age* lub *Salary* z przedziału (30000,80000), natomiast następnik reguły jest wartością atrybutu *Job*. Dodatkowo progi wsparcia i zaufania reguły są ustawione odpowiednio na 0.1 i 0.3.

```
GetRules(Emp)
into R
where Consequent in { (Age=*), (Salary=[30000,80000]) }
and Body has { (Job=*) }
and confidence > 0.3
and support > 0.1
```

Algorytm 1.5: Zapytanie MSQL – GetRules [41]

Innym rozwiązaniem redukującym wielkość modelu podczas jego budowy jest zaproponowana w [60] oraz [81] modyfikacja algorytmu budowy modelu reguł asocjacyjnych. Polega ona na tym, że wśród wygenerowanych zbiorów częstych przy użyciu algorytmu Apriori (algorytm 1.1) wybierane są tylko te, które spełniają założone kryteria. Kryteria, przedstawione w postaci wyrażeń logicznych, wskazują, które wartości atrybutów mają wejść w skład zbiorów częstych. Wybrane zbiory częste stają się bazą dla generowanych reguł, których liczba jest mniejsza, niż w przypadku, gdy analizowane są wszystkie możliwe zbiory częste.

Można zauważyć, że w wyżej przedstawionym podejściu, jak również tym omówionych w podrozdziale 1.4.1, występują utrudnienia podobne jak przy doborze odpowiednich parametrów budowy modelu. W tym przypadku również nałożenie warunków początkowych na budowany model może ograniczyć jego przydatność do rozwiązania zadanego problemu, a co za tym idzie konieczność tworzenia kolejnych wersji modeli drążenia.

1.4.3 Redukcja przy użyciu zapytań do modelu

Kolejną metodą redukcji modelu drążenia jest wybór odpowiednich reguł z już utworzonego modelu. Realizuje się to poprzez zadawanie zapytań do modelu, których wynikiem jest podzbiór reguł interesujących z punktu widzenia analityka. Redukcja, w szczególności modelu o dużej liczbie reguł, do małego ich podzbioru, daje możliwość łatwiejszej analizy i pozyskiwania użytecznej wiedzy. Do grupy narzędzi wykorzystujących to podejście należą: omawiany w podrozdziale 1.4.2 język zapytań MSQL oraz SMARTSKIP System [37].

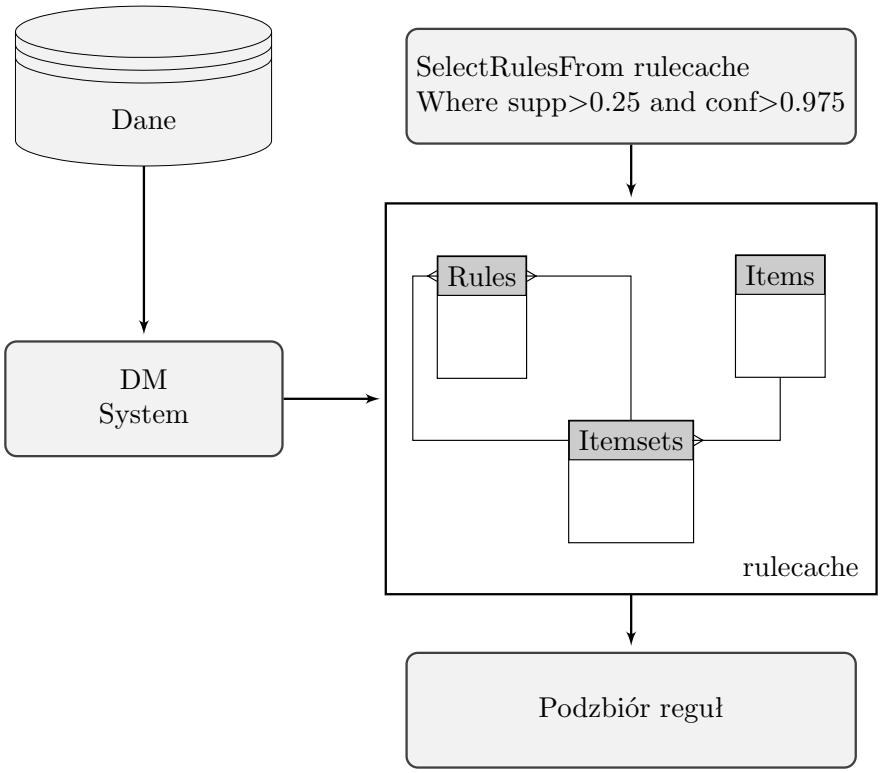
Język zapytań MSQL jest językiem, który oprócz generowania, daje również możliwość przetwarzania już utworzonych reguł asocjacyjnych.

SelectRules(R)
where Body **has** { (Age=[30,40]), (Sex=*)}
and Consequent **is** { (Car=*) }

Algorytm 1.6: Zapytanie MSQL – SelectRules [41]

W składni języka MSQL, poza pokazanym na algorytmie 1.5 poleceniem *GetRules* tworzącym zbiór reguł z ograniczeniami, istnieje również podobne składniowo polecenie *SelectRules*. Daje ono możliwość wyboru ze zbioru reguł R tylko tych reguł, które spełniają zadane w klauzuli *where* warunki. W algorytmie 1.6 warunek ograniczający dla reguły należy interpretować w sposób następujący: w podzbiorze pozycji w poprzedniku reguły muszą wystąpić dwie pozycje – wartość atrybutu *Age* z przedziału (30,40) oraz dowolna wartość atrybutu *Sex*, natomiast w następniku reguły dowolna wartość atrybutu *Car*.

Kolejnym przykładem wykorzystania zapytań dla potrzeb redukcji zbioru reguł jest System SMARTSKIP. Został on stworzony dla firmy Daimler-Chrysler na potrzeby analizy informacji o częściach montowanych w samochodach tej firmy. W systemie tym, w procesie drążenia danych generowany jest zbiór reguł, który następnie zapisywany jest w specjalnie do tego celu przygotowanej relacyjnej tabeli reguł (rysunek 1.5).



Rysunek 1.5: System SMARTSKIP

Zapytanie *SelectRulesFrom* skierowane do pamięci podręcznej reguł *rulecache* jest zamieniane na odpowiednie polecenie języka SQL, zwracające reguły, które spełniają zadany warunek. W przypadku, gdy w pamięci tymczasowej poszukiwany podzbiór nie zostanie znaleziony, są one pobierane z głównej tabeli reguł. Rozwiązanie to znacznie przyśpiesza zadawanie zapytań podobnych do siebie, w szczególności, gdy liczba reguł w głównej tabeli jest bardzo duża.

Porównując przedstawione metody redukcji (tabela 1.1 s. 22), można stwierdzić, iż opisana w tym podrozdziale metoda przy użyciu zapytań do modelu jest bardziej obiecująca niż przedstawione wcześniej, ze względu na możliwość wielokrotnego zadawania zapytań i badania podzbiorów reguł, bez potrzeby ponownego budowania modelu. Należy zwrócić uwagę, że żadne ze znanych rozwiązań nie umożliwia automatycznego zadawania

Tabela 1.1: Porównanie rozwiązań redukcji modeli drążenia

Typ redukcji	**Zalety**	**Wady**
ograniczenie danych wejściowych	– budowa modelu odbywa się tylko dla wyspecyfikowanych danych – liczba wygenerowanych reguł jest mniejsza niż w przypadku analizy całości danych	– zmiana założeń analizy wymusza konieczność budowy nowego modelu – analiza danych cząstkowych może doprowadzić do niepoprawnych wyników i wniosków
ograniczenie dla generowanych reguł	– we wszystkich regułach modelu występują wskazane przez użytkownika pozycje – łatwość analizy małej (zazwyczaj) liczby reguł	– zmiana założeń analizy wymusza konieczność budowy nowego modelu
ograniczenie przez użycie zapytań	– generowane są wszystkie reguły (powyżej progu wsparcia i zaufania) - nie ma potrzeby budowy nowego modelu przy zmianie założeń analizy	– trudność doboru odpowiedniego zapytania zwracającego oczekiwany wynik

zapytań w celu uzyskiwania odpowiedzi adekwatnych do potrzeb analityka. Nie są to również rozwiązania uniwersalne, umożliwiające wykorzystanie ich dla różnych platform sprzętowych i programowych. Dlatego też, stworzenie metodyki umożliwiającej analizę dużych zbiorów reguł asocjacyjnych, wraz z zestawem narzędzi programowych dających możliwość realizacji tej metodyki, wydaje się zarówno z badawczego jak i praktycznego punktu widzenia zadaniem zasadnym i interesującym.

1.5 Teza i zakres pracy

Tezą niniejszej pracy jest następujące stwierdzenie:

> *Zastosowanie adaptacyjnych procedur formułowania zapytań umożliwia skuteczne pozyskiwanie problemowo zorientowanej wiedzy z dużych zbiorów danych analitycznych.*

Prawdziwość tezy zostanie wykazana poprzez:

1. opracowanie metody pozyskiwania wiedzy poprzez automatyczne generowanie zapytań do modelu reguł asocjacyjnych,
2. konstrukcję i implementację algorytmów i narzędzi dla realizacji zaproponowanej metody
3. stworzenie prototypowej aplikacji do weryfikacji opracowanej metody,
4. przeprowadzenie eksperymentów dla danych rzeczywistych, umożliwiających weryfikację opracowanego rozwiązania.

1.6 Przegląd zawartości pracy

W rozdziale drugim została przedstawiona proponowana metoda generowania zapytań do modelu drążenia danych, opisano algorytm programowania genetycznego stanowiący podstawę tej metody. Dodatkowo zostały omówione: standard PMML, wykorzystany do tekstowego zapisu modeli drążenia danych oraz język zapytań XQuery, który daje możliwość zadawania zapytań do tak przechowywanych modeli.

Rozdział trzeci zawiera szczegółowy opis zastosowania algorytmu programowania genetycznego pozwalający na generowanie zapytań do modelu reguł asocjacyjnych. Zaproponowano w nim definicję kryterium oceny wygenerowanych zapytań na podstawie zwracanego przez nie podzbioru reguł asocjacyjnych.

Rozdział czwarty opisuje założenia implementacyjne oraz strukturę aplikacji opracowanej na potrzeby testowania skuteczności proponowanej

metody. Przedstawiono tu zadania poszczególnych modułów aplikacji oraz wskazano biblioteki programistyczne, które zostały wykorzystane do ich przygotowania.

W rozdziale piątym zademonstrowano wyniki przeprowadzonych z wykorzystaniem opracowanej metody eksperymentów obliczeniowych, dotyczących trzech rzeczywistych baz danych: (i) energetycznych, (ii) medycznych oraz (iii) wypadków samochodowych .

Na końcu pracy zamieszczono podsumowanie, bibliografię oraz dodatki zawierające opisy danych rzeczywistych wykorzystanych podczas badań, elementy eksperymentalnego systemu pozyskiwania wiedzy oraz szczegółowe wyniki eksperymentów.

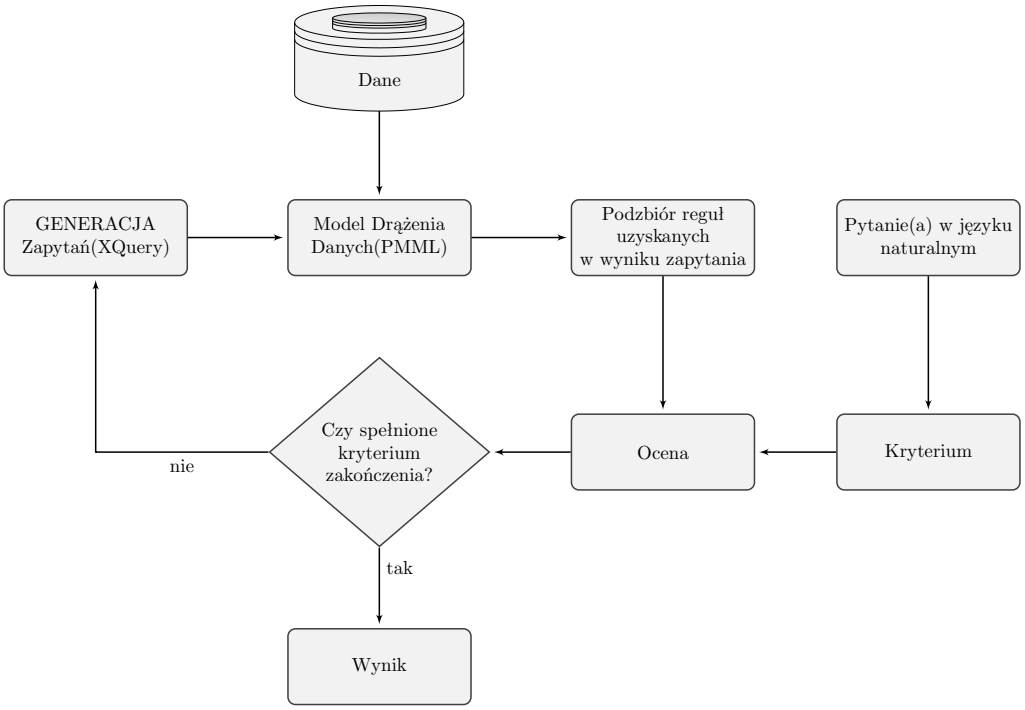
Rozdział 2

Metoda automatycznego generowania zapytań

2.1 Opis metody

Z rozważań przeprowadzonych w podrozdziale 1.4 można wysnuć wniosek, że najkorzystniejszą metodą analizy modelu reguł asocjacyjnych jest zadawanie zapytań do zbioru reguł, który został zbudowany bez ograniczeń dotyczących danych źródłowych oraz generowanych reguł. Jednak z powodu dużej liczby reguł zadanie prawidłowego zapytania nie jest rzeczą łatwą i często uzyskany efekt jest niezadawalający. Prowadzi to do sytuacji. w której otrzymanie oczekiwanego wyniku wymaga zadania wielu zapytań, co w sposób „ręczny" jest zajęciem czasochłonnym i często niewykonalnym ze względów czasowych. Dlatego też, zadaniem proponowanej metody jest automatyczne generowanie coraz lepszych zapytań do bazy reguł, a co za tym idzie poszukiwanie optymalnego rozwiązania pod względem zadanego kryterium. Schemat opracowanej metody przedstawiono na rysunku 2.1.

W pierwszej kolejności budowany jest model reguł asocjacyjnych. Zapisywany jest on w standardzie PMML opisanym w podrozdziale 2.4. Następnie zadawane jest zapytanie do modelu w języku XQuery przedstawionym w podrozdziale 2.5.2. Zapytanie to zwraca pewien podzbiór reguł, który to



Rysunek 2.1: Schemat metody automatycznego generowania zapytań

następnie poddawany jest ocenie względem przyjętego wcześniej kryterium. Jeśli warunek zakończenia jest spełniony, zwracane jest uzyskane rozwiązanie, w przeciwnym wypadku generowane jest kolejne zapytanie i cały cykl powtarza się.

W przypadku tworzenia kolejnego zapytania podstawowym zadaniem jest znalezienie takiego nowego zapytania, aby było ono lepsze od poprzedniego. Drugim problemem proponowanego rozwiązania jest dobranie odpowiedniego kryterium oceny. Musi ono łączyć w sobie elementy wskazane przez użytkownika oraz ograniczające liczbę reguł wynikowych do najważniejszych z punktu widzenia analizy. W dalszej części pracy przedstawione zostaną rozwiązania wskazanych problemów przy wykorzystaniu algorytmów programowania genetycznego oraz zdefiniowanego kryterium oceny.

2.2 Programy ewolucyjne

Programy ewolucyjne stały się w ostatnim czterdziestoleciu jedną z bardzo szybko rozwijających się gałęzi wiedzy w dziedzinie sztucznej inteligencji. Pod tym pojęciem należy rozumieć całą klasę systemów opartych na mechanizmie doboru naturalnego i dziedziczenia [57, 73].

Mechanizm ten polega na stworzeniu populacji osobników, która jest pewnym zbiorem rozwiązań zadanego problemu. Każdy z nich poddawany jest ocenie, która określa stopień, w jakim dane rozwiązanie spełnia założenia funkcji celu. Następnie na podstawie reprodukcji i modyfikacji najlepszych osobników tworzona jest kolejna populacja, w której spodziewamy się znaleźć rozwiązanie lepsze od dotychczas uzyskanych.

Reprodukcję należy interpretować jako przeniesienie osobnika do populacji potomnej bez dokonywania zmian w jego strukturze. Wśród metod modyfikujących nowopowstałą populację można wskazać najczęściej wykorzystywane, tzw. operatory genetyczne: mutację i krzyżowanie. Mutacja jest procesem zamiany wybranego fragmentu osobnika, na nowy, losowo wygenerowany. Natomiast w przypadku krzyżowania następuje wymiana fragmentów „materiału genetycznego" pomiędzy dwoma osobnikami populacji. W wyniku operacji modyfikujących otrzymujemy nowe osobniki (rozwiązania), a co za tym idzie obszar poszukiwań najlepszego wyniku cały czas podlega zmianom, dzięki czemu zwiększa się prawdopodobieństwo jego znalezienia.

Można wyróżnić następujące metody wykorzystujące mechanizmy ewolucyjne:

1. algorytmy genetyczne
2. strategie ewolucyjne
3. programowanie ewolucyjne
4. programowanie genetyczne.

Za głównego prekursora w dziedzinie algorytmów genetycznych powszechnie uznaje się Johna Hollanda. Jego praca [38], która powstała na bazie pomysłu obliczeń ewolucyjnych przedstawionych przez I. Rechenberga w [71], stała się podstawą dla innych programów ewolucyjnych. W algorytmach genetycznych definiuje się reprezentację osobników jako strukturę (genotyp) składającą się z ciągów kodowych (bitowych) będących odpowiednikami chromosomów. Genotyp stanowi podstawę do utworzenia fenotypu, który jest zbiorem parametrów, rozwiązaniem lub punktem w przestrzeni rozwiązań. Algorytmy genetyczne wykorzystywane są w wielu różnorodnych dziedzinach, takich jak: matematyka, medycyna, technika czy politologia. W kontekście optymalizacji funkcji można tu wyróżnić pracę DeJonga [23], która wskazała kierunek rozwoju algorytmów genetycznych [30].

W przypadku strategii ewolucyjnych zaproponowanych przez Schwefela we współpracy z Rechenbergiem [76] osobniki populacji mają reprezentację zmiennoprzecinkową. Jedynym mechanizmem modyfikującym, stosowanym w tych strategiach jest mutacja. Znalazły one zastosowanie głównie w zadaniach optymalizacji parametrycznej dla parametrów ciągłych, chociaż podjęto również próby wykorzystania tych strategii dla zadań dyskretnych [7, 36].

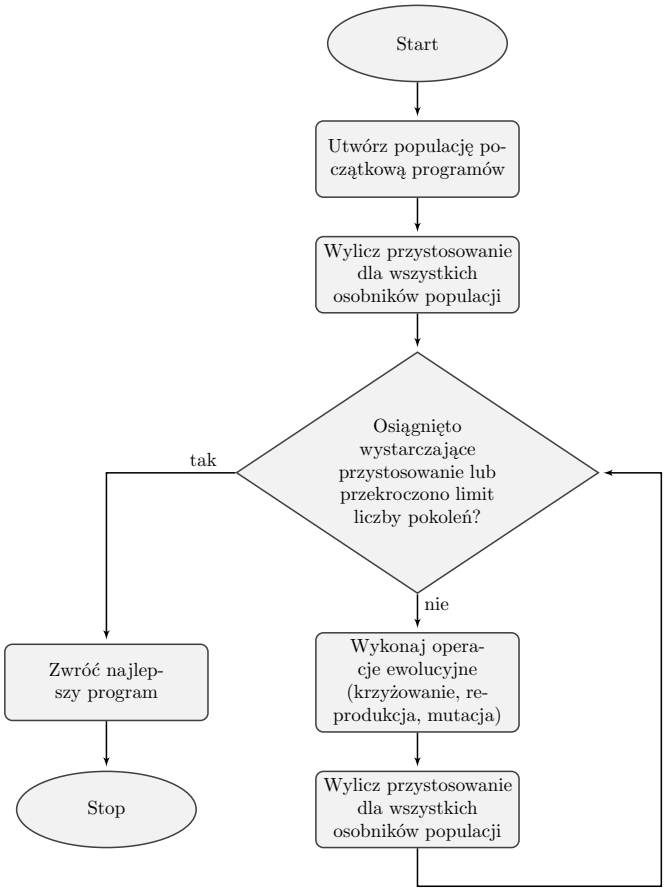
Programowanie ewolucyjne zostało wprowadzone w latach 60-tych przez Lawrence Fogela. Fogel zastosował tę ewolucyjną metodę w odniesieniu do automatów o skończonej pamięci z niewielką liczbą stanów [29]. Podobnie jak miało to miejsce dla strategii ewolucyjnych, jedynym operatorem genetycznym zastosowanym w tym podejściu jest mutacja, która dokonywana jest na elementach grafów opisujących stany automatów. Programowanie ewolucyjne znalazło zastosowanie w modelowaniu i identyfikacji systemów, sterowaniu, uczeniu sieci neuronowych, prognozowaniu, rozpoznawaniu obrazów i problemach kombinatorycznych.

Ostatnią z wymienionych metod ewolucyjnych jest, zaproponowane stosunkowo niedawno, bo na początku lat dziewięćdziesiątych, programowanie

genetyczne. Ponieważ stanowi ono podstawę rozwiązania problemu automatycznego generowania zapytań zaproponowanego w tej pracy, zostanie szerzej omówione w kolejnym podrozdziale.

2.3 Programowanie genetyczne

W metodzie programowania genetycznego, przedstawionej przez Johna R. Kozę w [49], zaproponował on, aby zamiast budowania jednego programu ewolucyjnego rozwiązującego zadany problem, przeszukiwać przestrzeń programów. Programy te stanowią populację osobników, które ewoluują w kolejnych pokoleniach w celu znalezienia jak najlepszego, w sensie przyjętego kryterium, rozwiązania.



Rysunek 2.2: Algorytm programowania genetycznego

Zasada programowania genetycznego (rysunek 2.2) w pełni opiera się na algorytmie programów ewolucyjnych, z tą jednak różnicą, że w tym przypadku inna jest reprezentacja osobnika populacji, która jest programem komputerowym.

2.3.1 Struktura osobnika w programowaniu genetycznym

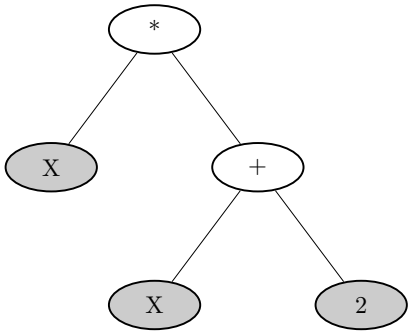
Struktura osobnika w programowaniu genetycznym, która podlega procesowi adaptacji, przypomina zastosowaną w przypadku programowania ewolucyjnego, z tą różnicą, że zamiast grafu stanów, ma ona formę drzewa. Drzewo to reprezentuje program komputerowy, w którym funkcje programu znajdują się w węzłach, natomiast argumenty funkcji (tzw. terminale) w liściach drzewa. Funkcjami mogą być, w zależności od zastosowanego języka programowania, operatory: matematyczne, warunkowe, logiczne lub funkcje: matematyczne, iteracyjne i inne. Argumentami funkcji mogą być zmienne, stałe, struktury, obiekty itp.

Przy doborze funkcji i terminali konieczne jest spełnienie tzw. wymagań domknięcia i wystarczalności.

Definicja 4. *Własność domknięcia zbioru funkcji F i zbioru terminali T jest spełniona, gdy każda funkcja ze zbioru funkcji F może przyjąć jako swoje argumenty dowolną wartość zwracaną przez funkcje zbioru F, jak również wartość ze zbioru terminali T.*

Definicja 5. *Własność wystarczalności zbioru funkcji F i zbioru terminali T jest spełniona, gdy zbiór F i T zawierają wszystkie elementy niezbędne do rozwiązania rozpatrywanego zadania.*

Na rysunku 2.3 przedstawiono przykład drzewiastej struktury osobnika. W pokazanym przykładzie osobniki populacji są wzorami funkcji zmiennej X. Ze zbioru funkcji operatorów matematycznych $F = \{*, /, +, -\}$ (mnożenie, dzielenie, dodawanie i odejmowanie) oraz zbioru terminali $T = \{\text{stała} \in N, \text{zmienna} \in \{X\}\}$ (stała należąca do liczb naturalnych oraz X



Rysunek 2.3: Drzewiasta struktura osobnika w programowaniu genetycznym

należące do zbioru zmiennych) tworzone jest drzewo, które jest odpowiednikiem wyrażenia matematycznego X*(X+2).

2.3.2 Typy przystosowania

Przystosowanie stanowi podstawę teorii ewolucji Darwina i określa zdolność osobnika do przetrwania i reprodukcji. W programach ewolucyjnych natomiast miara przystosowania definiuje stopień dopasowania danego osobnika do optymalnego rozwiązania danego problemu. Na podstawie oceny przystosowania dokonywana jest selekcja osobników do kolejnych etapów ewolucyjnych. W przypadku programowania genetycznego przystosowanie określa użyteczność oraz dokładność działania programu. W pracy [50] J. R. Koza wskazuje jako najczęściej stosowane następujące miary przystosowania:

- przystosowanie naturalne,
- przystosowanie standaryzowane,
- przystosowanie skorygowane,
- przystosowanie znormalizowane.

Przystosowanie naturalne jest określane jako miara, która wyrażana jest w terminologii zależnej od (naturalnej dla) rozważanego zadania. Jest ona definiowana na podstawie zbioru próbek testowych (**choć nie zawsze**), jako miara odległości wartości zwracanej przez wyrażenie reprezentujące

danego osobnika w punktach próbkowania od wartości poprawnej w tych punktach (2.1).

$$r(i) = \sum_{j=1}^{N} |S(i,j) - C(j)| \tag{2.1}$$

gdzie:

$r(i)$ – przystosowanie naturalne i-tego osobnika,

$S(i,j)$ – wartość zwracana przez i-tego osobnika dla j-próbki,

$C(j)$ – wartość poprawna dla j-próbki,

N – liczba próbek.

Ponieważ wynikiem zwracanym przez osobnika mogą być wyrażenia różnego typu (np. logiczne, wektory, rzeczywiste, zespolone itp.) może się zdarzyć, że lepsze przystosowanie może być wskazywane przez wyższe wartości (np. dla zadań maksymalizacji zysków) lub niższe (np. odległość punktów od wykresu).

Przystosowanie standaryzowane to przystosowanie naturalne, w którym niższe wartości przystosowania są traktowane jako lepsze z punktu widzenia rozwiązania problemu. W przypadku, gdy wartość przystosowania naturalnego jest lepsza dla wyższych wartości, dla uzyskania przystosowania standaryzowanego należy zastosować zależność (2.2).

$$s(i) = r_{max} - r(i) \tag{2.2}$$

gdzie:

$s(i)$ – przystosowanie standaryzowane i-tego osobnika,

r_{max} – maksymalne przystosowanie naturalne,

$r(i)$ – przystosowanie naturalne i-tego osobnika.

