Na prawach rękopisu

POLITECHNIKA WROCŁAWSKA
INSTYTUT INFORMATYKI STOSOWANEJ

Raport Serii PRE nr 2

Pozyskiwanie wiedzy w systemie agenckim z wykorzystaniem metod eksploracji danych

(rozprawa doktorska)

Damian Dudek

Promotor: dr hab. inż. Aleksander Zgrzywa, prof. PWr

Słowa kluczowe: agent, eksploracja danych, metody inkrementacyjne, statystyczne uczenie się

Wrocław 2005

Podziękowania

Pragnę podziękować Promotorowi pracy, Panu prof. Aleksandrowi Zgrzywie za opiekę merytoryczną, a także za wielką życzliwość, cierpliwość i zaufanie.

Pracownikom, doktorantom, absolwentom i studentom Zakładu Systemów Informacyjnych dziękuję za cenne dyskusje i różnego rodzaju wsparcie. Dziękuję zwłaszcza Panu Michałowi Kubiszowi za pomoc w przygotowaniu środowiska do badań eksperymentalnych.

Dziękuję Radkowi Katarzyniakowi za pomoc, niezwykle inspirujące rozmowy i wiedzę, której mogłem dzięki nim zaczerpnąć.

Członkom mojej rodziny, zwłaszcza żonie i synkowi – dziękuję za to, że zawsze są ze mną.

Spis treści

Wykaz ważniejszych oznaczeń

A_i	atrybut
A^S_i, A^M_j	atrybut jednowartościowy i wielowartościowy
$con\,(r, B)$	pewność (ang. *confidence*) reguły r w bazie transakcji B
$con_{overlap}\,(R_1, R_2)$	współczynnik zgodności pewności zbiorów reguł R_1 i R_2
D_K	dziedzina klucza schematu historii
D^M_j	dziedzina atrybutu wielowartościowego A^M_j
D^S_i	dziedzina atrybutu jednowartościowego A^S_i
D_T	zbiór punktów czasowych
$D^W_i\,(KB_H\,(t_1, t_2))$	*dziedzina właściwa atrybutu* A_i w historii KB_H, dla interwału czasowego $[t_1; t_2]$
$\mathscr{F}$	rodzina wszystkich funkcji wpływu czasowego f_T
$F\,(B, \sigma)$	zbiór wszystkich częstych zbiorów atrybutów w bazie transakcji B, przy progu σ
$FAKT\,(S_H)$	zbiór wszystkich faktów o schemacie S_H
$freq\,(X, B)$	częstość (ang. *frequency*) zbioru atrybutów X w bazie transakcji B
f_T	funkcja wpływu czasowego
γ	próg minimalnej pewności
$\hat{\gamma}$	estymator oczekiwanej pewności reguł
η	maksymalna, dopuszczalna liczba wartości N w jednym fakcie
$K\,(s)$, $T\,(s)$, $A^S_i(s)$, $A^M_j(s)$	wartości (odpowiednio): klucza, czasu, atrybutu jednowartościowego i wielowartościowego w fakcie s
$KB^{\#}_H(t_1; t_2)$	porcja faktów o schemacie $S^{\#}$ dla interwału czasowego $[t_1; t_2]$
$KB^B_R\,(h)$	zbiór reguł odkrytych wsadowo w zbiorze faktów h
KB_G	wiedza ogólna (pamięć długoterminowa)
KB_H	historia
$KB_H(t_1, t_2)$	porcja faktów dla interwału czasowego $[t_1; t_2]$
$KB^I_R\,(h)$	zbiór reguł odkrytych inkrementacyjnie w zbiorze faktów h
$KB^O_R(R_1, R_2)$	przecięcie semantyczne zbiorów reguł R_1 i R_2

KB_R	baza reguł
KB_T	wiedza chwilowa (pamięć krótkoterminowa)
$L\ (y, \hat{y})$	funkcja straty (ang. *loss function*), wyrażająca błąd odpowiedzi systemu uczącego się
m_x, m_y	maksymalna liczba atrybutów, które mogą wystąpić w poprzedniku i następniku reguły;
$p \equiv r$	semantyczna równość reguł p i r
$PI\ (d, P)$	zbiór częstych, interesujących wzorców opisujących zbiór danych d
R	zbiór reguł
$R\ (B, \sigma, \gamma)$	zbiór wszystkich częstych i wiarygodnych reguł w bazie B, przy progach minimalnego poparcia σ i pewności γ
$r = (X, Y, sup, con, b, t_m)$	reguła (rozszerzona definicja)
$r\colon X \Rightarrow Y$	reguła o poprzedniku X i następniku Y
$rule_{overlap}\ (R_1, R_2)$	współczynnik zgodności semantycznej zbiorów reguł R_1 i R_2
σ	próg minimalnego poparcia
$\hat{\sigma}$	estymator oczekiwanego poparcia reguł
$s = (tid, i_1, i_2, ..., i_m)$	transakcja (fakt)
$S^{\#}$	przekształcony schemat historii, w którym wszystkie atrybuty przyjmują wartości ze zbioru $\{0, 1, N\}$
$S_H = \{K, T, U\}$	schemat historii
S^S_H, S^M_H	zbiór atrybutów jedno- i wielowartościowych, należących do schematu S_H
$sup\ (r, B)$	poparcie (ang. *support*) reguły r w bazie transakcji B
$sup_{overlap}\ (R_1, R_2)$	współczynnik zgodności poparcia zbiorów reguł R_1 i R_2
$time_{dev}\ (R_1, R_2)$	średnie odchylenie czasowe zbiorów reguł R_1 i R_2
t_{now}	bieżący czas systemowy
T_{ref}	jednostka odniesienia w średnim odchyleniu czasowym $time_{dev}$
$U = \{I_1, I_2, ..., I_n\}$	zbiór atrybutów
$\vec{v}_c\ , v_c$	wektor danych przebiegu
$\vec{v}_g\ , v_g$	wektor parametrów globalnych

Wprowadzenie

Ogólna charakterystyka dziedziny

Technologie agenckie od kilkunastu lat stanowią rozległy obszar prac, których celem jest budowa inteligentnych, autonomicznych systemów, wspomagających człowieka w rozwiązywaniu problemów rozproszonych lub złożonych obliczeniowo, takich jak wyszukiwanie informacji w sieci WWW lub zarządzanie produkcją.

Głównym postulatem formułowanym w odniesieniu do systemu agenckiego jest zdolność autonomicznego, celowego i elastycznego działania, zgodnie ze stawianymi przed nim zadaniami. Spełnienie tych założeń wymaga wyposażenia agenta w odpowiedni aparat poznawczy, struktury przechowywania wiedzy, mechanizmy wnioskowania i wykonywania akcji w otoczeniu. W ostatnich latach można zaobserwować rosnące zainteresowanie rozwijaniem mechanizmów uczenia się. Tendencja ta wynika w naturalny sposób z postulatów autonomiczności i elastyczności, które sprawiają, iż zaprojektowanie dla agenta kompletnego i poprawnego algorytmu działania w rzeczywistym, złożonym otoczeniu często jest bardzo trudne lub nieopłacalne. Z tego względu celowe jest wyposażenie agenta w mechanizm samodzielnego zdobywania wiedzy i umiejętności podczas działania w docelowym środowisku.

Jeśli system agencki osadzony jest w dynamicznym, niedeterministycznym otoczeniu (np. sieć Internet), pozyskiwanie przez niego wiedzy o świecie i zachodzących tam zależnościach może być utrudnione, gdyż elementarne fakty bezpośrednio rejestrowane ze środowiska są bardzo liczne i nieraz sprzeczne ze sobą. W takiej formie obserwacje te nie mogą być bezpośrednio wykorzystywane przez agenta w procesie wnioskowania i planowania ciągu akcji. Dopiero analiza odpowiednio dużej liczby faktów, w celu odkrycia ogólnych, popartych statystycznie zależności, może dostarczyć agentowi nowej wiedzy przydatnej w procesie wnioskowania. Ze względu na charakter problemu oraz wymogi wydajnościowe, do analizy bazy obserwacji agenta celowe może się okazać wykorzystanie metod *eksploracji danych* (ang. *data-mining*), obecnie intensywnie rozwijanych w ramach dziedziny *wydobywania wiedzy z baz danych* (ang. *Knowledge Discovery in Databases – KDD*). Metody te pozwalają na wydajne znajdowanie ogólnych zależności na podstawie dużych zbiorów danych trenujących. Poza tym można ustalić wiarygodność wiedzy odkrywanej w ten sposób za pomocą odpowiednich miar statystycznych.

Cel pracy

Celem niniejszej pracy doktorskiej jest opracowanie metody inkrementacyjnego pozyskiwania reguł przez agenta na podstawie zgromadzonych obserwacji, wykorzystującej mechanizmy eksploracji danych. **Metoda ma zapewniać inkrementacyjne przetwarzanie dużej liczby obserwacji, przy zachowaniu ograniczonego rozmiaru przechowywanych danych historycznych i zbiorze uzyskanych w ten sposób reguł, który byłby porównywalny ze zbiorem reguł otrzymanym przy wsadowym przetwarzaniu całego zbioru danych.**

Osiągnięcie głównego celu pracy doktorskiej wiąże się z rozwiązaniem poniższych problemów szczegółowych.

1. Zdefiniowanie struktur bazy wiedzy agenta, potrzebnych do pozyskiwania reguł.
2. Opracowanie cyklu metody pozyskiwania reguł – etapów przetwarzania danych, ich wzajemnych relacji oraz interakcji ze strukturami bazy wiedzy agenta.
3. Zaproponowanie algorytmów wykorzystywanych w cyklu metody.
4. Weryfikacja formalna i eksperymentalna zaproponowanej metody.

Propozycja rozwiązania

W pracy zaproponowano rozwiązanie postawionego problemu pozyskiwania reguł w postaci nowej metody APS (Analiza Przeszłych Stanów, ang. *Analysis of Past States*). Jest to metoda inkrementacyjna, która pozwala na odkrywanie i dodawanie do bazy wiedzy agenta reguł związku pomiędzy atrybutami opisu rejestrowanego stanu świata. Metoda APS opiera się na podejściu *pamięci niedoskonałej* (ang. *imperfect recall*), ponieważ agent po przetworzeniu określonej porcji faktów historycznych i wydobyciu na ich podstawie odpowiednich reguł, usuwa bezpowrotnie te fakty ze swojej bazy wiedzy. Nowo odkryte reguły są jednak dodawane do bazy reguł w taki sposób, który uwzględnia reguły znalezione we wcześniejszych przebiegach uczenia się. W rezultacie, pomimo usuwania faktów, wynikowa baza reguł jest w przybliżeniu taka sama, jak w przypadku analizy wszystkich obserwacji od początku. Podejście takie pozwala na znaczącą redukcję rozmiaru bazy obserwacji, co ma duże znaczenie przy ograniczonych zasobach agenta.

Proponowana metoda pozwala na eliminację nieznanych wartości atrybutów poprzez generowanie *światów możliwych* (ang. *possible worlds*). Technika ta oparta jest na założeniu *światów dowolnych* (ang. *random-worlds*), w myśl którego wszystkie światy możliwe traktowane są jako jednakowo prawdopodobne. Generowanie światów możliwych wiąże się ze wzrostem złożoności obliczeniowej, ale jednocześnie wyklucza lub zmniejsza utratę informacji, która występowałaby przy usuwaniu niekompletnych faktów ze zbioru trenującego.

Metoda APS jest oddzielona od samego algorytmu eksploracji danych, co pozwala na zastosowanie różnych algorytmów i porównanie ich działania w różnych aplikacjach.

Plan rozprawy

Pierwszy rozdział poświęcony jest statystycznym metodom uczenia się bez nadzoru. Wyspecyfikowano w nim problem uczenia się bez nadzoru z perspektywy metod statystyki matematycznej i przedstawiono główne techniki eksploracji danych. Omówiono szczegółowo model reguł związku (ang. *association rules*) i ważniejsze metody ich znajdowania.

W drugim rozdziale pracy zawarto zwięzły przegląd współczesnych badań nad technologiami agenckimi. Omówiono definicję agenta i najważniejsze zastosowania agentów. Podane zostały podstawowe definicje dotyczące pojęcia wiedzy. Przytoczony został przegląd ważniejszych modeli reprezentacji wiedzy w systemach agenckich. Dalsza część rozdziału zawiera szczegółowy przegląd prac dotyczących uczenia się agentów. Przegląd ma służyć umiejscowieniu proponowanej metody pozyskiwania reguł w kontekście procesu uczenia się systemu agenckiego.

Trzeci rozdział zawiera szczegółową prezentację zaproponowanej metody APS. Rozdział rozpoczyna ogólny, nieformalny przegląd metody. Dalej scharakteryzowane zostały założenia metody. Podane zostały formalne definicje struktur wiedzy agenta, które są wykorzystywane w metodzie. Zdefiniowany został cykl pozyskiwania reguł w metodzie APS oraz algorytmy, które go realizują. Przedstawiono własności algorytmów, w szczególności algorytmu inkrementacyjnej aktualizacji bazy reguł. Dalsza część rozdziału stanowi dokumentację przeprowadzonej weryfikacji eksperymentalnej metody APS. Opisane zostały: cel i plan eksperymentu, środowisko testowe, zbiory danych testowych. Przedstawiono wyniki badań jakości i wydajności odkrywania reguł dla różnych wariantów danych wejściowych i ustawień parametrów metody.

Końcowa część rozprawy zawiera zestawienie i omówienie najważniejszych wyników, uzyskanych w pracy ze wskazaniem ich możliwych zastosowań i propozycji dalszych badań.

W Dodatku A przedstawiony został przykład obliczeniowy, który ilustruje sposób działania cyklu zaproponowanej metody APS. Dodatek B stanowi zestawienie liczbowych wyników badań eksperymentalnych.

1. Eksploracja danych jako metoda maszynowego uczenia się

W niniejszym rozdziale scharakteryzowano techniki maszynowego uczenia się. Zarysowano problematykę nauki o uczeniu się, w tym główne nurty badań z tej dziedziny. Podana została definicja systemu uczącego się i motywacja do tworzenia tego typu systemów. Przedstawiono podział metod uczenia się ze względu na trzy kryteria: metodę reprezentacji wiedzy, sposób jej wykorzystania oraz informację trenującą. Opisano podstawowe tryby uczenia się systemu: wsadowy, inkrementacyjny, epokowy i korekcyjny.

Dalsza część rozdziału, ze względu na cel pracy, poświęcona została grupie metod eksploracji danych. Podano definicję technik odkrywania zależności w bazach danych oraz ich podział. Szczegółowo omówiono, wykorzystywane w proponowanej metodzie APS, techniki odkrywania reguł związku. Z dziedziny tej przedstawiono ważniejsze osiągnięcia opracowane w ostatnich latach, w tym jeden z najbardziej znanych algorytmów – Apriori.

1.1. Maszynowe uczenie się

Maszynowe uczenie się, zwane także *uczeniem się maszyn* (ang. *machine learning*) jest obecnie jednym z najsilniej rozwijanych i obiecujących działów *słabej sztucznej inteligencji* (ang. *weak artificial intelligence*), którego celem jest stworzenie sztucznych systemów posiadających zdolność uczenia się. W dziedzinie tej wykorzystywane są osiągnięcia z wielu obszarów, takich jak [Cic2000]: statystyka, teoria prawdopodobieństwa, teoria informacji, modele baz danych, logika formalna, teoria sterowania, psychologia, neurofizjologia. W dziedzinie maszynowego uczenia się można wyróżnić trzy główne działy:

- *teoretyczny* – obejmujący badania nad teoretycznymi podstawami algorytmów uczenia się, oceną ich złożoności i jakości wyników ich działania; w ramach tego nurtu wypracowywane jest także jednolite nazewnictwo dotyczące technik uczenia się;
- *biologiczny* – mający na celu tworzenie obliczeniowych modeli procesów uczenia się, które występują w naturalnych systemach biologicznych (np. u ludzi); rozważania prowadzone są na różnych poziomach struktury tych organizmów i biorą pod uwagę aspekty biologiczne oraz psychologiczne; przykładem znanej pracy z tego zakresu jest zunifikowana teoria poznawania Newella [New1990], która stoi u podstaw systemu *Soar* [Lai1987];

- *systemowy* – obejmujący prace, których celem jest rozwijanie konkretnych metod i algorytmów uczenia się oraz tworzenie wykorzystujących je systemów; nurt ten dominuje w dziedzinie maszynowego uczenia się.

Wspólnym mianownikiem wymienionych nurtów jest badanie, wyjaśnianie i rozwijanie mechanizmów uczenia się. Aby uniknąć niejednoznaczności, przyjmujemy następującą definicję uczenia się, podaną przez Cichosza w pracy [Cic2000].

Uczeniem się systemu jest każda autonomiczna zmiana w systemie zachodząca na podstawie doświadczeń, która prowadzi do poprawy jakości jego działania.

([Cic2000] str. 34)

Autor powyższej definicji zwraca uwagę na trzy ważne aspekty uczenia się: jest ono wynikiem samodzielnych procesów zachodzących w systemie (w odróżnieniu od przekazywania gotowej wiedzy systemowi na przykład na etapie projektowania), opiera się ono na doświadczeniu, a więc informacji zaobserwowanej przez system w trakcie jego działania, i w końcu prowadzi, według pewnych kryteriów, do polepszenia działania systemu (w odróżnieniu od zwykłych zmian stanu wiedzy systemu, które nie mają bezpośredniego wpływu na zmianę jego skuteczności lub efektywności).

Meystel i Albus [Mey2002] na podstawie literatury przytaczają aż kilkanaście różnych definicji uczenia się. W oparciu o nie autorzy wskazują jednak na pewien zestaw wspólnych cech tego procesu. Poniżej przytoczone są niektóre z nich.

- Uczenie się jest procesem nabywania umiejętności, przy czym termin *umiejętność* zakłada istnienie pewnego programu, który wykazuje użyteczne zachowanie.
- Uczenie się powoduje zmianę algorytmu programu.
- Powoduje zmianę indywidualnego zachowania programu.
- Powoduje, że program uczący się staje się lepiej dostosowany do swoich zadań, to znaczy wykonuje zadania w sposób bardziej efektywny. Zakłada się przy tym, że w mechanizm uczenia się powinna być wbudowana miara jakości działania programu.
- Buduje lub modyfikuje reprezentację wiedzy.
- Jest procesem odkrywania. Tworzy nowe klasy i uogólnione kategorie.
- Wymusza na systemie określoną odpowiedź na zadany sygnał wejściowy.