Przystosowanie skorygowane wykorzystywane jest w przypadku, gdy istnieje potrzeba przeskalowania przystosowania i umieszczenie jego wartości w pewnych granicach – zazwyczaj w przedziale wartości $< 0, 1 >$ oraz uwrażliwienie na niewielkie różnice między wartościami $s(i)$. W tym celu zazwyczaj stosuje się zależność (2.3).

$$a(i) = \frac{1}{1 + s(i)} \tag{2.3}$$

gdzie:

$a(i)$ – przystosowanie skorygowane i-tego osobnika,

$s(i)$ – przystosowanie standaryzowane i-tego osobnika.

Cechą przystosowania skorygowanego jest to, że jego wyższe wartości oznaczają osobnika lepszego, co wydaje się rozwiązaniem bardziej intuicyjnym niż ma to miejsce w przypadku przystosowania standaryzowanego.

Przystosowanie znormalizowane (2.4) ma zastosowanie w przypadku, gdy proces selekcji odbywa się przy użyciu metody proporcjonalnej do wartości przystosowanie.

$$n(i) = \frac{a(i)}{\sum_{k=1}^{M} a(k)} \tag{2.4}$$

gdzie:

$n(i)$ – przystosowanie znormalizowane i-tego osobnika,

$a(i)$ – przystosowanie skorygowane i-tego osobnika,

M – liczebność populacji.

Wśród zalet tego typu przystosowania można wskazać następujące właściwości:

- wartości przystosowania znormalizowanego osobnika zawiera się w przedziale $< 0; 1 >$,
- suma wartości przystosowania wszystkich osobników populacji jest równa 1,
- lepszym osobnikom przyporządkowana jest wyższa wartość przystosowania.

2.3.3 Przegląd operacji ewolucyjnych w programowaniu genetycznym

Zasada działania operacji ewolucyjnych w programowaniu genetycznym jest podobna do tych stosowanych w innych programach ewolucyjnych. Różnica polega na tym, że w tym przypadku operacje modyfikacji wykonywane są na strukturach drzewiastych.

Operacja reprodukcji przebiega w programowaniu genetycznym w sposób standardowy (roz. 2.2). Dzięki temu, że podczas przenoszenia osobnika do populacji potomnej tworzona jest jego kopia, może on brać udział również w innych operacjach ewolucyjnych. Dodatkową zaletą reprodukcji jest to, iż osobnik nie zmienia wartości przystosowania, co znacznie przyśpiesza proces tworzenia nowego pokolenia.

Jedną z najczęściej stosowanych metod reprodukcji jest reprodukcja proporcjonalna, często nazywana ruletkową. Polega ona na wyborze osobnika na podstawie wartości jego funkcji przystosowania. W tym celu definiuje się zmienną losową, która określa prawdopodobieństwo reprodukcji dla każdego osobnika, które jest wprost proporcjonalne do wartości funkcji przystosowania (2.5).

$$p(i) = \frac{r(i)}{\sum_{k=1}^{M} r(k)}, \tag{2.5}$$

gdzie:

$p(i)$ – prawdopodobieństwo reprodukcji,
$r(i)$ – wartość funkcji przystosowania i-tego osobnika,
M – wielkość populacji

Metoda ta jest stosunkowo prosta w implementacji i daje w wielu przypadkach zadowalające rezultaty, ma jednak pewne wady:

- wymaga przeskalowania funkcji przystosowania do wartości nieujemnych [57],
- można ją stosować jedynie do jednej klasy zadań, tzn. tylko do maksymalizacji lub tylko minimalizacji [25],
- może doprowadzić do całkowitego zdominowania populacji przez najlepsze osobniki poprzez usuniecie tych o mniejszym przystosowaniu, które mogą być kluczowe do rozwiązania problemu, a co się z tym wiąże do zbyt wczesnej zbieżności algorytmu [5].

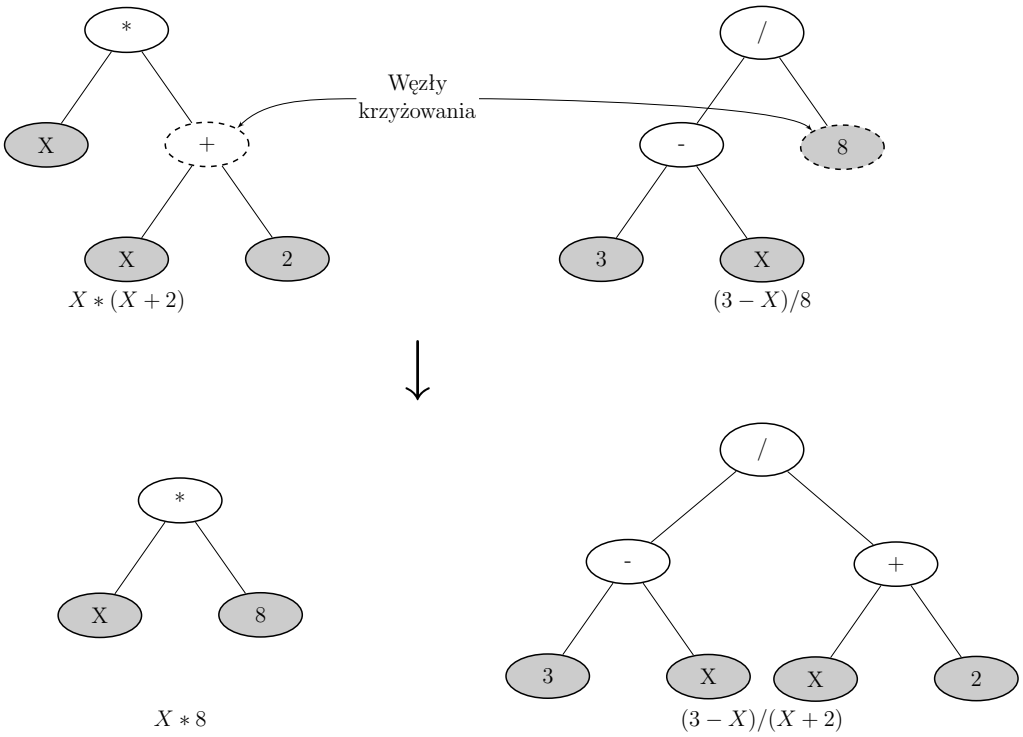
Dlatego też, w praktyce stosuje się również inne metody selekcji, takie jak: strategia elitarna, selekcja rankingowa i selekcja turniejowa.

Strategia elitarna zaproponowana w [23] jest oparta na zachowaniu co najmniej najlepszego osobnika w populacji bieżącej i wprowadzeniu go bez zmian do populacji potomnej (metoda ruletki nie gwarantuje „przeżycia" najlepszego osobnika). Główną zaletą tej metody jest to, że pomaga w uniknięciu przedwczesnej zbieżności oraz przyśpiesza proces poszukiwania optymalnego rozwiązania.

Selekcja rankingowa przedstawiona w [8, 89] polega na wyborze osobników na podstawie przypisanych im rang. Przydział rangi odbywa się na uporządkowanej malejąco pod względem przystosowania populacji i wartość jej uzależniona jest od pozycji danego osobnika. Prawdopodobieństwo wyboru osobnika do reprodukcji definiuje (najczęściej spotykana [5]) zależna od rangi funkcja liniowa. Zaletą tej metody jest to, że może mieć zastosowanie zarówno do problemu maksymalizacji jak i minimalizacji oraz nie wymaga konieczności skalowania funkcji przystosowania. Wadą natomiast jest pomijanie informacji o ocenie osobników na podstawie wartości ich przystosowania.

Selekcja turniejowa [31] polega na podziale populacji na podgrupy k-elementowe (k to rozmiar turnieju — zwykle 2 lub 3) i wyborze z każdej podgrupy osobnika o najlepszym przystosowaniu. Można to zrobić poprzez wybór losowy lub wybór deterministyczny. Metoda turniejowa, podobnie jak metoda rankingowa, nadaje się zarówno do rozwiązywania problemów maksymalizacji jak i minimalizacji, jak również może zostać wykorzystana w zadaniach optymalizacji wielokryterialnej.

Wymienione metody selekcji są uważane za podstawowe, natomiast istnieje wiele ich modyfikacji pozwalających rozwiązać problem selekcji dla konkretnych zadań [57].

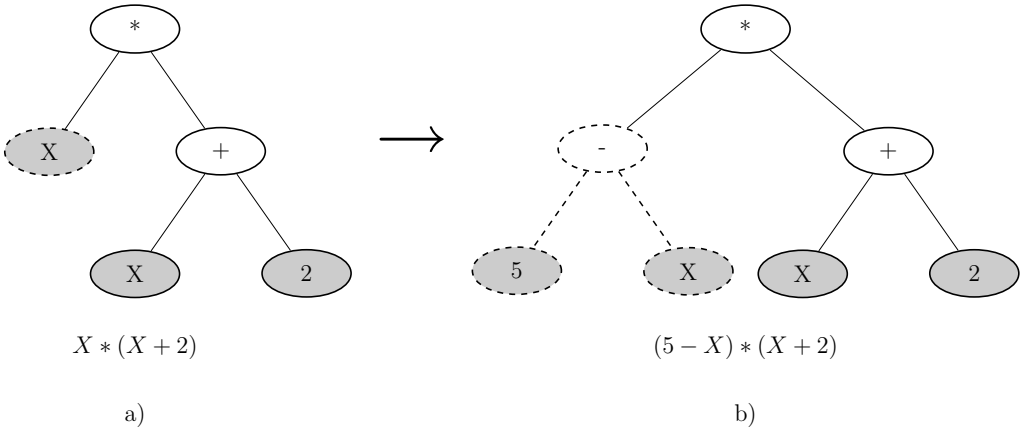


Rysunek 2.4: Operacja krzyżowania

Kolejna operacja ewolucyjna – krzyżowanie w programowaniu genetycznym przebiega w sposób następujący: wybierane są w sposób losowy (wykorzystuje się podobnie jak miało to miejsce w przypadku reprodukcji jedną z metod selekcji) dwa osobniki z populacji, a następnie punkty krzyżowania (węzły), w których następuje zamiana fragmentów drzew (rysunek 2.4). Uwidacznia się tutaj kolejna różnica w stosunku do innych programów ewolucyjnych. Mianowicie, punkty krzyżowania mogą być różne dla każdego z obu osobników, a nie jak ma to miejsce np. w algorytmach genetycznych, gdzie występuje tylko jeden punkt krzyżowania, taki sam dla obu osobników. Dzięki tej właściwości nowopowstałe osobniki charakteryzują się większą zmiennością, co może przyśpieszyć poszukiwanie rozwiązania.

W operacji mutacji losowany jest tylko jeden osobnik, a następnie węzeł drzewa, który ma zostać zmieniony (węzeł X na rysunku 2.5a). Następnie

gałąź drzewa jest „obcinana" w dół i zastępowana nową, losowo wygenerowaną (rysunek 2.5b). Dzięki operacji mutacji istnieje możliwość zmiany materiału genetycznego osobników, co w wielu przypadkach przyśpiesza, a niekiedy nawet jest niezbędne do znalezienie poszukiwanego rozwiązania.

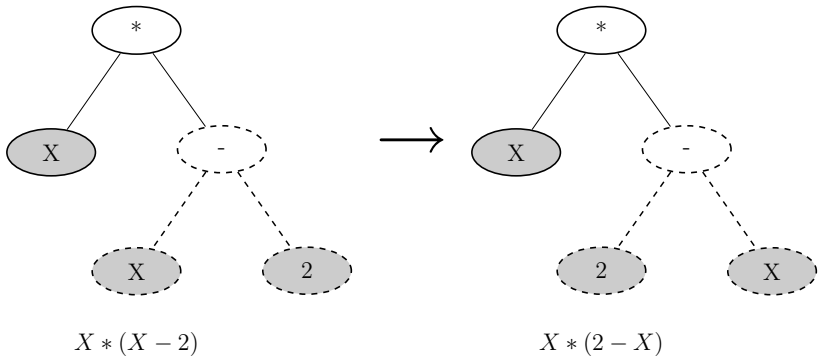


Rysunek 2.5: Operacja mutacji

Oprócz powszechnie używanych operacji ewolucyjnych, takich jak krzyżowanie czy mutacja, w procesie programowania genetycznego wykorzystywane są (choć nie są niezbędne) również inne, takie jak:

- dziesiątkowanie,
- permutacja,
- edycja.

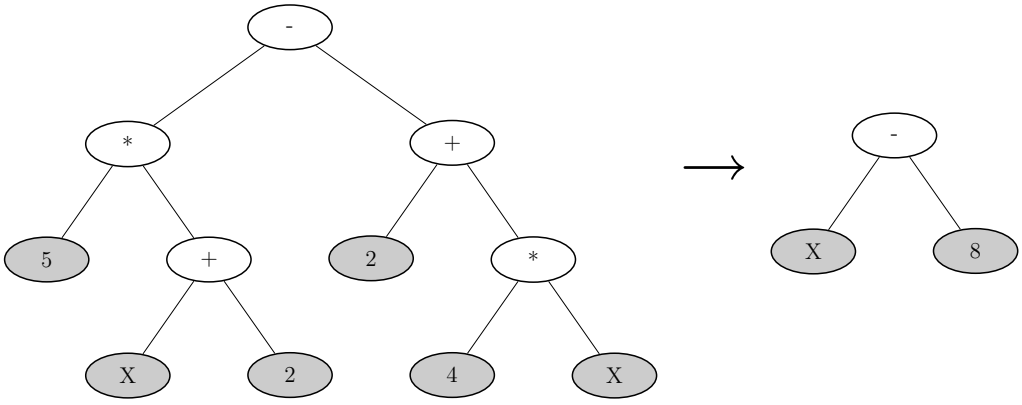
Dziesiątkowanie jest dość często wykonywaną operacją ewolucyjną i polega na wygenerowaniu większej liczby osobników w pierwszym kroku ewolucji, niż przewidziana docelowo wielkość populacji. Następnie wybierana jest odpowiednia liczba najlepszych osobników do populacji początkowej, dzięki temu w procesie ewolucyjnym już na starcie uczestniczą osobniki lepiej przystosowane. Zastosowanie tej operacji może w znacznym stopniu przyśpieszyć osiągniecie wyniku.



$X * (X - 2)$ $X * (2 - X)$

Rysunek 2.6: Operacja permutacji

Permutacja polega na zamianie miejscami gałęzi (parametrów) losowo wybranej funkcji (rysunek 2.6). Operacja ta możliwa jest i daje pewne rezultaty w przypadku funkcji nieprzemiennych (np. funkcji dzielenia, odejmowania itp.) i innych, w których zmiana taka jest dozwolona ze względu na charakter funkcji. Permutacja daje możliwość zwiększenia różnorodności osobników i w niektórych przypadkach może poprawić skuteczność poszukiwań, choć ze względu na ograniczone możliwości zastosowań wykorzystywana jest stosunkowo rzadko.



$(5 * (X + 2)) - (2 + (4 * X))$ $X - 8$

Rysunek 2.7: Operacja edycji

Zadaniem ostatniej z wymienionych operacji – edycji jest upraszczanie, bardzo skomplikowanych drzew (rysunek 2.7), których przetwarzanie

jest czasochłonne i wymagające dużej mocy obliczeniowej np. poprzez redukcję wyrażeń symbolicznych itp. Zastosowanie tej operacji oprócz przyśpieszenia procesu ewolucyjnego, może również spowodować usunięcie pewnych fragmentów „materiału genetycznego" osobników, które mogą być kluczowe dla poszukiwanego rozwiązania, a to z kolei może spowolnić lub też uniemożliwić znalezienie optymalnego wyniku. Z tego powodu stosowanie tej operacji wymaga dokładnej analizy pod względem zysków i strat jakie ona przyniesie.

Inną operacją genetyczną dającą możliwość zróżnicowania populacji, w szczególności w kolejnych krokach ewolucyjnych, kiedy osobniki stają się podobne do siebie, jest „wybijanie". Polega ono na usunięciu z populacji części osobników (np. o najgorszym przystosowaniu) i zastąpieniu ich nowymi – losowo wygenerowanymi. Dzięki tej operacji istnieje szansa wprowadzenia do populacji nowego materiału genetycznego, który może poprawić skuteczność poszukiwania rozwiązania.

2.3.4 Parametry kontrolne procesu programowania genetycznego

Kluczowe znaczenie dla prawidłowego przebiegu procesu programowania genetycznego ma odpowiedni dobór szeregu parametrów kontrolnych, których wartość określa się zazwyczaj w sposób eksperymentalny. Zestawienie najważniejszych parametrów pokazano w tabeli 2.1.

Podstawowym parametrem kontrolnym procesu programowania genetycznego jest wielkość populacji, która określa liczbę osobników (programów) wchodzących w skład populacji. Wartość tego parametru uzależniona jest od złożoności rozwiązywanego problemu – im problem bardziej skomplikowany tym jest większa.

Ponieważ w rzeczywistym świecie proces ewolucyjny trwa w sposób ciągły, natomiast tworzony przez komputer musi mieć swoje zakończenie, dlatego też definiowane są dwa parametry, których przekroczenie kończy

Tabela 2.1: Zestawienie najważniejszych parametrów kontrolnych programowania genetycznego

Parametr	**Opis**
wielkość populacji	określa liczbę osobników populacji
liczba pokoleń	określa maksymalną liczbę iteracji algorytmu programowania genetycznego
oczekiwane przystosowanie	określa wartość progową funkcji przystosowania, której osiągnięcie przerywa proces ewolucji
współczynnik reprodukcji	określa, jaką część nowej populacji stanowić będą osobniki będące efektem operacji reprodukcji
współczynnik krzyżowania	określa, jaką część nowej populacji stanowić będą osobniki będące efektem operacji krzyżowania (1 − *współczynnik reprodukcji*)
współczynnik mutacji	określa, z jaką częstotliwością ma wystąpić w procesie ewolucyjnym operacja mutacji
minimalna wielkość osobnika	określa minimalną głębokość (liczba poziomów) drzewa
maksymalna wielkość osobnika	określa maksymalną głębokość (liczba poziomów) drzewa
minimalna początkowa wielkość osobnika	określa początkową głębokość (liczba poziomów) drzewa

proces ewolucyjny. Są to: maksymalna liczba pokoleń oraz wartość progowa funkcji przystosowania. Liczba pokoleń, podobnie jak wielkość populacji determinowana jest złożonością problemu, natomiast wartość progowa funkcji przystosowania określa, w jakim stopniu oczekiwany wynik może odbiegać od maksymalnej (minimalnej) wartości funkcji celu.

Kolejne współczynniki: reprodukcji, krzyżowania i mutacji wskazują, jaka część nowej, generowanej w kolejnym kroku populacji ma powstać na podstawie tych operacji ewolucyjnych lub, jak ma to miejsce w przypadku mutacji, z jaką częstotliwością jest ona wykonywana. Współczynniki te dają możliwość wpływu na zmienność populacji i mają bardzo duże znaczenie dla

uzyskania prawidłowego wyniku oraz czasu, w jakim ten rezultat zostanie osiągnięty.

Ostanie trzy z wymienionych w tabeli 2.1 parametrów kontrolnych dotyczą wielkości osobnika (głębokości drzewa). Wprowadzenie tych ograniczeń ma na celu zróżnicowanie osobników, jak ma to miejsce w przypadku minimalnej początkowej wielkości osobników. Dodatkowo zapobiegają one tworzeniu osobników zbyt prostych, składających się na przykład z jednego terminatora, które zazwyczaj cechują się niską wartością funkcji przystosowania lub zbyt rozbudowanych, które dodatkowo wymagają dużych zasobów pamięciowych oraz są czasochłonne w przetwarzaniu.

2.4 PMML – standard zapisu modeli drążenia danych

Szybki rozwój systemów drążenia danych w ostatnich latach oraz niechęć do współpracy pomiędzy głównymi producentami tego typu oprogramowania stały się powodem braku kompatybilności pomiędzy poszczególnymi systemami. Każdy z dostawców narzędzi do eksploracji baz danych wykorzystywał swój własny system zapisu i przechowywania modeli drążenia, co nastręczało wiele trudności w przypadku próby dostępu do modelu w inny sposób niż przewidzieli to twórcy programu. Często również utrudnione, a w wielu przypadkach niemożliwe, było przenoszenie modeli pomiędzy poszczególnymi aplikacjami czy systemami. Powodowało to częstokroć konieczność zmiany całego systemu analizy danych w przypadku, gdy tylko jeden z elementów systemu był niewystarczający z punktu widzenia stawianych wymagań. Dla rozwiązania powyższych problemów powstał w 1998 roku standard PMML.

PMML (*Predictive Model Markup Language*) jest opartym na języku XML, standardem tekstowego zapisu modeli statystycznych i drążenia danych. Język ten został wprowadzony przez Data Mining Group [22] –

organizację, która definiuje nowe standardy w dziedzinie drążenia danych. W jej skład wchodzi wiele wiodących firm takich jak: IBM, Microsoft, Oracle, SAS, SPSS i inne. Standard PMML, podobnie jak i inne oparte na bazie XML zdobywają w ostatnim okresie coraz większą popularność i przewiduje się [18], iż staną się one w najbliższym czasie podstawą systemów drążenia danych i pozyskiwania wiedzy.

Tabela 2.2: Porównanie standardów zapisu modeli drążenia

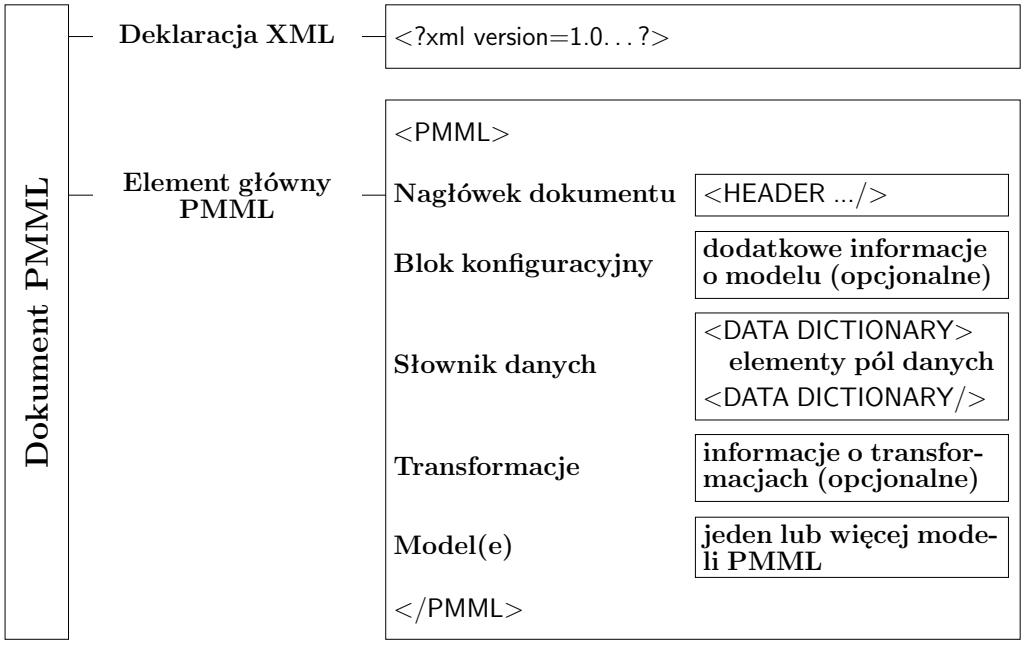
	Dotychczasowe rozwiązania		**Standard PMML**
Format zapisu	Relacyjna baza danych	Wewnętrzny format zapisu	Plik tekstowy
Przenoszalność modelu	Utrudniona	Niemożliwa lub bardzo utrudniona	Łatwa
Zależność od aplikacji DM	Duża	Pełna	Brak
Dostęp do elementów modelu	Język zapytań SQL	Uzależniony od aplikacji	Języki zapytań XQuery, XPath lub DOM, SAX

W porównaniu ze stosowanymi dotychczas sposobami zapisu modeli drążenia danych standard PMML ma wiele zalet. W tabeli 2.2 przedstawiono porównanie języka PMML z innymi metodami zapisu modeli drążenia danych. Do korzyści związanych z wykorzystaniem PMML'a można zaliczyć to, że informacje opisujące model są przechowywane w pliku tekstowym, dzięki czemu można w łatwy sposób przenosić modele drążenia pomiędzy różnymi aplikacjami i systemami data mining. W przypadku modeli zapisywanych w relacyjnych bazach danych proces ich przenoszenia jest utrudniony i często wymaga dodatkowych przekształceń dla potrzeb docelowej aplikacji. Dla oprogramowania, które przechowują modele w wewnętrznym formacie zapisu, istnieje pełna zależność pomiędzy systemem drążenia danych, a sposobem zapisu modelu, skutkiem czego w wielu przypadkach nie

ma możliwości dostępu do modelu w inny sposób niż ten, który przewidzieli twórcy danego oprogramowania. W przypadku relacyjnych baz danych dostęp do poszczególnych elementów modelu odbywa się przy pomocy języka SQL lub innego języka bazującego na tym standardzie. Natomiast, dla języka PMML wykorzystywane są języki zapytań dla struktur XML takie jak XQuery, XPath lub też interfejsy programowania takie jak DOM (*Document Object Model*) czy SAX (*Simple API for XML*).

2.4.1 Struktura dokumentu PMML

Każdy dokument PMML (najnowsza wersja 3.1) ma określoną strukturę, która zdefiniowana jest przy pomocy standardu XML Schema. Struktura ta, pokazana na rysunku 2.8, składa się nie tylko z bloków opisujących sam model, ale również z informacji dotyczących danych wykorzystanych do budowy modelu.



Rysunek 2.8: Struktura dokumentu PMML [83, 84]

Można tu wyróżnić takie elementy jak [64, 70]: **Nagłówek** (Header) oraz **Blok konfiguracyjny** – ogólny opis aplikacji tworzącej model, parametrów takich jak czas, prawa autorskie itp., **Słownik danych** (Data Dictionary) – opis pól będących źródłem danych dla modelu, **Transformacje** – opis modyfikacji danych źródłowych (np. normalizacja, dyskretyzacja, agregacje itp.). Dzięki możliwości zapisu dodatkowych informacji powiązanych z modelem, język PMML może być wszechstronnie wykorzystany na poszczególnych etapach procesu pozyskiwania wiedzy, nie tylko jako standard zapisu modeli [14].

Ostatnim elementem dokumentu PMML jest model lub modele danych. Standard PMML (w wersji 3.1) wspiera szereg najpopularniejszych modeli drążenia danych, jak również modeli statystycznych. Należą do nich między innymi modele: reguł asocjacyjnych, drzewa decyzyjnego, sieci neuronowych, segmentacji, sekwencji, Bayesa i inne.

2.4.2 Model reguł asocjacyjnych w standardzie PMML

Wśród szeregu modeli statystycznych i drążenia danych, zdefiniowanych w języku PMML dostępny jest model reguł asocjacyjnych. Struktura zapisu tego modelu składa się z pięciu elementów (tabela 2.3).

Głównym elementem opisującym model asocjacyjny w języku PMML jest *AssociationModel*. Przy jego pomocy zapisywane są parametry budowy utworzonego modelu, w szczególności wartości progowe wsparcia i zaufania, jak również liczba reguł, zbiorów częstych, pozycji w modelu itp. Kolejnymi elementami są: *MiningSchema* – opis atrybutów, które były źródłem danych dla modelu, *Item* – opis pozycji modelu wraz z identyfikatorami, *Itemset* – opis zbiorów kandydujących (wartość wsparcia oraz referencje do pozycji składowych zbioru) oraz *AssociationRule* – opis reguł asocjacyjnych (referencje do zbiorów częstych dla następnika i poprzednika reguły oraz parametry wsparcia i zaufania dla reguły).

Tabela 2.3: Elementy modelu reguł asocjacyjnych w PMML

Element	**Opis**
AssociationModel	Główny element modelu, przechowuje parametry modelu np. liczba reguł, zbiorów częstych, pozycji, wielkości progowe wsparcia i zaufania itp.
Przykład	
<**AssociationModel** minimumConfidence="0.0" numberOfRules="4" avgNumberOfItemsPerTA="5.0" minimumSupport="0.3" numberOfTransactions="74150" numberOfItemsets="6" modelName="New association rules model" functionName="associationRules" numberOfItems="4" maxNumberOfItemsPerTA="5">	
MiningSchema	Zawiera opis atrybutów wykorzystanych do budowy modelu
Przykład	
<MiningSchema> <MiningField name="item" outliers="asIs" usageType="active" /> </MiningSchema>	
Item	Identyfikuje pozycje w modelu
Przykład	
<Item value="frontpage" id="0" /> <Item value="news" id="1" />	
Itemset	Identyfikuje zbiory kandydujące w modelu
Przykład	
<Itemset numberOfItems="2" support="0.07556" id="2"> <ItemRef itemRef="0" /> <ItemRef itemRef="1" /> </Itemset>	
AssociationRule	Identyfikuje reguły w modelu
Przykład	
<AssociationRule confidence="0.58821" support="0.01477" consequent="1" antecedent="25" /> <AssociationRule confidence="0.11911" support="0.01477" consequent="2" antecedent="36" />	

Porównując zapis modelu asocjacyjnego w języku PMML (tabela 2.2) z zapisem modelu w relacyjnych bazach danych (np. System SMART-Skip [37]) można stwierdzić, że w języku PMML wszystkie informacje dotyczące modelu przechowywane są w jednym zbiorze danych, a nie jak ma to miejsce w kilku tabelach danych. Łatwy dostęp oraz kompletność informacji o parametrach modelu stanowi o wyższości tego języka dla potrzeb przechowywania modelu reguł asocjacyjnych nad jego odpowiednikiem dla baz danych.

2.5 Przetwarzanie dokumentów w XML

Wraz z rozwojem języka XML oraz opartych na nim technologii wykorzystywanych w coraz to nowych dziedzinach zastosowań, takich jak: bazy danych, sieci WWW, multimedia, komunikacja, drążenie danych itp., koniecznym stało się zapewnienie odpowiednich standardów dostępu i przetwarzania tego typu danych. Wśród wielu rozwiązań, opracowanych dla wciąż rosnących potrzeb w tej dziedzinie, można wskazać kilka typów narzędzi:

- interfejsy API,
- języki transformacji dokumentów,
- języki zapytań.

2.5.1 Rodzaje narzędzi przetwarzania XML

Interfejsy API są bibliotekami procedur, które pozwalają na dostęp do poszczególnych elementów zbioru XML i są implementowane dla wieku języków programowania. Jednym z tego typu interfejsów jest Document Object Model (DOM). Prace nad tym standardem rozpoczęły się w połowie lat 90-tych, a najnowsza rekomendacja World Wide Web Consortium (W3C), DOM level 3, została przedstawiona w kwietniu 2004 roku [91]. Cechą charakterystyczną standardu DOM jest to, że dokument XML w całości przechowywany jest w pamięci operacyjnej w postaci drzewa węzłów, co

umożliwia stały dostęp do danych XML. Wadą tego rozwiązania jest zapotrzebowanie na pamięć operacyjną, które w przypadku plików o dużych rozmiarach, może być znaczące. Odmienne podejście do przetwarzania dokumentów XML zostało zastosowane w konkurencyjnym do DOM interfejsie Simple API for XML (SAX). W tym nieoficjalnym, ale popularnym, standardzie zbiór danych analizowany jest w sposób sekwencyjny, poprzez operowanie na kolejnych fragmentach dokumentu XML. Dzięki temu rozwiązaniu, wymagania związane z pamięcią operacyjną są mniejsze oraz prędkość przetwarzania jest wyższa, niż ma to miejsce w przypadku Document Object Model. Minusem interfejsu SAX jest natomiast to, że przetwarzanie odbywa się tylko w jednym kierunku (brak możliwości powrotu do elementów wcześniejszych) oraz brak kontroli nad poprawnością dokumentu, która wymaga całościowej analizy zbioru XML [39].

Drugą grupę narzędzi operujących na danych XML są języki transformacji dokumentów. Głównym zadaniem, w którym są wykorzystywane, jest zautomatyzowany proces wyboru, dokonywania zmian oraz przenoszenia informacji, których źródłem są zbiory XML. Wynik działania może być zapisany w dokumencie zgodnym z XML lub innym formacie tekstowym np. strony HTML itp. Przedstawicielem tego typu narzędzi jest język XSLT, którego oficjalna rekomendacja organizacji W3C w wersji 2.0 została opublikowana w styczniu 2007 roku [94]. W języku XSLT tworzone są tzw. arkusze styli, które określają warunki wyboru elementów XML, ich transformacji oraz sposób formatowania dokumentu docelowego. Choć zaimplementowane w XSLT mechanizmy pozwalają na wykorzystanie go nie tylko jako narzędzia do transformacji czy wizualizacji dokumentów XML, ale również jako języka zapytań do danych XML, to dość skomplikowana składnia stanowi jego wadę w tego typu zastosowaniu [47].

Ostatnią z wymienionych grup stanowią języki zapytań dla dokumentów XML. Pełnią one podobną rolę jak powszechnie wykorzystywany w relacyjnych bazach danych język SQL. Pierwszym tego typu językiem dla

danych XML był język Lorel opracowany w ramach przedsięwzięcia LORE w Uniwersytecie Stanforda w 1997 roku, jako język zapytań dla danych półstrukturalnych [1]. W późniejszym okresie (rok 1999), język ten został dostosowany do potrzeb przetwarzania danych XML [32]. W roku 1998 powstaje, wprowadzony przez firmę AT&T, język XML-QL [24], rozwiązanie open source pod nazwą XQL [72], oraz w kolejnych latach, opracowany na Politechnice Mediolańskiej, graficzny język XML-GL [15] oraz standard XPath, wykorzystywany do adresowania elementów dokumentów XML [92]. Na podstawie tych prac, Chamberlin i in. zdefiniował język Quilt [16], który stał się pierwowzorem języka zapytań XQuery.

2.5.2 Język zapytań XQuery

Pierwsza wersja języka XQuery, jako tzw. working draft, została przedstawiona w lutym 2001 roku. Od tego czasu trwały prace nad rozwojem tego języka, aby w styczniu 2007 stał się on oficjalnym standardem, wskazywanym przez organizację World Wide Web Consortium, jako język zapytań dla dokumentów XML [93].

Główną zaletą języka XQuery jest zwięzła, przypominająca stylem język zapytań SQL, struktura zapytania nazywana w skrócie „FLWOR". Skrót ten pochodzi od podstawowych instrukcji występujących w zapytaniach XQuery: **for** (instrukcja pętli), **let** (instrukcja przypisania wartości), **where** (instrukcja warunku), **order by** (instrukcja sortowania) i **return** (instrukcja zwracająca wynik). Oprócz wymienionych wyrażeń język XQuery wyposażony jest również w instrukcję warunkową oraz wiele wbudowanych funkcji bibliotecznych, które można uzupełnić o deklaracje własnych.

Jako przykład wykorzystania standardu XQuery w dziedzinie drążenia danych można wymienić prace Bragi [9] i Wana [88], w których wykorzystują ten język do budowy modeli reguł asocjacyjnych, dla których źródłem danych są zbiory XML. W proponowanym w niniejszej pracy rozwiązaniu

XQuery zostało wykorzystane do selekcji reguł z modelu danych zapisanych w PMML, które przedstawiono w [83].

W treści algorytmu 2.1 przedstawiono przykład zapytania, którego zadaniem było wyselekcjonowanie wszystkich reguł z modelu zapisanego w zbiorze *model.xml*, których parametr wsparcia przekracza wartość 0.2 oraz wynik zapytania jest uporządkowany malejąco według parametru zaufania.

```
for $i in doc("model.xml")//AssociationRule
where $i/@support>0.2
order by $i/@confidence descending
return $i
```

Algorytm 2.1: Przykład zapytania XQuery

Zapytanie XQuery składa się z przynajmniej jednego tzw. wyrażenia ścieżkowego XPath, którego zadaniem jest wybór odpowiednich elementów z dokumentu XML. W przedstawionym przykładzie (algorytm 2.1) zapis ścieżki (*doc("model.xml")//AssociationRule*) jest interpretowany, jako wybór wszystkich elementów *AssociationRule* ze zbioru danych *model.xml*. Następnie w pętli **for** elementy te są kolejno przypisywane zmiennej *$i*. W przypadku, kiedy wartość atrybutu *support* jest większa od 0.2, wtedy wartość zmiennej *$i* zostaje zwrócona przez zapytanie. Dodatkowo, wynik (wybrane elementy *AssociationRule*) są uporządkowane malejąco według atrybutu *confidence*.

Rozdział 3

Postać osobnika i kryterium jego oceny

W zaproponowanej w rozdziale 2 metodzie automatycznego generowania zapytań do modelu reguł asocjacyjnych stwierdzono, że podstawowym zadaniem wymagającym szczególnej uwagi jest zdefiniowanie struktury osobnika, która będzie poddawana działaniom ewolucyjnym oraz kryterium oceny osobników populacji. W rozdziale tym zostaną przedstawione propozycje rozwiązań tych zagadnień.

3.1 Zapytanie XQuery jako osobnik w programowaniu genetycznym

Analizując zapytanie z algorytmu 2.1 można zauważyć, że podstawą selekcji reguł ze zbioru danych jest zadanie warunku w klauzuli **where**. Możliwe jest uściślenie tego zapytania poprzez wprowadzenie dodatkowego ograniczenia do warunku, co pokazano w formule zaprezentowanej jako algorytm 3.1.

Kryterium wyboru reguł (algorytm 3.1) zostało rozszerzone o klauzulę, której zadaniem jest wyselekcjonowanie reguł, które spełniają dodatkowo warunek zaufania reguły na poziomie większym niż 0,6.

```
for $i in doc("model.xml")//AssociationRule
where   (@support>0.2 and @confidence>0.6)
order by $i/@confidence descending
return $i
```

Algorytm 3.1: Złożone zapytanie XQuery I

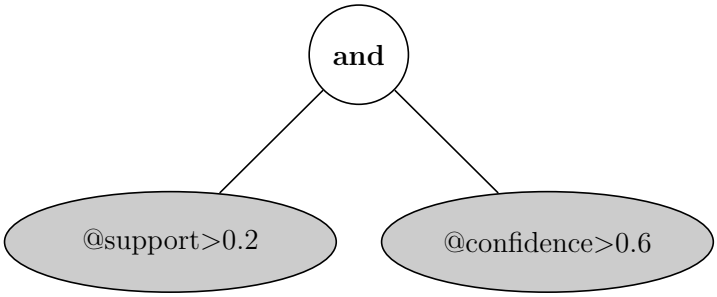
Oczywistym jest, że liczba reguł zwróconych przez zapytanie rozszerzone o dodatkowy warunek (algorytm 3.1) będzie mniejsza lub równa tej, którą zwróci zapytanie zapisane w postaci algorytmu 2.1. Można przewidywać, że poprzez odpowiednie sformułowanie wyboru reguł uzyska się możliwość oddziaływania na liczbę oraz rodzaj reguł, które otrzymamy w wyniku zapytania. Poddając warunek zapytania działaniom mechanizmów ewolucyjnych, opisanych w podrozdziale 2.3.3, można dążyć do znalezienia zapytania zwracającego najlepszy z punktu widzenia użytkownika podzbiór reguł. Aby uzyskać możliwość zastosowania takiej procedury, konieczne jest zdefiniowanie struktury osobnika, który będzie reprezentował zapytanie XQuery.

3.1.1 Definicja struktury osobnika

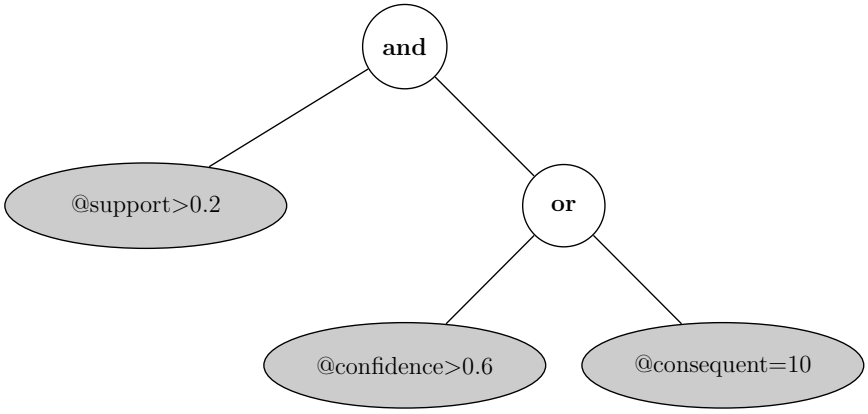
Według przedstawionych w podrozdziale 2.3.1 założeń i definicji, struktura osobnika w programowaniu genetycznym ma formę drzewa składającego się z funkcji i terminatorów. Analizując algorytm 3.1, warunek zapytania można przedstawić jako strukturę drzewiastą z operatorem logicznym **and** w węźle drzewa i wyrażeniami **@support>0.2** i **@confidence>0.6** w liściach. Struktura taka została przedstawiona na rysunku 3.1.

Pokazana na rysunku 3.1 struktura jest prostym drzewem o dwóch poziomach głębokości. Drzewo to można w łatwy sposób rozbudować, poprzez dodanie kolejnych węzłów będących operatorami logicznymi, tworząc w ten sposób rozbudowane wyrażenia warunkowe, jak to pokazano na rysunku 3.2. Odpowiada to warunkowi zawartemu w formule z algorytmu 3.2.

W tym przypadku, dla zachowania odpowiedniej kolejności wykonywania poszczególnych funkcji logicznych, zastosowano nawiasy okrągłe.



Rysunek 3.1: Warunek zapytania w strukturze drzewiastej



for $i **in** doc("model.xml")//AssociationRule
[@support>0.2 and (@confidence>0.6 or @consequent=10)]
order by $i/@confidence **descending**
return $i

Rysunek 3.2: Rozbudowany warunek zapytania w strukturze drzewiastej

for $i **in** doc("model.xml")//AssociationRule
where [@support>0.2 and (@confidence>0.6 or @consequent=10)]
order by $i/@confidence **descending**
return $i

Algorytm 3.2: Złożone zapytanie XQuery II

Na podstawie zaprezentowanych zasad tworzenia wyrażeń warunkowych zapisanych w postaci drzewa, można zdefiniować zbiór funkcji F, które będą wykorzystywane do tworzenia osobnika reprezentującego warunek zapytania do modelu.

$$F \equiv \{and, or, not\} \tag{3.1}$$

W zbiorze (3.1) uwzględniono 3 operatory logiczne: **and** – iloczyn logiczny, **or** – suma logiczna oraz **not** – negacja, które są dostępne w ramach składni języka XQuery.

Drugim elementem wymaganym do zdefiniowania struktury osobnika jest zbiór terminatorów. W proponowanym rozwiązaniu zbiór terminatorów (T) składa się z jednego elementu: *wyrażenie*.

$$T \equiv \{wyrażenie\} \tag{3.2}$$

Element *wyrażenie* w (3.2) zawiera trzy składniki:

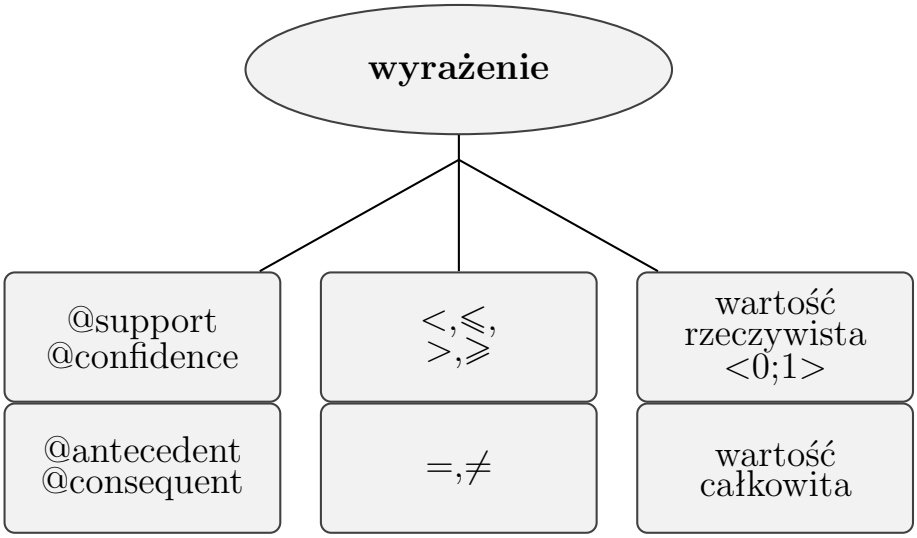
- nazwa atrybutu,
- operator porównania,
- wartość rzeczywista $\in < 0, 1 >$ lub wartość całkowita.

Na rysunku 3.3 przedstawiono dwa warianty wyrażenia:

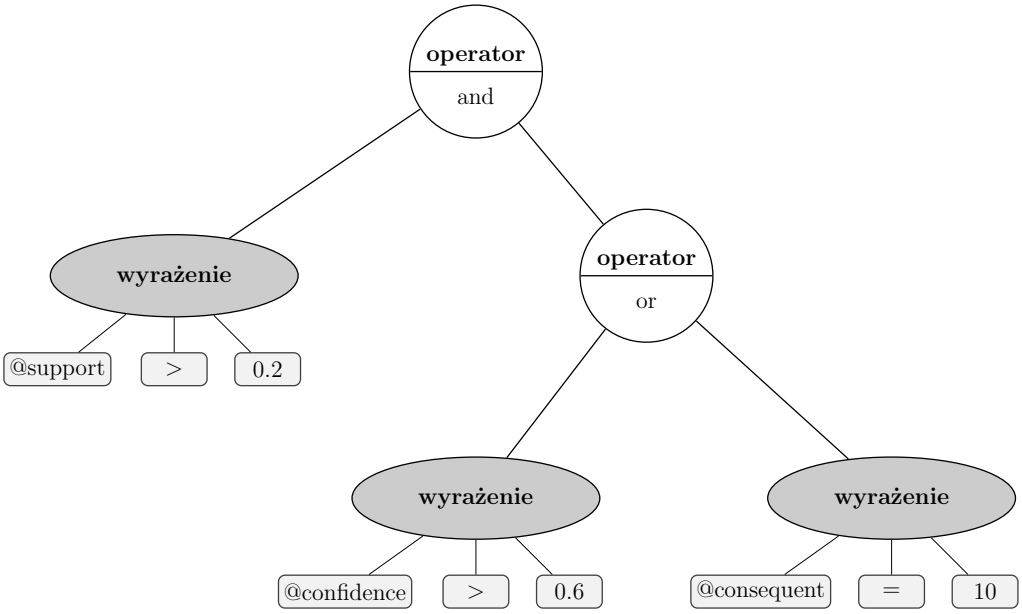
1. atrybutom wsparcia lub zaufania reguły przypisywany jest operator ze zbioru $\{<, \leqslant, >, \geqslant\}$ oraz wartość rzeczywista z przedziału $< 0, 1 >$,
2. atrybutom poprzednika lub następnika reguły przypisywany jest operator ze zbioru $\{=$ (równy), $\neq$ (różny)$\}$ oraz wartość całkowita, która jest identyfikatorem zbioru częstego.

Dla potrzeb dostosowania operatorów: $\leqslant, \geqslant$ i $\neq$ do składni języka XQuery, w wyrażeniach warunku stosuje się następujące ich odpowiedniki: $<=, >=$ oraz $!=$.

Na rysunku 3.4 pokazano kompletną strukturę osobnika dla rozbudowanego wyrażenia warunkowego 3.2.



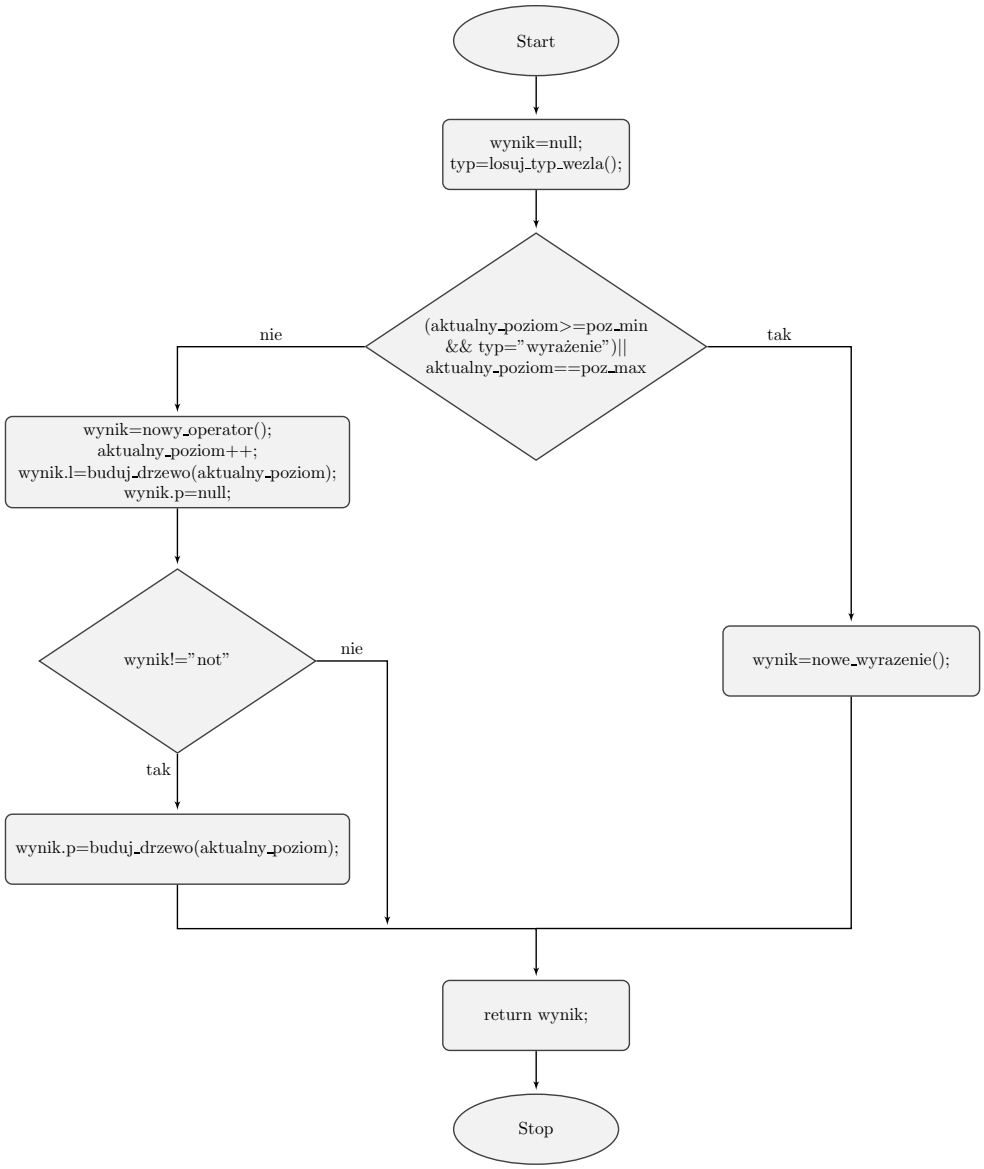
Rysunek 3.3: Struktura wyrażenia terminatorowego



Rysunek 3.4: Przykład struktury osobnika

3.1.2 Algorytm budowy osobnika

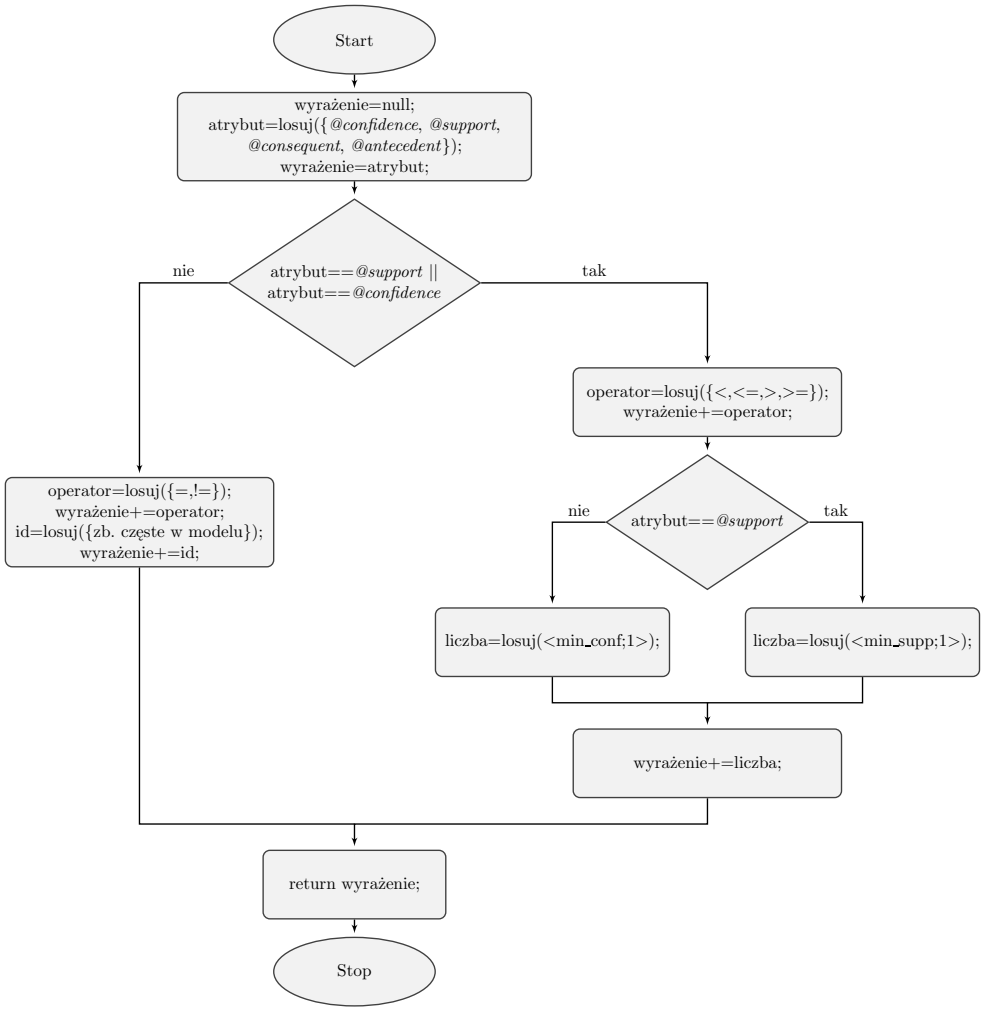
Dla potrzeb generowania osobników populacji opracowano algorytm budowy pojedynczego osobnika przedstawiony na rysunku 3.5.