Istnieje wiele powodów intensywnego rozwoju metod maszynowego uczenia się [Cic2000], [Die2003], [She2000], [Thr1995]. W wielu współczesnych dziedzinach zastosowań wymagany jest wysoki stopień autonomiczności tworzonych systemów informatycznych, aby mogły one spełniać stawiane przed nimi cele wspomagania człowieka w sterowaniu skomplikowanymi procesami i rozwiązywaniu problemów złożonych obliczeniowo. Jednocześnie systemy te funkcjonują często w dynamicznych i trudnych do modelowania środowiskach, co sprawia, iż zaprojektowanie kompletnych i prawidłowych algorytmów ich działania może być niemożliwe, bardzo trudne lub nieopłacalne. Nawet jeśli projektant systemu jest w stanie przewidzieć wszystkie możliwości, z którymi system może się zetknąć w trakcie działania, przypadków tych może być tak wiele, że przeszukiwanie ich przestrzeni podczas wykonywania algorytmu byłoby nieefektywne. Tymczasem w rzeczywistości system i tak

może zetknąć się jedynie z niewielką częścią tych przypadków spośród wszystkich możliwych, a więc wystarczyłoby, aby system został zaprojektowany i zoptymalizowany właśnie ze względu na nie. Okazuje się jednak, że na etapie projektowania często nie można z góry określić, z którymi przypadkami zetknie się system. W tej sytuacji pojawia się potrzeba wyposażenia systemu w mechanizm samodzielnego zdobywania wiedzy w oparciu o informacje docierające do systemu już po jego wdrożeniu w konkretnym środowisku. Podobna motywacja przyświeca rozwijaniu technik uczenia się w dziedzinie systemów agenckich, których podstawowymi cechami są właśnie autonomiczność oraz adaptacyjność w dynamicznym środowisku [Mae1994].

1.2. Podział metod uczenia się

Istnieje wiele sposobów podziału metod maszynowego uczenia się [Mul1996]. Meystel i Albus [Mey2002] przytaczają aż 20 różnych typów uczenia się. Z kolei Cichosz [Cic2000] proponuje wielowymiarową klasyfikację, opartą na czterech kryteriach, omówionych niżej.

- *Reprezentacja wiedzy* – w oparciu o tę cechę można wyróżnić metody *symboliczne* oraz *subsymboliczne*. W pierwszej grupie wiedza zapisywana jest za pomocą symboli, które posiadają określone znaczenie, zgodnie z przyjętą semantyką; w grupie drugiej informacja jest zapisana jako zbiory lub ciągi elementów, które brane osobno nie posiadają interpretacji zrozumiałej dla człowieka.
- *Sposób wykorzystania wiedzy lub umiejętności* – wynika on z metody reprezentacji wiedzy i celu jej gromadzenia. Ze względu na to kryterium można wymienić między innymi: *klasyfikację*, *aproksymację*, *rozwiązywanie problemów*, *sekwencyjne podejmowanie decyzji*, *modelowanie środowiska* i *przedstawienie wiedzy użytkownikowi do dalszego wykorzystania*.
- *Źródło i postać informacji trenującej* – istnieją dwie podstawowe rodziny metod: *uczenie się z nadzorem* (ang. *supervised learning*) oraz *uczenie się bez nadzoru* (ang. *unsupervised learning*). Obie grupy są omawiane w dalszej części niniejszego rozdziału.
- *Mechanizm nabywania i doskonalenia wiedzy lub umiejętności* – podstawowymi grupami są: *uczenie się indukcyjne* (ang. *inductive learning*), które polega na wyciąganiu ogólnych wniosków na podstawie poszczególnych przypadków trenujących, oraz *mechanizmy nieindukcyjne*, w tym *wyjaśnianie* (ang. *explanation-based learning*), służące ukonkretnianiu wiedzy wrodzonej systemu uczącego się.

Dietterich [Die2003] zwraca uwagę na inny, ważny sposób podziału metod maszynowego uczenia się, wyróżniając dwie grupy.

- *Uczenie się na podstawie doświadczenia* (ang. *empirical learning*) – nabywanie wiedzy wymaga dostarczania informacji trenującej z zewnątrz.
- *Uczenie się na podstawie analizy* (ang. *analytical learning*, *speedup learning*) – nie jest wymagana informacja trenująca z zewnątrz; system uczący się doskonali swoją wiedzę lub umiejętności poprzez analizę i przetwarzanie wiedzy już posiadanej.

W każdej z wymienionych wyżej grup mamy do czynienia z wieloma algorytmami i technikami o charakterze ogólnym lub dostosowanymi do konkretnych aplikacji.

Uczenie się zazwyczaj przebiega w jednym z trzech głównych trybów [Cic2000].

- *Wsadowy* (ang. *batch mode*) – system uczący się w tym trybie otrzymuje do przetworzenia kompletny zbiór informacji trenującej. Podczas uczenia się nie są dostarczane do systemu żadne dodatkowe przykłady trenujące i nie ma żadnej interakcji z nauczycielem. Jeżeli po zakończeniu nauki pojawiają się nowe przykłady, uczenie musi być wykonywane jeszcze raz od początku – na całym zaktualizowanym zbiorze trenującym. Tryb wsadowy nakłada najmniejsze wymagania na algorytmy uczenia się, ale jednocześnie ma bardzo ograniczone zastosowanie praktyczne.
- *Inkrementacyjny* (ang. *incremental mode*) – w tym trybie system nie dostaje całego zbioru trenującego na raz, lecz kolejno przetwarza poszczególne jego elementy, czyli pojedyncze przypadki, doskonaląc swoją wiedzę. W dowolnym momencie proces uczenia się może być przerwany w celu przedstawienia dotychczasowych jego wyników. Tryb inkrementacyjny jest stosowany w przeważającej większości rzeczywistych zastosowań i eksperymentów obliczeniowych.
- *Epokowy* (ang. *epoch mode*) – stanowi hybrydę, która łączy w sobie cechy trybów wsadowego i inkrementacyjnego. Proces uczenia się przebiega w cyklach zwanych *epokami*, w których kolejno przetwarzane są pewne zbiory przykładów trenujących. Do czasu przetworzenia kolejnej epoki, system uczący się może korzystać z wyników uzyskanych we wcześniejszych cyklach. Praktyczne wykorzystanie tego typu algorytmów wynika ze specyfiki niektórych zastosowań oraz z faktu, iż niektóre algorytmy działają wydajniej w trybie wsadowym, niż inkrementacyjnym.

Maloof i Michalski [Mal2000], [Mal2004] omawiają trzy grupy metod uczenia się w zależności od sposobu przechowywania przeszłych przykładów trenujących w pamięci.

- *Kompletna pamięć przykładów* (ang. *full instance memory*) – system uczący się przechowuje w pamięci wszystkie przykłady trenujące.
- *Częściowa pamięć przykładów* (ang. *partial instance memory*) – system przechowuje w pamięci część przykładów trenujących. Wykorzystywany jest pewien mechanizm selekcji najlepszych przykładów z punktu widzenia bieżącego zadania uczenia się lub mechanizm zapominania, usuwający z pamięci przykłady, które z punktu widzenia zadania uczenia się są nierelewantne lub nieaktualne.
- *Brak pamięci przykładów* (ang. *no instance memory*) – system nie przechowuje żadnych przykładów trenujących w pamięci.

1.3. Uczenie się z nadzorem i bez nadzoru

Nadzór lub jego brak podczas procesu zdobywania wiedzy stanowi podstawowe kryterium podziału metod uczenia się maszyn. Obie, omówione niżej, grupy technik są obecnie intensywnie rozwijane i odgrywają ważną rolę w systemach uczących się.

1.3.1 Uczenie się z nadzorem

Uczenie się z nadzorem (ang. *supervised learning*) jest niekiedy określane jako *uczenie się z nauczycielem* (ang. *learning with a teacher*). Zadaniem systemu uczącego się jest

obserwowanie informacji wejściowej i odpowiadanie na nią właściwą informacją wyjściową, natomiast proces uczenia się ma na celu znalezienie algorytmu podawania właściwych odpowiedzi [Cic2000]. Możemy, posługując się notacją podaną w [Has2001], zapisać informację wejściową i wyjściową w postaci wektorów zmiennych, odpowiednio: $X = (X_1, X_2, ..., X_p)$ oraz $Y = (Y_1, Y_2, ..., Y_m)$. Oznaczamy przez $x_i = (x_{i1}, x_{i2}, ..., x_{ip})$ wektor wartości zmiennych wejściowych dla i-tego przypadku (obserwacji) oraz przez $y_i = (y_{i1}, y_{i2}, ..., y_{im})$ odpowiadający mu wektor wartości zmiennych wyjściowych. System uczący się ma za zadanie przewidywać, jakie wartości będą przyjmowały zmienne wyjściowe w zależności od podawanych zmiennych wejściowych. Przewidywanie to jest oparte na zbiorze trenującym, czyli przykładach prawidłowych odpowiedzi y_i dla określonych wektorów wejściowych x_i, postaci: (x_1, y_1), (x_2, y_2), ..., (x_N, y_N). Informacja trenująca, określana także mianem *ciągu uczącego* [Bub1993], jest podawana do systemu uczącego się przez źródło zwane *nauczycielem*. Dla każdego przypadku trenującego x_i system uczący się zwraca odpowiedź $\hat{y}_i$, a nauczyciel podaje prawidłową odpowiedź albo błąd odpowiedzi systemu uczącego się, który może być wyrażony za pomocą pewnej *funkcji straty* (ang. *loss function*): $L(y, \hat{y})$. W ujęciu probabilistycznym uczenie się z nadzorem może być formalnie określone jako problem estymacji gęstości rozkładu prawdopodobieństwa warunkowego $Pr(Y|X)$, gdzie (X, Y) są zmiennymi losowymi, posiadającymi pewien wspólny rozkład $Pr(X, Y)$ [Has2001].

Odmianą uczenia się z nadzorem są: *uczenie się na podstawie zapytań* [Cic2000], w którym nauczyciel pełni rolę *wyroczni*: dostarcza systemowi uczącemu się informacji trenującej, lecz ma ona postać wyłącznie odpowiedzi na jawne zapytania systemu uczącego się. Jako inny rodzaj uczenia się z nadzorem niekiedy wskazywane jest *uczenie się ze wzmocnieniem* (ang. *reinforcement learning*) [Cic2000], jednak tutaj kwalifikujemy je raczej jako uczenie się bez nadzoru, ponieważ w większości systemów uczących się w ten sposób źródłem oceny odpowiedzi jest wyłącznie wewnętrzna funkcja nagrody, nie zaś źródło zewnętrzne. Dlatego też uczenie się ze wzmocnieniem jest omawiane w następnym podrozdziale: *Uczenie się bez nadzoru*.

1.3.2 Uczenie się bez nadzoru

W procesie *uczenia się bez nadzoru* (ang. *unsupervised learning*) lub inaczej *uczenia się bez nauczyciela* (ang. *learning without a teacher*) nie jest dostępna informacja trenująca, a jedynie sekwencje wektorów zmiennych wejściowych X, na podstawie których system uczący się ma samodzielnie określić właściwe odpowiedzi Y [Cic2000]. W ujęciu probabilistycznym, opisanym w pracy [Has2001], system uczący się otrzymuje zbiór N obserwacji $(x_1, x_2, ..., x_N)$ wektora $X = (X_1, X_2, ..., X_p)$, który posiada rozkład prawdopodobieństwa $Pr(X)$. Celem uczenia się jest bezpośrednie wywnioskowanie przez system własności tego rozkładu bez pomocy nauczyciela, który mógłby dostarczać prawidłowych odpowiedzi lub oceny błędu. Ponieważ jednak w problemach uczenia się bez nadzoru zazwyczaj wymiar wektora X jest znacznie większy, niż w przypadku metod uczenia się z nadzorem, zamiast dokładnego estymowania rozkładu $Pr(X)$ stosuje się pewne *statystyki opisowe* (ang. *descriptive statistics*), charakteryzujące wartości wektora X lub zbiory takich wartości, dla których gęstość rozkładu $Pr(X)$ jest stosunkowo wysoka. Przykładami technik uczenia się bez nadzoru są: *samoorganizujące się mapy* (ang. *self-organizing maps*), *grupowanie* (ang. *cluster analysis*) i *reguły związku* (ang. *association rules*). Metody te wiążą się z analizą i przekształcaniem przestrzeni

zmiennych wejściowych, opartym na zasadach będących integralną częścią samego algorytmu uczenia się [Bub1993], [Cic2000].

O ile w przypadku metod uczenia się z nadzorem istnieje wyraźna miara oceny jakości uczenia się systemu (np. wspomniana wyżej funkcja straty), o tyle w odniesieniu do uczenia się bez nadzoru tego typu bezpośrednia miara nie istnieje. W większości algorytmów uczenia się bez nadzoru trudno jest ustalić prawidłowość wiedzy uzyskiwanej w ten sposób na podstawie obserwacji. Z tego powodu opisane wyżej metody uczenia się bez nadzoru trudniej upowszechniają się, niż algorytmy z nadzorem [Has2001].

Szczególnym przypadkiem uczenia się bez nadzoru jest *uczenie się poprzez eksperymentowanie* [Cic2000], w którym system uczący się wykonuje akcje na otoczeniu, obserwuje ich wyniki i na tej podstawie modyfikuje swoją wewnętrzną strategię podejmowania decyzji. Do tej grupy metod można zaliczyć *uczenie się ze wzmocnieniem* (ang. *reinforcement learning*, RL) [Cic2000], [Die1997], [Dor1994], [Kwa2004], [Rib2002], w którym nie jest dostępne żadne zewnętrzne źródło informacji trenującej (*nauczyciel*), a jedynie doświadczenie systemu uczącego się, pochodzące z jego interakcji z otoczeniem, oraz wewnętrzna funkcja oceny akcji. Metoda ta jest w ostatnim czasie przedmiotem bardzo dużego zainteresowania w dziedzinie maszynowego uczenia się.

W wielu pracach definicja uczenia się ze wzmocnieniem jest oparta na formalizmie *procesów decyzyjnych Markova* (ang. *Markov Decision Processes*, *MDPs*). Podstawą tego aparatu formalnego jest *warunek Markova*, który mówi, że każda obserwacja zewnętrznego procesu (środowiska), uczyniona przez system uczący się, musi być wyłącznie funkcją poprzedniej obserwacji i akcji systemu (z uwzględnieniem pewnego, losowego zakłócenia): $o_{t+1} = f(o_t, a_t, w_t)$, gdzie o_t jest obserwacją w chwili t, a_t – wykonaną akcją, w_t – zakłóceniem [Rib2002].

System uczący się ze wzmocnieniem przeprowadza eksperymentowanie na środowisku, wykonując akcję a, która zmienia środowisko ze stanu s_t do innego stanu s_{t+1}. Działanie systemu jest oceniane przez jego wewnętrzną informację trenującą, mającą postać liczbowej *funkcji nagrody* (ang. *reward function*) $R(s_t, a, s_{t+1})$ [Die1997]. Informacja trenująca ma więc charakter wartościujący, nie zaś instruktażowy [Cic2000]. Funkcja nagrody pozwala na lokalną ocenę poszczególnych akcji, przeprowadzanych w konkretnych stanach otoczenia i dlatego jest określana jako *nagroda natychmiastowa* (ang. *immediate reward*). Natomiast do oceny działania systemu uczącego się w dłuższym okresie stosowana jest powszechnie miara *oczekiwanej łącznej nagrody* (ang. *cumulative discounted reward*) [Die1997]:

$$\sum_{t=0}^{\infty} \gamma^t R(s_t, a, s_{t+1})$$

przy czym $0 < \gamma^t < 1$ jest współczynnikiem odpowiednio zmniejszającym znaczenie nagród spodziewanych w przyszłości. Reguła lub procedura wyboru akcji w konkretnym stanie nazywana jest *strategią* (ang. *policy*). Celem uczenia się ze wzmocnieniem jest zatem znalezienie optymalnej strategii, która maksymalizuje łączną nagrodę [Die1997].

Istnieją dwie podstawowe odmiany uczenia się ze wzmocnieniem [Die2003]: uczenie się oparte na modelu (ang. *model-based* RL) i uczenie się bez modelu (ang. *model-free* RL). W pierwszym przypadku system uczący się po wykonaniu każdej akcji zapisuje doświadczenie w postaci czwórek $\langle s_t, a, r, s_{t+1} \rangle$, gdzie s_t jest obserwowanym stanem środowiska w chwili t; a – akcją wykonaną przez system, r – uzyskanym wzmocnieniem, a s_{t+1} – stanem

środowiska w chwili $t + 1$. Po zgromadzeniu wystarczająco dużej liczby takich czwórek, system może się nauczyć *funkcji prawdopodobieństwa przejścia* (ang. *probability transition function*) $P(s_{t+1}| s_t, a)$, która określa prawdopodobieństwa przejścia środowiska ze stanu s_t do stanu s_{t+1} po wykonaniu akcji a. Drugą funkcją, której uczy się system, jest wspomniana wcześniej *funkcja nagrody* $R(s_t, a, s_{t+1})$. Jeśli znane są te dwie funkcje, system może obliczyć optymalną strategię, wykorzystując algorytmy *programowania dynamicznego* (ang. *dynamic programming*, DP). Z kolei w uczeniu się ze wzmocnieniem bez modelu (ang. *model-free* RL) system uczy się strategii bezpośrednio podczas interakcji ze środowiskiem, bez zapisywania doświadczenia w postaci $\langle s_t, a, r, s_{t+1}\rangle$ i bez uczenia się modelu – funkcji P i R. Najbardziej znaną i rozpowszechnioną metodą uczenia się bez modelu jest *Q-learning* [Die2003], [Kwa2004], [Rib2002].