Rysunek 3.5: Schemat algorytmu budowy osobnika

Procedura *buduj_drzewo* jest procedurą rekurencyjną, generującą poszczególne węzły drzewa w dół (od korzenia do liści). W pierwszej kolejności losowany jest typ węzła, który może przyjąć dwie wartości: *wyrażenie* lub

operator. Następnie sprawdzany jest warunek, czy *aktualny_poziom* głębokości drzewa programu przekroczył ustalony poziom minimalny oraz czy wylosowany rodzaj węzła jest wyrażeniem lub też został osiągnięty poziom maksymalny, w takim przypadku jako węzeł generowane jest wyrażenie. Jeżeli żaden z powyższych warunków nie został spełniony, jako węzeł osobnika przypisywany jest operator ze zbioru funkcji. W przypadku, gdy wylosowany zostanie operator jednoargumentowy **not**, drzewo na kolejnym poziomie jest budowane tylko dla lewego podwęzła. Jeżeli operator jest dwuargumentowy (**and** lub **or**), rozrost drzewa następuje również dla prawego podwęzła.



Rysunek 3.6: Schemat algorytmu budowy wyrażenia

Procedura odpowiedzialna za generowanie wyrażenia, przedstawiona na rysunku 3.6, wykorzystuje opisane w podrozdziale 3.1.1 dwa warianty jego budowy. W pierwszej kolejności losowany jest atrybut ze zbioru {*@confidence*, *@support*, *@consequent*, *@antecedent*}, a następnie, w zależności od wybranego atrybutu, odpowiedni operator logiczny oraz wartość liczbowa. Wartość liczbowa dla wsparcia i zaufania losowana jest z przedziału liczb rzeczywistych, ograniczonego do minimalnych i maksymalnych wartości tych parametrów w modelu. Dla atrybutów poprzednika i następnika wybierana jest wartość losowa ze zbioru identyfikatorów zbiorów częstych, które wystąpiły w regułach modelu.

W tabeli 3.1 pokazano w kolejnych krokach przykład realizacji algorytmów budowy drzewa dla osobnika z rysunku 3.2.

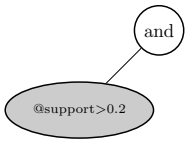
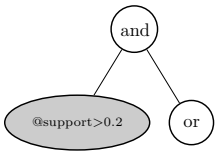
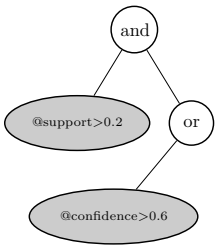
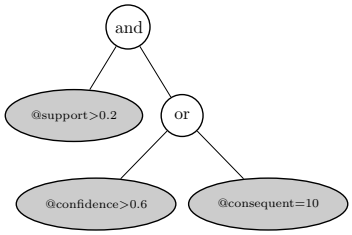
3.2 Typy kryteriów oceny reguł

Ocena wartości informacyjnej, jaką przedstawiają wzorce pozyskane w procesie drążenia danych jest jednym z głównych problemów w dziedzinie pozyskiwania wiedzy [77] i stanowi przedmiot wielu badań. Szereg publikacji dotyczących tej tematyki rozpatruje dwa podstawowe typy miar określających, jak bardzo interesująca dla badacza jest dana reguła (ang. *interestingness measures*): miary obiektywne (ang. *objective measures*) i subiektywne (ang. *subjective measures*).

3.2.1 Miary obiektywne

Miary obiektywne można zdefiniować jako kryteria determinowane przez strukturę danych (ang. data-driven) i są one określane w oparciu o metody matematyczne i statystyczne. Jako główne zalety tego typu kryteriów można wskazać ich niezależność od rozpatrywanej domeny oraz to, że nie wymagają ingerencji ze strony użytkownika [78]. Dzięki tym właściwościom miary obiektywne można w łatwy sposób zaimplementować w systemach

Tabela 3.1: Przykład działania algorytmów budowy osobnika

krok	**typ węzła**	**wylosowane wartość**	**postać drzewa**
1	operator	and	
2	wyrażenie	@support > 0.2	
	operator	or	
3	wyrażenie	@confidence > 0.6	
	wyrażenie	@consequent = 0.6	

pozyskiwania wiedzy dla potrzeb automatycznej oceny otrzymanych wzorców.

Wśród najczęściej stosowanych w procesie budowy modelu reguł asocjacyjnych miar obiektywnych wykorzystuje się współczynnik wsparcia i zaufania reguły przedstawione w podrozdziale 1.3.1.

Miara wsparcia przydatna jest w szczególności do nadawania dolnych ograniczeń dla algorytmu *Apriori* (1.1), w celu zmniejszenia liczby generowanych reguł. Zastosowanie parametru wsparcia do redukcji liczby reguł nie zawsze jednak daje pożądane rezultaty w przypadku, gdy poszukiwane

zależności występują w niewielkim podzbiorze transakcji (np. wykrywanie oszustw w danych bankowych, które stanowią niewielką część wszystkich transakcji).

W celu określenia dokładności reguły stosuje się parametr zaufania wskazujący, jak duża grupa rekordów spośród rozpatrywanych transakcji, w których wystąpił poprzednik, zawiera również następnik. Parametr zaufania można opisać zależnością:

$$conf(A \rightarrow B) = \frac{sup(A \rightarrow B)}{sup(A)} \tag{3.3}$$

Kolejną podobną do kryterium zaufania jest zaprezentowana w [19] miara *Laplace'a*:

$$lapl(A \rightarrow B) = \frac{sup(A \rightarrow B) + 1}{sup(A) + 2} \tag{3.4}$$

Jak wykazują badania [11, 81] parametry zaufania oraz *Laplace'a* mogą być często błędnie interpretowane w przypadkach, gdy przyjmują duże wartości pomimo, że powiązanie pomiędzy poprzednikiem i następnikiem reguły nie jest silne. Można to stwierdzić poprzez analizę reguł przeciwnych, w których zanegowanie poprzednika reguły daje wyższe wartości tych współczynników. Wynika to ze sposobu obliczania wartości tego parametru – zbiory częste w następniku reguły o dużej częstości występowania są powodem dużej wartości wymienionych kryteriów.

Jedną z miar określających stopień powiązania pomiędzy zbiorami częstymi poprzednika i następnika reguły jest zdefiniowana w [11] miara *lift* (3.5) (często w literaturze określana również jako miara *interest*).

$$lift(A \rightarrow B) = \frac{conf(A \rightarrow B)}{sup(B)} \tag{3.5}$$

Mierzony stopień powiązania może przyjąć następujące wartości:

- $lift = 1$ – zdarzenia niezależne
- $lift < 1$ – zdarzenia skorelowane negatywnie – wymaga rozważenia reguły odwrotnej, jako bardziej wiarygodnej

– $lift > 1$ – zdarzenia skorelowane pozytywnie

Miara *lift* jest jedną z częściej stosowanych w profesjonalnych systemach drążenia danych.

Kolejną alternatywną miarą, określającą powiązanie zbiorów częstych w regule, jest zdefiniowana przy pomocy wyrażenia (3.6) miara *conviction* [12].

$$conv(A \rightarrow B) = \frac{1 - sup(B)}{1 - conf(A \rightarrow B)} \tag{3.6}$$

Jako główną przewagę tego kryterium nad miarą lift wymienia się [6] jej niesymetryczność tj. $(conv(A \rightarrow B) \neq conv(B \rightarrow A))$. Drugą wskazywaną zaletą jest to, że wykorzystuje miarę wsparcia dla poprzednika oraz następnika reguły, co rozwiązuje problem nakreślony dla miar zaufania (3.3) i *Laplacea* (3.4). Natomiast wadę tej miary stanowi fakt, że dla wartości zaufania równej 1, przyjmuje ona wartość nieokreśloną.

Jakościowo innym kryterium od wyżej wymienionych miar jest wywodząca się z teorii informacji *J-miara*. Miara ta została zaproponowana w [79] i określana jest wyrażeniem:

$$\begin{aligned} J\text{-}miara(A \rightarrow B) = sup(A) \cdot \Bigg[& conf(A \rightarrow B) \cdot ln\left(\frac{conf(A \rightarrow B)}{sup(B)}\right) \\ & + (1 - conf(A \rightarrow B)) \cdot ln\left(\frac{1 - conf(A \rightarrow B)}{1 - sup(B)}\right)\Bigg] \end{aligned} \tag{3.7}$$

Do głównych zalet tego obiektywnego kryterium należy zaliczyć to, że podobnie jak *conviction* jest niesymetryczne oraz zakres przyjmowanych przez nie wartości należy do przedziału $< 0, 1 >$. Kolejną zaletę *J-miary* przedstawiono w pracy [51] wskazując, że przyjmuje ona większe wartości dla reguł o skrajnym wsparciu następnika i miary zaufania. Dzięki tej właściwości można wykorzystać tę miarę do potwierdzenia lub zaprzeczenia postawionych hipotez. Modyfikując wyrażenie (3.7), można zdefiniować *J-miarę* odwrotną:

$$J\text{-}miara'(A \rightarrow B) = 1 - J\text{-}miara(A \rightarrow B) \tag{3.8}$$

która powinna preferować reguły o średnich wartościach: wsparcia następnika i zaufania reguły. Dzięki tej właściwości, reguły oczywiste, dobrze znane badaczowi, powinny zostać ocenione gorzej, co zwiększa prawdopodobieństwo odkrycia reguł wcześniej nieznanych lub zaskakujących.

W wielu publikacjach, na przykład [6, 85], omówiono szereg innych miar obiektywnych. Tutaj przedstawiono te najczęściej omawiane i wykorzystywane w systemach pozyskiwania wiedzy.

3.2.2 Miary subiektywne

Zastosowanie miar obiektywnych (opisanych w podrozdziale 3.2.1) nie zawsze daje pożądane rezultaty [63], w szczególności, gdy duża liczba reguł spełnia wartości progowe kryteriów. Wysoka wartość miar obiektywnych częstokroć wskazuje na powiązania, które są dobrze znane analitykowi, a co za tym idzie ich wartość informacyjna jest mała. Kolejnym problemem jest to, że miary obiektywne analizują reguły bez powiązania tego procesu ze wskazaniami badacza co do „zawartości" reguł – wartości atrybutów, które powinny lub nie pojawić się w regule.

Dzięki miarom subiektywnym analityk ma możliwość określenia kryteriów poszukiwania reguł na podstawie swojej wiedzy dziedzinowej, przy wykorzystaniu doświadczeń, przesłanek czy też przypuszczeń z tej wiedzy wynikających.

Jedną z propozycji miary subiektywnej [48] jest zdefiniowanie jej jako szablonu w postaci wyrażeń (3.9) i porównywanego z otrzymanymi w procesie drążenia wzorcami.

$$A_1, A_2, \ldots, A_k \rightarrow A_{k+1} \tag{3.9}$$

gdzie:

A_i może być nazwą atrybutu, wartością atrybutu lub też wyrażeniem typu:

$C+$ – jedna lub więcej wartości ze zbioru wartości atrybutu C

$C*$ – zero lub więcej wartości ze zbioru wartości atrybutu C.

Poszczególne reguły porównywane są z wyrażeniami wzorca, przy czym analityk ma możliwość zdefiniowania wzorca „pozytywnego" – akceptującego reguły i „negatywnego" – odrzucającego reguły. Główną niedogodnością tego kryterium jest to, że badacz nie ma możliwości wskazania stopnia pewności z jaką określa poszczególne wyrażenia w definicji wzorca.

Podobne rozwiązanie zaprezentowano w publikacji [52], przy czym wzorce poszukiwanych reguł wykorzystują zagadnienia związane z teorią logiki rozmytej. W tym przypadku poszczególne wzorce mają reprezentację następującą:

$$\textbf{If } P_1, P_2, \ldots, P_n \textbf{ then } C \tag{3.10}$$

gdzie:

P_i jest wyrażeniem typu: (*nazwa atrybutu*) OP (*dopuszczalna wartość*), OP $\in \{=, \neq, <, >, \leqslant, \geqslant\}$ – operator,

C – wyrażenie formułowane podobnie jak P_i, przy czym wartość atrybutu nie musi wystąpić w bazie danych, ale może zostać zdefiniowana przy pomocy reguł logiki rozmytej.

Główną wadą kryteriów subiektywnych jest to, że przy ich pomocy można wyszukiwać reguły spodziewane i oczekiwane [53], w wyniku czego możliwe jest jedynie potwierdzenie już istniejących przypuszczeń, a odkrywanie nowych powiązań, wcześniej nieznanych jest tylko przypadkowe.

3.3 Propozycja kryterium oceny

Dla prawidłowego działania algorytmu genetycznego niezbędny jest dobór odpowiedniego kryterium oceny osobników populacji. W proponowanej metodzie automatycznego generowania zapytań zadaniem kryterium jest ocena podzbioru reguł asocjacyjnych zwróconych przez zapytanie. Przy definiowaniu funkcji oceny przyjęto następujące założenia:

1. Kryterium oceny powinno być połączeniem miary obiektywnej z subiektywną,

   Jak wskazują badania np. [53, 78], połączenie miar obiektywnych – wskazujących na ważność reguły pod względem statystycznym – oraz miar subiektywnych – oceniających reguły z punktu widzenia wiedzy analityka, pozwoli ograniczyć wady obu tych rozwiązań.
2. Kryterium oceny powinno być definiowane w sposób intuicyjny, bez konieczności „ręcznego" wprowadzania skomplikowanych wyrażeń i wzorów.

   Metoda, aby mogła znaleźć zastosowanie w różnych dziedzinach wiedzy oraz była łatwo akceptowalna przez analityków, nie powinna wymagać od nich specjalistycznej wiedzy z zakresu informatyki oraz powinna być zrozumiała i prosta w użyciu.
3. Podczas definicji miary subiektywnej użytkownik powinien mieć możliwość stopniowania ważności poszczególnych powiązań.

   Eksperci nie zawsze potrafią przedstawić swoją wiedzę w sposób dokładny, ale raczej z pewnym przybliżeniem lub prawdopodobieństwem, dlatego też ważnym jest, aby określenie kryterium było elastyczne i pozwalało na wskazanie stopnia pewności, z jaką definiowane jest dany wzorzec miary subiektywnej.
4. Kryterium oceny powinno preferować te zapytania, które zwracają liczbę reguł ograniczoną do pewnego, wskazanego przez użytkownika, poziomu liczności.

   Zbyt wiele reguł zwracanych przez zapytanie będzie, podobnie jak rozbudowany model, zbyt trudne w analizie, dlatego kryterium powinno oceniać zarówno liczbę zwracanych reguł, jak też ich zgodność z preferencjami badacza.

3.3.1 Definicje

Przy formułowaniu proponowanego przez autora kryterium wykorzystane zostaną następujące pojęcia i definicje.

Definicja 6. *Zbiorem wag W jest nazwany, zdefiniowany przez użytkownika, rosnący ciąg liczb całkowitych, którego i-ty element w_i określa stopień ważności powiązania, przy czym najmniejsza wartość ciągu wskazuje na element najmniej ważny.*

Definicja 7. *Wagę znormalizowaną $\bar{w}_i$ i-tego elementu ze zbioru wag W definiuje zależność:*

$$\bar{w}_i = \frac{w_i - w_{min} + 1}{w_{max} - w_{min} + 1}, \quad \bar{w}_i \in (0, 1>$$

gdzie:

w_i – *waga i-tego elementu ze zbioru wag W*

w_{min} – *wartość minimalna ze zbioru wag W*

w_{max} – *wartość maksymalna ze zbioru wag W*

Definicja 8. *Powiązaniem jest nazwane wyrażenie typu:*

$$X \twoheadrightarrow Y, \ \textit{przy} \ X \neq Y$$

gdzie:

X, Y – *odpowiednio poprzednik i następnik powiązania, które są wybranymi przez użytkownika wartościami atrybutów źródłowej bazy danych.*

Definicja 9. *Wzorcem kryterium WK jest nazwany, niepusty zbiór powiązań, którym przypisano wagi znormalizowane.*

Definicja 10. *Wagą A_{p_i} i-tej pozycji dla poprzednika reguły modelu asocjacyjnego jest nazwana średnia wartość wag znormalizowanych tych powiązań ze wzorca kryterium WK, w których dana pozycja występuje w poprzedniku powiązania lub 0, gdy nie występuje w żadnym poprzedniku powiązania.*

Definicja 11. *Wagą A_{n_i} i-tej pozycji dla następnika reguły modelu asocjacyjnego jest nazwana średnia wartość wag znormalizowanych tych powiązań ze wzorca kryterium WK, w których dana pozycja występuje w następniku powiązania lub 0, gdy nie występuje w żadnym następniku powiązania.*

Definicja 12. *Regułą mocną R_M w zbiorze reguł modelu asocjacyjnego określa się jako regułę, w której występują co najmniej dwie pozycje, których waga jest większa od 0.*

Definicja 13. *Regułą słabą R_S w zbiorze reguł modelu asocjacyjnego określa się jako regułę, w której występuje tylko jedna pozycja, której waga jest większa od 0.*

Definicja 14. *Wagę SW_{p_i} i-tego zbioru częstego Z_i dla poprzednika reguły modelu asocjacyjnego definiuje zależność:*

$$SW_{p_i} = \sum_{Z_i} A_{pZ_i}$$

Definicja 15. *Wagę SW_{n_i} i-tego zbioru częstego Z_i dla następnika reguły modelu asocjacyjnego definiuje zależność:*

$$SW_{n_i} = \sum_{Z_i} A_{nZ_i}$$

Definicja 16. *Przystosowanie P_Q reguł zwróconych przez zapytanie Q definiuje zależność:*

$$P_Q = \sum_{Q} SW_{p_i} + \sum_{Q} SW_{n_i}$$

Definicja 17. *Przystosowanie P_G wszystkich reguł ze zbioru reguł G definiuje zależność:*

$$P_G = \sum_{G} SW_{p_i} + \sum_{G} SW_{n_i}$$

Definicja 18. *Współczynnikiem oceniającym liczbę reguł ważnych (słabych lub mocnych, w zależności od poszukiwanego typu reguł) D_R zwróconych*

przez zapytanie definiuje zależność:

$$D_R = \begin{cases} 0, & N_Q = 0 \\ \frac{N_R}{N_Q}, & N_Q > 0 \end{cases}$$

gdzie:

N_Q – *liczba reguł zwróconych przez zapytanie*

N_R – *liczba reguł ważnych*

Definicja 19. *Współczynnik oczekiwanej liczby reguł D_N zwróconych przez zapytanie definiuje zależność:*

$$D_N = \begin{cases} 1, & N_Q \leqslant N_O \\ \frac{N_O}{N_Q}, & N_Q > N_O \end{cases}$$

gdzie:

N_Q – *liczba reguł zwrócona przez zapytanie*

N_O – *oczekiwana liczba reguł wskazana przez badacza ($N_O \in \mathbb{N}$).*

3.3.2 Procedura definiowania kryterium oceny

Proponowane definiowanie kryterium oceny przebiega w kilku etapach:

1. Definiowanie zbioru wag

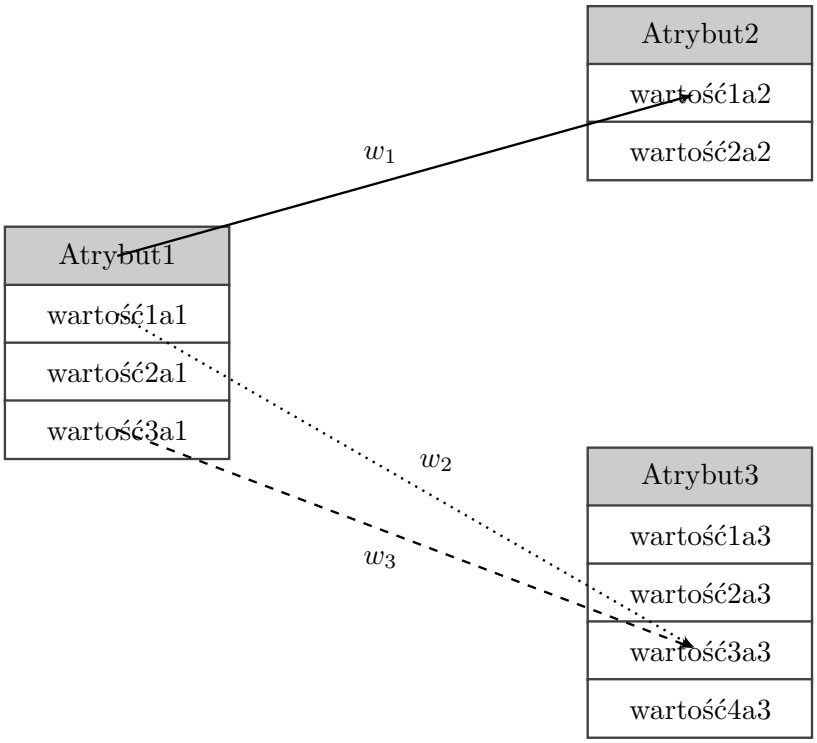
   Na tym etapie użytkownik określa zbiór wag zgodnie z definicją 6. Zbiór ten może być definiowany dla każdego eksperymentu (przeprowadzanej analizy) oddzielnie, dzięki czemu może uwzględniać specyfikę dziedziny rozwiązywanego problemu.

   Dla każdej wagi, oprócz jej wartości, przypisywane są dodatkowo atrybuty koloru oraz rodzaju linii, dla potrzeb reprezentacji graficznej.

2. Definiowanie wzorca kryterium

   Etap definiowania wzorca kryterium przebiega w następujących krokach:

– wskazanie zbioru danych do analizy,
– wybór atrybutów ze zbioru danych,
– zdefiniowanie powiązań pomiędzy wartościami atrybutów.



Rysunek 3.7: Definicja wzorca kryterium; w_1,w_2,w_3 – wagi powiązań, $w_i \in W$, wartość(j)a(x) – j-ta wartość atrybutu x

Jak pokazano na rysunku 3.7 poszczególne powiązania definiowane są poprzez wybór odpowiedniej wagi, a następnie wskazanie poprzednika i następnika powiązania.

Udostępnione są dwie możliwości wskazania powiązania:

– pomiędzy dwiema pojedynczymi wartościami atrybutów (waga w_2, w_3)
– pomiędzy atrybutem (wszystkimi jego wartościami) a wartością pojedynczą atrybutu lub odwrotnie (waga w_1).

Dla poprawnego działania proponowanej metody konieczne jest wskazanie przynajmniej jednego powiązania.

Tabela 3.2: Wzorzec kryterium

Powiązanie	**Waga znormalizowana**
(wartość1a1) $\twoheadrightarrow$ (wartość1a2)	$\bar{w}(1)$
(wartość2a1) $\twoheadrightarrow$ (wartość1a2)	$\bar{w}(1)$
(wartość3a1) $\twoheadrightarrow$ (wartość1a2)	$\bar{w}(1)$
(wartość1a1) $\twoheadrightarrow$ (wartość3a3)	$\bar{w}(2)$
(wartość3a1) $\twoheadrightarrow$ (wartość3a3)	$\bar{w}(3)$

W wyniku zdefiniowanych powiązań przedstawionych na rysunku 3.7 tworzony jest zgodnie z definicją 9 wzorzec kryterium. Wzorzec ten składa się z 5 powiązań pokazanych w tabeli 3.2.

3. Parametry wyszukiwania

   Oprócz definicji wag oraz wzorca kryterium, konieczne jest również wskazanie dodatkowych parametrów, których zadaniem będzie uszczegółowienie rodzaju oczekiwanych rezultatów.

   Do parametrów wyszukiwania należy podać liczbę oczekiwanych reguł N_O, której oczekuje analityk. Wartość ta określa, jak maksymalnie duży podzbiór reguł ma zwracać zapytanie, które badacz potrafi przeanalizować i zinterpretować. Na jego podstawie definiowany jest współczynnik oczekiwanej liczby reguł D_N (definicja 19), dzięki któremu zapytania zwracające zbyt dużą liczbę reguł będą „karane" zmniejszeniem funkcji przystosowania.

   Drugim parametrem definiowanym przez użytkownika jest typ poszukiwanych reguł. Na podstawie tego parametru następuje wybór odpowiedniego kryterium oceny zapytań w procesie ewolucyjnym, które

jest rozszerzeniem kryterium przedstawionego w [84]. Wyróżnia się tutaj dwie możliwości:

– poszukiwanie reguł, które mają potwierdzić wskazane wzorce lub im zaprzeczyć, zgodnie ze zdefiniowaną funkcją przystosowania F:

$$F = \left(P_{S_{R_M}} + P_{O_{R_M}}\right)/2 \cdot D_R \cdot D_N \tag{3.11}$$

gdzie:

$N_{R_{MQ}}$ – liczba reguł mocnych zwrócona przez zapytanie

$P_{S_{R_M}} = \begin{cases} 0, & P_{G_{R_M}} = 0 \\ \frac{P_{Q_{R_M}}}{P_{G_{R_M}}}, & P_{G_{R_M}} \neq 0 \end{cases}$ – miara subiektywna

$P_{O_{R_M}} = \begin{cases} 0, & N_{R_{MQ}} = 0 \\ \frac{\sum_{i=1}^{N_{R_{MQ}}} J\text{-}miara(R_{MQi})}{N_{R_{MQ}}}, & N_{R_{MQ}} \neq 0 \end{cases}$ – miara obiektywna

D_R – współczynnik oceniający liczbę reguł ważnych (definicja 18)

D_N – współczynnik oczekiwanej liczby reguł (definicja 19)

– poszukiwanie reguł nowych, wcześniej nieznanych, które wiążą się z wzorcem kryterium (F – funkcja przystosowania):

$$F = \left(P_{S_{R_S}} + P_{O_{R_S}}\right)/2 \cdot D_R \cdot D_N \tag{3.12}$$

gdzie:

$N_{R_{SQ}}$ – liczba reguł słabych zwrócona przez zapytanie

$P_{S_{R_S}} = \begin{cases} 0, & P_{G_{R_S}} = 0 \\ \frac{P_{Q_{R_S}}}{P_{G_{R_S}}}, & P_{G_{R_S}} \neq 0 \end{cases}$ – miara subiektywna

$P_{O_{R_S}} = \begin{cases} 0, & N_{R_{SQ}} = 0 \\ \frac{\sum_{i=1}^{N_{R_{SQ}}} J\text{-}miara'(R_{SQi})}{N_{R_{SQ}}}, & N_{R_{SQ}} \neq 0 \end{cases}$ – miara obiektywna

D_R – współczynnik oceniający liczbę reguł ważnych (definicja 18)

D_N – współczynnik oczekiwanej liczby reguł (definicja 19)

3.3.3 Interpretacja kryterium oceny

Zaproponowane w podrozdziale 3.3.2 kryterium łączy w sobie trzy elementy, które oceniają zapytanie: miarę obiektywną, miarę subiektywną oraz parametry wyszukiwania.

Miara obiektywna opiera się na przedstawionej w rozdziale 3.2.1 *J-mierze* oraz zaproponowanej jej wartości odwrotnej. W zależności od wybranego typu poszukiwanych reguł obliczana jest średnia wartości *J-miary* dla reguł zwróconych przez zapytanie i będących regułami mocnymi R_{MQ} lub *J-miary'* dla reguł słabych R_{SQ}. Ponieważ wartości *J-miary* i jej odwrotności należą do przedziału <0,1>, również średnia wartość tych miar dla reguł wchodzących w skład podzbioru reguł zwróconego przez zapytanie będzie zawierać się w tym zakresie. Im wyższe wartości *J-miary* (*J-miary'*) reguł wynikowych, tym wartość części obiektywnej większa. W przypadku, gdy liczba reguł mocnych (lub słabych) $N_{R_{MQ}}$ ($N_{R_{SQ}}$) zwróconych przez zapytanie jest równa 0, część obiektywna kryterium przyjmuje również wartość 0.