Innym, często stosowanym podejściem do rozwiązywania problemu uczenia się ze wzmocnieniem są *systemy klasyfikatorów* (ang. *classifier systems*, CS) – systemy regułowe, które uczą się poprzez odpowiednie modyfikowanie siły reguł na podstawie informacji z otoczenia oraz poprzez znajdowanie lepszych reguł z wykorzystaniem algorytmów genetycznych [Sen1999]. System CS posiada zbiór reguł, nazywanych *klasyfikatorami* (ang. *classifiers*), z których każda posiada część warunkową (opis przykładu trenującego albo sytuacja), część wynikową (klasa, do której jest przydzielany przykład albo akcja) i dane wyrażające jej *siłę* (ang. *strength*) [Mae1994b]. W trakcie uczenia się system wykorzystuje algorytmy ewolucyjne jako zaawansowaną metodę eksperymentowania (zob. przegląd algorytmów ewolucyjnych przedstawiony w pracy [Kwa1999]), zamieniającą klasyfikatory o małej sile na mutacje i krzyżowania (ang. *crossover*) silnych klasyfikatorów. Systemy CS mają wbudowany mechanizm generalizacji – symbol '#', *pomiń* (ang. *don't care*), pozwalający na uogólnianie reguł. Dorigo i Bersini [Dor1994] zaprezentowali ciekawe porównanie technik *Q-learning* oraz CS, wykazując, że po nałożeniu odpowiednich ograniczeń (m.in. brak zapamiętywanych stanów wewnętrznych i wyeliminowanie symbolu '#') metody CS stają się równoważne *Q-learning*.

Uczenie się ze wzmocnieniem (RL) różni się od typowych metod uczenia się z nadzorem [Hag2002]. Po pierwsze system uczący się nie otrzymuje z zewnątrz informacji trenującej w postaci par (wejście, wyjście), lecz po wykonaniu akcji uzyskuje obserwację stanu, który powstał w jej wyniku oraz natychmiastową nagrodę oceniającą akcję w kontekście poprzedniego stanu. System nie otrzymuje zatem informacji, która akcja w danym stanie jest najlepsza z punktu widzenia długoterminowego działania, lecz aby móc działać efektywnie system musi samodzielnie gromadzić i przetwarzać doświadczenie: możliwe stany systemu, akcje, przejścia z jednego stanu do innego oraz nagrody. W końcu, w przeciwieństwie do uczenia się z nadzorem, w metodach RL nie ma podziału na fazę uczenia się (ang. *learning phase*) i fazę działania systemu (ang. *performance phase*), lecz oba procesy przebiegają równolegle. Jest to podejście szczególnie ważne i przydatne w systemach działających w środowiskach dynamicznych i trudnych do strukturalnego opisania, gdzie ważna jest efektywność działania w czasie rzeczywistym (ang. *on-line performance*) [Hag2002].

Szczegółowa prezentacja uczenia się ze wzmocnieniem jest zawarta w pracy [Rib2002].

1.4. Metody eksploracji danych

Odkrywanie wiedzy w bazach danych (ang. *knowledge discovery in databases*, KDD), jest interdyscyplinarną dziedziną, która łączy w sobie osiągnięcia z obszaru maszynowego uczenia się, statystyki oraz baz danych [Cic2000], [Man1996]. Celem systemów KDD jest przetwarzanie rzeczywistych danych, które są powstają w wyniku działalności różnych organizacji i są przechowywane w bazach danych. Potencjalnie zasoby tych baz danych zawierają pewną ukrytą wiedzę – istotne statystycznie, a nieznane dotąd zależności, które po odpowiednim przetworzeniu mogą być wykorzystane do usprawnienia działania organizacji. Systemy KDD mają za zadanie odkrywać te regularności i dostarczać je użytkownikowi w takiej postaci, aby mogły one być prawidłowo interpretowane i wykorzystane w procesach decyzyjnych – na przykład poprzez predykcję. Za *interesujące* można uznać te zależności, które odnoszą się do atrybutów ważnych dla użytkownika danych, natomiast ich *użyteczność* zależy między innymi od ich istotności statystycznej i dokładności [Cic2000].

Fayyad, Piatetsky-Shapiro i Smyth definiują proces KDD jako:

nietrywialny proces znajdowania prawidłowych, *nowych*, *użytecznych i zrozumiałych wzorców w danych* ([Fay1996a] za [Fay1996b]).

Definicja ta obejmuje szereg aspektów odkrywania zależności w bazach danych. Po pierwsze proces KDD ma być *nietrywialny* to znaczy musi on wykraczać poza analizę, w której badane są tylko z góry określone zależności tak, jak to ma miejsce na przykład w prostych aplikacjach OLAP (ang. *Online Analytical Processing*) [Koh1998]. Proces KDD powinien wykorzystywać poszukiwanie struktur, modeli, wzorców lub parametrów. Ponadto wymagane jest, aby odkrywane wzorce były: *prawidłowe* – czyli z określonym przybliżeniem odpowiadające rzeczywistości opisanej w danych; *nowe* – to znaczy nie znane wcześniej; *potencjalnie użyteczne* – takie, które mogą być wykorzystane przez użytkownika do usprawnienia działalności; *zrozumiałe* – czyli w takiej formie, która pozwala na ich właściwą interpretację – bezpośrednio po ich odkryciu lub po odpowiednim przetworzeniu.

Proces KDD jest złożony z wielu etapów, które mogą być powtarzane iteracyjnie. Główne kroki, które można wyodrębnić to [Fay1996b], [Man1996], [Man1997]:

1. zrozumienie dziedziny;
2. przygotowanie zbioru danych do przetworzenia;
3. odkrywanie zależności – eksploracja danych (ang. *data mining*);
4. przetworzenie znalezionych zależności;
5. wykorzystanie uzyskanych wyników w praktyce.

Zależnie od dziedziny zastosowania i celów analizy, każdy z tych etapów może przybierać różną postać i wykorzystywać rozmaite algorytmy. Zakłada się, że w tak opisanym procesie KDD wymagana jest interakcja z użytkownikiem i podejmowanie przez niego wielu istotnych decyzji [Fay1996b]. A zatem, patrząc z perspektywy maszynowego uczenia się, aplikacje KDD jako całość należy ulokować w obszarze systemów uczących się z nadzorem.

Metody *eksploracji danych* (ang. *data mining*, DM) są wykorzystywane w procesie KDD podczas etapu właściwego odkrywania zależności. Mimo iż same techniki DM nie są wystarczające dla działania rzeczywistych aplikacji KDD, stanowią one niezwykle ważne ogniwo całego procesu i w sposób bardzo znaczący wpływają na jego wydajność. Fakt ten sprawia, że od kilkunastu lat algorytmy DM są przedmiotem bardzo intensywnych badań. W odróżnieniu od procesu KDD jako całości, przynajmniej część algorytmów DM można uznać za techniki uczenia się bez nadzoru, ponieważ przetwarzają one zbiór wejściowy i znajdują w nim zależności bez żadnej informacji trenującej, pochodzącej od zewnętrznego nauczyciela.

1.4.1 Podział metod eksploracji danych

Zostało opracowanych bardzo wiele metod DM, które mogą być klasyfikowane według różnych kryteriów [Deo1997]. Fayyad, Piatetsky-Shapiro i Smyth [Fay1996b] zaproponowali uogólnione spojrzenie, według którego większość metod DM zawiera trzy podstawowe elementy:

- *model* – zawiera parametry, które mają być znalezione w danych; model ma określoną funkcję (np. klasyfikacja) i posiada pewną reprezentację (np. drzewo decyzyjne);
- *kryterium wyboru* (ang. *preference criterion*) – definiuje metodę oceny modelu lub jego parametrów;
- *algorytm poszukiwania* (ang. *search algorithm*) – specyfikuje sposób znajdowania modeli i ich parametrów, na podstawie danych wejściowych i kryterium wyboru.

We współczesnych pracach z zakresu DM można wyróżnić następujące *funkcje modelu*:

- *klasyfikacja* (ang. *classification*) – przydziela przykłady (ang. *data items*) do predefiniowanych klas;
- *regresja* (ang. *regression*) – przypisuje przykładom liczby rzeczywiste – wartości zmiennej predykcyjnej (ang. *prediction variable*);
- *grupowanie* (ang. *clustering*) – przyporządkowuje przykłady do klas (inaczej grup lub klastrów), które są automatycznie znajdowane w danych na podstawie miar podobieństwa lub rozkładu gęstości prawdopodobieństwa (w przeciwieństwie do klasyfikacji, w której klasy są predefiniowane);
- *podsumowywanie* (ang. *summarization*) – daje w wyniku zwięzły opis zbioru danych (np. zależności funkcyjne między zmiennymi);
- *modelowanie zależności* (ang. *dependency modeling*) – określa istotne zależności pomiędzy zmiennymi wraz z liczbowymi miarami ich siły;
- *analiza powiązań* (ang. *link analysis*) – znajduje relacje pomiędzy wieloma atrybutami opisującymi przykłady (np. reguły związku), spełniające wymagania statystycznej istotności i dokładności;
- *analiza sekwencji* (ang. *sequence analysis*) – pozwala na znalezienie zależności sekwencyjnych (np. podobnych serii czasowych), trendów i odchyleń.

Reprezentacja modelu ma duży wpływ na jego siłę semantyczną, elastyczność i możliwości zrozumienia przez człowieka. Do często stosowanych reprezentacji należą:

- *reguły* (ang. *rules*);
- *drzewa decyzyjne* (ang. *decision trees*);
- *modele liniowe* (ang. *linear models*);
- *modele nieliniowe* (ang. *nonlinear models*) – na przykład sieci neuronowe;
- *metody oparte na przykładach* (ang. *example-based methods*);
- *modele zależności probabilistycznych* (ang. *probabilistic dependency models*) – na przykład sieci Bayesa;
- *modele relacyjne* (ang. *relational attribute models*).

Kryterium wyboru (ang. *preference criterion*) – definiuje metodę oceny modelu lub jego parametrów i zazwyczaj ma postać pewnej jawnej, liczbowej funkcji (np. miary dopasowania modelu do danych), która jest umieszczona w algorytmie poszukiwania.

Algorytmy poszukiwania można podzielić na dwie kategorie: (i) poszukiwanie parametrów dla znanego modelu (często zbliżone do problemu optymalizacji) oraz (ii) poszukiwanie modelu wśród wielu dostępnych.

Przedstawiając powyższy podział metod DM na różne kategorie autorzy stwierdzają, iż nie istnieje jedna, najlepsza technika dla wszystkich zastosowań, lecz jej wybór powinien być uzależniony od wymogów konkretnej aplikacji.

Mannila [Man1997] zwrócił uwagę, że dużą część problemów DM można opisać jako poszukiwanie w danych interesujących i często występujących wzorców. Jeśli oznaczyć przez P klasę wzorców lub zdań opisujących własności zbioru danych d, wówczas ogólne zadanie algorytmu DM polega na znalezieniu zbioru:

$$PI(d, P) = \{p \in P: (p \text{ występuje wystarczająco często w } d) \wedge (p \text{ jest interesujące})\}.$$

Tak więc w ogólnym algorytmie wyznaczania $PI(d, P)$ najpierw znajdowane są wszystkie częste wzorce, a następnie spośród nich wybierane te, które są interesujące. Należy zwrócić uwagę, że o ile kryterium częstościowe może być określone za pomocą precyzyjnych miar liczbowych, o tyle rozstrzygnięcie, czy dany wzorzec jest interesujący, jest trudne do formalizacji i może wymagać podjęcia decyzji przez użytkownika w trakcie interakcji z systemem. Mannila [Man1997] proponuje poniższy, ogólny algorytm znajdowania częstych wzorców, który wykorzystuje pewną relację $<$, porządkującą wzorce w zbiorze P. Relacja $p < q$ oznacza, że jeśli wzorzec q jest częsty, to wzorzec p również jest częsty. Innymi słowy, znając częsty wzorzec p, w oparciu o niego można znaleźć nowy, częsty wzorzec q.

Algorytm FFP: *Znajdowanie wszystkich częstych wzorców*

Wejście: d – zbiór danych;
P – zbiór możliwych wzorców, na którym określona jest relacja $<$.

Wyjście: F – zbiór częstych wzorców.

1. $C := \{p \in P$: nie istnieje $q \in P$, takie że $q < p\}$. //początkowe wzorce ze zbioru P
2. Dopóki $C \neq \varnothing$ wykonuj kroki 3–6.
3. Dla każdego $p \in C$ wykonaj krok 4.
4. Wyznacz liczbę wystąpień wzorca p w danych d.
5. $F := F \cup \{p \in C$: p jest wystarczająco częsty w $d\}$.
6. $C := \{p \in P$: p jest potencjalnie częsty i zostały przeanalizowane wszystkie wzorce $q < p\}$.
7. Koniec dopóki.
8. Zwróć F.

Ogólna idea powyższego algorytmu może być wykorzystana do znajdowania reguł związku, powtarzających się epizodów w sekwencji zdarzeń i zależności funkcyjnych w relacjach.

1.4.2 Metody usuwania wartości nieznanych

Często spotykanym problemem w rzeczywistych zastosowaniach technik DM, są nieznane wartości atrybutów, które w naturalny sposób wynikają z niepewności i niepełności przetwarzanych danych. Ponieważ algorytmy DM z reguły nie pozwalają na bezpośrednie podawanie niekompletnych danych wejściowych, stosowane są różne metody ich eliminacji. Techniki te można podzielić na trzy główne grupy, omówione poniżej [Cic2000].

- *Przykłady trenujące, które zawierają nieznane wartości, są usuwane*. Jest to najprostsze i najszybsze rozwiązanie, które jednak może być stosowane w bardzo ograniczonym zakresie, gdyż z założenia prowadzi do utraty potencjalnie ważnych danych trenujących. Metoda ta może mieć jednak sens, jeśli w określonym przykładzie brakuje wartości bardzo wielu atrybutów (np. powyżej pewnego, ustalonego progu). Wówczas przykład taki należy całkowicie odrzucić, ponieważ i tak jego wartość informacyjna jest znikoma.
- *Znalezienie i wykorzystanie zależności atrybutów o nieznanych wartościach od atrybutów o wartościach znanych w dostępnym zbiorze trenującym*. Ten sposób można określić jako bardziej *inteligentny* od poprzedniego, ale możliwości jego zastosowania są również ograniczone do przypadku, w którym nieznane wartości dotyczą jedynie niewielkiego odsetka wszystkich przykładów trenujących. Jeśli natomiast wartości nieznane są często spotykane i równomiernie rozłożone w całym zbiorze trenującym (to znaczy dotyczą zarówno wielu przykładów, jak i wielu atrybutów), metoda ta może nie zdać egzaminu. Poza tym, stosując tę technikę, musimy zwrócić uwagę na wzrost złożoności obliczeniowej procesu KDD w związku z dodatkowymi przebiegami analizy danych wejściowych.
- *Przykład z wartościami nieznanymi jest zastępowany wieloma przykładami, które zawierają wszystkie możliwe wartości brakujących atrybutów*. Jeśli algorytm DM przyjmuje ułamkową charakterystykę (wagi) dla danych wejściowych, nowe przykłady można opatrzyć wartościami ułamkowymi, które wyrażają prawdopodobieństwo występowania określonej wartości na podstawie proporcji częstościowych, obliczonych w bieżącym zbiorze trenującym. To podejście również wymaga dodatkowego

przetwarzania danych wejściowych. Jeśli jednak algorytm DM nie pozwala na stosowanie wag dla poszczególnych przykładów, w celu wyeliminowania wartości nieznanych w danym przykładzie można na jego podstawie wygenerować nowe przykłady, odzwierciedlające wszystkie możliwe wartości nieznanego atrybutu, które w tym przypadku są traktowane jako jednakowo prawdopodobne. Złożoność obliczeniowa samego procesu generowania nowych przykładów jest wówczas stosunkowo niewielka (o ile liczba wartości nieznanych nie jest duża), może ona jednak prowadzić do poważnego powiększenia zbioru trenującego. (Jeden przykład z nieznanymi wartościami zastępowany jest potencjalnie wieloma innymi przykładami, wygenerowanymi na jego podstawie). To z kolei może powodować istotne spowolnienie właściwego przetwarzania przygotowanego zbioru trenującego przez algorytmy DM.

Widać zatem, że nie można wskazać jednej, zawsze najlepszej metody usuwania wartości nieznanych. Należy natomiast wybierać określone rozwiązanie dla konkretnej aplikacji i danych trenujących.

W dalszej części tego rozdziału opisane są algorytmy znajdowania reguł związku (ang. *association rules*), wykorzystywane w proponowanej metodzie APS.

1.5. Odkrywanie reguł związku

Znajdowanie *reguł związku* (ang. *association rules*) jest ważnym działem eksploracji danych, który jest silnie rozwijany od ponad dziesięciu lat. Na początku algorytmy te były stosowanie do analizy dużych baz komercyjnych, w których dane pochodziły z systemów sprzedaży stosowanych w supermarketach. Z czasem reguły związku nabrały ogólniejszego znaczenia i zaczęły być stosowane także w innych dziedzinach.

Reguły związku, zwane także *regułami asocjacyjnymi*, wyrażają zależności współwystępowania wartości atrybutów w badanej bazie danych – zbiorze przykładów [Cic2000]. Każda reguła związku zawiera dwie części – listy wartości atrybutów: *warunkujących* i *warunkowanych*, przy czym określony atrybut w danej regule może wystąpić tylko w jednej części. Reguła związku reprezentuje zatem informację, że jeśli występują dane wartości atrybutów warunkujących, to często w tym samym przykładzie występują także określone wartości atrybutów warunkowanych. Częstość współwystępowania wartości atrybutów jest wyrażana za pomocą odpowiednich miar statystycznych, które decydują o znaczeniu i wiarygodności reguły. W odniesieniu do maszynowego uczenia się, odkrywanie reguł związku można zaliczyć do metod uczenia się bez nadzoru [Has2001].

1.5.1 Model formalny

Opracowane zostały różne modele formalne reprezentacji reguł związku. Przedstawiony niżej opiera się na formalizacji, zaproponowanej przez prekursorów tej dziedziny: Agrawala, Imielinskiego, Swamiego i Srikanta [Agr1993], [Agr1994], a także na notacji podanej w pracy Goethalsa [Goe2002].

Niech $U = \{I_1, I_2, ..., I_n\}$ będzie zbiorem atrybutów, z których każdy ma dziedzinę $\{0, 1\}$. *Schematem transakcji na zbiorze U* nazywamy parę $S = (TID, I)$, gdzie *TID* jest zbiorem identyfikatorów transakcji, a I zbiorem atrybutów $I \subseteq U$.

Transakcja o schemacie S jest wektorem $s = (tid, i_1, i_2, ..., i_m)$, gdzie $tid \in TID$ oraz $i_k \in \{0, 1\}$ dla $k \in [1; m]$, jest wartością atrybutu I_k w transakcji s, co oznaczamy $i_k = s(I_k)$.

Mówimy, że zbiór $X \subseteq U$ jest *spełniony* przez transakcję s, co oznaczamy $s \vdash X$, jeżeli dla każdego atrybutu $x_j \in X$ zachodzi $s(x_j) = 1$. *Bazą transakcji B* nazywamy zbiór transakcji o schemacie S.

Częstością (ang. *frequency*) zbioru X w bazie transakcji B, co oznaczamy *freq* (X, B), nazywamy liczbę z przedziału [0; 1]:

$$freq(X, B) = \frac{card\{s : s \vdash X\}}{card\ B}.$$

Częstość zbioru atrybutów X ma intuicyjnie zrozumiałą interpretację probabilistyczną, reprezentuje ona bowiem estymację prawdopodobieństwa spełniania zbioru X przez losowo wybraną transakcję s ze zbioru B.

Zbiór $X \subseteq U$ nazywamy *częstym zbiorem atrybutów* (ang. *frequent itemset*), jeśli jego częstość w bazie B jest nie mniejsza, niż wartość pewnego przyjętego *progu minimalnego poparcia* $\sigma \in [0; 1]$.

Zbiór wszystkich częstych zbiorów atrybutów w bazie transakcji B, przy progu σ, oznaczamy następująco [Goe2002]:

$$F(B, \sigma) = \{X \subseteq U : freq(X, B) \geq \sigma\}.$$

W oparciu o te oznaczenia możemy zdefiniować problem znajdowania częstych zbiorów atrybutów.

Problem: *Znajdowanie częstych zbiorów atrybutów*

Dane są: zbiór atrybutów U, baza B transakcji o schemacie S oraz próg minimalnego poparcia σ.

Wyznaczyć: zbiór $F(B, \sigma)$.

Regułą związku nazywamy wyrażenie postaci $X \Rightarrow Y$, gdzie $X \subset U$, $Y \subset U$ oraz $X \cap Y = \emptyset$.

Poparciem (ang. *support*) reguły r: $X \Rightarrow Y$ w bazie transakcji B, co oznaczamy *sup* (r, B), nazywamy liczbę z przedziału [0; 1]:

$$sup(r, B) = freq(X \cup Y, B).$$

Poparcie reguły może być traktowane jako estymacja prawdopodobieństwa jednoczesnego spełniania zbioru X i Y przez losowo wybraną transakcję z bazy B [Has2001].

Reguła r: $X \Rightarrow Y$, jest nazywana *częstą regułą* w bazie transakcji B, jeśli zbiór $X \cup Y$ jest częstym zbiorem atrybutów w bazie B.

Pewnością (ang. *confidence*) reguły r: $X \Rightarrow Y$ w bazie transakcji B, co oznaczamy *con* (r, B), nazywamy liczbę z przedziału [0; 1]:

$$con(r, B) = \frac{freq(X \cup Y, B)}{freq(X, B)} = \frac{sup(r, B)}{freq(X, B)}.$$

Pewność reguły reprezentuje estymację prawdopodobieństwa warunkowego $Pr\ (Y \mid X)$: spełnienia zbioru Y przez losowo wybraną transakcję ze zbioru B, pod warunkiem spełnienia przez tę transakcję części warunkującej X [Has2001].

Reguła r: $X \Rightarrow Y$, jest nazywana *wiarygodną regułą* w bazie transakcji B, jeśli jej pewność jest nie mniejsza, niż wartość pewnego przyjętego *progu minimalnej pewności* $\gamma \in [0; 1]$.

Zbiór wszystkich częstych i wiarygodnych reguł w bazie B, określonych na zbiorze atrybutów U, przy progach minimalnego poparcia σ i pewności γ, oznaczamy następująco [Goe2002]:

$$R\ (B, \sigma, \gamma) = \{X \Rightarrow Y: X, Y \subset U \wedge X \cap Y = \varnothing \wedge X \cup Y \in F\ (B, \sigma) \wedge con\ (X \Rightarrow Y, B) \geq \gamma\}.$$

Stąd też możemy sformułować problem znajdowania reguł związku następująco.

Problem: *Znajdowanie reguł związku*

Dane są: zbiór atrybutów U, baza B transakcji o schemacie S oraz progi minimalnego poparcia σ i pewności γ.

Wyznaczyć: zbiór $R\ (B, \sigma, \gamma)$.

1.5.2 Metody znajdowania reguł związku

Powstało bardzo wiele prac, prezentujących rozwiązanie powyższego problemu znajdowania reguł związku lub problemów do niego zbliżonych. Osiągnięcia te mogą podzielone wielowymiarowo, według różnych kryteriów. Proponowany jest tutaj umowny podział na następujące grupy: (i) metody wsadowe, przetwarzające całą bazę transakcji; (ii) metody inkrementacyjne, które pozwalają na aktualizację bazy reguł po wystąpieniu zmian w bazie transakcji, bez konieczności ponawiania pełnej analizy tej bazy; (iii) metody ograniczania zbioru reguł, w tym metody wyboru reguł interesujących dla użytkownika; (iv) inne metody, nie należące do żadnej z poprzednich grup.

Metody wsadowe

Najbardziej znany algorytm rozwiązania problemu znajdowania reguł związku o nazwie Apriori, zaproponowany został przez Agrawala i Srikanta [Agr1994]. Algorytm ten składa się z dwóch głównych faz: (1) znajdowania częstych zbiorów atrybutów i (2) generowania reguł na podstawie częstych zbiorów. Realizacja pierwszego etapu jest w dużej mierze zbieżna z ogólnym algorytmem FFP, zaproponowanym przez Mannilę [Man1997] i przedstawionym w podrozdziale 1.4.1. Najpierw znajdowane są częste zbiory wśród pojedynczych atrybutów, a następnie zbiory te są rozszerzane do zbiorów zawierających dwa i więcej atrybutów. Zakłada się przy tym, że jeżeli zbiór atrybutów A zawiera się w innym zbiorze B, który jest częsty, to zbiór A również musi być częsty. W algorytmie FFP zależność tę reprezentuje relacja $<$, szeregującą wzorce w zbiorze P, natomiast w kategoriach częstościowych można ją wyrazić następująco [Has2001]: dla każdych, niepustych zbiorów A, $B \subseteq U$, jeśli $A \subseteq B$, to $freq(A) \geq freq(B)$. W drugim etapie algorytmu Apriori z każdego znalezionego zbioru częstego generowane są reguły (na zasadzie kombinatorycznej), dla których musi być obliczona tylko

pewność, ponieważ poparcie jest równe poparciu samego zbioru częstego. Dokładny opis algorytmu można znaleźć w pracy [Agr1994] oraz [Goe2002].

Shen i jego współpracownicy [ShS1999] zaproponowali algorytm znajdowania zbiorów częstych o nazwie MINER, wykorzystujący techniki grupowania atrybutów. W porównaniu z Apriori algorytm ten poważnie zmniejsza liczbę przebiegów przez dane i związanych z nimi operacji odczytu i zapisu. Autorzy opracowali także algorytm PAR_MINER, który jest wersją algorytmu MINER, przeznaczoną do obliczeń równoległych.

Protaziuk i Rybiński w pracy [Pro2003] poruszają zagadnienie odkrywania reguł związku w bazach transakcji, zawierających wartości nieznane. Autorzy zaproponowali algorytm DFSIT (ang. *Discovering Frequent itemSets in Incomplete Transaction*), który umożliwia znajdowanie reguł w bazie, zawierającej transakcje z nieznaną wartością co najwyżej jednego atrybutu. Algorytm ten jest oparty na algorytmie Apriori oraz miarach probabilistycznych: minimalnego i maksymalnego możliwego poparcia, minimalnej i maksymalnej możliwej pewności, szacowanego poparcia i szacowanej pewności zbioru atrybutów w badanej bazie transakcji. Autorzy wykazali, że ich rozwiązanie spełnia kryteria tzw. *uprawnionego podejścia do niekompletnych danych* (ang. *legitimate approach to data incompleteness*), zdefiniowanego przez Kryszkiewicz i Rybińskiego [Kry2000] (odsyłacz podany za [Pro2003]).

W większości metod znajdowania reguł związku stosowany jest pojedynczy próg minimalnego poparcia, wyznaczający zbiory częste. Tymczasem w rzeczywistych zastosowaniach dla każdego atrybutu mogą być określone inne kryteria oceny statystycznej wiarygodności. Nawiązując do tej koncepcji Lee, Hong i Lin [LeH2005] zaproponowali algorytm stanowiący modyfikację Apriori, który pozwala na zastosowanie różnych progów minimalnego poparcia dla poszczególnych atrybutów. Algorytm ten wykorzystuje miarę tzw. *maksymalnego ograniczenia* (ang. *maximum constraint*), definiowaną dla danego zbioru atrybutów jako maksimum wartości minimalnego poparcia, przypisanych podzbiorom tego zbioru.

Jednym z problemów, podczas realizacji algorytmów znajdowania zbiorów częstych, jest wydajne porównywanie identyfikatorów transakcji, które znajdują się na różnych listach. Gardarin, Pucheral i Wu [GaP1998] zaproponowali dwa algorytmy obliczania poparcia zbiorów atrybutów: N-BM i H-BM, w których porównywanie zbiorów identyfikatorów transakcji (wspierających określone zbiory atrybutów) opiera się na wektorach binarnych i operacji AND, w odróżnieniu od innych podejść, wykorzystujących łączenie posortowanych list. Autorzy wykazali metodami analitycznymi i eksperymentalnymi, że ich propozycja jest znacznie wydajniejsza od tradycyjnych technik, opartych na listach.

Algorytmy znajdowania reguł związku i inne metody DM są nieodłącznie związane z systemami baz danych, które przechowują potrzebne struktury (np. tabele analizowanych transakcji). W związku z tym część prac dotyczy współpracy metod DM z systemami bazodanowymi, optymalizacji procesorów zapytań i realizacji kluczowych operacji (np. odkrywania reguł) bezpośrednio na poziomie baz danych. Potencjalne korzyści z osadzenia procesu DM na poziomie systemu baz danych obejmują automatyczne zrównoleglanie obliczeń, łatwość programowania i wykorzystanie operatorów relacyjnych [Tho1998]. Meo, Psaila i Ceri [Meo1998] zaproponowali rozszerzenie języka zapytań SQL w postaci operatora `MINE RULE` do znajdowania reguł związku. Thomas i Sarawagi [Tho1998] opracowali szczegółowo różne etapy odkrywania reguł związku w postaci zapytań w języku SQL-92 oraz w języku SQL-OR z rozszerzeniami obiektowo-relacyjnymi (ang. *object-relational extensions*).

Autorzy wykorzystali uogólniony model pozwalający na reprezentację hierarchii reguł związku i wzorców sekwencyjnych. Nanopoulos i Manolopoulos [Nan2004] opracowali metodę odkrywania reguł związku, która dynamicznie dopasowuje przebieg algorytmu do zmieniającego się rozmiaru dostępnego bufora pamięci operacyjnej. Rozwiązanie to pozwala na uniknięcie zwiększonego dostępu do pamięci wirtualnej na dysku oraz zawieszania procesu odkrywania reguł, w sytuacji, gdy wyższy priorytet w systemie zarządzania bazą danych uzyskuje obsługa bieżących transakcji OLTP.

W ostatnich latach można zauważyć dużą popularyzację metod opartych na różnego rodzaju grafach. Zaki [Zak2000] zaproponował szereg grafowych algorytmów znajdowania częstych zbiorów: *Eclat*, *MaxEclat*, *Clique*, *MaxClique*, *TopDown*, *AprClique*. Stosowany jest w nich między innymi graf podzbiorów atrybutów, w którym wyodrębniane są kompletne podgrafy (takie, w których wszystkie wierzchołki są ze sobą połączone), zwane *klikami* (ang. *clique*). Coenen, Goulbourne i Leng [Coe2004] zaproponowali metodę znajdowania reguł związku, której pierwszym etapem jest pojedynczy przebieg przez dane testowe. W trakcie tego przebiegu wykonywane są obliczenia tzw. *częściowego poparcia* (ang. *partial support*) zbiorów atrybutów, których wyniki są zapisywane w strukturze *drzewa wyliczenia zbiorów* (ang. *set enumeration tree*). Autorzy opracowali algorytmy oparte na Apriori, pozwalające na przetwarzanie tej struktury w celu obliczania właściwych miar poparcia zbiorów.

W ostatnim czasie Tsay i Chiang [TsC2005] wprowadzili nową metodę odkrywania reguł o nazwie CBAR (ang. *clustered-based association rule*). W rozwiązaniu tym baza transakcji, po pojedynczym odczycie, jest dzielona k grup, z których każda zawiera transakcje o takiej samej długości (tzn. liczbie atrybutów). W ten sposób przy iteracyjnym rozszerzaniu rozmiaru kandydujących zbiorów częstych, badane są kolejno coraz mniejsze fragmenty początkowej bazy. Prowadzi do znacznej redukcji czasu analizy w porównaniu z algorytmem Apriori.

Metody inkrementacyjne

Inkrementacyjne metody znajdowania reguł związku cieszą się dużym zainteresowaniem, ponieważ w stosunku do metod wsadowych potencjalnie pozwalają one na poważne zmniejszenie złożoności obliczeniowej, dzięki ograniczeniu przebiegów przez dane, które już były przetwarzane.

Jeden z bardziej znanych, inkrementacyjnych algorytmów znajdowania reguł związku, o nazwie FUP_2, został zaproponowany przez Cheunga, Lee i Kao [Che1997] (odsyłacz za [LeS1998]). Algorytm ten wykorzystuje procedurę `apriori-gen` z algorytmu Apriori do generowania kandydujących zbiorów atrybutów na podstawie analizy całej, zaktualizowanej bazy transakcji. Dla każdego k w kolejnych iteracjach, zbiór kandydatów C_k dzielony jest na dwie partycje: P_k zawiera zbiory, które wystąpiły w poprzednim przebiegu odkrywania reguł (wyniki poprzedniej analizy są zapamiętywane), Q_k zawiera zaś nowe zbiory, których nie było poprzednio. Dla kandydatów z P_k znane są częstości z poprzedniego przebiegu odkrywania reguł, stąd przetworzone transakcje nie muszą być ponownie analizowane, wystarczy jedynie uwzględnić częstości w transakcjach zmodyfikowanych. Z kolei zbiór nowych kandydatów Q_k jest najpierw zawężany na podstawie częstości tych zbiorów w transakcjach dodanych Δ^+ i usuniętych Δ^- , oraz progu minimalnego poparcia. Przetwarzanie starej bazy transakcji w celu naliczenia nieznanych częstości wykonywane jest zatem tylko dla tych zbiorów kandydujących, które mogą być częste. Zmniejsza to liczbę koniecznych przebiegów analizy

danych, gwarantując jednocześnie wysoką precyzję wynikowego zbioru reguł. Ci sami autorzy, Lee, Cheung i Kao w pracy [LeS1998] zaproponowali inny algorytm inkrementacyjnego aktualizowania reguł związku o nazwie DELI (ang. *Difference Estimation for Large Itemsets*), w którym również wykorzystana jest funkcja `apriori_gen` z algorytmu Apriori. W algorytmie DELI użyto technik próbkowania (ang. *sampling*) do szacowania rozmiaru zmian, jakie zachodzą w bazie transakcji. Jeśli zmiany są niewielkie, dotychczasowe reguły są uznawane za wystarczająco dobre przybliżenie rzeczywistych reguł i są one pozostawiane bez modyfikacji. Aktualizacja reguł następuje dopiero po zgromadzeniu odpowiednio dużej liczby zmian w bazie danych. Rozpatrywane są tutaj zbiory: D (pierwotna baza transakcji), D^* (zbiór transakcji nie zmienionych po aktualizacji), Δ^+ (transakcje dodane do bazy) oraz Δ^- (transakcje usunięte z bazy). DELI można zakwalifikować jako algorytm przybliżony, gdyż dopuszcza on rozbieżność między zbiorem reguł, a aktualnym stanem bazy transakcji. Dzięki temu jednak w ogólnym przypadku wymaga on mniejszych nakładów obliczeniowych, niż FUP_2.

Tsai, Lee i Chen [Tsa1999] opracowali inkrementacyjną metodę znajdowania zbiorów częstych, zbliżoną do algorytmu FUP [Che1996] (odsyłacz za [Tsa1999]). Rozwiązanie to polega na przechowywaniu zbiorów, które w wyniku ostatniego przebiegu analizy zostały uznane jako potencjalnie częste, to znaczy ich częstość jest co prawda mniejsza od progu minimalnego poparcia (*min_sup*), ale mieści się w przedziale [*min_sup* – *t*; *min_sup*), dla pewnego przyjętego *stopnia tolerancji t*, takiego, że $0 < t < min\_sup$. Idea metody opiera się zatem na przewidywaniu, że dany zbiór atrybutów może stać się częsty po dokonaniu modyfikacji w bazie transakcji. W ten sposób, na podstawie analizy zbioru transakcji nie zmienionych, dodanych i usuniętych, można wyprowadzić nowy zbiór reguł przy zmniejszonej liczbie koniecznych przebiegów przez dane.

Wiele prac z zakresu metod inkrementacyjnych opiera się na reprezentacji zbiorów atrybutów i relacji między nimi w postaci grafu. Rozwiązania te często pozwalają na bardzo poważne ograniczenie liczby operacji wejścia i wyjścia w porównaniu z metodami niegrafowymi, niekiedy jednak są one ograniczone rozmiarem generowanego grafu. Metody dokładne, eliminujące powroty do raz przetworzonych transakcji wymagają zapisania w grafie informacji o częstości wszystkich możliwych podzbiorów. To wiąże się z potencjalnie bardzo dużym rozmiarem grafu. Z kolei metody ograniczające rozmiar grafu pozwalają na ograniczenie, ale nie na pełną eliminację ponownego przetwarzania transakcji.