Część subiektywna kryterium $P_{S_{R_M}}$ ($P_{S_{R_S}}$) jest wynikiem ilorazu przystosowania reguł mocnych (lub słabych) $P_{Q_{R_M}}$ ($P_{Q_{R_S}}$) zwróconych przez zapytanie i przystosowania wszystkich reguł mocnych (słabych) $P_{G_{R_M}}$ ($P_{G_{R_S}}$) występujących w zbiorze G wszystkich reguł modelu. Zgodnie z definicjami 16,17 wartości przystosowań wyliczane są jako suma wag zbiorów częstych występujących w regułach, odpowiednio, wyników zapytania oraz całego modelu. Przystosowanie zapytania przyjmuje wartości $<0, P_{G_{R_M}}>$ (lub $< 0, P_{G_{R_S}}>$), przy czym najwyższa wartość osiągana jest wtedy, gdy zapytanie zwróci wszystkie reguły mocne (słabe), które występują w modelu. W przypadku, gdy wartość przystosowania wszystkich reguł modelu (która jest wyliczana na samym początku procesu ewolucyjnego) jest równa 0, oznacza to, że w modelu nie występują żadne reguły mocne (słabe), co wskazuje na brak powiązań zdefiniowanych we wzorcu kryterium. W przypadku,

gdy liczba reguł mocnych (słabych) $N_{R_{MQ}}$ ($N_{R_{SQ}}$) zwróconych przez zapytanie jest równa 0, część subiektywna kryterium przyjmuje również wartość 0.

Aby kryterium przystosowania poszczególnych osobników nie zostało zdominowane przez część obiektywną lub subiektywną, suma wartości obu tych części jest uśredniana.

Dla potrzeb oceny liczby reguł zwróconych przez zapytanie wprowadzono dodatkowy współczynniki D_R. Współczynnik ten na podstawie definicji 18 określa stosunek reguł ważnych – w zależności od typu wyszukiwania: silnych lub słabych – zwróconych przez zapytanie do całkowitej liczby reguł uzyskanych przez to zapytanie. Dzięki temu współczynnikowi będą preferowane te zapytania, które zwracają mniejszą liczbę reguł, ale reguł interesujących. Współczynnik ten przyjmuje wartości z przedziału <0, 1>, 0 – w przypadku, gdy zapytanie nie zwróciło reguł lub w wyniku nie było reguł znaczących, a 1, gdy wszystkie reguły pozyskane w wyniku zapytania były znaczące.

Ponieważ w wyniku zapytania może wystąpić duża liczba reguł spełniających zadany wzorzec kryterium, w kryterium oceny występuje omówiony wcześniej współczynnik oczekiwanej liczby reguł D_N, którego przedział wartości jest (0, 1>, przy czym wartość maksymalną uzyskujemy w przypadku, gdy liczba reguł zwróconych przez zapytanie jest mniejsza lub równa wskazanej przez użytkownika oczekiwanej liczbie reguł.

Rozdział 4

Implementacja

Dla potrzeb testowania metody automatycznego generowania zapytań, przedstawionej w rozdziale 2, została opracowana oraz zaimplementowana aplikacja komputerowa GAZdRA (Generowanie Automatyczne Zapytań do Reguł Asocjacyjnych), dzięki której możliwe było sprawdzenie działania tej metody na danych rzeczywistych.

4.1 Założenia projektowe

Podczas projektowania aplikacji przyjęto szereg założeń, które określały jakie funkcjonalności zostaną zaimplementowane oraz jakie narzędzia programistyczne zostaną do tego celu wykorzystane.

Założono, że głównym zastosowaniem tworzonego oprogramowania będzie pozyskiwanie wiedzy ze zbiorów danych z wykorzystaniem metody automatycznego generowania zapytań do modelu reguł asocjacyjnych. Główne zadania, jakie zostały postawione przed aplikacją są następujące:

– możliwość obsługi danych w formacie relacyjnym (każdy rekord bazy danych ma tę samą liczbę wartości) oraz transakcyjnym (rekordy o zmiennej liczbie wartości),

- obsługa baz danych zapisanych w tekstowym formacie CSV (ang. Comma Separated Values) obsługiwanym przez większość arkuszy kalkulacyjnych oraz aplikacji baz danych,
- dostarczanie łatwego w obsłudze interfejsu do definiowania kryterium subiektywnego,
- budowa modeli reguł asocjacyjnych z możliwością ograniczenia liczby reguł poprzez wskazania wartości progowych współczynników wsparcia oraz zaufania, lub też przybliżone określenie oczekiwanej liczby reguł,
- modele drążenia danych przechowywane w plikach tekstowych w formacie PMML,
- możliwość parametryzacji algorytmu programowania genetycznego oraz krokowe wykonywanie procesu ewolucyjnego, z możliwością wstecznego powrotu do dowolnego pokolenia,
- przedstawienie wyników działania algorytmu w postaci tabelarycznej oraz w formie wykresu.

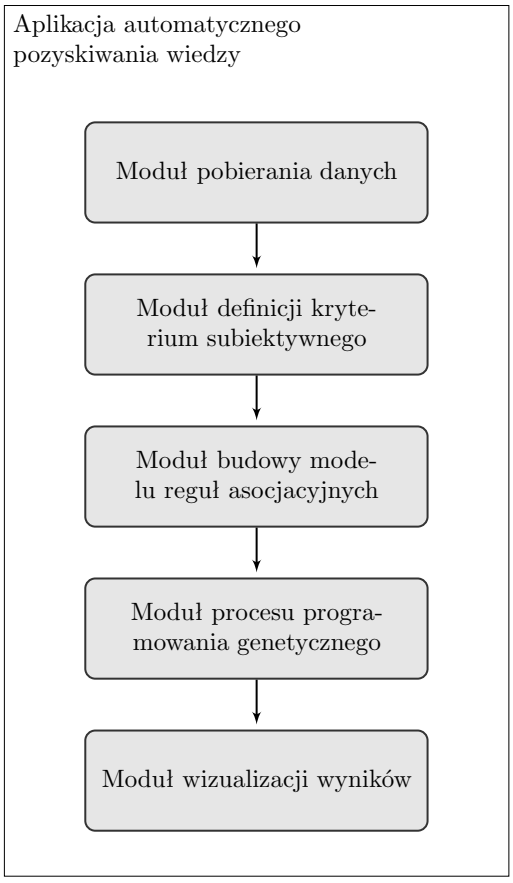
Aplikacja realizująca przedstawione założenia stworzona została z wykorzystaniem języka oprogramowania Java. Jest to język obiektowy, dla którego istnieje duży zbiór gotowych bibliotek programistycznych, ułatwiających tworzenie kodu aplikacji. Dzięki temu można w łatwy i szybki sposób tworzyć rozbudowane rozwiązania programistyczne. Kolejną, bardzo ważna zaletą języka Java jest jego niezależność od platformy systemowej, co umożliwia wykorzystanie przygotowanego oprogramowania w różnych środowiskach sprzętowych i systemowych. Java charakteryzuje się również łatwością tworzenia nawet bardzo skomplikowanych interfejsów, a co za tym idzie aplikacji przyjaznych w obsłudze dla użytkownika.

Przy tworzeniu oprogramowania w języku Java wykorzystano środowisko programistyczne NetBeans IDE w wersji 6.0. Jest to stale rozwijane

i udoskonalane, jedno z najpopularniejszych i najbardziej zaawansowanych narzędzi programistycznych dla Javy, udostępniane na zasadzie licencji open-source.

4.2 Struktura programu

Zrealizowana aplikacja składa się z pięciu modułów odpowiedzialnych za poszczególne funkcje wskazane w założeniach projektowych (podrozdział 4.1). Struktura współdziałania modułów została przedstawiona na rysunku 4.1.



Rysunek 4.1: Modułowa struktura programu

Poszczególne elementy graficznego interfejsu użytkownika dla modułów aplikacji zostały pokazane w dodatku B.

4.2.1 Moduł pobierania danych

Pierwszym składnikiem programu jest moduł pobierania danych źródłowych. Głównym zadaniem tego modułu jest przetwarzanie wskazanego przez użytkownika pliku w formacie CSV.

Format CSV charakteryzuje się tym, że dane w pliku tekstowym rozdzielone są przy pomocy separatora, natomiast poszczególne rekordy znakiem końca linii. Znakiem separatora jest zazwyczaj znak przecinka, choć mogą wystąpić również inne, takie jak średnik, tabulacja itp. Dodatkowo różne aplikacje mogą generować pliki CVS o różnych znakach separatora łańcuchów znakowych (np. cudzysłów, apostrof). Plik CVS może zawierać również nagłówek opisujący poszczególne kolumny danych. Dla potrzeb pobierania danych z plików CVS o różnych konfiguracjach, moduł wczytywania został wyposażony w interfejs umożliwiający parametryzację tego procesu, bez konieczności dodatkowego przetwarzania źródłowej bazy danych.

Ponadto, moduł wczytywania ma możliwość przetwarzania danych nie tylko w formacie relacyjnym, ale również transakcyjnym. Dzięki temu będzie możliwe w przyszłości zastosowanie proponowanej aplikacji w innych dziedzinach niż te, które zostały przedstawione w niniejszej pracy (np. do przetwarzania danych dotyczących sprzedaży w hipermarketach itp.).

W module wykorzystano darmową bibliotekę opencsv, która w prosty sposób umożliwia pobieranie danych z plików zapisanych w formacie CVS.

4.2.2 Moduł definicji kryterium subiektywnego

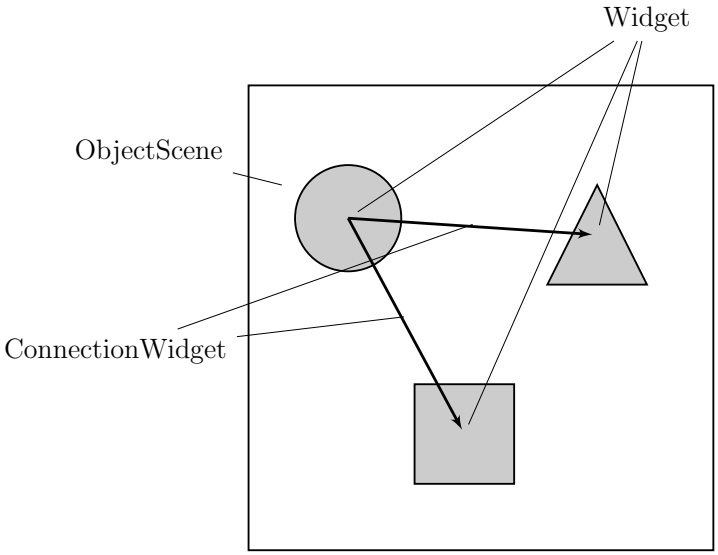
Moduł definicji kryterium subiektywnego jest kolejnym składnikiem aplikacji, który spełnia ważną rolę w procesie definiowania kryterium oceny, opisanego w podrozdziale 3.3.

Moduł ten implementuje graficzny interfejs, dzięki któremu użytkownik ma możliwość zdefiniowania własnego zbioru wag (zgodnie z definicją 6)

oraz przypisanie tym wagom reprezentacji graficznej (kolor, rodzaj i grubość linii).

Kolejny element interfejsu udostępnionego w module stwarza możliwość intuicyjnego (za pomocą myszki) definiowania wzorca kryterium w sposób analogiczny do pokazanego na rysunku 3.7. Użytkownik wybiera z menu podręcznego interesujące go atrybuty oraz odpowiednią wagę, która ma być im przypisana. W następnym kroku wskazywane jest, poprzez narysowanie strzałki, powiązanie, które zostaje dodane do tabeli powiązań. Istnieje możliwość śledzenia na bieżąco zmian w tabeli powiązań, dodawania oraz usuwania atrybutów lub powiązań.

W module wykorzystano funkcjonalności, jakie udostępnia biblioteka programistyczna Visual Library, która jest częścią aplikacji NetBeans i znajduje zastosowanie w wizualizacji grafów i schematów blokowych.



Rysunek 4.2: Podstawowe klasy biblioteki Visual Library

Biblioteka Visual Library definiuje trzy podstawowe klasy (rys. 4.2):

- ObjectScene – przechowuje wszystkie elementy sceny,
- Widget – element sceny,
- ConnectionWidget – połączenie pomiędzy elementami sceny.

Wymienione klasy, aby mogły być wykorzystane, musiały zostać specjalnie dostosowane dla potrzeb niniejszego projektu. Klasa Widget została rozszerzona o możliwość wizualizacji atrybutów oraz ich wartości, tak jak to zaproponowano w definicji wzorca kryterium na rysunku 3.7. Natomiast klasa ConnectionWidget została zmodyfikowana, tak aby mogła reprezentować powiązania wzorca kryterium, zarówno pomiędzy pojedynczymi, jak również wszystkimi wartościami atrybutów. Dodatkowo zaimplementowano algorytm pozwalający na zastosowanie różnej reprezentacji graficznej powiązań, w zależności od zdefiniowanych wartości wag, kolorów i rodzajów linii.

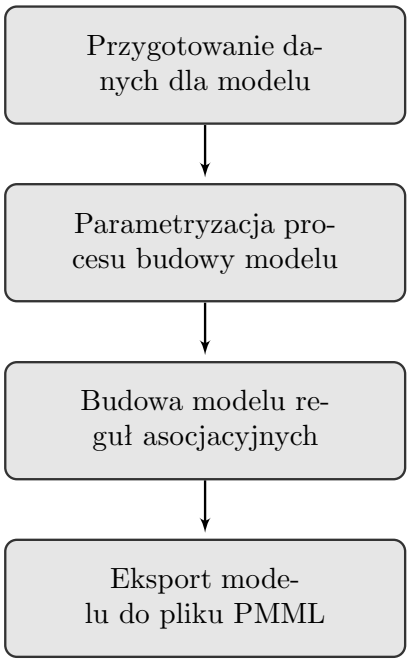
W programie wykorzystano również standardową bibliotekę komponentów Swing języka Java do tabelarycznego przedstawienia powiązań zdefiniowanych w kryterium oceny.

4.2.3 Moduł budowy modelu

W kolejnym etapie działania aplikacji uruchamiany jest moduł budowy modelu reguł asocjacyjnych. Poszczególne fazy działania modułu budowy modelu przedstawiono na rysunku 4.3.

Przygotowanie danych odbywa się na podstawie zdefiniowanego kryterium subiektywnego – w przypadku danych relacyjnych wybierane są tylko te atrybuty, które w tym kryterium zostały wskazane. Dla danych zapisanych w sposób transakcyjny, w którym nie można wyodrębnić poszczególnych atrybutów, źródłowym zbiorem danych dla modelu są wszystkie wartości bazy danych. Za prawidłowy przebieg tego etapu budowy modelu odpowiadają funkcje (napisane dla potrzeb projektu), które przygotowują tymczasowe pliki danych dla modelu, zgodnie z dokumentacją biblioteki Xelopes [86].

Parametryzacja modelu odbywa się poprzez wskazanie wartości progowych zaufania i wsparcia. Drugą możliwością, jaką daje biblioteka programistyczna Xelopes (wersja 1.3.1 była udostępniana bezpłatnie przez firmę



Rysunek 4.3: Etapy działania modułu budowy modelu

Prudsys AG.), wykorzystaną na potrzeby budowy modelu, jest ograniczenie wielkości modelu poprzez określenie przybliżenia, oczekiwanej liczby reguł, jaka powinna zostać wygenerowana [86]. Obie możliwości zostały udostępnione poprzez interfejs użytkownikom aplikacji.

Budowa modelu odbywa się z wykorzystaniem klasy *AssociationRulesBuild* biblioteki Xelopes, której funkcje zostały tak dostosowane, aby możliwa była zewnętrzna parametryzacja procesu budowy.

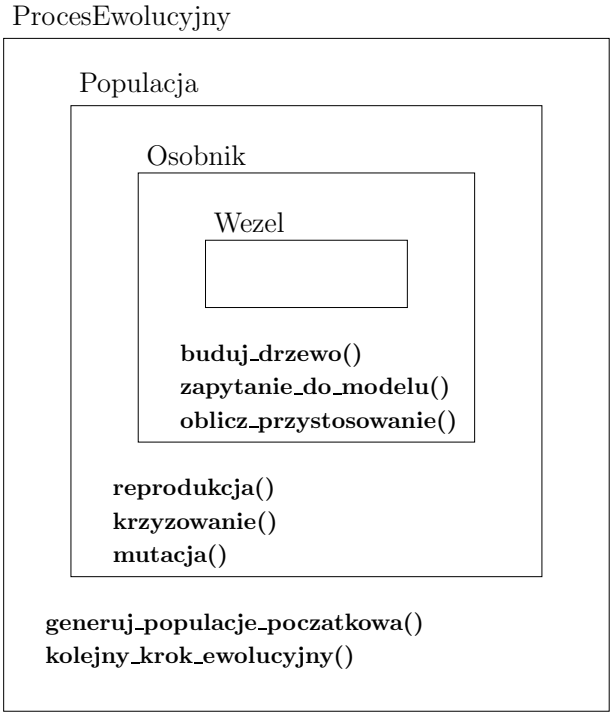
Proces budowy przebiega etapowo. W pierwszej kolejności następuje inicjowanie algorytmu Apriori zaimplementowanego w bibliotece Xelopes, poprzez wskazanie źródła danych dla modelu oraz parametrów podanych podczas procesu parametryzacji modelu. Następnie weryfikowana jest prawidłowość zmiennych parametrycznych oraz źródła danych i rozpoczyna się główny proces budowy modelu. W zależności od wybranej opcji funkcja budowy modelu analizuje minimalne wsparcie i zaufanie lub też próbuje dostosować te wartości, tak aby otrzymany model zawierał sugerowaną przez

użytkownika liczbę reguł. Podczas generowania modelu na bieżąco wyświetlane są informacje na temat przebiegu procesu oraz samego modelu m. in. zbiory pozycji, reguły asocjacyjne itp.

W ostatniej fazie działania modułu budowy modelu pozyskany model zapisywany jest w postaci zbioru plikowego w formacie PMML. W tym przypadku również wykorzystywana jest odpowiednia funkcja biblioteki Xelopes, która przetwarza wewnętrzny zapis modelu do postaci tekstowej, użytecznej do dalszego przetwarzania przy pomocy zapytań XQuery.

4.2.4 Moduł procesu programowania genetycznego

Kolejnym modułem, uznanym za kluczowy dla całego procesu automatycznego generowania zapytań, jest moduł procesu programowania genetycznego. W module tym zostały zaimplementowane wszystkie elementy potrzebne dla algorytmu programowania genetycznego (rysunek 2.2).



Rysunek 4.4: Struktura klas modułu procesu programowania genetycznego

Na rysunku 4.4 przedstawiono najważniejsze klasy oraz funkcje modułu programowania genetycznego.

Dla potrzeb reprezentacji osobnika w postaci drzewa (rys. 3.1) zdefiniowano klasę **Wezel**, której zadaniem jest przechowywanie poszczególnych węzłów drzewa, zarówno funkcji (operatorów logicznych), jak również terminatorów – wyrażeń, które pokazano na rysunku 3.3. Elementy klasy **Wezel** wykorzystywane są w klasie **Osobnik**, w której zaimplementowano szereg funkcji odpowiedzialnych za budowę osobnika oraz generowanie zapytań do modelu na podstawie struktury drzewiastej osobnika.

Kolejna klasa (**Populacja**) reprezentuje populację osobników procesu ewolucyjnego. W klasie tej zaprogramowano funkcje odpowiedzialne za poszczególne operacje genetyczne, których przegląd przedstawiono w podrozdziale 2.3.3 oraz funkcje do obliczania przystosowania zapytania zgodnie z zdefiniowanym kryterium oceny. Najwyżej w poziomie hierarchii klas znajduje się klasa **ProcesEwolucyjny**, która kontroluje przebieg kolejnych kroków procesu programowania genetycznego. Do jej zadań należą m. in. wygenerowanie populacji początkowej oraz wykonywanie kolejnych kroków ewolucyjnych. Na rysunkach B.4 i B.5 w dodatku B przedstawiono diagramy klas dla wymienionych elementów modułu procesu programowania genetycznego.

Dodatkowo, obok zdefiniowania wymienionych klas, moduł został wyposażony w interfejs, umożliwiając pełną parametryzację procesu programowania genetycznego (tabela 2.1). Dzięki temu możliwe jest kontrolowanie przebiegu procesu oraz dostosowanie jego działania do potrzeb analizy zróżnicowanych modeli reguł asocjacyjnych.

Zewnętrznym oprogramowaniem wykorzystanym do zadawania zapytań XQuery do modelu jest darmowa biblioteka Qexo (Query Expressions for XML Objects). Pozostałe elementy modułu zostały napisane od podstaw, z powodu braku odpowiednich gotowych rozwiązań programistycznych.

4.2.5 Moduł wizualizacji wyników

Ostani moduł programu odpowiedzialny jest za wizualizację oraz archiwizację wyników eksperymentów analitycznych.

Przedstawianie wyników odbywa się w postaci tabelarycznej, jako lista wszystkich osobników populacji z ich przystosowaniem, liczbą reguł znaczących oraz liczba wszystkich reguł zwracanych przez zapytanie. Dostępny jest również podgląd reguł zwracanych przez poszczególne osobniki (zapytania) populacji, dzięki czemu możliwe jest ich porównanie i analiza.

Kolejnym elementem wizualizacji wyników jest wykres przedstawiający przebieg procesu genetycznego – kształtowanie się przystosowania najlepszego osobnika oraz średniego przystosowania wszystkich osobników populacji w poszczególnych krokach ewolucyjnych. Do realizacji wykresu została wykorzystana biblioteka JFreeChart (oprogramowanie open-source).

Ostatnim zadaniem tego modułu jest archiwizacja rezultatów analiz, tak aby mogły być one poddawane późniejszemu przetwarzaniu dla celów prezentacji i porównania.

Rozdział 5

Badania eksperymentalne

W celu sprawdzenia skuteczności opracowanej i przedstawionej w rozdziałach 2 i 3 metody automatycznego generowania zapytań przeprowadzono serię badań eksperymentalnych, wykorzystując aplikację opisaną w rozdziale 4. Do budowy modeli drążenia koniecznym było pozyskanie danych rzeczywistych, które następnie poddawane były analizie. Uzyskanie interesujących danych rzeczywistych nie jest zadaniem łatwym, gdyż w większości przypadków dane tego typu są chronione tajemnicą firm i instytucji, które niechętnie udostępniają swoje zasoby do badań.

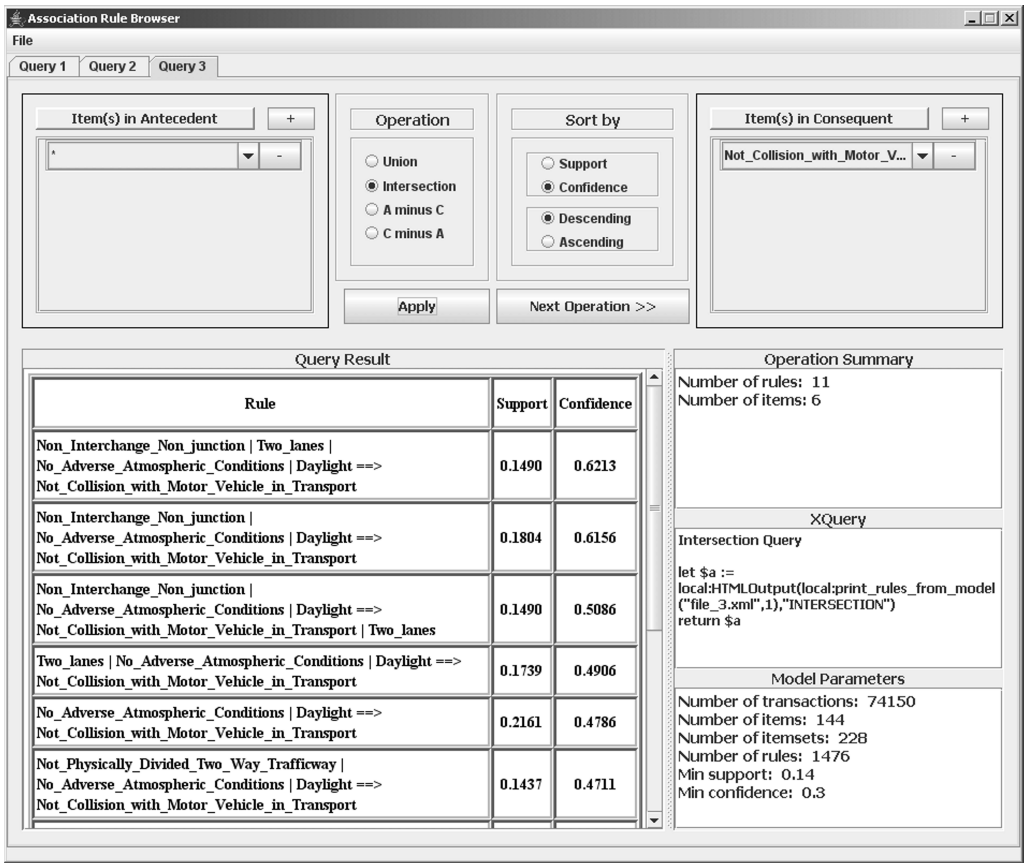
W danym przypadku, udało się pozyskać dane z trzech źródeł:

- dane o zużyciu energii elektrycznej, udostępnione przez Zakład Energetyczny,
- dane medyczne z badań alergologicznych, przeprowadzonych przez Zakład Alergologii Szpitala Uniwersyteckiego w Krakowie,
- dane o wypadkach samochodowych w Stanach Zjednoczonych (ogólnie dostępne w sieci internet).

Pełny opis powyższych danych znajduje się w dodatku A.

Kolejnym krokiem w procesie badawczym było odpowiednie przygotowanie danych do analizy, co wymagało napisania odpowiednich skryptów

transformacji danych. Po wykonaniu eksperymentów numerycznych, uzyskane wyniki porównano z rezultatami „ręcznej" analizy, dokonanej przy pomocy zaproponowanej w [84] przeglądarki reguł asocjacyjnych (Rys. 5.1).



Rysunek 5.1: Przeglądarka reguł asocjacyjnych

Przeglądarka reguł asocjacyjnych wykorzystuje język zapytań XQuery, na który tłumaczone są zadawane przez użytkownika kryteria wyboru reguł z modelu. Możliwe jest również krokowe „odpytywanie" modelu, w którym podzbiór reguł uzyskany w zapytaniu staje się podstawą dla kolejnych zapytań. Możliwy jest również powrót do wcześniejszych kroków analizy i rozpoczęcie zadawania zapytań od dowolnego punktu procesu wyszukiwania.

5.1 Przygotowanie danych do analizy

Aby była możliwa budowa modeli reguł asocjacyjnych, przed przystąpieniem do eksperymentów obliczeniowych pozyskane dane wymagały odpowiedniego przygotowania. W tym celu wykonano szereg operacji, takich jak:

- ujednolicenie wartości atrybutów,
- dyskretyzacja wartości numerycznych,
- zamiana wartości atrybutów z postaci kodu na wartości typu opisowego,
- zachowanie jednoznaczności wartości atrybutów.