Aumann, Feldman, Lipshtat i Manilla [Aum1999] opracowali algorytm *Borders*, który pozwala na inkrementacyjne aktualizowanie zbioru reguł związku, zgodnie ze zmianami wprowadzanymi w źródłowej bazie transakcji (takimi, jak wstawianie, modyfikowanie i usuwanie wierszy). Ważną cechą algorytmu jest nie analizowanie przetworzonych wcześniej danych, jeżeli zarejestrowane zmiany nie skutkują powstaniem nowych zbiorów częstych. Jeśli z kolei cała baza musi być analizowana, wówczas odpowiednio minimalizowana jest liczba przebiegów przez dane oraz rozmiar zbioru kandydatów, dla których obliczane jest poparcie. Algorytm *Borders* opiera się na tak zwanych *zbiorach granicznych* (ang. *border sets*), to znaczy takich zbiorach atrybutów, których wszystkie podzbiory właściwe są zbiorami częstymi. Dla wszystkich zbiorów granicznych i zbiorów częstych ustawicznie aktualizowane są miary częstościowe. Gdy pojawia się przyrost (zmiana) danych tylko on jest analizowany w celu aktualizacji poparcia (czyli częstości) wszystkich zbiorów granicznych i częstych. Zmodyfikowane miary odzwierciedlają poparcie w całym, zaktualizowanym zbiorze

transakcji. Przetwarzanie całej bazy jest konieczne tylko wtedy, gdy po aktualizacji, poparcie któregoś ze zbiorów granicznych osiąga próg minimalnego poparcia, przez co zbiór ten staje się zbiorem częstym. Również wtedy jednak koszt obliczania częstości jest zredukowany dzięki wykorzystaniu informacji o zbiorach granicznych.

Hu, Lu i Shi [HuL1999] opracowali algorytm inkrementacyjnego odkrywania reguł oparty na *kracie pojęciowej* (ang. *concept lattice*), który wymaga pojedynczego przebiegu przez dane testowe. Złożoność algorytmu, mierzona liczbą węzłów grafu, które muszą być wygenerowane, gdy dodawana jest nowa transakcja, jest rzędu $O(2^k |U|)$, gdzie k jest średnią liczba atrybutów transakcji, U zaś jest rozmiarem bazy. Można stwierdzić zatem, że duża liczba węzłów jest ceną za pojedynczy przebieg przez dane.

Aggarwal i Yu [Agg2001] zaproponowali algorytmy generowania reguł związku w *trybie bezpośrednim* (ang. *online generation*), to znaczy w odpowiedzi na zapytania użytkownika, w których zmieniają się parametry wejściowe (np. progi minimalnego poparcia i pewności). Podejście to jest oparte na *grafie sąsiedztwa* (ang. *adjacency lattice*), który jest tworzony dla częstych zbiorów atrybutów. Przy zmieniających się parametrach minimalnego poparcia i pewności nowy zbiór reguł może być zwracany bez konieczności ponownego przetwarzania zbioru testowego od początku. Ponadto możliwe jest zadawanie zapytań o reguły, które zawierają określone atrybuty w poprzedniku lub następniku.

Zhou i Ezeife [ZhE2001] zaproponowali inkrementacyjny algorytm MAAP (ang. *maintaining association rules with apriori property*), który opiera się na własności Apriori. Algorytm wykorzystuje zbiory częste wysokiego poziomu (to znaczy zawierających największe liczby atrybutów), uzyskane w wyniku poprzedniego przebiegu analizy, i w pierwszej fazie bada analogiczne zbiory atrybutów w zmienionych transakcjach. Z kolei część zbiorów niskiego poziomu jest obliczanych bez konieczności skanowania bazy na podstawie własności Apriori (każdy podzbiór zbioru częstego musi być również zbiorem częstym), co zmniejsza ogólną złożoność algorytmu.

Lee G., Lee K.L. i Chen [LeG2001] opracowali grafowy algorytm znajdowania zbiorów częstych DLG*, stanowiący udoskonaloną wersję algorytmu DLG, który zaproponowali Yen i Chen [Yen1996] (odsyłacz podany za [LeG2001]). Autorzy zaproponowali inkrementacyjny algorytm DUP aktualizacji częstych zbiorów, oparty na DLG*. Podobnie, jak w przypadku [LeS1998], rozpatrywane są w nim zależności między liczbą wystąpień danego zbioru atrybutów w zbiorach: D (oryginalna baza), d^+ (transakcje dodane do bazy), d^- (transakcje usunięte z bazy), rozmiarem tych zbiorów i progiem minimalnego poparcia. Zależności te pozwalają na ograniczenie liczby przebiegów przez dane.

Ezeife i Su [Eze2002] wprowadzili dwa inkrementacyjne algorytmy odkrywania reguł związku: *DB-Tree* oraz *PotFP-Tree*, oparte na *drzewach częstych wzorców* (ang. *frequent pattern tree*, *FP-tree*). Pierwszy z algorytmów zapisuje w formie drzewa *FP-tree* częstości wszystkich atrybutów w zbiorze testowym, dzięki czemu przy modyfikacji danych nie jest wymagane powtórne przetwarzanie całego zbioru, a jedynie analiza samych zmian. Ceną za to jest potencjalnie duży rozmiar drzewa *FP-tree* w stosunku do metod zapisujących tam jedynie częste wzorce. Drugi z proponowanych algorytmów, *PotFP-Tree* stanowi rozwiązanie pośrednie, w którym drzewo przechowuje częstości zbiorów aktualnie częstych oraz tych, dla których istnieje duże prawdopodobieństwo, że staną się częste po zmianie danych. Wyznacznikiem tego jest miara średniego poparcia, na którym jest oparta większość procesów

odkrywania reguł w pewnym okresie (ang. *watermark*). Zakładając pewną tolerancję można przyjąć, że wszystkie zbiory, które aktualnie nie są częste, ale ich częstość mieści się w przedziale [*t*; *średnie_poparcie*], dla pewnego *t*, są zbiorami potencjalnie częstymi. Badania eksperymentalne wskazują, że tak skonstruowany algorytm *PotFP-Tree* zmniejsza liczbę powrotów do uprzednio przetworzonych danych. Zauważmy, że jest to podejście bardzo podobne do algorytmu prezentowanego w pracy [Tsa1999].

Do metod inkrementacyjnych można zaliczyć także *aktywną eksplorację danych* (ang. *active data mining*), która została wprowadzona przez Agrawala i Psailę [Agr1995]. W podejściu tym dane testowe są przetwarzane ustawicznie przy zadanych parametrach częstościowych. Odkrywane reguły są dodawane do bazy reguł, dla tych zaś reguł, które już tam występują, aktualizowana jest historia parametrów statystycznych. Jeżeli historia zaczyna wykazywać określone trendy, które są definiowane tzw. *zapytaniami o kształt* (ang. *shape queries*), uruchamiane są procedury wyzwalane (ang. *triggers*), wykonujące określone akcje. Fong, Wong i Huang [Fon2003] zaproponowali metodę inkrementacyjnego aktualizowania zbioru reguł związku, która należy do metod aktywnej eksploracji danych. U podstaw metody leży *model metadanych* (ang. *frame metadata model*), składający się z czterech klas, które opisują: tabele faktów, atrybuty, metody i ograniczenia (ang. *constraints*). W oparciu o ten model autorzy zaproponowali algorytm, który pozwala na sterowaną zdarzeniami (ang. *event-driven*), ustawiczną aktualizację zbioru reguł na podstawie zmian, które zachodzą w źródłowej tabeli faktów. Metoda ta zakłada zapamiętywanie dla każdej reguły $X \Rightarrow Y$ częstości zarówno zbioru X, jak i zbioru $X \cup Y$.

Sung, Li, Tan i Ng [Sun2003] poruszają problem możliwości zastosowania zbioru reguł związku, uzyskanego w oparciu o określony zbiór danych, do opisu innej sytuacji, to znaczy środowiska, które może być charakteryzowane przez inne dane testowe. Autorzy zaproponowali model wpływu zmiany środowiska (sytuacji) na zbiór reguł, który je opisuje. W modelu tym rozpatrywane są tzw. *czynniki* (ang. *factors*), to znaczy atrybuty charakteryzujące środowisko, które wpływają na odkrywane reguły, choć nie są jawnie reprezentowane w danych testowych. Zmodyfikowany zbiór reguł dla nowej sytuacji wyprowadzany jest z istniejącego zbioru w oparciu o tzw. *kliki* (ang. *caucuses*) – grupy czynników wraz ich wartościami.

Au i Chan [AuC2005] opracowali metodę badania zmian w regułach związku, opartą na zbiorach rozmytych (ang. *fuzzy sets*). Reguły związku są odkrywane wybranym algorytmem (np. Apriori) w kolejnych partycjach bazy faktów. Reguły te następnie są przetwarzane algorytmem FID (rozszerzenie algorytmu ID3), dającym w wyniku rozmyte drzewo decyzyjne, które może być przekształcone do zbioru *meta-reguł* rozmytych (a więc reguł opisujących inne reguły). Meta-reguły mogą być wykorzystane do scharakteryzowania zmian, jakim ulegają reguły, a także do przewidywania ich przyszłych modyfikacji.

Metody ograniczania zbioru reguł

Jednym z problemów, dotyczących algorytmów odkrywania reguł związku takich, jak Apriori, jest zwracanie zbyt dużej ich liczby, określane niekiedy mianem *eksplozji reguł* (ang. *rule explosion*). Duża liczba reguł uniemożliwia ich zrozumienie i wykorzystanie na przykład w procesie podejmowania decyzji. Jednocześnie znacząca część tych reguł często i nie przedstawia wartości z punktu widzenia użytkownika. Powstało zatem wiele prac, których celem jest zniwelowanie tego efektu. Można je zgrubnie podzielić na dwie grupy, omawiane

kolejno poniżej: (i) metody ograniczania zbioru zwracanych reguł do podzbioru reguł najbardziej interesujących dla użytkownika, względem przyjętych kryteriów; (ii) metody oparte na tak zwanej *zwięzłej reprezentacji* (ang. *concise representation*) reguł lub zbiorów częstych.

Znajdowanie interesujących reguł

W jednej z wczesnych prac z pierwszej grupy metod Klementinen, Mannila, Ronkainen, Toivonen i Verkamo [Kle1994] wprowadzili formalizm *szablonów reguł* (ang. *rule templates*), służący do opisu struktury interesujących reguł związku. Szablon opisuje zbiór reguł poprzez listę atrybutów, które występują w poprzedniku i następniku reguły. Użytkownik może określać interesujące go reguły poprzez tzw. szablony *włączające* (ang. *inclusive templates*), a także ograniczać ich zbiór poprzez tzw. szablony *ograniczające* (ang. *restrictive templates*). Obie grupy szablonów reprezentują odpowiednio pozytywne i negatywne zainteresowania użytkownika. W oparciu o wprowadzoną metodę autorzy opracowali program do wizualizacji odkrywanych reguł, o nazwie *Rule Visualizer*.

Bayardo i Agrawal [Bay1999] zaproponowali ogólną metodę odkrywania najbardziej interesujących reguł związku z punktu widzenia różnych metryk (np. poparcia, pewności, wzrostu entropii). Autorzy wykazali, że najlepsze reguły, względem każdej z omawianych miar oceny, wyznaczane są przez granicę optymalnego poparcia i pewności. Granica ta jest definiowana przez relacje częściowego porządku $<_{sc}$ oraz $<_{s\neg c}$, na podstawie których powstają odpowiednio zbiory reguł *sc-optymalnych* (ang. *sc-optimal*) oraz *s¬c-optymalnych* (ang. *s¬c-optimal*).

Bayardo, Agrawal i Gunopulos [Bay2000] opracowali algorytm DENSE-MINER przeznaczony do odkrywania reguł związku w *gęstych zbiorach danych* (ang. *dense data sets*), to znaczy zbiorach, które charakteryzowane są przez następujące cechy: (i) duża liczba częstych zbiorów, (ii) silne korelacje pomiędzy wieloma atrybutami, (iii) duża liczba atrybutów w każdej transakcji (fakcie). Są to zatem zbiory odmienne od klasycznych baz koszyków (ang. *market-basket databases*), w których jest duża liczba możliwych atrybutów, lecz średnia liczba atrybutów w pojedynczej transakcji jest stosunkowo niewielka. Podczas przetwarzania gęstych zbiorów danych tradycyjnymi algorytmami, takimi jak Apriori, może pojawiać się wykładniczy wzrost złożoności obliczeniowej. Algorytm DENSE-MINER bezpośrednio korzysta z nakładanych przez użytkownika ograniczeń: minimalnego poparcia i pewności, oraz z dodatkowej, nowo wprowadzonej miary *minimalnego polepszenia* (ang. *minimum improvement*, *min_imp*), która pozwala na uzyskanie odpowiednio zwiększonej siły predykcyjnej odkrywanych reguł w stosunku do pochodnych reguł uproszczonych. Reguła uproszczona powstaje w wyniku usunięcia jednego lub więcej warunków z poprzednika pierwotnej reguły. W algorytmie DENSE-MINER odkrywane są tylko te reguły, których pewność jest o co najmniej *min_imp* większa, niż pewność każdej ich reguły uproszczonej. Dodatkowy próg pozwala zatem na ograniczenie problemu eksplozji reguł (ang. *rule explosion*). Proponowany algorytm został zweryfikowany na rzeczywistych zbiorach testowych: dotyczących spisu ludności (PUMS) oraz danych telekomunikacyjnych (CONNECT-4).

Lin, Alvarez i Ruiz w pracy [Lin2002] zwracają uwagę, że klasyczne algorytmy odkrywania reguł związku w szczególności nie są właściwe dla systemów rekomendujących (ang. *recommender systems*), ponieważ zwracają one dużo reguł nieinteresujących dla użytkownika, a ich ogólna liczba podlega trudnym do przewidzenia wahaniom, jako wynik różnych wartości minimalnego poparcia, które zawsze musi być podawane na początku, przed wykonaniem

analizy. Autorzy zaproponowali alternatywny algorytm ASARM (ang. *Adaptive-Support Association Rule Mining*), w którym zamiast wprowadzania progu minimalnego poparcia, podaje się zakres liczby reguł, które mają być znalezione. Na tej podstawie algorytm sam dopasowuje minimalne poparcie do zadanego przedziału. Ponadto algorytm uwzględnia próg minimalnej pewności reguł oraz ograniczenia semantyczne w postaci atrybutu, który ma się pojawić w następniku reguły.

Rastogi i Shim [Ras2002] omawiają zagadnienie *optymalizowanych reguł związku* (ang. *optimized association rules*), które pozwalają na znajdowanie zależności interesujących z punktu widzenia aplikacji marketingowych i reklamowych. Optymalizowana reguła związku ma ogólną postać: $(A_1 \in [l_1, u_1]) \wedge C_1 \Rightarrow C_2$, gdzie A_1 jest atrybutem liczbowym, l_1 i u_1 są zmiennymi (ang. *uninstantiated variables*), C_1 i C_2 są zaś koniunkcjami warunków postaci $A_1 = v_i$ lub $A_1 \in [l_i, u_i]$ (przy czym v_i, l_i, u_i są wartościami należącymi do dziedziny atrybutu A_i). Na przykład reguła „(*data* $\in [l_1, u_1]$) $\wedge$ *miasto_źródłowe* = Warszawa $\Rightarrow$ *kraj_docelowy* = Francja" może być wykorzystana przez firmę telekomunikacyjną do określenia w jakim okresie roku najwięcej rozmów telefonicznych do Francji jest wybieranych w Warszawie. Autorzy zaproponowali rozszerzenia modelu reguł optymalizowanych i algorytmy ich znajdowania.

Tsai i Chen w pracy [Tsa2004] wprowadzili szczególny przypadek problemu znajdowania reguł związku, w którym reguły są odkrywane na podstawie bazy transakcji oraz bazy klientów. Jako rozwiązanie problemu autorzy zaproponowali algorytm *LigCid* znajdowania częstych zbiorów oraz algorytm grafowy *Rule_Discovery*, służący do generowania reguł z uwzględnieniem zdefiniowanych przez użytkownika priorytetów, które dotyczą atrybutów warunkowych z bazy klientów. Wang, Perng, Ma i Yu [WgP2005] opracowali architekturę HIFI (ang. *heterogeneous items in frequent itemsets*), umożliwiającą znajdowanie reguł związku zgodnie z zadanymi przez użytkownika wzorcami atrybutów i innymi ograniczeniami.

Zwięzła reprezentacja reguł i zbiorów częstych

Druga grupa metod ograniczania liczby reguł związana jest z poszukiwaniem bezstratnej, *zwięzłej reprezentacji* (ang. *concise representation*) reguł lub zbiorów częstych, na podstawie której można wyprowadzić wszystkie reguły pochodne.

Chen, Wei, Liu i Wets [ChW2002] wykazali formalnie, że złożone reguły związku mogą wyprowadzane ze zbioru tzw. *prostych reguł związku* (ang. *simple association rules*, SAR), których następniki są złożone z pojedynczych atrybutów, a zatem mają postać: $A_1 \wedge A_2 \wedge ... \wedge A_n \Rightarrow A_j$. Ponieważ zaś zbiór SAR jest stosunkowo mały w porównaniu ze zbiorami reguł zwracanymi przez klasyczne algorytmy, prezentowane podejście pozwala na znaczącą redukcję złożoności obliczeniowej i eliminację problemu eksplozji reguł.

Jedną z metod zwięzłej reprezentacji jest zawężenie zbioru wszystkich reguł do *nienadmiarowego zbioru reguł* (ang. *non-redundant rules*). Z kolei Li, Shen i Topor [LiJ2004] wprowadzili reprezentację w postaci *informacyjnego zbioru reguł* (ang. *informative rule set*), który jest mniejszy od zbioru nienadmiarowego. Autorzy zaproponowali algorytm bezpośredniego tworzenia zbioru informacyjnego bez konieczności generowania wszystkich zbiorów częstych. Zarówno liczba reguł jak i czas ich znajdowania przez ten algorytm są mniejsze od analogicznych wyników dla Apriori.

Kryszkiewicz, Rybiński i Gajek w pracy [Kry2004] przedstawili szczegółowo szereg rodzajów bezstratnej, zwięzłej reprezentacji zbiorów częstych. Z punktu widzenia zastosowań praktycznych najbardziej użyteczna jest reprezentacja oparta na *zbiorach domkniętych* (ang. *closed itemsets*) i *generatorach zbiorów częstych* (ang. *frequent generators*). Tymczasem jednak istnieją inne rodzaje reprezentacji, na przykład oparte na *uogólnionych generatorach bezdysjunkcyjnych* (ang. *generalized disjunction-free generators*, GDFGR), które są bardziej zwięzłe i wydajne. Autorzy zaproponowali zatem algorytmy przejścia między różnymi reprezentacjami zbiorów częstych, bez konieczności ponownego przetwarzania źródłowego zbioru transakcji.

W jednej z nowszych prac Pasquier i jego współpracownicy [Pas2005] zaproponowali zwięzłą reprezentację reguł związku w postaci tak zwanych *reguł min-max* o minimalnych poprzednikach i maksymalnych następnikach. Autorzy przedstawili algorytmy generujące zbiór reguł *min-max* oraz odtwarzające pełny zbiór reguł pochodnych bez odwoływania się do bazy transakcji.

Inne metody znajdowania reguł związku

Klasyczny model reguł związku, wprowadzony przez Agrawala, Imielinskiego, Swamiego i Srikanta [Agr1993], [Agr1994] oparty był na koszykach z zakupami (ang. *market basket data*) i uwzględniał wyłącznie pozytywne przypadki łącznego występowania poszczególnych artykułów lub ich brak. W ostatnich latach zaproponowane zostały różne rozszerzenia i modyfikacje tego modelu. Na przykład Silverstein, Brin i Motwani [Sil1998] wprowadzili rozszerzenie, w którym reguły związku są uogólnione do *reguł zależności* (ang. *dependence rules*), wyznaczanych statystycznym testem χ^2 na podstawie wystąpień zarówno pozytywnych, jak i negatywnych. Autorzy zaproponowali także odpowiedni algorytm znajdowania reguł zależności.

Aumann i Lindell [Aum2003] zaproponowali model *ilościowych reguł związku* (ang. *quantitative association rules*), to znaczy reguł zawierających w poprzedniku lub następniku atrybuty z wartościami liczbowymi. W przeciwieństwie do innych podejść, autorzy nie stosują technik dyskretyzacji w celu zamiany atrybutów liczbowych na wyliczeniowe (ang. *categorical*), lecz badają statystyczny rozkład ciągłych wartości atrybutów za pomocą takich miar, jak wartość oczekiwana i wariancja. Ilościowa reguła związku ma ogólną postać: „*podzbiór_populacji* ⇒ *średnia wartość dla tego podzbioru*", przy czym średnia wartość dla podzbioru powinna się znacząco różnić od odpowiedniej średniej dla całej bazy danych. Założenie to gwarantuje, że reguła jest potencjalnie interesująca, gdyż wskazuje na własność nietypową, nierozpowszechnioną. Przykładami reguł ilościowych są: „*płeć = kobieta* i *wykształcenie = wyższe* ⇒ *średnia stawka wynagrodzenia* = 7.90 EUR / *godzinę*" albo „*czas kształcenia* = [14, 18] *lat* ⇒ *średnia stawka wynagrodzenia* = 11,64 EUR / *godzinę*". Autorzy zdefiniowali dwa główne typy reguł: (i) reguły *atrybuty wyliczeniowe* ⇒ *atrybuty ilościowe*, oraz (ii) *atrybuty ilościowe* ⇒ *atrybuty ilościowe*. Zaproponowane zostały także odpowiednie algorytmy znajdowania reguły z tych kategorii. Prezentowane podejście zostało zweryfikowane eksperymentalnie.

Harms i Deogun [Har2004] opracowali metodę MOWCATL odkrywania *sekwencyjnych reguł związku* (ang. *sequential rules*), które na podstawie wystąpienia danego wzorca

pozwalają przewidywać wystąpienie innego wzorca po pewnym czasie. Pojawiły się także propozycje połączenia modelu reguł związku z paradygmatem zbiorów rozmytych, w wyniku czego powstają *rozmyte reguły związku* (ang. *fuzzy association rules*) [LeL2004], [Kay2005a]. Do innych propozycji należą: model reguł z wieloma poziomami pojęciowymi (ang. *multi-level association rules*) [Lis2004] oraz wprowadzony ostatnio przez Raucha [Rau2005] nowy model logiczny reguł związku, które reprezentują uogólnioną relację między dwoma atrybutami Boolowskimi.

Pojawiają się także równoległe algorytmy przetwarzania reguł związku, oparte na odpowiednim partycjonowaniu źródłowej bazy transakcji. Obszerny przegląd metod z tego zakresu został przedstawiony w pracy Zakiego [Zak1999], natomiast przykładem nowych osiągnięć jest algorytm D-ARM, wprowadzony przez Schustera, Wolffa i Trocka [Scu2005].

2. Pozyskiwanie wiedzy przez agenta

W niniejszym rozdziale omawiane jest zagadnienie pozyskiwania wiedzy przez systemy agenckie. Rozdział rozpoczyna ogólny przegląd technologii agenckich oraz metod reprezentacji wiedzy agentów. Po wprowadzeniu podstawowych pojęć, w dalszej części tekstu zestawione są techniki uczenia się agentów.

2.1. Technologie agenckie

Prowadzone od lat 1980 badania nad *technologiami agenckimi* (ang. *agent technologies*, *intelligent agents, software agents*) mają na celu stworzenie systemów informatycznych, które by w inteligentny sposób wspomagały człowieka w interakcji z komputerem oraz w rozwiązywaniu problemów rozproszonych lub złożonych obliczeniowo [Bra1997], [Nwa1996], [Nwa1999], [Oli1999], [Par1998]. Wybrane dziedziny zastosowań agentów obejmują [Jen1998]:

- wyszukiwanie, filtrowanie i rekomendowanie informacji w sieci WWW [Ber1999], [ChC2002], [Eli2003], [Ene2005], [Etz1994], [God2004], [Kim2002], [Klu2001], [Kno1997], [Men2000], [Muk2003], [NeS1997], [Paz1997], [Paz2002], [Piv2002], [Yao2002];
- rynek elektroniczny, wspomaganie zakupów przez Internet, automatyczne negocjacje, wywiad gospodarczy [Car2000], [Dec1997], [Kep2002], [LiT2000], [LiY2003], [Men2002], [Per1997], [Rip2000];
- wspomaganie elektronicznej obsługi urzędów państwowych i systemów prawnych [Hee1997];
- planowanie spotkań [Ces1999];
- planowanie podróży, telematyka [Ndu1998];
- zarządzanie komunikacją miejską [Ezz2005];
- zarządzanie sieciami komputerowymi i telekomunikacyjnymi [App2000], [Dav1998], [Mav2004];
- zarządzanie instalacjami przemysłowymi i produkcją w przedsiębiorstwie [Lgo2004], [Par1998], [Wag2003];

- zarządzanie dystrybucją energii elektrycznej [Par1998];
- kontrolę ruchu lotniczego [Rao1995].

Ze względu na duży zakres i zróżnicowanie prac nad agentami, nie ma zgodności co do samej definicji *agenta*. Aby uniknąć niejednoznaczności przyjmiemy poniższą definicję, zaproponowaną przez Jenningsa i Wooldridge'a, która trafnie oddaje istotę systemu agenckiego[1].

Agent jest to system komputerowy, który jest osadzony w pewnym środowisku i jest zdolny do podjęcia autonomicznego, elastycznego działania, zgodnie z jego celami projektowymi [Jen1997], [Woo1995a], [Woo1999c].

Badania nad rozwojem narzędzi formalnych i technicznych, umożliwiających realizację systemów agenckich są bardzo rozległe i zróżnicowane. Od lat 1990 prace te zaczęły osiągać poziom języków i architektur programowania agentów [Sho1993], [Woo1997b] oraz praktycznych metodologii tworzenia systemów agenckich [Del2001], [GaL2004], [Lan1997], [Lie1998], [Woo1997a], [Woo1999a], [Woo1999b], opisywanych niekiedy odrębną nazwą *inżynierii agenckiej* (ang. *agent-oriented engineering*) [Mue1997]. Ponieważ jest to dyscyplina stosunkowo młoda, mimo dużego zaangażowania środowiska naukowego oraz znaczących inwestycji dużych firm, ciągle brakuje jednolitych, dominujących standardów w obszarze platform i architektur agenckich, metodologii projektowania oraz specjalizowanych narzędzi programistycznych.

2.2. Reprezentacja wiedzy agenta

Zgodnie z przytoczoną definicją, głównym postulatem formułowanym w odniesieniu do systemu agenckiego jest zdolność autonomicznego, celowego i elastycznego działania, zgodnie ze stawianymi przed nim zadaniami. Spełnienie tych założeń wymaga wyposażenia agenta w odpowiedni aparat poznawczy, struktury przechowywania wiedzy, mechanizmy wnioskowania i wykonywania akcji w otoczeniu. Poziom wiedzy jest podstawowym elementem agenta, który decyduje o jego własnościach i sposobie działania.

2.2.1 Pojęcie wiedzy

Wiedza jest jednym z pojęć bardzo podstawowych, stąd też jej zdefiniowanie przysparza szczególnych trudności. Meystel i Albus [Mey2002] wskazują na ważną relację między: *danymi* (ang. *data*), stanowiącymi pewien zapis fizyczny, *informacją* (ang. *information*), która powstaje w wyniku dodania do danych pewnego elementu opisowego, oraz *wiedzą* (ang. *knowledge*), która jest strukturą o pewnym znaczeniu, zbudowaną z wielu jednostek informacji. Na tej podstawie Meystel i Albus zaproponowali poniższą definicję wiedzy.

Wiedza jest systemem wzorców, (...) *opartych na jednostkach informacji, które pochodzą z doświadczenia. Wzorce te są uogólnione, etykietowane i zorganizowane w sieć relacyjną o pewnej architekturze. Wiedza implikuje zdolność efektywnego wykorzystywania jej przez posiadacza do wnioskowania i podejmowania decyzji*.

([Mey2002], str. 109, tłumaczenie wolne)

[1] W niniejszej rozprawie terminy *agent*, *program agencki* oraz *system agencki* są używane zamiennie i oznaczają to samo. W niektórych pracach *system agencki* jest traktowany jako termin szerszy od *agenta* i odnoszony jest do *systemów wieloagenckich*.

Dalej, autorzy sformułowali szereg dodatkowych definicji, przydatnych do opisu wiedzy, jej własności i przetwarzania [Mey2002].

Wzorzec (ang. *pattern*) *to rozróżnialny system połączonych składników i cech, oparty na zaobserwowanych lub zamierzanych relacjach między jego częściami.* ([Mey2002], str. 109)

Stan (sytuacja) jest zbiorem parametrów i zmiennych, które pozwalają na obserwowanie, rejestrowanie lub sterowanie obiektem lub zbiorem obiektów. Zbiór ten jest opatrzony momentem czasowym, w którym został on zaobserwowany lub zmierzony.
([Mey2002], str. 110)

Doświadczenie jest zapisem zdarzenia (zmiany stanu), lub grupy zdarzeń, zaobserwowanych w przeszłości (...). ([Mey2002], str. 110)

Doświadczenie składa się między innymi z poniższych składników:

- *stan początkowy* – sytuacja przed obserwowanymi zmianami;
- *stan końcowy* – sytuacja po zajściu obserwowanych zmian;
- *akcje* – zbiór zmian stanów wraz z przyczynami ich zajścia.

Meystel i Albus [Mey2002] podają także szereg funkcji, w które powinien być wyposażony inteligentny system z reprezentacją wiedzy, między innymi:

- aktualizacja wiedzy;
- wydajne przechowywanie wiedzy;
- udoskonalanie wiedzy (np. poprzez wnioskowanie);
- wyszukiwanie i zwracanie wiedzy;
- kompresja wiedzy poprzez kodowanie lub uogólnianie.

Są to ważne wyznaczniki dla metody pozyskiwania wiedzy, której zaprojektowanie jest głównym celem niniejszej pracy doktorskiej.

2.2.2 Formalizacja wiedzy agenta

Opracowanych zostało bardzo wiele modeli formalizujących wiedzę agentów. Są to prace silnie zróżnicowane obejmujące różne podejścia i koncepcje. Na tym tle jednym z wyróżniających się nurtów są modele opisujące wiedzę i działanie agentów, których strona formalna oparta jest na języku logiki matematycznej. Część tych modeli czerpie inspirację z ogólnych rozważań filozoficznych, część zaś opiera się na stosunkowo prostych, intuicyjnych założeniach, określanych niekiedy mianem *psychologii ludowej* (ang. *folk psychology*). Mimo to, wiele proponowanych modeli stanowi istotny wkład w tłumaczenie postulatów teoretycznych, dotyczących agentów, na język, który pozwala na ich techniczną realizację. Staranne opracowanie od strony formalnej, wraz z zapewnieniem niesprzeczności, zupełności i rozstrzygalności logicznych systemów aksjomatycznych, pozwala na ich przeniesienie na poziom architektur, środowisk implementacyjnych i w końcu – konkretnych systemów agenckich. Poniżej omawiane są wybrane, znaczące prace z tej dziedziny. Dla części z przytoczonych prac teoretycznych podane są także rozwiązania techniczne, które do nich nawiązują lub wręcz są na nich bezpośrednio oparte.

2.2.3 Semantyka światów możliwych

Fagin, Halpern, Moses i Vardi [Fag1995] opracowali bardzo dogłębny przegląd modeli opartych na *semantyce światów możliwych* (ang. *possible world semantics*). W koncepcji tej zakłada się, że w przypadku, gdy agent nie posiada pełnej wiedzy o stanie świata, buduje on zbiór *światów możliwych*, to znaczy wszystkich możliwych stanów, które są niesprzeczne z jego obecną wiedzą. O tym, które światy możliwe są dopuszczalne w danym stanie, decyduje tak zwana *relacja dostępności* (ang. *accessibility relation*). Do formalizacji opisanej tutaj idei na poziomie syntaktycznym wykorzystywany jest język logiki modalnej [Fag1995], [Haj1996]. Na poziomie semantycznym model oparty jest *na strukturze Kripke'go* [Kri1963], złożonej ze zbioru stanów, interpretacji (funkcji wartościującej prawdziwość formuł zdaniowych w każdym stanie) i binarnej relacji dostępności. Model taki może być rozbudowywany dla przypadku wielu agentów, *wiedzy wspólnej* (ang. *common knowledge*) i *wiedzy rozproszonej* (ang. *distributed knowledge*). Własności logicznych modeli wiedzy agentów są weryfikowane metodami algorytmicznymi albo (częściej) metodami dowodzenia twierdzeń. W tym drugim przypadku wyznacza się *aksjomaty* (ang. *axioms*), a więc formuły logiczne, które są zawsze prawdziwe w danym modelu. Istnieje szereg standardowych, powszechnie rozpoznawanych systemów aksjomatycznych, które odpowiadają modelom wiedzy o określonych własnościach, na przykład: *K*, *S*4 (znany także jako *KT*4), *S*5 (oznaczany także przez *KT*45). Doskonały wykład, dotyczący tych modeli wraz z ich licznymi odmianami i rozszerzeniami, jest zawarty w pracy [Fag1995].

Cohen i Levesque [Coh1990] zaproponowali model, którego głównym elementem jest formalizacja intencji agenta w oparciu o teorię Bratmana i założenie *racjonalnej równowagi* (ang. *rational balance*) [Brt1987] (odsyłacz za [Coh1990]). Założenie to ma służyć osiągnięciu, zgodnego z racjonalnym działaniem, złotego środka pomiędzy przesadną, *fanatyczną* konsekwencją agenta w dążeniu do obranego celu, a zbyt łatwym jego porzucaniem. Model oparty jest na wielomodalnej logice I rzędu, w której operatory modalne dotyczą zarówno czasu, jak i przekonań agentów. Semantyka powyższej logiki opiera się na semantyce ewoluujących światów możliwych [Haj1996]. Autorzy sformalizowali szereg własności intencjonalnego działania agenta, takich jak: kompetencja, cel i intencja. Do modelu Cohena i Levesque [Coh1990] nawiązuje architektura agencka BDI (ang. *Belief Desire Intention*) o nazwie JAM, opracowana przez Hubera [Hub1999].

Singh [Sin1992] przeprowadził szczegółową, krytyczną analizę teorii Cohena i Levesque, wskazując na słabości niektórych jej elementów, między innymi niewłaściwą formalizację kompetencji agenta i zdolności osiągnięcia celu. Aby wyeliminować te wady Singh [Sin1991] zaproponował odmienny model oparty na logice czasu rozgałęzionego CTL* [Eme1990], rozszerzony o operatory modalne dotyczące przekonań, intencji i akcji wykonywanych przez agenta. Ponieważ w modelu tym duży nacisk położony jest na ograniczoność agenta: pod względem zdolności wnioskowania, percepcji i wykonywania akcji, Singh wprowadza dodatkowo funkcje: prawdopodobieństwa zaistnienia określonego scenariusza (*Prob*) oraz użyteczności (*Utility*) i kosztu akcji wykonywanej przez agenta (*Cost*). Omawiany tutaj model Singha został przez niego obszernie rozwinięty w późniejszych pracach, między innymi [Sin1995a], [Sin1995b], [Sin1998].

Rao i Georgeff [Rao1991] opracowali model częściowo zgodny z propozycją Cohena i Levesque [Coh1990], nawiązujący do teorii intencji według Bratmana, a w warstwie

formalnej oparty na logice CTL* [Eme1990] i temporalnej strukturze *drzewa czasu* (ang. *time tree*). Główny wkład autorów polega na sformalizowaniu różnych strategii i własności intencjonalnego działania agenta. Warto zwrócić uwagę, że jest to jedna z nielicznych, znaczących prac teoretycznych, które miały także kontynuację techniczną. W oparciu o powyższy model formalny oraz architekturę agencką BDI o nazwie dMARS [DIn1998], Rao i Georgeff opracowali wieloagencki system OASIS, stanowiący uproszczoną wersję systemu PRS (ang. *Procedural Reasoning System*) [Ing1992], służący do kontroli ruchu lotniczego, który był próbnie wdrożony na lotnisku w Sydney [Rao1995]. Semantyczna ekspresja niektórych elementów modelu formalnego została celowo osłabiona, aby umożliwić praktyczną realizację systemu.

Mimo starannego opracowania teoretycznego, większość modeli opartych na klasycznej koncepcji światów możliwych, jest trudnych do bezpośredniego przeniesienia na poziom techniczny. Podstawowym powodem jest ukryte w nich założenie *logicznej wszechwiedzy* (ang. *logical omniscence*) agenta, zgodnie z którym do zbioru przekonań agenta należą nie tylko formuły jawnie zapisane, ale także wszystkie ich logiczne konsekwencje [Fag1995], [Hua1991]. Mimo postępu techniki obliczeniowej, założenie to ciągle nie jest możliwe do zrealizowania w rzeczywistych systemach ze względu na zbyt dużą złożoność. Do prób przezwyciężenia problemu wszechwiedzy agenta, należy koncepcja *wiedzy jawnej* (ang. *explicit knowledge*) i *niejawnej* (ang. *implicit knowledge*), wprowadzona przez Levesque [Lev1984]. Zgodnie z jej założeniami, przekonania (wiedza) agenta są podzielone na dwa podzbiory: *jawne* (opisywane operatorem B), to znaczy zdania i formuły rzeczywiście zapisane w bazie wiedzy oraz *niejawne* (reprezentowane operatorem L) – zdania i formuły, które mogą być wywnioskowane przez agenta na podstawie wiedzy jawnej. To rozróżnienie pozwala na zmniejszenie złożoności obliczeniowej problemu rozstrzygalności, czy dane zdanie zawiera się logicznie w innym zdaniu (z niewielomianowej, ang. *NP-complete* do wielomianowej rzędu $O(nm)$, gdzie n i m są rozmiarami badanych zdań).