Problem niejednolitych wartości atrybutów wystąpił w szczególności w danych medycznych, ponieważ dane ankietowe oraz wyniki badań były wprowadzane do bazy „ręcznie" w różnych ośrodkach alergologicznych miasta Krakowa. Lepiej pod tym względem wypadły dane o wypadkach zbierane poprzez wypełnianie formularzy oraz dane o zużyciu energii, które są pobierane w sposób automatyczny.

Dyskretyzacji wymagały dane energetyczne, ponieważ wartości atrybutów zużycia energii oraz temperatury i nasłonecznienia były w postaci numerycznej.

Konieczność zamiany wartości atrybutów z postaci kodu na wartość typu opisowego wystąpiła w sposób szczególny w danych o wypadkach samochodowych, ponieważ wszystkie wartości w bazie były zakodowane, a opis kodowania był dostępny w specjalnie dołączonym pliku danych.

Problem jednoznaczności wartości atrybutów wystąpił w przypadku danych medycznych oraz zużycia energii.

Dla zautomatyzowania procesu obróbki danych zostały przygotowane skrypty w języku Python, które w znacznym stopniu usprawniły przeprowadzenie tego etapu badań eksperymentalnych.

5.2 Eksperymenty dotyczące pozyskiwania wiedzy z danych rzeczywistych

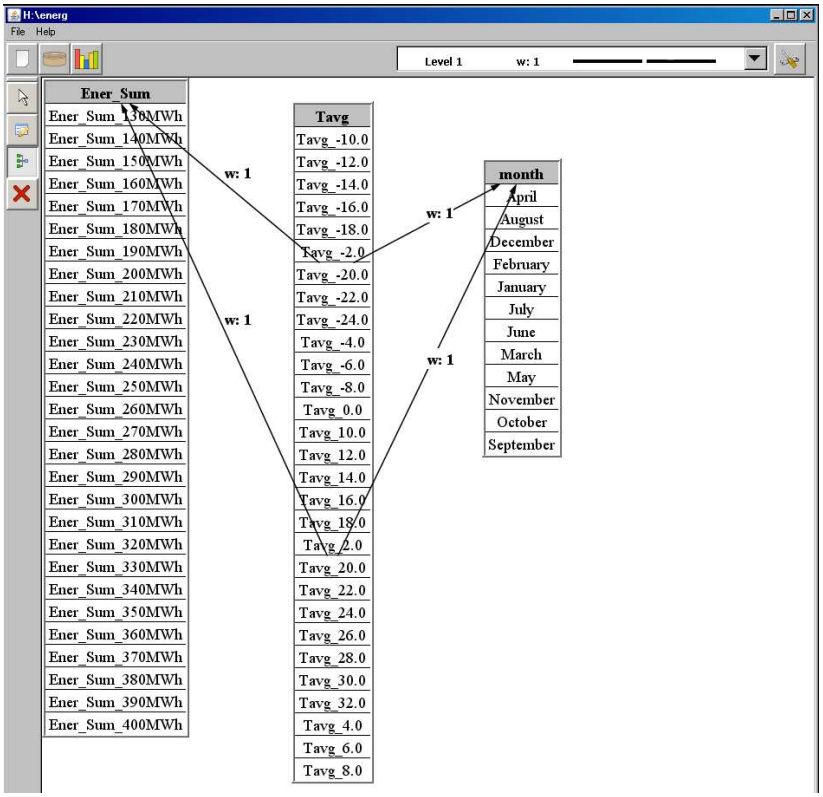
5.2.1 Dane energetyczne

Problem I [45]

Poszukiwana jest odpowiedź na pytanie:

Jak kształtuje się zużycie energii w Zakładzie przy średniej temperaturze 20℃ lub -20℃, oraz w jakich miesiącach te temperatury występują?

Na rysunku 5.2 przedstawiono kryterium subiektywne zdefiniowane dla potrzeb tego problemu.



Rysunek 5.2: Kryterium subiektywne – danych energetycznych – problem I

Kryterium subiektywne zostało określone poprzez wybranie z źródłowej bazy 3 atrybutów: *Ener_sum* (zużycie energii dla całego Zakładu Energetycznego), *Tavg* (średnia temperatura zmierzona w regionie) oraz *month*

(miesiąc w którym dokonano pomiaru). Następnie wskazano powiązania pomiędzy dwiema wartościami temperatury: -20℃ oraz 20℃ a wszystkimi wartościami atrybutów *Ener_sum* i *month*. Powiązania te miały przypisaną wagę o wartości 1. W rezultacie otrzymano wzorzec kryterium składający się z szeregu powiązań łączących wskazane temperatury ze zużyciem energii oraz miesiącami.

Kolejnym krokiem było zbudowanie modelu reguł asocjacyjnych, którego parametry przedstawiono w tabeli 5.1.

Tabela 5.1: Parametry modelu – dane energetyczne – problem I

Parametr	**Wartość**
Liczba rekordów	26301
Liczba pozycji	69
Liczba zbiorów częstych	430
Liczba reguł	752
Min wsparcie	0.003125
Min zaufanie	0

Zbudowany model został zapisany w standardzie PMML, a następnie poddany procesowi przeszukiwania przy pomocy algorytmu automatycznego generowania zapytań. Parametry procesu programowania genetycznego przedstawiono w tabeli 5.2.

W pierwszej kolejności zostało utworzone 2000 osobników, z których wybrano 1000 najlepszych jako populację początkową. Po przejściu 20 cykli procesu ewolucji został znaleziony osobnik reprezentujący zapytanie pokazane w treści algorytmu 5.1, którego rezultatem jest podzbiór reguł przedstawionych w tabeli 5.3.

```
(not((@consequent>426) and (@consequent<=172)))
and ((@confidence>=0.1343) and (@antecedent=367))
```

Algorytm 5.1: Najlepsze zapytanie dla danych energetycznych – problem I

Tabela 5.2: Parametry procesu programowania genetycznego – dane energetyczne – problem I

Parametr	**Wartość**
Wielkość populacji programów	1000
Minimalna głębokość drzewa	1
Maksymalna głębokość drzewa	5
Limit pokoleń	20
Dziesiątkowanie	2000 os.
Reprodukcja	5%
Mutacja	co 5 pokoleń
Wybijanie	co 5 pokoleń – 20% populacji
Liczba reguł oczekiwanych	30
Najlepszy osobnik do następnej populacji	Tak

Tabela 5.3: Reguły uzyskane dla danych energetycznych – problem I

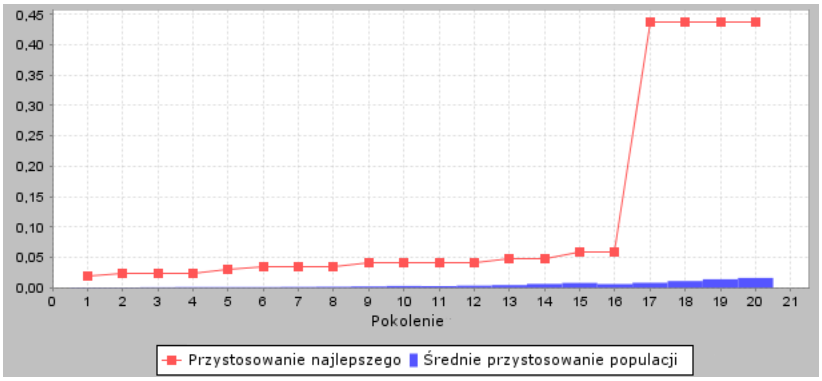
Uzyskane reguły	**Wsparcie**	**Zaufanie**	$\mathbf{L_z/L_r}$
(Tavg_20.0) → (August)	0.0078	0.2894	6/6
(Tavg_20.0) → (July)	0.0072	0.2681	
(Tavg_20.0) → (June)	0.0066	0.2454	
(Tavg_20.0) → (Ener_Sum_230MWh)	0.0040	0.1504	
(Tavg_20.0) → (Ener_Sum_240MWh	0.0053	0.1986	
(Tavg_20.0) → (Ener_Sum_250MWh)	0.0057	0.2199	

Najlepsze zapytanie zwróciło 6 reguł (L_r), z których wszystkie były znaczące (L_z). Analizując uzyskany wynik można stwierdzić, że średnia temperatura 20°C występuje najczęściej w sierpniu oraz zużycie energii waha się w przedziale 230 MWh do 250 MWh, przy czym najwyższa wartość występuje najczęściej. Jak można zaobserwować, w podzbiorze tym nie ma reguł, w których wystąpiłaby średnia wartość temperatury -20°C, co należy zinterpretować, że występuje ona zbyt rzadko, aby pojawić się w modelu.

Dla porównania uzyskanego wyniku przeprowadzono wyszukiwanie reguł za pomocą Przeglądarki reguł asocjacyjnych. W tym przypadku uzyskano wynik zbliżony do tego, jaki został wygenerowany w sposób automatyczny, z tą różnicą, że w podzbiorze reguł znalazła się jeszcze jedna dodatkowa reguła:

$$\text{Tavg\_20.0} \rightarrow \text{September} \quad \text{wsparcie}=0.0034 \quad \text{zaufanie}=0.1276$$

Przebieg procesu ewolucyjnego ilustruje rysunek 5.3.



Rysunek 5.3: Przebieg procesu ewolucyjnego, dane energetyczne – problem I (oś rzędnych określa wartość funkcji przystosowania)

Analizując wykres z rys. 5.3 można zauważyć stały trend wzrostowy wartości funkcji przystosowania najlepszego osobnika w kolejnych pokoleniach, jak również wzrost średniej wartości przystosowania wszystkich osobników populacji.

```
not(@confidence<=0.2186)
```

Algorytm 5.2: Osobnik w I kroku ewolucyjnym dla danych energetycznych – problem I

Dla porównania, w pierwszym kroku ewolucyjnym zapytanie, które było najlepszym osobnikiem populacji (algorytm 5.2), było mało rozbudowane i zwracało 42 reguły, z których tylko 4 były znaczące. Wartość funkcji przystosowania tego osobnika wynosiła w przybliżeniu 0.02, podczas gdy

wartość, dla najlepszego osobnika, pozyskanego w 17 kroku ewolucyjnym, była równa 0.44. Wskazuje to, że algorytm poprzez operacje ewolucyjne generuje osobniki (zapytania) coraz lepsze z punktu widzenia postawionego kryterium, co powinno doprowadzić do najkorzystniejszego (bliskiego optymalnemu) rozwiązania.

W celu potwierdzenia powtarzalności uzyskanych wyników przeprowadzono kolejne eksperymenty dla parametrów identycznych z podanymi w tabeli 5.2 oraz kryterium z rysunku 5.2 (ze strony 85). Uzyskane wyniki przedstawiono w dodatku C, w tabeli C.1. Kolejne próby wykazały, podobnie jak miało to miejsce dla pierwszego eksperymentu, że proces programowania genetycznego znajduje zapytania, które zwracają identyczny lub prawie identyczny podzbiór reguł. Można również stwierdzić, że w każdym przypadku najlepsze zapytanie jest inne, a struktury wyrażeń reprezentujących zapytanie w niektórych przypadkach mogą być dość złożone. Oczywistym jest, że te złożone wyrażenia mogłyby zostać uproszczone poprzez zastosowanie powszechnie znanych wzorów matematycznych i logicznych, co prawdopodobnie zwiększyłoby szybkość przetwarzania zapytań. Z drugiej jednak strony redukcja wyrażeń wiązałaby się z utratą „materiału genetycznego", który w połączeniu z fragmentami innych osobników pozwalałby uzyskać lepszy wynik na dalszym etapie procesu ewolucji.

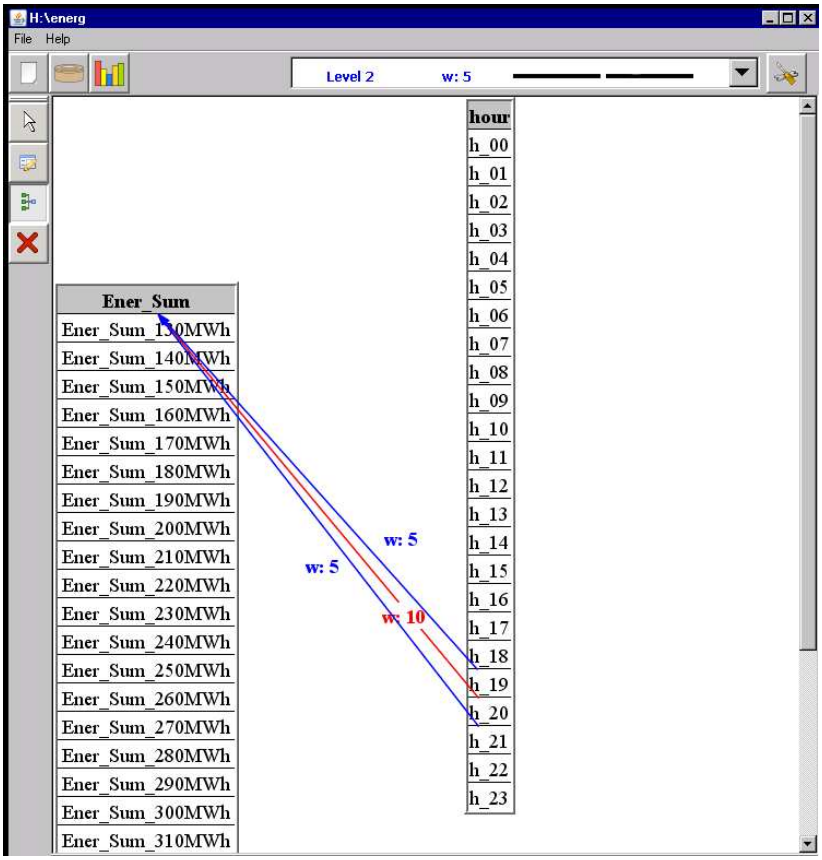
Ponadto można zauważyć, że najlepszy wynik może zostać uzyskany w dowolnej iteracji procesu ewolucyjnego (zarówno w pierwszej, jak ma to miejsce w przypadku eksperymentu nr 5, jak i w ostatniej z przewidzianych 20 dla eksperymentu nr 2).

Problem II

Poszukiwana jest odpowiedź na pytanie:

Jak kształtuje się zużycie energii w Zakładzie Energetycznym ok. godziny 20?

Na rysunku 5.4 przedstawiono kryterium subiektywne zdefiniowane dla potrzeb tego problemu.



Rysunek 5.4: Kryterium subiektywne dla danych energetycznych – problem II

W kryterium tym wskazano 2 atrybuty z bazy danych: *Ener_Sum* (zużycie energii w Zakładzie) oraz *hour* (godzina wykonania pomiaru). Następnie zdefiniowano 2 wagi o wartościach odpowiednio 10 i 5. Powiązanie o wadze 10 zostało określone dla godziny 20 i wszystkich wartości atrybutu *Ener_Sum*, a powiązanie o wadze 5 połączyło godzinę 19 i 21 również z atrybutem *Ener_Sum*.

Ponieważ atrybuty w kryterium subiektywnym zmieniły się w porównaniu do tych z problemu I, koniecznym było zbudowanie kolejnego modelu drążenia. Parametry tego modelu pokazano w tabeli 5.4.

Tabela 5.4: Parametry modelu dla danych energetycznych – problem II

Parametr	**Wartość**
Liczba rekordów	26301
Liczba pozycji	52
Liczba zbiorów częstych	308
Liczba reguł	522
Min wsparcie	0.000781
Min zaufanie	0

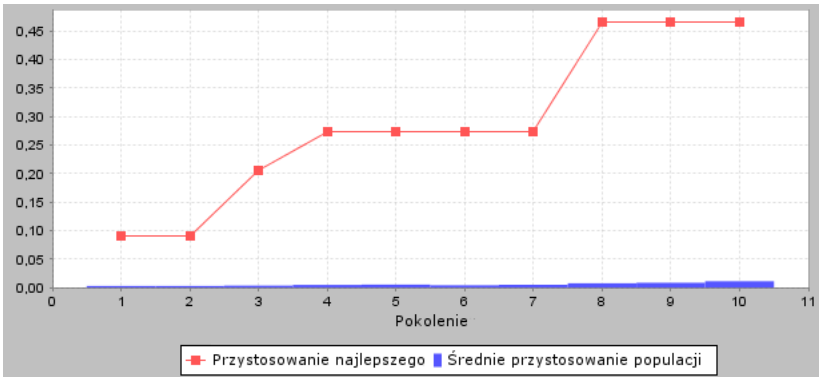
Parametry procesu programowania genetycznego były takie same jak dla problemu I (tab. 5.2) za wyjątkiem limitu pokoleń, który w tym przypadku został ustawiony na 10.

W wyniku działania algorytmu automatycznego wyszukiwania otrzymano zapytanie, które zwróciło 34 reguły (został przekroczony zakres liczby reguł oczekiwanych). W tym przypadku również przeprowadzono badanie wyniku działania algorytmu automatycznego z wynikiem otrzymanym przy pomocy Przeglądarki reguł asocjacyjnych (rys. 5.1) i wykazało ono, że uzyskuje się dokładnie taką samą liczbę reguł ze wskazanymi w kryterium subiektywnym następnikami i poprzednikami. Tabela 5.5 przedstawia 3 reguły najlepsze pod względem wartości współczynników wsparcia i zaufania wybrane z podzbioru reguł zwróconych przez zapytanie. Można stwierdzić, że najwyższe zużycie energii elektrycznej występuje ok. godziny 19, a następnie w kolejnych godzinach stopniowo maleje.

Tabela 5.5: Reguły uzyskane dla danych energetycznych – problem II

Uzyskane reguły	**Wsparcie**	**Zaufanie**	$\mathbf{L}_z/\mathbf{L}_r$
(h_19) → (Ener_Sum_320MWh)	0.0053	0.1259	34/34
(h_20) → (Ener_Sum_290MWh)	0.0066	0.1670	
(h_21) → (Ener_Sum_270MWh)	0.0108	0.2591	

Przebieg procesu programowania genetycznego przedstawiono na wykresie 5.5.



Rysunek 5.5: Przebieg procesu ewolucyjnego dla danych energetycznych – problem II

5.2.2 Dane medyczne

W ramach tej grupy badań przeprowadzono analizę danych medycznych, z udziałem pracowników Zakładu Alergologii Collegium Medicum Uniwersytetu Jagiellońskiego w Krakowie. Przeprowadzone eksperymenty dotyczyły problemu analizy skuteczności przygotowanych pytań ankietowych, dla potrzeb wstępnej diagnostyki uczuleń alergicznych [44].

Badanie ankietowe przeprowadzane w szkołach daje podstawowe informacje na temat symptomów, które mogą wskazywać na możliwość występowania choroby alergicznej u badanego dziecka oraz w jego najbliższej rodzinie. Na podstawie odpowiedzi na pytania ankietowe, lekarze alergolodzy określają tzw. rozpoznanie wstępne choroby alergicznej, które to w przypadku pozytywnego wyniku jest następnie potwierdzane przy pomocy testów medycznych. Rozpoznanie ostateczne określa końcową diagnozę stawianą po przeprowadzeniu badań w poradni alergologicznej.

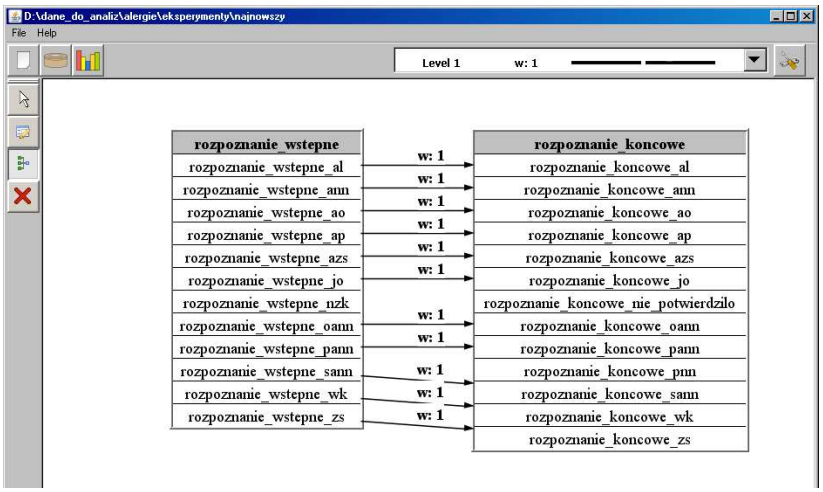
Przygotowanie ankiety dla potrzeb badania wstępnego nie jest zadaniem prostym, gdyż pytania ankietowe muszą być na tyle łatwo zrozumiałe dla

ankietowanych, aby umożliwiały udzielenie na nie jednoznacznej odpowiedzi, a zarazem muszą dawać podstawę do postawienia wstępnej diagnozy. Dlatego też ważna jest ocena wyników uzyskanych przy pomocy ankiet poprzez porównanie opartego na nich rozpoznania wstępnego z rozpoznaniem ostatecznym.

Dla potrzeb analizy przeprowadzono eksperymenty w odniesieniu do dwóch sytuacji:

– rozpoznanie wstępne choroby alergicznej było takie samo jak rozpoznanie końcowe,
– rozpoznanie końcowe nie potwierdziło choroby alergicznej.

Na rysunku 5.6 przedstawiono kryterium zdefiniowane dla pierwszego przypadku, w którym wskazano powiązania pomiędzy rozpoznaniem wstępnym oraz rozpoznaniem końcowym, przy czym stwierdzona wstępnie choroba była taka sama, jak ta, która została zdiagnozowana ostatecznie na podstawie badań medycznych.



Rysunek 5.6: Kryterium: rozpoznanie wstępne=rozpoznanie końcowe

W rezultacie w skład kryterium subiektywnego weszło 11 powiązań z przypisaną im wagą 1. Po zdefiniowaniu kryterium został zbudowany model reguł, którego parametry przedstawiono w tabeli 5.6.

Tabela 5.6: Parametry modelu – dane medyczne

Parametr	**Wartość**
Liczba rekordów	1114
Liczba pozycji	25
Liczba zbiorów częstych	276
Liczba reguł	1806
Min wsparcie	0.0015625
Min zaufanie	0

Jak można zauważyć, liczba reguł modelu w porównaniu z eksperymentami opisanymi w podrozdziale 5.2.1 została zwiększona do 1806, co powinno utrudnić wyszukiwanie zapytania zwracającego na tyle mały podzbiór reguł spełniających kryterium, aby był łatwy do analizy.

W tym przypadku zdecydowano się również na sprawdzenie, jaki wpływ na działanie procesu automatycznego pozyskiwania wiedzy ma wielkość populacji osobników. Związane jest to bezpośrednio z wydajnością systemu oraz zapotrzebowaniem na pamięć operacyjną sprzętu komputerowego. Duża liczebnie populacja znacznie spowalnia proces programowania genetycznego, z drugiej jednak strony duża liczba osobników powinna zwiększać prawdopodobieństwo szybszego (pod względem liczby kroków procesu ewolucji) znalezienia najlepszego rozwiązania.

Przeprowadzono badanie na trzech wielkościach populacji: 100, 500 i 1000 osobników. Pozostałe parametry algorytmu programowania genetycznego były identyczne z tymi z tabeli 5.2. W ramach obliczeń wykonano po 10 prób dla każdej liczby populacji, a następnie porównano średnią wartość kryterium oceny (tabela 5.7).

Porównując wartość średnią przystosowania do wielkości populacji można stwierdzić, że najlepsze rezultaty osiągnięto przy populacji 1000 osobników, natomiast najgorszy wynik dla populacji o wielkości 100. Jest to

Tabela 5.7: Wpływ wielkości populacji na średnią wartość przystosowania osobników

Wielkość populacji	**Średnia wartość kryterium oceny**
100	0,1269
500	0,1817
1000	0,1927

rezultat oczywisty. Natomiast, dość zaskakująco, uzyskane wyniki dla populacji 500 osobników niewiele odbiegają od tych, które otrzymano dla 1000 osobników, co może wskazywać, że zbiór 500 osobników może być zupełnie wystarczający dla poszukiwania rozwiązania danej klasy problemów. Ponadto, czas przetwarzania mniejszej populacji jest znacznie krótszy.

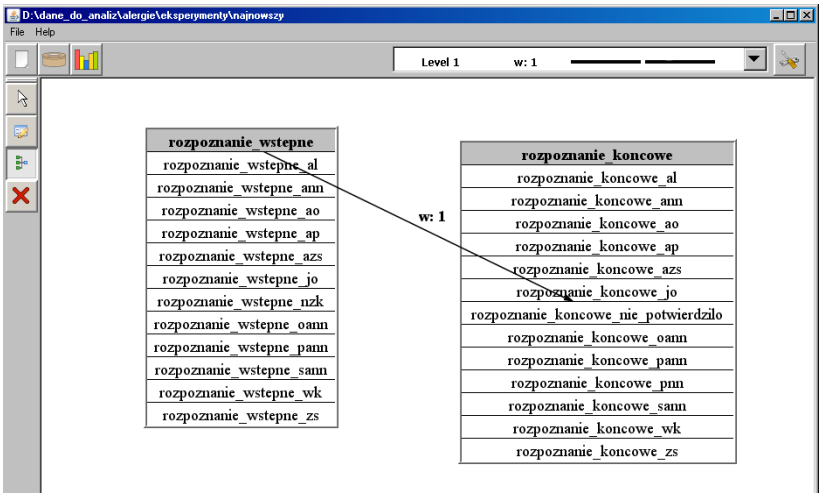
W wyniku przeprowadzonych eksperymentów najlepszy uzyskany rezultat (o najwyższej wartości kryterium oceny) zwracał 31 reguł, z których wszystkie były znaczące. W tabeli 5.8 pokazano 5 reguł o najwyższym współczynniku wsparcia.

Jak można zauważyć, najczęściej występującym rozpoznaniem wstępnym jest całoroczny atopowy nieżyt nosa (pann), okresowy atopowy nieżyt nosa (oann) oraz atopowe zapalenie skóry (azs). We wszystkich przypadkach przynajmniej jedno z rozpoznań wstępnych zostało potwierdzone przez rozpoznanie ostateczne, choć w uzyskanych regułach rozpoznanie końcowe pojawia się w poprzednikach reguły, co nie było zamierzone podczas definicji kryterium subiektywnego. Prawdopodobnie w tych przypadkach przeważyła wartość kryterium obiektywnego, ponieważ wartości współczynników zaufania dla tych reguł są bardzo duże. Można również zauważyć, że w pojedynczych regułach występują różne typy rozpoznania końcowego – co spowodowane jest tym, że w części przypadków stwierdzono więcej niż jedną chorobę alergiczną.

Drugi eksperyment został wykonany dla tego samego modelu, ale dla innego kryterium subiektywnego (rys. 5.7).