Inną propozycją rozwiązania problemu wszechwiedzy agenta jest *logika świadomości* (ang. *logic of awareness*) wprowadzona przez Fagina i Halperna [Fag1988] (odsyłacz za [Fag1995]) i rozwijana przez Huang i Kwast [Hua1991]. W logice tej występuje dodatkowy modalny operator *świadomości* $A_i\varphi$, którego interpretacja mówi, iż *agent i jest świadomy formuły* φ, to znaczy może stwierdzić lub obliczyć (w przypadku baz wiedzy) jej prawdziwość.

Wooldridge [Woo1995b] zaproponował logikę L_B, przeznaczoną do modelowania wnioskujących agentów o ograniczonych zasobach (ang. *resource-bounded reasoners*), która również stanowi próbę rozwiązania problemu wszechwiedzy agenta. Autor wprowadził *model przekonania* (ang. *belief model*) jako strukturę reprezentującą przekonania agenta, której ważną część stanowi *relacja rozszerzania przekonań* (ang. *belief extension relation*).

Van der Hoek, Van Linder i Meyer [Van1997], [Van1999] stworzyli bardzo rozbudowany formalizm wiedzy i działania agentów, w którym rozszerzyli klasyczny zestaw operatorów modalnych, między innymi o operatory: *zdolności* (ang. *abilities*), *okazji* (ang. *opportunities*), *zobowiązania* (ang. *commitment*). Autorzy poświęcili także sporo uwagi zagadnieniu utrwalania przekonań agenta na podstawie obserwacji.

Katarzyniak [Kat1999] zaproponował oryginalny model, w którym wiedza agenta jest reprezentowana i przetwarzana z punktu widzenia samego agenta. Podejście to bardzo różni się od, powszechnie przyjętego w innych modelach, widzenia systemu z perspektywy zewnętrznego i wszechwiedzącego obserwatora. Autor wprowadził oddzielne operatory

konieczności (ang. *necessity*) i możliwości (ang. *possibility*) dla przeszłości, teraźniejszości i przyszłości. Na modelu tym oparte są dalsze prace, dotyczące między innymi procesu *utrwalania przekonań* (ang. *grounding belief formulas*) przez agenta na podstawie jego percepcji [Kat2002], [Kat2003], [Pie2003].

Jednym z nowszych modeli wiedzy i zachowania agentów jest logika ADL (ang. *agent dynamic logic*) wprowadzona przez Schmidt, Tishkovsky'ego i Hustadta [Sch2004]. Rozbudowany przegląd logicznych modeli wiedzy, która zmienia się w czasie, zawarty jest w pracy Hajnicz [Haj1996]. Omówienie modeli [Coh1990], [Rao1991], [Sin1991], traktowanych jako przykłady formalizacji paradygmatu BDI [Geo1999], można znaleźć w pracy [Dud2000].

Daniłowicz, Nguyen i Jankowski [Dan2002] przedstawili szereg metod opisu stanu wiedzy agentów w systemie wieloagenckim. Autorzy zdefiniowali miary spójności wiedzy wielu agentów oraz kryteria wyboru takiej reprezentacji wiedzy, która jest najbardziej zbliżona do reprezentacji stanów wiedzy poszczególnych agentów. Metody i algorytmy, prezentowane w pracy, są oparte na ogólnych metodach wyboru konsensusu [Ngu2002].

2.2.4 Reprezentacja wiedzy niepewnej

Agent osadzony w rzeczywistym, dynamicznym środowisku często styka się z wiedzą niepewną lub niepełną [Bac1996], [Bac1999]. Źródłami niepewności wiedzy są między innymi [Dud2001]:

- *percepcja* – aparat odbierania bodźców z otoczenia (ang. *sensors*), w który wyposażony jest agent, ma określoną precyzję, stąd też wiedza pozyskiwana w ten sposób może być zaszumiona (ang. *noisy*), niekompletna lub niedokładna [Bac1999];
- *informacja od innych agentów* – może być nie w pełni wiarygodna [Lia2000], albo źle zrozumiana z powodu rozbieżności ontologicznych;
- *indukcja wiedzy* – agent może przetwarzać zbiór dostępnych, licznych obserwacji i znajdować na ich podstawie uogólnione wnioski (np. wzorce, reguły), których wiarygodność statystyczna odzwierciedla stopień ich niepewności [Bac1996];
- *wnioskowanie przeprowadzane na wiedzy niepewnej* – zawsze daje w wyniku wiedzę niepewną, niezależnie od zastosowanych reguł wnioskowania.

Z tego względu reprezentacja wiedzy niepewnej i niepełnej odgrywa ważną rolę w modelowaniu agentów. Opisywane wyżej modele w bardzo ograniczony sposób pozwalają na reprezentację wiedzy tego typu – często tylko za pomocą operatorów *możliwości* (ang. *possibility*) i *konieczności* (ang. *necessity*). W wielu aplikacjach jest to niewystarczające. Poniżej zestawione są wybrane prace, które rozszerzają możliwości reprezentacji wiedzy niepewnej.

Van der Hoek i Meyer [Van1991a], [Van1991b] opracowali logikę epistemiczną $Gr(S5)$ ze *stopniowanymi modalnościami* (ang. *graded modalities*), opartą na rachunku zdań i rozszerzającą standardową strukturę Kripke'go w systemie aksjomatycznym $S5$. W modelu tym autorzy wyróżniają *wiedzę absolutną* (ang. *absolute knowledge*), obejmującą zdania, które są traktowana jako prawdziwe tylko wtedy, gdy nie ma od nich żadnych wyjątków. Oprócz tej wiedzy, można jednak wyrażać także prawdziwość zdania φ, gdy jest co najwyżej n wyjątków

od φ. Autorzy wskazują, że ich logika może być wykorzystywana w systemach wspomagania decyzji, stanowiąc pomost pomiędzy klasyczną logiką epistemiczną i metodami ilościowymi (w tym nawet probabilistycznymi), które są stosowane w sztucznej inteligencji.

Nieco podobną koncepcję zaprezentowali Demolombe i Liau [Dem2001] proponując *logikę stopniowanego zaufania i przekonań* (ang. *logic of graded trust and belief*) jako formalną podstawę modelowania agentów. Logika ta umożliwia reprezentowanie wiarygodności przekonań agenta za pomocą dyskretnych *stopni zaufania* (ang. *levels of trust*). Intencją autorów było umożliwienie gradacji zaufania agenta do różnych źródeł informacji (w tym innych agentów) w systemie wieloagenckim.

Zaproponowane zostały także modele oparte na liczbowych miarach niepewności, takich jak prawdopodobieństwo (wyznaczane zgodnie z różnymi teoriami, np. według teorii Bayesa), *funkcje przekonania* (ang. *belief functions*) w teorii Dempstera-Shafera; miary *możliwości* (ang. *possibility measures*), które są stosowane na przykład w logice rozmytej, wprowadzonej przez Zadeha [Bol1991], [Wal1996].

Fagin i Halpern [Fag1994] zaproponowali, oparty na rachunku zdań (ang. *propositional logic*), probabilistyczny model wiedzy, będący rozszerzeniem klasycznej struktury Kripke'go [Kri1963] o reprezentację prawdopodobieństwa. Autorzy omówili własności modelu i podali jego kompletną aksjomatyzację. Podobną logikę opisuje Halpern w [Hal1998]. Z kolei we wcześniejszej pracy [Hal1990] ten sam autor wprowadził logikę probabilistyczną opartą na logice pierwszego rzędu, która ma większą siłę ekspresji od zwykłego rachunku zdań. Halpern zaproponował tam trzy rodzaje struktur: pierwszy do reprezentacji prawdopodobieństwa dotyczącego dziedziny (ang. *probability on the domain*), drugi do wyrażania stopni pewności (ang. *degrees of belief*) opartych na prawdopodobieństwie światów możliwych, trzeci zaś – będący połączeniem obu poprzednich. Autor zaproponował kompletną aksjomatyzację dla wszystkich trzech rodzajów struktur i przeanalizował ich własności, wskazując na niepełną rozstrzygalność. Kontynuacja tego nurtu zawarta jest między innymi w pracach [Aba1994], [Fag1994]. Bacchus [Bac1990] zaproponował, opartą na logice pierwszego rzędu, logikę probabilistyczną *Lp*, podając dla niej kompletną aksjomatyzację. Logika ta jest podobna do omawianej wyżej logiki Halperna [1990].

Halpern i Pucella [Hal2002] wprowadzili, opartą na rachunku zdań, *logikę wnioskowania o oczekiwaniach* (ang. *logic of expectations*), której semantyka jest zależna od zastosowanej miary niepewności. W szczególności autorzy podali pełną aksjomatyzację tej logiki dla miar: prawdopodobieństwa, funkcji przekonania (ang. *belief functions*) i miary możliwości (ang. *possibility measure*).

Milch i Koller [Mil2000] zaproponowali *epistemiczną logikę probabilistyczną* (ang. *Probabilistic Epistemic Logic*, PEL), będącą szczególnym przypadkiem logiki [Fag1994]. Umożliwia ona modelowanie niepewności przekonań agentów przy założeniu *doskonałej pamięci* (ang. *perfect recall assumption*), które oznacza, że agent nie zapomina zapisanych obserwacji. Jako reprezentację dla logiki PEL autorzy wykorzystali *sieci Bayesa* (ang. *Bayesian networks*).

2.3. Uczenie się systemu agenckiego

Dalsza część niniejszego rozdziału jest poświęcona maszynowemu uczeniu się w kontekście systemów agenckich i zawiera przegląd, analizę i ocenę współczesnych osiągnięć w tej dziedzinie.

Techniki uczenia się w systemach agenckich są w ostatnich latach przedmiotem dużego zainteresowania i licznych prac, prowadzonych w wielu ośrodkach naukowych. Tak intensywny rozwój tej dziedziny w naturalny sposób wynika z postulatów stawianych przed systemem agenckim, przede wszystkim autonomiczności i elastyczności jego działania. Ta druga cecha jest wiązana z pojęciem systemu adaptacyjnego, a więc zdolnego do samodzielnego dostosowywania się do zmieniających się warunków w otoczeniu tak, aby jak najefektywniej realizować stawiane przed nim cele. Według Maes agent jest adaptacyjny, jeśli wraz z rosnącym doświadczeniem potrafi on udoskonalać swoje działanie, czyli polepszać zdolność osiągania swoich celów [Mae1994b].

Prace z dziedziny uczących się agentów są zróżnicowane i mogą być uporządkowane według różnych kryteriów. Podstawowy podział obejmuje dwie grupy, omawiane w dalszej części tekstu:

1) *metody uczenia się pojedynczego agenta* – stanowią punkt wyjścia dla działania adaptacyjnego indywidualnego agenta oraz jakiegokolwiek bardziej złożonego procesu uczenia się grupowego; nacisk położony jest na samą technikę pozyskiwania i przetwarzania wiedzy oraz jej wykorzystania w procesie decyzyjnym agenta;
2) *metody uczenia się systemów wieloagenckich* – dotyczą adaptacyjnego zachowania agenta, które uwzględnia istnienie innych agentów; zaliczane są tutaj metody *silnego* i *słabego* uczenia się systemów wieloagenckich.

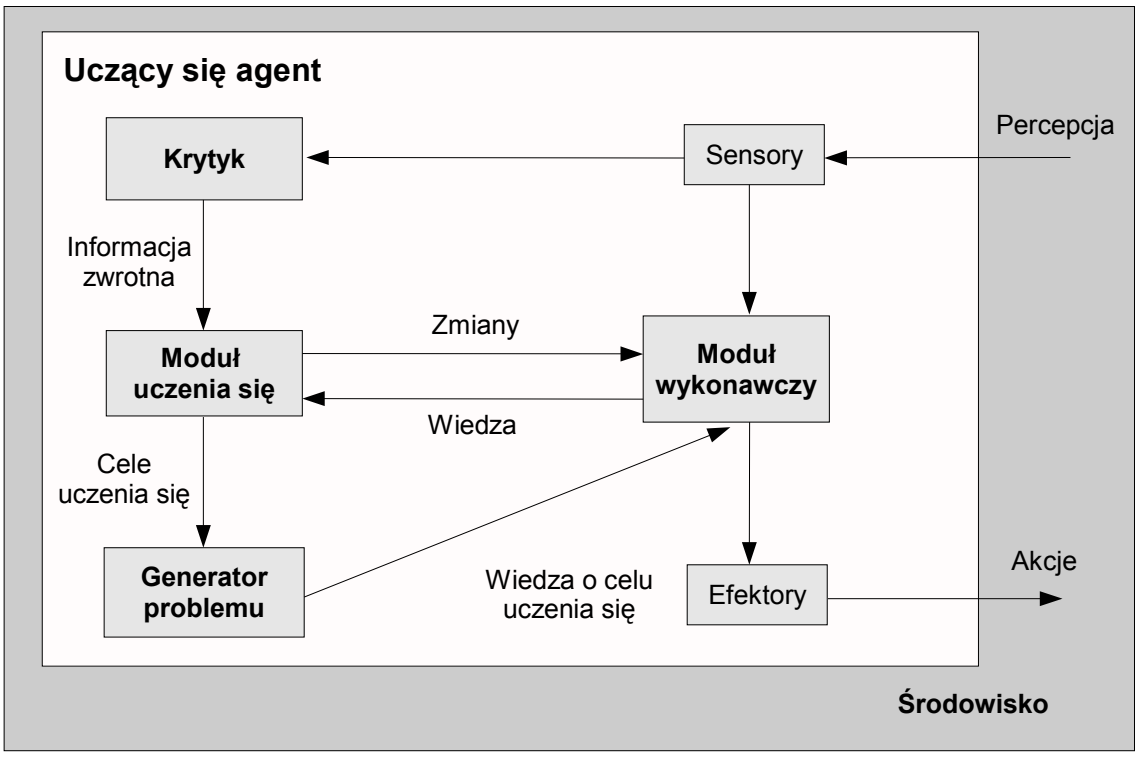
2.4. Uczenie się pojedynczego agenta

W tym przypadku, określanym także mianem *uczenia się scentralizowanego* (ang. *centralized learning*) lub *izolowanego* (ang. *isolated learning*) [Sen1999],[Wei1996], proces uczenia się ma charakter indywidualny i dotyczy tylko określonego agenta. Interakcja z innymi agentami jest możliwa, ale nie stanowi ona warunku koniecznego do tego, aby agent mógł się uczyć.

Russell i Norvig [Rus1995] opisali ogólną architekturę uczącego się agenta, przedstawioną na Rys. 2.1. Według tego schematu agent składa się z czterech głównych modułów [Rus1995], [Gue2003].

- *Moduł wykonawczy* (ang. *performance element*) – odpowiada za przetwarzanie informacji uzyskiwanej ze środowiska zewnętrznego za pośrednictwem sensorów i wybór akcji, które mają być wykonywane na tym środowisku przez efektory. Moduł ten stanowi zbiór wszystkich funkcji agenta z wyjątkiem uczenia się i może być zbudowany na wiele sposobów, zależnie od konkretnej architektury agenckiej. W szczególności może on zawierać: (i) wiedzę ogólną o dziedzinie; (ii) informację o świecie zewnętrznym i jego dynamice; (iii) funkcję użyteczności, która wartościuje stany otoczenia w kontekście celów agenta; (iv) przyporządkowanie akcji do warunków w bieżącym stanie środowiska oraz (v) informację o możliwych skutkach wykonania poszczególnych akcji.

- *Krytyk* (ang. *critic*) – dostarcza modułowi uczenia się informacji trenującej oraz oceny efektywności dotychczasowego działania agenta. W przypadku uczenia się z nadzorem (ang. *supervised learning*) ocena ta pochodzi o nauczyciela. Jeśli natomiast agent uczy się bez nadzoru (ang. *unsupervised learning*), ocena działania jest możliwa na podstawie ustalonej miary wydajności, która zależy od konkretnej dziedziny zastosowania i jest ustalana przez projektanta systemu.
- *Moduł uczenia się* (ang. *learning element*) – na podstawie wiedzy o module wykonawczym oraz informacji zwrotnej od krytyka, dokonuje on w module wykonawczym zmian, które potencjalnie mogą udoskonalić działanie agenta. Modyfikacje te dotyczą przede wszystkim procedury wyboru akcji na podstawie warunków otoczenia. Przy projektowaniu modułu uczenia się należy wyodrębnić przeznaczone do zmian elementy modułu wykonawczego, ich reprezentację oraz sposób dokonywania modyfikacji. Trzeba także uwzględnić wiedzę ogólną, posiadaną przez agenta już w momencie implementacji (ang. *prior knowledge*), zdefiniować postać danych trenujących z modułu wykonawczego oraz informacji zwrotnej od krytyka.
- *Generator problemu* (ang. *problem generator*) – odpowiada za strategię eksploracji (ang. *exploration strategy*), czyli sugerowanie akcji w poszukiwaniu nowych doświadczeń, które mogą poprawiać działanie agenta w perspektywie długoterminowej, mimo iż w danej chwili są nieoptymalne względem posiadanej przez agenta wiedzy.



Rys. 2.1. Ogólna architektura uczącego się agenta (na podstawie [Rus1995], str. 526).

Warto zwrócić uwagę, że duża część dotychczasowych, ogólnych prac z zakresu maszynowego uczenia się nie bierze pod uwagę osadzenia algorytmu uczenia się w systemie agenckim, to znaczy nie uwzględnia autonomicznego sterowania procesem, współpracy modułu uczenia się z innymi modułami agenta i jego interakcji z konkretnym środowiskiem zewnętrznym [Kaz2001]. Tymczasem problemy te są ważne przy próbie zastosowania technik uczenia się

w systemie agenckim. Kazakov i Kudenko [Kaz2001] podają szereg kwestii, które muszą być wówczas rozstrzygnięte:

- zdefiniowanie celu uczenia się;
- określenie źródła i charakteru danych trenujących;
- wybór techniki uczenia się;
- sposób integracji metody maszynowego uczenia się z architekturą agencką.