Tabela 5.8: Reguły uzyskane dla danych medycznych – eksperyment I

Uzyskane reguły	**Wsparcie**	**Zaufanie**	**L_z/L_r**
(rozpoznanie_wstepne_oann) i (rozpoznanie_wstepne_pann) i (rozpoznanie_koncowe_pann) → (rozpoznanie_koncowe_oann)	0,0179533	0,8333333	31/31
(rozpoznanie_wstepne_oann) i (rozpoznanie_wstepne_pann) i (rozpoznanie_koncowe_oann) → (rozpoznanie_koncowe_pann)	0,0179533	0,8333333	
(rozpoznanie_wstepne_azs) i (rozpoznanie_wstepne_pann) i (rozpoznanie_koncowe_oann) → (rozpoznanie_koncowe_pann)	0,0152603	0,68	
(rozpoznanie_wstepne_pann) i (rozpoznanie_koncowe_azs) → (rozpoznanie_koncowe_pann)	0,0134650	0,8333333	
(rozpoznanie_wstepne_azs) i (rozpoznanie_wstepne_pann) i (rozpoznanie_koncowe_azs) → (rozpoznanie_koncowe_pann)	0,0134650	0,8333333	



Rysunek 5.7: Kryterium: rozpoznanie wstępne nie zostało potwierdzone

W tym przypadku jako poprzedniki powiązań wystąpiły wszystkie wartości atrybutu rozpoznania wstępnego, a jako poprzednik powiązań wystąpiła wartość *rozpoznanie_koncowe_nie_potwierdzilo*. Parametry wyszukiwania były takie same jak w przypadku pokazanym w tabeli 5.1. W wyniku działania algorytmu wyszukującego znaleziono najlepsze zapytanie z punktu widzenia kryterium, które zwróciło 24 reguły, z których wszystkie były regułami znaczącymi. W tabeli 5.9 pokazano 3 reguły o najwyższym współczynniku wsparcia.

Tabela 5.9: Reguły uzyskane dla danych medycznych – eksperyment II

Uzyskane reguły	**Wsparcie**	**Zaufanie**	$\mathbf{L_z/L_r}$
(rozpoznanie_wstepne_azs) → (rozpoznanie_koncowe_nie_ potwierdzilo)	0,1912028	0,6493902	24/24
(rozpoznanie_wstepne_oann) → (rozpoznanie_koncowe_nie_potwierdzilo)	0,1310593	0,6008230	
(rozpoznanie_wstepne_ap) → (rozpoznanie_koncowe_nie_potwierdzilo)	0,0610413	0,8095238	

Analizując te wyniki można stwierdzić, że najczęściej błędnie diagnozowane są atopowe zapalenie skóry (azs), okresowy atopowy nieżyt nosa (oann) oraz alergia pokarmowa (ap).

Jak można zauważyć na podstawie parametru wsparcia reguł, w których nie potwierdzono wstępnej diagnozy, udział nieprawidłowo zdiagnozowanych przypadków jest dość znaczący. Dlatego też konieczne jest przeanalizowanie pytań ankietowych, aby w przyszłości podjąć próbę poprawy skuteczności rozpoznania wstępnego.

5.2.3 Dane o wypadkach samochodowych

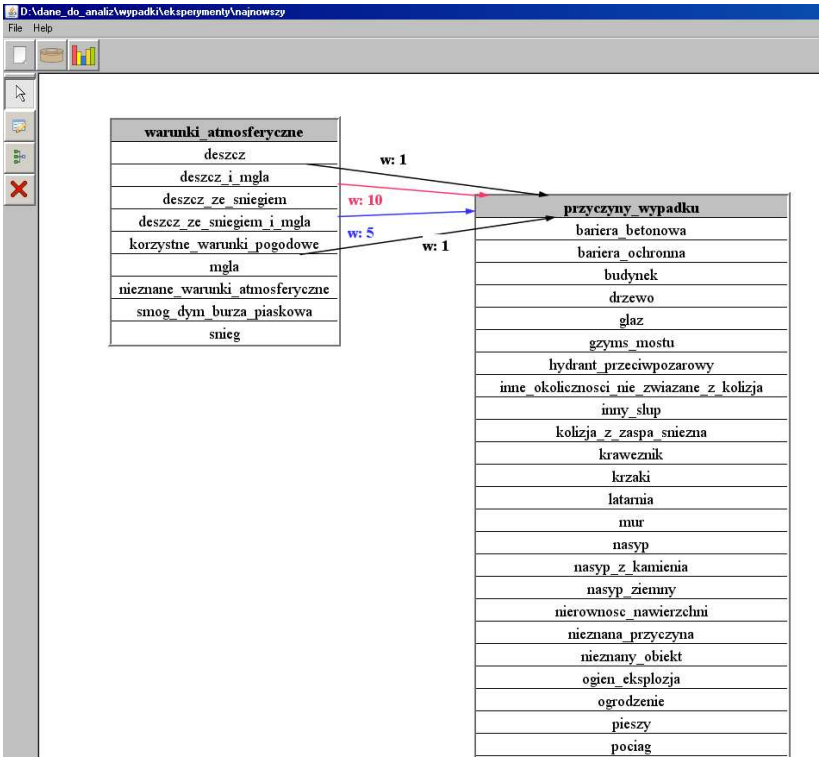
Kolejny eksperyment dotyczący analizy danych za pomocą metody automatycznego generowania zapytań przeprowadzono dla danych o wypadkach samochodowych, których opis znajduje się w dodatku A.3.

Problem

Znalezienie odpowiedzi na pytanie:

Co było najczęstszą przyczyną wypadków drogowych podczas deszczu i mgły?

Na rysunku 5.8 przedstawiono kryterium subiektywne zdefiniowane dla tego problemu.



Rysunek 5.8: Kryterium subiektywne dla danych o wypadkach

W kryterium wskazano powiązania o różnych wagach:

- wartość atrybutu warunki_pogodowe: *deszcz_i_mgla* ze wszystkimi wartościami atrybutu przyczyny_wypadku w następniku, powiązania z wagą 10
- wartość atrybutu warunki_pogodowe: *deszcz_ze_sniegiem_i_mgla* ze wszystkimi wartościami atrybutu przyczyny_wypadku w następniku, powiązania z wagą 5

– wartość atrybutu warunki_pogodowe: *deszcz* oraz oddzielnie *mgla* ze wszystkimi wartościami atrybutu przyczyny_wypadku w następniku, powiązania z wagą 1.

Na podstawie tego kryterium został zbudowany model, którego parametry przedstawiono w tabeli 5.10.

Tabela 5.10: Parametry modelu – dane o wypadkach samochodowych

Parametr	**Wartość**
Liczba rekordów	74150
Liczba pozycji	57
Liczba zbiorów częstych	330
Liczba reguł	546
Min wsparcie	0,0000122
Min zaufanie	0

W kolejnym kroku analizy dokonano sprawdzenia działania algorytmu automatycznego generowania zapytań przy ograniczeniu oczekiwanej liczby reguł do 10. Pozostałe parametry eksperymentu były takie same jak w tabeli 5.2.

W rezultacie uzyskano najlepsze zapytanie (algorytm 5.3), które zwróciło 13 reguł przedstawionych w tabeli 5.11.

```
(((@antecedent=323) and (not((@consequent<=196)
and (@consequent<82)))) and
(not(((@support=0.2172) or (@consequent<61)) and
(@antecedent!=81)))) or ((((@consequent<=134)
or (@consequent=282)) or ((@consequent!=100) and
(@consequent<=80))) and (@confidence>=0.9847))
```

Algorytm 5.3: Postać najlepszego osobnika dla danych o wypadkach samochodowych

Liczba reguł zwróconych przez zapytanie przekroczyła zalecaną w parametrach poszukiwań wartość, lecz przekroczenie to było nieznaczne i wszystkie pozyskane reguły były powiązane ze wskazanym kryterium subiektywnym.

Tabela 5.11: Reguły uzyskane – dane o wypadkach samochodowych

Uzyskane reguły	**Wsparcie**	**Zaufanie**	$\mathbf{L_z/L_r}$
(deszcz_i_mgla) → (poruszajacy_sie_pojazd)	0,0005529	0,3904762	13/13
(deszcz_i_mgla) → (pieszy)	0,0002562	0,1809524	
(deszcz_i_mgla) → (wywrotka)	0,0000809	0,0571429	
(deszcz_i_mgla) → (slup_reklamowy)	0,0000674	0,0476190	
(deszcz_i_mgla) → (nasyp)	0,0000270	0,0190476	
(deszcz_i_mgla) → (nasyp_z_kamienia)	0,0000270	0,0190476	
(deszcz_i_mgla) → (pojazd_na_przeciwnej_jezdni)	0,0000270	0,0190476	
(deszcz_i_mgla) → (nasyp_ziemny)	0,0000135	0,0095238	
(deszcz_i_mgla) → (ogrodzenie)	0,0000135	0,0095238	
(deszcz_i_mgla) → (pojazd_zaparkowany)	0,0000135	0,0095238	
(deszcz_i_mgla) → (przyczyna_inny_nie_zmotoryzowany)	0,0000135	0,0095238	
(deszcz_i_mgla) → (slup_znaku_drogowego)	0,0000135	0,0095238	
(deszcz_i_mgla) → (staly_obiekt)	0,0000135	0,0095238	

Na podstawie reguł przedstawionych w tabeli 5.11 można stwierdzić, że najczęstszą przyczyną wypadków *w deszczu i we mgle* jest inny *poruszający się pojazd* lub też *pieszy*, w mniejszym stopniu natomiast były to *wywrotka* czy też *słup reklamowy*. Pozostałe przyczyny wystąpiły w bardzo małej liczbie przypadków.

5.3 Ocena wyników eksperymentów

Podsumowując wyniki przeprowadzonych badań można zauważyć, że proponowana metoda może mieć zastosowania w wielu dziedzinach, w których poszukuje się zależności pomiędzy danymi, do przedstawienia których w szczególności przydatne są modele reguł asocjacyjnych. W tej pracy analizie poddano dane z dziedziny energetyki, medycyny oraz ruchu drogowego.

Dane te wymagały wstępnego przetworzenia dla potrzeb projektowanych analiz korzystających z modelu reguł asocjacyjnych, za pomocą specjalnych skryptów napisanych w języku Python, które dokonały wymaganych konwersji. Przewiduje się, że w przyszłości aplikacja zostanie rozszerzona o moduł graficznego definiowania tego typu skryptów, aby ułatwić dostosowanie aplikacji do danych pochodzących z różnych źródeł, tak aby była możliwość zautomatyzowania również i tego etapu procesu pozyskiwania wiedzy.

Jak można stwierdzić poprzez analizę wyników badań, metoda automatycznego generowania zapytań, w kolejnych krokach procesu ewolucyjnego pozwalała znaleźć takie zapytania do modelu, aby zwrócony przez nie podzbiór reguł był łatwiejszy w analizie niż liczące setki czy nawet tysiące reguł model analizowany we wstępnym etapie badań. Liczba reguł pozyskanych w wyniku zadania „optymalnego" zapytania do modelu była zredukowana do wartości oczekiwanej przez użytkownika lub nieznacznie się od niej różniła. Dzięki temu specjaliści dziedzinowi mają ułatwione zadanie podczas analizy, często bardzo dużych zbiorów danych. Podzbiór reguł, zwracanych przez najlepsze zapytanie znalezione przez algorytm, zawierał w większości przypadków tylko reguły znaczące, które jak należy sądzić powinny być interesujące dla użytkownika.

Zakończenie

W pracy zaproponowano metodę pozyskiwania wiedzy z dużych zbiorów danych z wykorzystaniem ewolucyjnych procedur generowania zapytań. Proponowane rozwiązanie stanowi połączenie dwóch dziedzin: odkrywania wiedzy za pomocą reguł asocjacyjnych oraz algorytmów ewolucyjnych, a w szczególności algorytmów programowania genetycznego.

Rozprawa zawiera rozbudowane wprowadzenie, w którym przedstawiono aktualny stan badań w dziedzinie pozyskiwania wiedzy ze zbiorów danych oraz omówiono problem redukcji rozbudowanych modeli drążenia, ze szczególnym uwzględnieniem modelu reguł asocjacyjnych. Redukcja ta odbywała się do tej pory przy pomocy różnych języków zapytań, takich jak: DMQL, MINE RULE, MSQL, SQL, itp., które wykorzystywane były na kolejnych etapach procesu budowy modelu. Użycie tych języków wymagało opracowania specjalnych metod przechowywania reguł asocjacyjnych oraz znajomości przez użytkownika ich, dość często rozbudowanej, składni.

Zasadniczą część pracy stanowią rozdziały poświęcone opisowi metody automatycznego generowania zapytań przy pomocy algorytmów programowania genetycznego. Zaproponowano tutaj zapis modeli reguł asocjacyjnych w opartym na XML standardzie PMML, który dzięki swojej tekstowej formie jest łatwy w dostępie i przetwarzaniu. Jako język zapytań do modelu zaproponowano XQuery, którego reprezentację w postaci osobnika w populacji procesu programowania genetycznego przedstawiono w rozdziale 3.

Kolejnym zadaniem zrealizowanym w niniejszej rozprawie było zdefiniowanie kryterium oceny. W tym celu połączono kryterium subiektywne, definiowane przez użytkownika z kryterium obiektywnym, jakim jest wywodząca się z teorii informacji *J-miara*. Kryterium subiektywne zdefiniowane zostało w postaci wzorca kryterium i nie wymaga od użytkownika wprowadzania specjalistycznych wzorów, a jedynie w sposób intuicyjny wskazywania (przy pomocy myszki) interesujących powiązań. W ramach dalszych prac planowane jest rozszerzenie możliwości definiowania kryterium, np. poprzez dodanie opcji wskazywania w kryterium powiązań koniecznych lub opcjonalnych.

W celu weryfikacji zaproponowanej metody opracowano aplikację GAZdRA zaimplementowaną w języku Java, której opis znajduje się w rozdziale 4. Oprogramowanie składa się z pięciu modułów, do budowy których oprócz darmowych bibliotek programistycznych, wykorzystano szereg zaimplementowanych rozwiązań własnych.

W rozdziale 5 przedstawiono opis oraz wyniki badań eksperymentalnych przeprowadzonych na trzech zbiorach danych rzeczywistych. Wskazują one na to, że zaproponowana metoda automatycznego generowania zapytań może skutecznie służyć do poszukiwania coraz to lepszego zapytania do modelu reguł asocjacyjnych. Zapytanie to ogranicza liczbę reguł wynikowych do takich, które są interesujące z punktu widzenia analityka. Dzięki temu, użytkownik ma możliwość analizy tylko pewnego podzbioru reguł powiązanych z problemem jaki zamierza rozwiązać.

W wyniku przeprowadzonych obserwacji można dostrzec pewne ograniczenia proponowanej metody. Pierwszym z nich jest działanie metody tylko na modelu jakościowych reguł asocjacyjnych. Planuje się, że w ramach dalszych prac metoda zostanie rozszerzona o inne modele np. o zaproponowane w [80] ilościowe reguły asocjacyjne. Drugim problemem związanym z wdrożeniem metody jest jej zapotrzebowanie na pamięć oraz moc obliczeniową sprzętu komputerowego, które, jak wykazały testy, wydatnie wzrasta przy

większej liczbie osobników populacji. Dlatego też, kolejnym zadaniem na przyszłość będzie podjęcie próby zaimplementowania metody w środowisku rozproszonym, co pozwoli na przyśpieszenie wyszukiwania bardziej skomplikowanych zależności.

Reasumując, jako najważniejsze i oryginalne osiągnięcia wynikające z przeprowadzonych badań, zdaniem autora, można wskazać:

- stworzenie koncepcji zwiększenia efektywności pozyskiwanej wiedzy z dużych zbiorów danych, opartej na automatycznym generowaniu zapytań, z zastosowaniem programowania genetycznego;
- skonstruowanie kryterium oceny otrzymanych rozwiązań, stanowiącego połączenie kryterium subiektywnego (definiowanego przez użytkownika) z kryterium obiektywnym opartym na zastosowaniu *J-miary*;
- zaprojektowanie i implementację aplikacji, stanowiącej narzędzie realizacji opracowanej metody, umożliwiającej zarówno weryfikację samej metody, jak też rozwiązywanie określonej klasy zadań praktycznych (co potwierdzają przeprowadzone eksperymenty).

Pomimo wspomnianych powyżej niedoskonałości opracowanego rozwiązania, zasadnym wydaje się stwierdzenie, że uzyskane rezultaty stanowią znaczący krok w kierunku doskonalenia metod i narzędzi pozyskiwania wiedzy z dużych zbiorów danych – co pośrednio dowodzi prawdziwości tezy sformułowanej w początkowej części rozprawy.

Dodatki

Dodatek A

Opis danych

A.1 Dane o zużyciu energii elektrycznej

Dane o zużyciu energii zostały pozyskane z Zakładu Energetycznego w Zamościu. Przedstawiają one wyniki pomiarów zużycia energii elektrycznej przeprowadzone w latach 2004-2006 w odstępach godzinowych. Pomiary te wykonywane są odpowiednio dla całego Zakładu, jak również oddzielnie dla trzech miast leżących na terenie Zakładu: Chełma, Przemyśla i Zamościa.

Oprócz zużycia energii zapisywane są również dane dotyczące zmierzonej temperatury w siedmiu miejscowościach regionu, a także maksymalne, minimalne oraz średnie wartości temperatury pomierzone dla całego obszaru działania Zakładu. Dodatkowo mierzone było nasłonecznienie w dwóch miejscach regionu. Ostatnie dwa atrybuty służyły do zarejestrowania daty i czasu dokonanego pomiaru.

W skład bazy danych wchodzi 65301 rekordów, z których każdy opisany jest przy pomocy 17 atrybutów.

A.2 Dane alergologiczne

Dane alergologiczne pochodzą z przeprowadzonych, w latach 2004-2006 przez Zakład Alergologii Szpitala Uniwersyteckiego w Krakowie, badań

w Ramach Miejskiego Programu Profilaktyki i Promocji Zdrowia „Zdrowy Kraków". Badaniem objęto 1918 dzieci w wieku od 7 do 9 lat oraz uczniów w wieku od 16 do 18 lat.

Badanie przeprowadzono w dwóch etapach. W I etapie posłużono się ankietą opracowaną przez prof. dr hab. med. Krystynę Obtułowicz zawierającą zestaw pytań dotyczących występowania objawów choroby alergicznej, takich jak: katar, zapalenie spojówek, napady duszności, kaszlu, świszczącego oddechu, zmian skórnych o typie wyprysku, pokrzywki oraz kolek brzusznych u badanych obecnie i w dzieciństwie. Ponadto ankietowanego pytano o uczulenie na jakieś substancje, występowanie innych chorób, stosowanie leków przewlekle, a także czy jest leczony w Poradni Alergologicznej. Następna grupa pytań dotyczyła obecności choroby alergicznej w rodzinie respondenta. Ankieta ta zawierała również prośbę kierowaną do rodziców i opiekunów o zgodę na wykonanie badań alergologicznych.

Na podstawie analizy wypełnionych przez uczniów odpowiednich ankiet, przeprowadzonego wywiadu i badania fizykalnego do drugiego etapu badań zakwalifikowano 867 dzieci w wieku od 7 do 9 lat i 328 uczniów w wieku od 16 do 18 lat z dodatnim wywiadem alergologicznym. W ramach tego etapu wykonano punktowe testy skórne (10 alergenów) oraz testy płatkowe z alergenami kontaktowymi.

Baza danych składa się z 1918 rekordów, z których każdy dotyczył jednego pacjenta opisanego za pomocą 50 atrybutów.

A.3 Dane o wypadkach drogowych w USA

Dane dotyczą wypadków samochodowych ze skutkiem śmiertelnym, które wydarzyły się w Stanach Zjednoczonych w latach 1998 i 1999.

Dane te zostały pobrane z serwera FTP organizacji National Highway Traffic Safety Administration (NHTSA) [59]. Opisują one zdarzenia ze

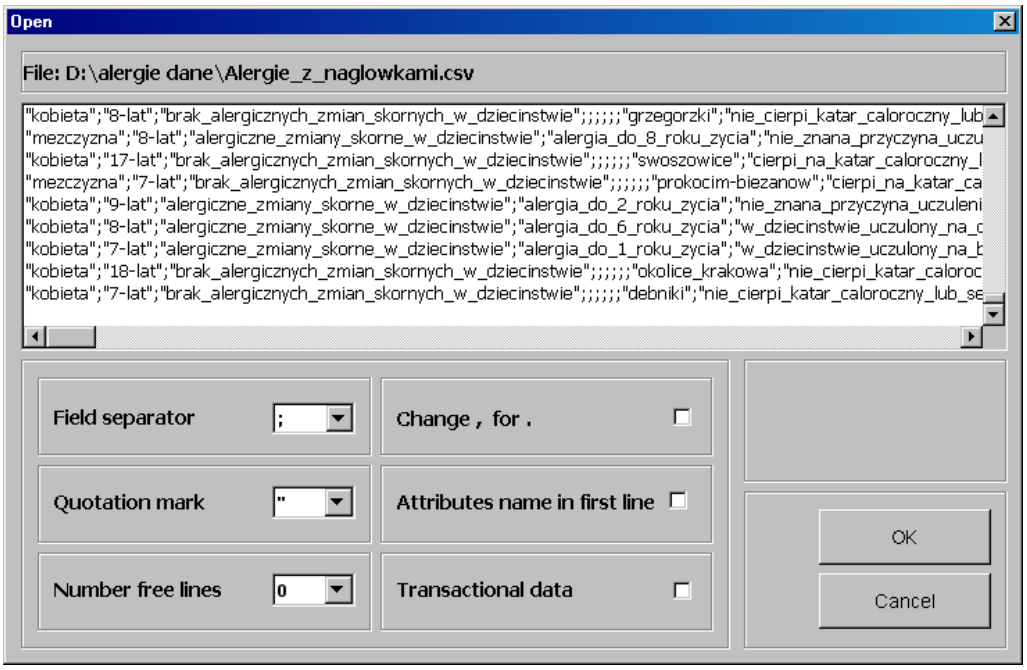
wszystkich stanów i zawierają dodatkowe informacje powiązane z wypadkami, takie jak: miejsce wypadku, data, rodzaj zdarzenia, warunki pogodowe, liczbę poszkodowanych, itp.

Baza danych gromadzi 74150 rekordów, z których każdy jest opisany za pomocą 16 cech (atrybutów).

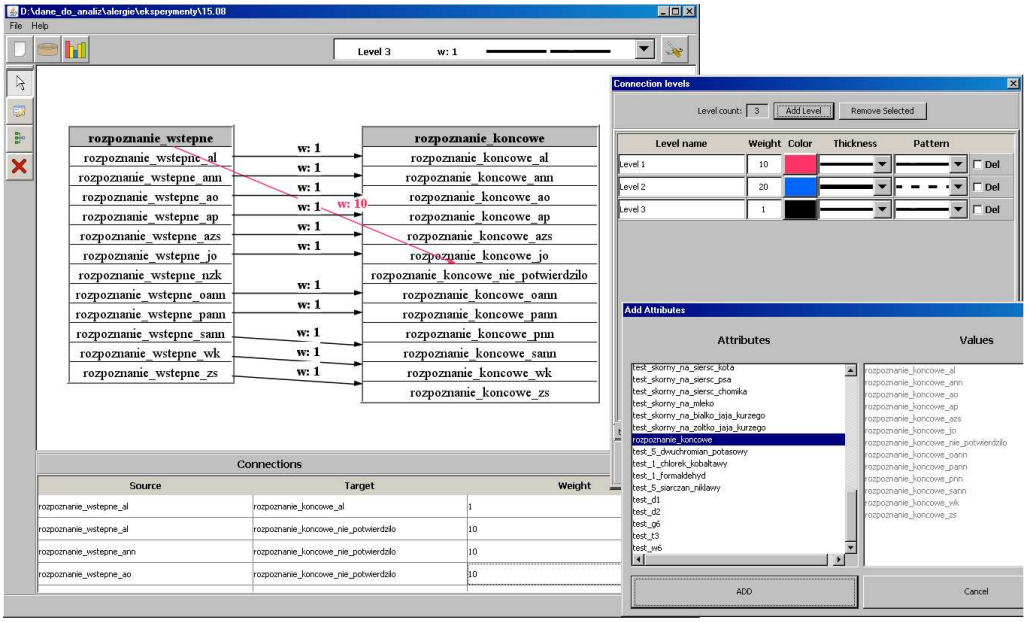
Dodatek B

Elementy aplikacji

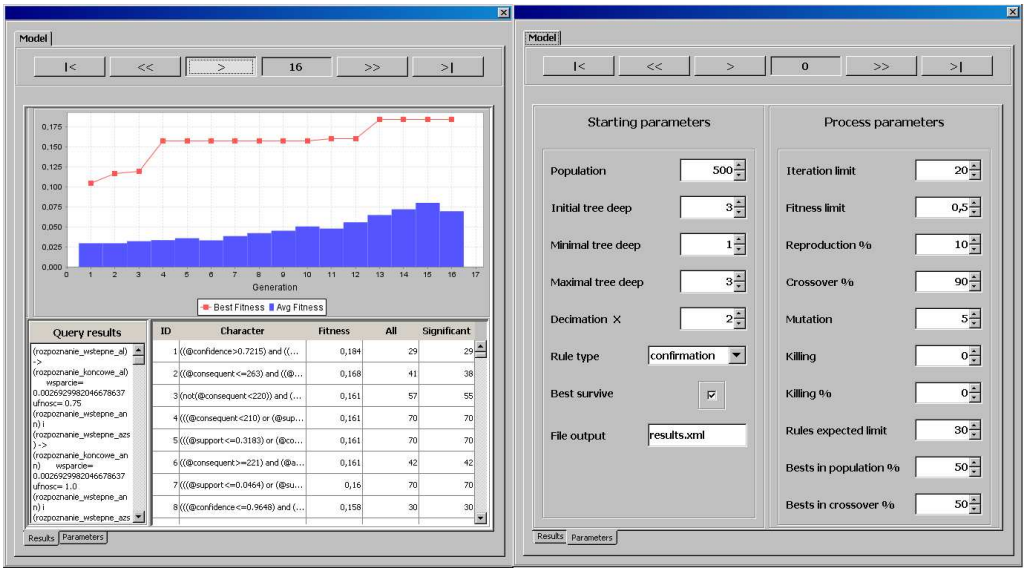
Przedstawione dalej fragmenty interfejsu użytkownika (zrzuty z ekranu) ilustrują najważniejsze etapy działania aplikacji GAZdRA.



Rysunek B.1: Moduł wczytywania i przygotowania danych

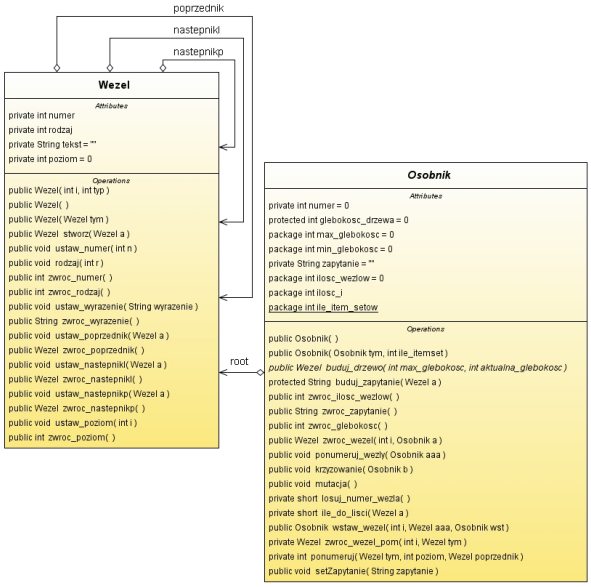


Rysunek B.2: Moduł definicji kryterium subiektywnego



Rysunek B.3: Moduł procesu programowania genetycznego i wizualizacji wyników

Przedstawione dalej diagramy obrazują fragment struktury klas aplikacji GAZdRA.



Rysunek B.4: Fragment diagramu klas aplikacji I

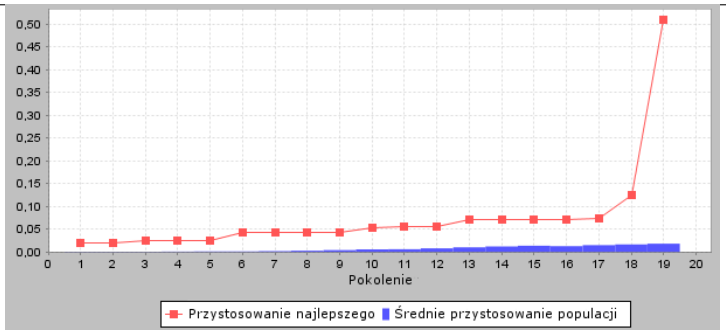
Rysunek B.5: Fragment diagramu klas aplikacji II

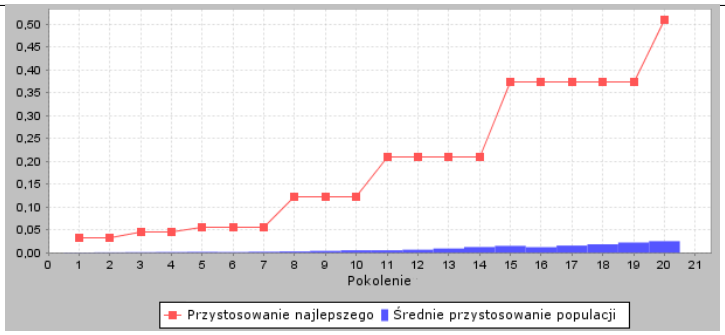
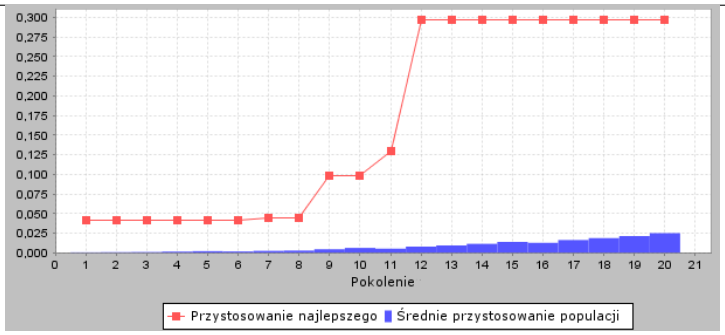
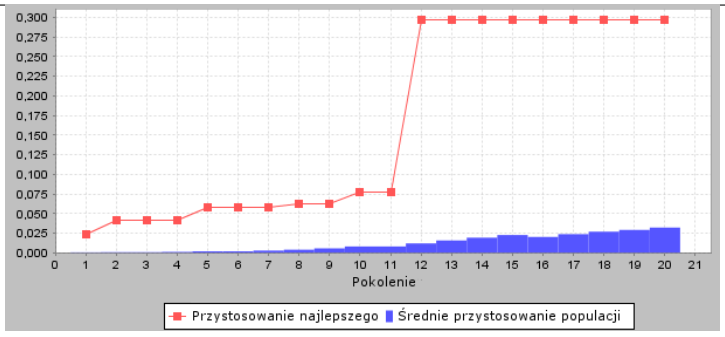
Dodatek C

Wyniki badań eksperymentalnych

W dodatku podano wybrane rezultaty badań eksperymentalnych, ilustrujące przebieg procesu ewolucyjnego, jak też sposób prezentacji otrzymanych wyników.

Tabela C.1: Wyniki eksperymentów obliczeniowych – dane energetyczne

<table>
<tr><th>Lp.</th><th>Najlepszy osobnik</th><th>L_z/L_r</th></tr>
<tr><td rowspan="3">1</td><td><code>((@antecedent>=358) and ((((@confidence<=0.0908) and (@antecedent>=277)) and (@support<0.2938)) or (@confidence>0.1036))) and ((((@consequent>22) and (@confidence>=0.6420)) and ((@consequent>370) and (@confidence>=0.9225))) or (@antecedent<=369))</code></td><td>7/7</td></tr>
<tr><td colspan="2">Przebieg procesu ewolucyjnego</td></tr>
<tr><td colspan="2"> 
</td></tr>
</table>

Lp.	Najlepszy osobnik	L_z/L_r
2	`((((@confidence<=0.4079) or (not(@consequent<=398))) or ((@antecedent!=111) and ((((@support<0.4882) and (@antecedent<=147)) or (not(@confidence>0.1221))))) and ((not(((@antecedent!=367) and ((@support<0.1055) or (@consequent<=287)))) and ((@antecedent!=81) or (@confidence>0.1922)))`	7/7
	Przebieg procesu ewolucyjnego	
		
3	`(@antecedent<374) and ((@antecedent>358) or ((@support>=0.0717) and (@consequent<=292)))`	7/12
	Przebieg procesu ewolucyjnego	
		
4	`(((@antecedent<374) or ((@confidence>0.1858) and (@support=0.3395))) and (not(@antecedent<=362))) or ((@support>=0.0761) or ((@confidence=0.5932) or (@confidence>=0.8053)))`	7/12
	Przebieg procesu ewolucyjnego	
		

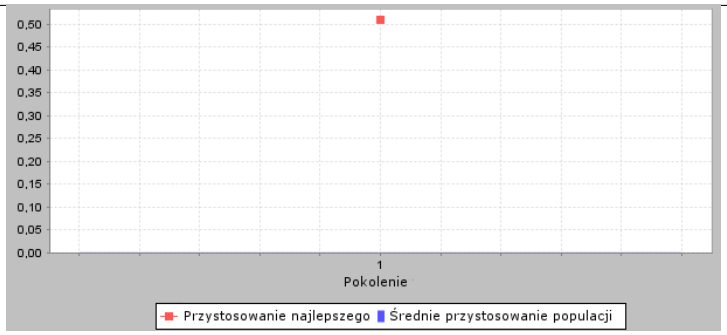
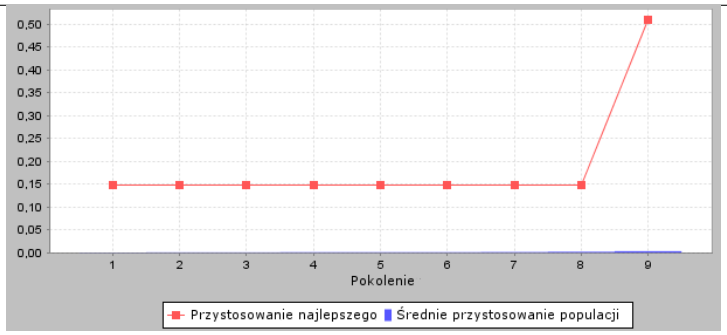
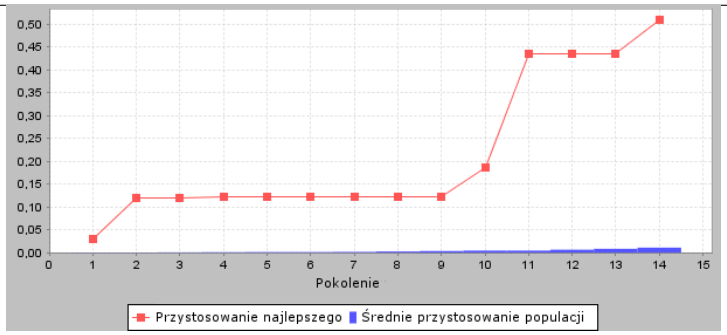
<table>
<tr><th>Lp.</th><th>Najlepszy osobnik</th><th>L_z/L_r</th></tr>
<tr><td rowspan="3">5</td><td colspan="2">not(not((@support>=0.6561) or (@antecedent=367)))</td></tr>
<tr><td colspan="2">Przebieg procesu ewolucyjnego</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td rowspan="3">6</td><td colspan="2">not(@antecedent!=367)</td></tr>
<tr><td colspan="2">Przebieg procesu ewolucyjnego</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td rowspan="3">7</td><td colspan="2">(((not(@antecedent=98)) and ((@confidence>0.0751) or (@antecedent>362))) and (((@antecedent<368) and ((@support>0.8431) or (@antecedent>362))) and (@antecedent<=415))) and (((not((@support=0.1782) and (@antecedent<=78))) or (not(not(@consequent<=284)))) or (not((not(@support>=0.6012)) and (@consequent!=139))))</td></tr>
<tr><td colspan="2">Przebieg procesu ewolucyjnego</td></tr>
<tr><td colspan="2"> 
</td></tr>
</table>

Tabela C.2: Wyniki eksperymentów obliczeniowych – dane medyczne

Liczba osobników	**Nr eksperymentu**	**Wartość kryterium oceny**	$\mathbf{L}_z/\mathbf{L}_r$
100	1	0,1398	28/34
	2	0,1213	99/108
	3	0,1216	26/33
	4	0,1509	26/32
	5	0,1113	32/42
	6	0,1206	33/42
	7	0,1337	28/36
	8	0,1175	30/38
	9	0,1098	33/44
	10	0,1421	44/53
500	1	0,1947	29/30
	2	0,1595	33/33
	3	0,1786	26/30
	4	0,1702	25/29
	5	0,1469	35/41
	6	0,1501	29/29
	7	0,1914	26/28
	8	0,1805	30/30
	9	0,2226	30/30
	10	0,2226	30/30
1000	1	0,1760	27/29
	2	0,1589	33/36
	3	0,1848	31/34
	4	0,1914	26/28
	5	0,2232	31/31
	6	0,2232	31/31
	7	0,1612	34/34
	8	0,1636	29/31
	9	0,2226	30/30
	10	0,2226	30/30

Bibliografia

[1] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener. The Lorel query language for semistructured data. *International Journal on Digital Libraries*, 1(1):68–88, 1997. [cytowanie na str. 48]

[2] R. Agrawal, T. Imieliński, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D. C., 26–28 1993. [cytowanie na str. 9]

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994. [cytowanie na str. 12]

[4] T. Anand, G. Kahn, and A. C. Nielsen. Spotlight: A data explanation system. In *Proc. of the Eighth Conference on Artificial Intelligence for Application CAIA-92*, pages 2–8, Monterey, CA, 1992. [cytowanie na str. 6]

[5] J. Arabas. *Wykłady z algorytmów ewolucyjnych.* WNT, Warszawa, 2004. [cytowanie na str. 34, 35]

[6] P. J. Azevedo and A. M. Jorge. Comparing rule measures for predictive association rules. In *ECML*, pages 510–517, 2007. [cytowanie na str. 60, 61]

[7] T. Bäck, F. Hoffmeister, and H. P. Schwefel. A survey of evolution strategies. In R. Belew and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 2–9, San Mateo, CA, 1991. Morgan Kaufman. [cytowanie na str. 28]

[8] J. E. Baker. Adaptive selection methods for genetic algorithms. In J. J. Grefenstette, editor, *Proceedings of the First International Conference on*

Genetic Algorithms and Their Applications. Lawrence Erlbaum Associates, Publishers, 1985. [cytowanie na str. 35]

[9] D. Braga, A. Campi, S. Ceri, M. Klemettinen, and P. Lanzi. Discovering interesting information in xml data with association rules. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 450–454, New York, NY, USA, 2003. ACM. [cytowanie na str. 48]

[10] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984. [cytowanie na str. 14, 15]

[11] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In J. Peckham, editor, *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pages 265–276. ACM Press, 1997. [cytowanie na str. 59]

[12] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In J. Peckham, editor, *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pages 255–264. ACM Press, 05 1997. [cytowanie na str. 60]

[13] J. Brzeziński, T. Morzy, M. Morzy, and Ł. Rutkowski. Algorytm optymalizacji zapytań eksploracyjnych z wykorzystaniem materializowanych perspektyw eksploracyjnych. Technical Report RB-006/02, Poznań University of Technology, 2002. [cytowanie na str. 12]

[14] A. G. Buchner, M. D. Mulvenna, S. S. Anand, M. Baumgarten, and R. Bohm. Data mining and xml: Current and future issues. *wise*, 02:127–131, 2000. [cytowanie na str. 44]

[15] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca. XML-GL: A graphical language for querying and restructuring XML documents. In *Sistemi Evoluti per Basi di Dati*, pages 151–165, 1999. [cytowanie na str. 48]

[16] D. Chamberlin, J. Robie, and D. Florescu. Quilt: An XML query language for heterogeneous data sources. *Lecture Notes in Computer Science*, 1997, 2001. [cytowanie na str. 48]

[17] P. Cichosz. *Systemy uczące się*. WNT, Warszawa, 2000. [cytowanie na str. 14]

[18] K. Cios and L. Kurgan. Trends in data mining and knowledge discovery, 2002. [cytowanie na str. 42]

[19] P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Proc. Fifth European Working Session on Learning*, pages 151–163, Berlin, 1991. Springer. [cytowanie na str. 59]

[20] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970. [cytowanie na str. 5]

[21] T. Connolly and C. E. Begg. *Database Systems: A Practical Approach to Design, Implementation and Management 2nd Ed.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998. [cytowanie na str. 4]

[22] Data Mining Group (DMG). http://www.dmg.org/, 1998-2007. [cytowanie na str. 41]

[23] K. A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975. [cytowanie na str. 28, 35]

[24] A. Deutsch, M. F. Fernandez, D. Florescu, A. Y. Levy, and D. Suciu. "XML-QL: A Query Language for XML". In *WWW The Query Language Workshop (QL)*, Cambridge, MA, 1998. [cytowanie na str. 48]

[25] L. Rutkowski D.Rutkowska, M. Piliński. *Sieci neuronowe, algorytmy genetyczne i systemy rozmyte*. Wydawnictwo Naukowe PWN, Warszawa, 1999. [cytowanie na str. 34]

[26] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *Ai Magazine*, 17:37–54, 1996. [cytowanie na str. 7]

[27] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996. [cytowanie na str. 7]

[28] U. M. Fayyad, N. Weir, and S. Djorgovski. Automated cataloging and analysis of sky survey image databases: the skicat system. In *CIKM '93: Proceedings of the second international conference on Information and knowledge management*, pages 527–536, New York, NY, USA, 1993. ACM Press. [cytowanie na str. 6]

[29] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, New York, 1966. [cytowanie na str. 28]

[30] D. E. Goldberg. *Algorytmy genetyczne i ich zastosowania*. WNT, 2003. [cytowanie na str. 28]

[31] D. E. Goldberg, K. Deb, and B. Korb. Do not worry, be messy. In *Proc. of the fourth Int. Conf. on Gen. Alg.*, pages 24–30. Morgan-Kaufmann publishers, Los Altos, CA, 1991. [cytowanie na str. 35]

[32] R. Goldman, J. McHugh, and J. Widom. From semistructured data to XML: Migrating the lore data model and query language. In *Workshop on the Web and Databases (WebDB '99)*, pages 25–30, 1999. [cytowanie na str. 48]

[33] J. Han, J. Chiang, S. Chee, J. Chen, Q. Chen, S. Cheng, W. Gong, M. Kamber, G. Liu, K. Koperski, Y. Lu, N. Stefanovic, L. Winstone, B. Xia, O. Zaiane, S. Zhang, and H. Zhu. Dbminer: A system for data mining in relational databases and data warehouses, 1997. [cytowanie na str. 16]

[34] J. Han, Y. Fu, W. Wang, K. Koperski, and O. Zaiane. DMQL: A data mining query language for relational databases, 1996. [cytowanie na str. 9, 15, 16]

[35] J. Han and M. Kamber. *Data Mining, Second Edition : Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems) (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, January 2006. [cytowanie na str. 7, 14, 16]

[36] M. Herdy. Application of the Evolutionsstrategie to Discrete Optimization Problems. In H. P. Schwefel and R. Männer, editors, *Parallel problem solving from nature: 1st Workshop, PPSN I*, pages 188–192, Berlin, 1991. Springer. [cytowanie na str. 28]

[37] J. Hipp, Ch. Mangold, U. Guntzer, and G. Nakhaeizadeh. Efficient Rule Retrieval and Postponed Restrict Operations for Association Rule Mining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 52–65, 2002. [cytowanie na str. 15, 20, 46]

[38] J. H. Holland. *Adaptation in natural artificial systems.* University of Michigan Press, Ann Arbor, 1975. [cytowanie na str. 28]

[39] D. Hunter, A. Watt, J. Rafter, K. Cagle, J. Duckett, and B. Patterson. *Beginning XML (Programmer to Programmer).* Wrox Press Ltd., 2004. [cytowanie na str. 47]

[40] T. Imieliński and A. Virmani. MSQL: A Query Language for Database Mining. *Data Min. Knowl. Discov.*, 3(4):373–408, 1999. [cytowanie na str. 15, 18]

[41] T. Imieliński, A. Virmani, and A. Abdulghani. DMajor – Application Programming Interface for Database Mining. *Data Min. Knowl. Discov.*, 3(4):347–372, 1999. [cytowanie na str. 15, 18, 19, 20]

[42] W. H. Inmon. *Building the Data Warehouse.* John Wiley & Sons, Inc., New York, NY, USA, 1992. [cytowanie na str. 5]

[43] W. H. Inmon. Data mart does not equal data warehouse. *DM Review*, 1996. [cytowanie na str. 6]

[44] B. Jędrzejec, E. Czarnobilska, G. Porębski, K. Obtułowicz, and E. Nawarecki. The Estimation of Reliability In The Preventive Examination Of Allergic Diseases With Knowledge Discovery Methods. In *Computers in Medical Activity*, 2008. [cytowanie na str. 92]

[45] B. Jędrzejec and E. Nawarecki. Discovery of knowledge in electrical power systems. In *9th IFAC Workshop on Intelligent Manufacturing Systems (IMS'08)*, pages 75–80, 2008. [cytowanie na str. 85]

[46] G. V. Kass. An Exploratory Technique for Investigating Large Quantities of Categorical Data. *j-APPL-STAT*, 29(2):119–127, 1980. [cytowanie na str. 14]

[47] M. Kay. Comparing XSLT and XQuery. In *XTech 2005: XML, the Web and beyond*, 2005. [cytowanie na str. 47]

[48] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. In Nabil R. Adam, Bharat K. Bhargava, and Yelena Yesha, editors, *Third International Conference on Information and Knowledge Management (CIKM'94)*, pages 401–407. ACM Press, 1994. [cytowanie na str. 61]

[49] J. R. Koza. Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STAN-CS-90-1314, Dept. of Computer Science, Stanford University, 1990. [cytowanie na str. 29]

[50] J. R. Koza. *Genetic Programming: On the programming of computers by means of natural selection.* Bradford Books, Cambridge, MA., 1992. [cytowanie na str. 31]

[51] D. T. Larose. *Odkrywanie wiedzy z danych.* Wydawnictwo Naukowe PWN, 2006. [cytowanie na str. 60]

[52] B. Liu, W. Hsu, L.-F. Mun, and H.-Y. Lee. Finding Interesting Patterns Using User Expectations. *Knowledge and Data Engineering*, 11(6):817–832, 1999. [cytowanie na str. 62]

[53] K. McGarry. A survey of interestingness measures for knowledge discovery. *Knowl. Eng. Rev.*, 20(1):39–61, 2005. [cytowanie na str. 62]

[54] D. Meltz, R. Long, M. Harrington, R. Hain, and G. Nicholls. *Introduction to IMS™, An: Your Complete Guide to IBM's Information Management System*. IBM Press, 2004. [cytowanie na str. 4]

[55] R. Meo, G. Psaila, and S. Ceri. A New SQL-like Operator for Mining Association Rules. In *The VLDB Journal*, pages 122–133, 1996. [cytowanie na str. 15, 16, 17]

[56] R. Meo, G. Psaila, and S. Ceri. A Tightly-Coupled Architecture for Data Mining. In *ICDE '98: Proceedings of the Fourteenth International Conference on Data Engineering*, pages 316–323, Washington, DC, USA, 1998. IEEE Computer Society. [cytowanie na str. 15, 16]

[57] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed.* Artificial Intelligence. Springer-Verlag, Berlin, 1996. [cytowanie na str. 27, 34, 35]

[58] T. Morzy and M. Zakrzewicz. SQL-Like Language for Database Mining. In *Proceedings of the First East-European Symposium on Advances in Databases and Information Systems (ADBIS'97), St.-Petersburg, September 2-5, 1997. Volume 1: Regular Papers*, pages 311–317. Nevsky Dialect, 1997. [cytowanie na str. 15, 16, 17]

[59] National Highway Traffic Administration. Fatality Analysis Reporting System Encyclopedia, 2003-2008. ftp://ftp.nhtsa.dot.gov/FARS. [cytowanie na str. 107]

[60] R. T. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. pages 13–24, 1998. [cytowanie na str. 19]

[61] T. W. Olle. *The Codasyl Approach to Data Base Management*. John Wiley & Sons, Inc., New York, NY, USA, 1978. [cytowanie na str. 4]

[62] G. Piatetsky-Shapiro and W. J. Frawley, editors. *Knowledge Discovery in Databases*. AAAI/MIT Press, 1991. [cytowanie na str. 12]

[63] G. Piatetsky-Shapiro and C. Matheus. The interestingness of deviations, 1994. [cytowanie na str. 61]

[64] Predictive Model Markup Language (PMML). http://www.dmg.org/v3-1/generalstructure.html. [cytowanie na str. 44]

[65] D. Pyle. *Data preparation for data mining*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999. [cytowanie na str. 8]

[66] J. R. Quinlan. Simplifying decision trees. *Int. J. Man-Mach. Stud.*, 27(3):221–234, 1987. [cytowanie na str. 15]

[67] J. R. Quinlan. Induction of Decision Trees. In J. W. Shavlik and T. G. Dietterich, editors, *Readings in Machine Learning*. Morgan Kaufmann, 1990. Originally published in *Machine Learning* 1:81–106, 1986. [cytowanie na str. 15]

[68] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. [cytowanie na str. 14]

[69] J. R. Quinlan and R. L. Rivest. Inferring decision trees using the minimum description length principle. *Inf. Comput.*, 80(3):227–248, 1989. [cytowanie na str. 15]

[70] S. Raspl. PMML Version 3.0 - Overview and Status. In *KDD-2004 Workshop on Data Mining Standards, Services and Platforms (DM-SSP 04), KDD-2004 The Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004. [cytowanie na str. 44]

[71] I. Rechenberg. *Evolutionsstrategie*, volume 15 of *problemata*. Friedrich Frommann Verlag (Günther Holzboog KG), Stuttgart, 1973. [cytowanie na str. 28]

[72] J. Robie, J. Lapp, and D. Schach. XML Query Language (XQL). In *Proc. of the query languages workshop*, 1998. [cytowanie na str. 48]

[73] L. Rutkowski. *Metody i techniki sztucznej inteligencji : inteligencja obliczeniowa*. Wydawnictwo Naukowe PWN, Warszawa, 2006. [cytowanie na str. 27]

[74] R. Sasisekharan, V. Seshadri, and S. M. Weiss. Data Mining and Forecasting in Large-Scale Telecommunication Networks. *IEEE Expert: Intelligent Systems and Their Applications*, 11(1):37–43, 1996. [cytowanie na str. 6]

[75] A. Savasere, E. Omiecinski, and S. B. Navathe. An Efficient Algorithm for Mining Association Rules in Large Databases. In *The VLDB Journal*, pages 432–444, 1995. [cytowanie na str. 12]

[76] H. P. Schwefel. *Numerical optimization of Computer models*. John Wiley & Sons, Ltd., Chichester, 1981. [cytowanie na str. 28]

[77] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *Ieee Trans. On Knowledge And Data Engineering*, 8:970–974, 1996. [cytowanie na str. 58]

[78] R. A. Sinoara and S. O. Rezende. A methodology for identifying interesting association rules by combining objective and subjective measures. *Inteligencia Artificial, Revista Iberoamericana de IA*, 10(32):19–27, 2006. [cytowanie na str. 58, 62]

[79] P. Smyth and R. M. Goodman. An information theoretic approach to rule induction from databases. *IEEE Transactions on Knowledge and Data Engineering*, 4(4):301–316, 1992. [cytowanie na str. 60]

[80] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. pages 1–12, 1996. [cytowanie na str. 103]

[81] R. Srikant, Q. Vu, and R. Agrawal. Mining Association Rules with Item Constraints. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining, KDD*, pages 67–73. AAAI Press, 14–17 1997. [cytowanie na str. 19, 59]

[82] L. D. Stevens. The Evolution of Magnetic Storage. *IBM Journal of Research and Development*, 25(5):663–676, 1981. [cytowanie na str. 3]

[83] K. Świder and B. Jędrzejec. A Query-Driven Exploration of Association Rule Models in PMML. In R. Tadeusiewicz, A. Ligęza, and M. Szymkat, editors, *Proc. 5th Int. Conference Computer Methods and Systems*, pages 409–414. Oprogramowanie Naukowo-Techniczne, 2005. [cytowanie na str. 43, 49]

[84] K. Świder, B. Jędrzejec, and M. Wysocki. A Query Driven Exploration of Multiple Association Rules. In C. Cotta, S. Reich, R. Schaefer, and A. Ligeza, editors, *Knowledge-Driven Computing, Knowledge Engineering and Intelligent Computations Series: Studies in Computational Intelligence*, volume 102, pages 283–288, 2008. [cytowanie na str. 43, 68, 83]

[85] P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the right objective measure for association analysis. *Inf. Syst.*, 29(4):293–313, 2004. [cytowanie na str. 61]

[86] M. Thess and M. Bolotnicov. *XELOPES Library Documentation, Version 1.3.1.* prudsys AG., 2005. [cytowanie na str. 77, 78]

[87] TwoCrows. Introduction to Data Mining and Knowledge Discovery. 1999. Two Crows Corporation, Third Edition, Online Publication. [cytowanie na str. 8]

[88] J. W. W. Wan and G. Dobbie. Mining association rules from XML data using XQuery. In *ACSW Frontiers '04: Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation*, pages 169–174, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc. [cytowanie na str. 48]

[89] D. Whitley. The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation. In J. D. Schaffer, editor, *Proc. of the Third Int. Conf. on Genetic Algorithms*, pages 116–121, San Mateo, CA, 1989. Morgan Kaufmann. [cytowanie na str. 35]

[90] M. Wojciechowski and M. Zakrzewicz. Itemset materializing for fast mining of association rules. In *Advances in Databases and Information Systems*, pages 284–295, 1998. [cytowanie na str. 12]

[91] World Wide Web Consortium. Document object model (DOM) level 3 core specification, 2004. http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/. [cytowanie na str. 46]

[92] World Wide Web Consortium. XML Path Language (XPath) 2.0, 2007. http://www.w3.org/TR/xpath20/. [cytowanie na str. 48]

[93] World Wide Web Consortium. XQuery 1.0: An XML Query Language, 2007. http://www.w3.org/TR/xquery/. [cytowanie na str. 48]

[94] World Wide Web Consortium. XSL transformations (XSLT) version 2.0, 2007. http://www.w3.org/TR/xslt20/. [cytowanie na str. 47]

[95] C. Yongho, K. Hosung, J. Myoung-ju, K. Dongjoo, C. Hyungsik, L. Jong-Min, K. In-Young, and K. I. Sun. ECG Acquisition and Management System for Knowledge Discovery in Database: Data Modeling, Design, and Implementation. In *ISMDA '02: Proceedings of the Third International Symposium on Medical Data Analysis*, pages 79–85, London, UK, 2002. Springer-Verlag. [cytowanie na str. 6]

Spis rysunków

Spis tabel

Spis algorytmów