2.4.1 Cel uczenia się

Podany przez Kazakova i Kudenkę [Kaz2001], cel uczenia się agenta, jest bardzo zbliżony do omawianego w poprzednim rozdziale, ogólnego celu systemu uczącego się. Jeśli przyjąć, że agent jest obiektem, który przy pomocy sensorów pobiera dane z otoczenia i w oparciu o te dane oraz wewnętrzny proces wnioskowania wykonuje akcje na otoczeniu, wówczas można określić cel uczenia się agenta jako znalezienie procedury decyzyjnej, która umożliwia wybór odpowiednich akcji.

Z kolei Maes [Mae1994b] definiuje problem uczenia się agenta na podstawie doświadczenia w następujący, silniejszy sposób. Dany jest agent z: (i) zbiorem dostępnych akcji (lub tzw. modułów kompetencji), (ii) danymi sensorycznymi, odbieranymi z otoczenia oraz (iii) wieloma celami (zmiennymi w czasie). Należy opracować sposób udoskonalania wyboru akcji przez agenta w oparciu o doświadczenie, tak, aby agent lepiej dążył do celów [Mae1994b]. Omawiane tutaj *polepszenie* osiągania celów może być różnie określone w zależności od dziedziny zastosowania agenta i może oznaczać na przykład zmniejszanie średniego kosztu osiągnięcia celu lub zwiększanie skumulowanej nagrody (w przypadku uczenia się ze wzmocnieniem). A zatem Maes wymaga, aby uczenie się agenta nie tylko pozwalało na znalezienie procedury decyzyjnej, lecz aby procedura ta była stopniowo ulepszana.

Zdaniem Maes każdy model uczenia się autonomicznego agenta powinien spełniać poniższe wymagania [Mae1994b].

- *Uczenie się powinno być inkrementacyjne* – nie powinien występować podział na fazę uczenia się i fazę działania.
- *Uczenie się powinno być zorientowane na wiedzę relewantną do celów agenta*, gdyż w rzeczywistych, złożonych środowiskach agent nie ma możliwości przyswajania wszystkich dostępnych informacji.
- *Model uczenia się powinien obsługiwać wiedzę niepewną i niepełną*, na przykład szumy i zakłócenia, zdarzenia zachodzące z pewnym prawdopodobieństwem, błędne odczyty danych pochodzących z percepcji.
- *Uczenie się powinno przebiegać bez nadzoru*, aby agent mógł działać autonomicznie.
- *Przekazywanie agentowi pewnej wiedzy wbudowanej* – powinno być zapewnione, aby nie musiał on uczyć się wszystkiego od początku.

Dodatkowo Gaines [Gai1997], wyrażając wiedzę w kategoriach zdolności wykonania pewnego zadania, proponuje, aby tak pojęta kompetencja agenta miała własności monotoniczności. Oznacza to, że jeśli agent potrafi wykonać pewne zadanie, to będzie on zawsze w stanie

wykonać to zadanie. Zdolności agenta nie mogą maleć, a jedynie wzrastać lub pozostawać na stałym poziomie.

Warto zwrócić uwagę w przytoczonych pracach, że w odniesieniu do systemu agenckiego uczenie się w ogólnym przypadku nie jest definiowane jako klasyczny problem optymalizacji procedury decyzyjnej agenta. Często mówi się natomiast o „polepszaniu" procesu decyzyjnego, o metodach przybliżonych i heurystycznych. Taki stan rzeczy wynika ze stawianych przed agentami wysokich wymagań, dotyczących ich poziomu autonomiczności i osadzenia w dynamicznym środowisku, które sprawiają, że klasyczna optymalizacja często nie może być zastosowana ze względu na zbyt dużą złożoność obliczeniową.

2.4.2 Dane trenujące

Należy określić, jakie fakty opisujące otoczenie agenta i jego interakcję z nim, będą podstawą uczenia się. Warto tutaj podkreślić wymóg selektywności danych dostarczanych agentowi. W rzeczywistych zastosowaniach bardzo rzadko do algorytmów uczących przekazuje się wszystkie informacje, które system jest w stanie zarejestrować lub pozyskać z innych źródeł. Zgodnie z omówionym wcześniej postulatem Maes [Mae1994b], agent nie powinien uczyć się wszystkiego, lecz tylko tego, co może być przydatne z punktu widzenia jego celów. Pierwszym ograniczeniem, uzasadniającym ten wymóg, jest złożoność obliczeniowa uczenia się, która może sprawiać, iż agent jest w stanie przetworzyć tylko część dostępnych danych tak, aby nie zakłócało to płynności jego normalnego działania. Po drugie, agent w trakcie swojego działania jest w stanie wykorzystać ograniczoną ilość wiedzy (np. ze względu na czas przeszukiwania struktur bazy wiedzy).

Możemy wyróżnić trzy rodzaje źródeł informacji trenującej dla agenta [Kaz2001].

- *Zewnętrzny nauczyciel* (ang. *external teacher*) – źródło zewnętrzne w stosunku do agenta, które dostarcza mu ciąg przykładów trenujących tak, jak to zostało opisane w poprzednim rozdziale, przy omawianiu uczenia się z nadzorem.
- *Informacja trenująca pochodząca z otoczenia* (ang. *environmental feedback*) – jest to sytuacja charakterystyczna dla uczenia się poprzez eksperymentowanie. Agent wykonuje akcję na otoczeniu i odbiera z niego informację, która wskazuje na odniesioną w ten sposób korzyść lub stratę. Informacja ta może być zdefiniowana jako *użyteczność* (ang. *utility*) określonego stanu otoczenia, który został zaobserwowany przez agenta. Jedną z metod charakterystycznych dla tego źródła informacji trenującej jest, omawiane we wcześniejszym rozdziale, uczenie się ze wzmocnieniem, które, przypomnijmy, zostało tam zakwalifikowane jako uczenie się bez nadzoru (mimo iż niekiedy jest traktowane jako uczenie się z nadzorem, choć nie bezpośrednim).
- *Wewnętrzna ocena agenta* (ang. *internal agent bias*) – funkcja pozwalająca na wartościowanie pozyskiwanej wiedzy (np. wzorców i zależności w danych rejestrowanych z otoczenia) bez jakiejkolwiek bezpośredniej, zewnętrznej informacji trenującej. Funkcja ta może być różnie definiowana w zależności od dziedziny zastosowania. Kazakov i Kudenko określają ten model jako *uczenie się bez nadzoru* (ang. *unsupervised learning*), stwierdzając przy tym, iż niewiele jest prac z tego zakresu [Kaz2001]. Sytuacja ta nie powinna dziwić, skoro, jak to już wcześniej zostało wyjaśnione, w przypadku metod uczenia się bez nadzoru trudno jest określić wyraźną miarę oceny jakości uczenia się systemu (tutaj – wewnętrzną funkcję oceny).

Poniżej przedstawiona jest analiza uczenia się z nadzorem i bez nadzoru w kontekście systemu agenckiego. Ocena przydatności obu grup technik uczenia się w architekturze agenckiej przeprowadzona jest przede wszystkim pod kątem zapewnienia autonomiczności, która jest podstawową cechą wymaganą od systemu agenckiego i odróżniającą go od innych systemów informatycznych.

Metody uczenia się z nadzorem cieszą się dużą popularnością, ponieważ pozwalają na osiągnięcie wysokiej dokładności i szybkości procesu uczenia się. Dodatkowo, dla tej grupy algorytmów istnieją wyraźne miary oceny jakości uczenia się. Uczenie się z nadzorem zakłada dostępność zewnętrznego w stosunku do systemu uczącego się źródła informacji trenującej, czyli nauczyciela. Wymóg autonomiczności agenta wyklucza możliwość bezpośredniego manipulowania na jego wiedzy (co może mieć miejsce wyłącznie w fazie implementacji), stąd też informacja trenująca może być przekazywana agentowi tylko poprzez komunikację z nim. W rzeczywistych zastosowaniach możemy wyodrębnić następujące źródła informacja trenującej dla autonomicznego agenta.

- *Projektant*. Na etapie tworzenia lub wdrażania systemu agenckiego projektant przeprowadza sekwencję uczenia systemu agenckiego, uzyskując stan jego wiedzy pożądany w konkretnym środowisku, po czym przestaje wpływać na działanie agenta.
- *Użytkownik końcowy*. Już po wdrożeniu systemu agenckiego w określonym środowisku, agent może zwracać się do użytkownika o dostarczenie pewnej informacji trenującej, na przykład o ocenę ostatniej akcji agenta. Tego typu pytania pojawiają się co pewien czas i są przedzielone okresami całkowicie autonomicznego działania agenta, kiedy korzysta on z dostępnej wiedzy lub doskonali ją bez interakcji z użytkownikiem.
- *Inny agent*. Jest to możliwe w systemie wieloagenckim (ang. *multi-agent system*, MAS), w którym poszczególne agenty[2] są zdolne do komunikowania się i wymiany wiedzy między sobą.

Pierwsza możliwość, a więc dostarczanie informacji trenującej przez projektanta lub wdrożeniowca, ma duże znaczenie na etapie instalowania systemu agenckiego w określonym środowisku. Nie może być ona jednak traktowana jako mechanizm uczenia się podczas normalnej, autonomicznej pracy agenta.

Uzyskiwanie informacji trenującej od użytkownika jest możliwe jedynie w pewnej klasie zastosowań, to znaczy tam, gdzie może zachodzić interakcja z użytkownikiem. Są to przede wszystkim programy pełniące rolę *osobistych asystentów* (ang. *personal assistants*), na przykład w takich dziedzinach, jak: wyszukiwanie i filtrowanie informacji, dokonywanie zakupów przez Internet lub udział w elektronicznych aukcjach. Dodatkowym warunkiem jest zgoda użytkownika na udzielanie informacji agentowi. W ogólnym przypadku agent nie powinien wymuszać na użytkowniku działań, które, mimo iż mogą być wykorzystane do optymalizacji systemu, nie są absolutnie konieczne do jego funkcjonowania, a w perspektywie krótkoterminowej mogą wręcz spowalniać interakcję systemu z użytkownikiem. Przypuśćmy, że rola agenta polega na rekomendowaniu użytkownikowi stron WWW w oparciu o jego zainteresowania, które zostały rozpoznane na podstawie oceny stron przeglądanych do tej pory.

[2] Autor niniejszej pracy, zgodnie z własną intuicją językową, stosuje *agenty* jako liczbę mnogą słowa *agent*, w odniesieniu do programów agenckich (ang. *software agents*), w przeciwieństwie zaś do agentów – ludzi. Jest to podobna odmiana, jak w przypadku słowa *pilot*: mówimy *ci piloci* (o ludziach sterujących samolotami), ale *te piloty* (o urządzeniach RTV).

Podczas wyświetlania danej strony agent może poprosić użytkownika o ocenę stopnia jej relewancji, ale użytkownik ma prawo nie udzielać odpowiedzi, na przykład dlatego, że w danym momencie się śpieszy i chce tylko szybko przejrzeć wybraną stronę. W tej sytuacji, właśnie ze względu na prawie całkowity brak możliwości wymuszania przez agenta na użytkowniku udzielania informacji, uczenie się z nauczycielem – użytkownikiem może być zastosowane w mocno ograniczonym zakresie. A zatem wykorzystanie tego typu metod uczenia się z nadzorem w rzeczywistych aplikacjach jako głównego mechanizmu adaptacyjnego, może okazać się nieefektywne, gdyż duża część potencjalnie ważnej informacji trenującej jest wtedy poza jego zasięgiem.

Pozostało nam rozważenie możliwości wykorzystania przez agenta informacji trenującej, która pochodzi od innego agenta. Możemy mieć do czynienia z dwoma głównymi sposobami uczenia się w systemie wieloagenckim [Wei1996]: (i) *silne* (ang. *strong multi-agent learning*) – wzajemne dzielenie się wiedzą i doświadczeniem przez agenty, które mają wspólny cel uczenia się oraz (ii) *słabe* (ang. *weak multi-agent learning*) – indywidualne uczenie się poszczególnych agentów, na które ma wpływ zachowanie innych agentów, obserwowane we wspólnym środowisku. Łatwo można stąd wywnioskować, że tylko silne uczenie się możemy uznać za uczenie się z nadzorem, gdzie agent posiadający większą wiedzę lub umiejętności przekazuje je innemu agentowi w formie informacji trenującej. W przypadku słabego uczenia się systemu wieloagenckiego, poszczególne agenty nie prowadzą ze sobą komunikacji po to, aby bezpośrednio przekazywać sobie nawzajem swoją wiedzę ze względu na wspólne rozwiązywanie pewnego problemu, lecz wyłącznie po to, aby zrealizować własne cele. Agenty zatem uczą się samodzielnie, bez pomocy innych agentów, natomiast ewentualna poprawa globalnej wydajności całego systemu wieloagenckiego ma co najwyżej charakter *zjawiskowy* (ang. *emergent*) – może pojawić się w systemie, ale w sposób nie do końca możliwy do przewidzenia. Uczenie się w systemie wieloagenckim jest omówione w dalszej części pracy, natomiast tutaj chcemy rozstrzygnąć możliwość wykorzystania agenta jako nauczyciela dla innego agenta, który uczy się z nadzorem. Tego typu rozwiązanie ma sens tylko wtedy, gdy agent – nauczyciel posiada wiedzę przydatną dla agenta uczącego się, której agent – uczeń nie posiada. Oto przykłady takiej sytuacji.

- *Specjalizacja agentów*. Poszczególne agenty mają odrębne funkcje lub obszary działania (np. przeszukują różne grupy domen w sieci Internet) i specjalizują się w nich, przez co różna jest także ich wiedza pochodząca z interakcji z fragmentami środowiska. Jeśli jednak zachodzi potrzeba, aby przez pewien czas dany agent działał poza swoim normalnym obszarem, może on, zamiast rozpoznawania tej części środowiska od zera, pozyskać wiedzę od wyspecjalizowanego tam agenta.
- *Różnice w doświadczeniu*. Wszystkie agenty działają w tym samym środowisku, ale między sobą różnią się one doświadczeniem – na przykład długością działania („życia agenta") lub intensywnością dotychczasowej interakcji z otoczeniem. W tej sytuacji również agenty o większym doświadczeniu mogą przekazywać swoją wiedzę innym agentom – mniej doświadczonym.

Naturalną metodą uczenia się agenta w powyższych sytuacjach jest omawiane wcześniej uczenie się na podstawie zapytań [Cic2000], w którym agent – nauczyciel występuje w roli wyroczni odpowiadającej na jawne pytania agenta – ucznia. Przy odpowiedniej konfiguracji systemu wieloagenckiego takie dzielenie się wiedzą nie tylko może poprawiać lokalnie

efektywność poszczególnych agentów, ale też całego systemu. Zwróćmy jednak uwagę, że dzieje się tak tylko wtedy, gdy zachodzą znaczące różnice pomiędzy wiedzą poszczególnych agentów. Samo współdzielenie wiedzy nie gwarantuje natomiast dostarczenia nowej wiedzy do systemu wieloagenckiego, gdyż musi to nastąpić w fazie projektowej lub podczas indywidualnego uczenia się poszczególnych agentów. (Agenty same muszą skądś zdobyć wiedzę, aby móc się nią dzielić z innymi agentami). Widzimy zatem, że niezależnie od pozytywnych i pożądanych skutków silnego uczenia się systemu wieloagenckiego, nie może być ono jedynym mechanizmem uczenia się agentów, lecz powinno raczej pełnić rolę wspomagającą i optymalizującą działanie całego systemu.

Podsumowując powyższą analizę możemy stwierdzić, że:

- w systemie agenckim istnieje możliwość zastosowania uczenia się z nadzorem, jednak informacja trenująca musi być przekazywana agentowi na drodze komunikacji, a nie bezpośredniej manipulacji na jego wiedzy;
- źródłem informacji trenującej dla agenta może być: projektant, użytkownik końcowy lub inny agent;
- we wszystkich przeanalizowanych przypadkach uczenie się z nadzorem może prowadzić do pożądanych skutków – to znaczy do poprawy indywidualnego działania agenta lub nawet całego systemu wieloagenckiego;
- w systemie autonomicznym, którym jest agent, uczenie się z nadzorem może być zastosowane w ograniczonym zakresie i dlatego nie powinno być jedynym mechanizmem uczenia się, lecz pełnić rolę wspomagającą.

Przestawione konkluzje prowadzą do ogólnego wniosku, że głównym mechanizmem adaptacji autonomicznego agenta powinno być jego indywidualne uczenie się bez nadzoru, które może być ewentualnie uzupełniane uczeniem się z nadzorem. Jest to wniosek w dużym stopniu zgodny z intuicją, skoro najważniejszą cechą agenta jest jego samostanowienie o sobie, czyli autonomiczność. Zauważmy, że ten wynik jest bardzo zbieżny z przytoczonym wcześniej postulatem Maes, iż w architekturze agenckiej należy stosować uczenie się bez nadzoru [Mae1994b].

Wymagamy zatem, aby agent, jako system uczący się, samodzielnie kontrolował cały proces: pozyskiwania informacji z otoczenia, przetwarzania jej, wpływania na stan bazy wiedzy, wykorzystywania zdobytej wiedzy do udoskonalania swojego działania i oceny uczenia się. Spełnienie tych wymagań jest poważnym wyzwaniem, jeśli wziąć pod uwagę, że w uczeniu się bez nadzoru istnieje, wspomniana już kilkakrotnie, trudność oceny jakości wiedzy w ten sposób pozyskiwanej [Has2001]. W dalszej części tekstu rozważane są różne zagadnienia związane z osadzeniem metod uczenia się w architekturze agenckiej.

2.4.3 Techniki uczenia się agentów

Wśród opracowanych dotychczas technik uczenia się agentów można zauważyć zdecydowaną dominację metod symbolicznych, rzadkością zaś są propozycje takie, jak [Rin1997], wykorzystujące techniki niesymboliczne, na przykład sieci neuronowe. Dostępne prace z zakresu uczenia się agentów mogą być uporządkowane na wiele sposobów i według różnych kryteriów. W niniejszym przeglądzie proponujemy, częściowo opierając się na [Gue2003] oraz [Mae1994b], podział technik na poniższe grupy:

- uczenie się ze wzmocnieniem (ang. *reinforcement learning*, RL);
- uczenie się oparte na wyjaśnianiu (ang. *explanation-based learning*);
- uczenie się pojęć (ang. *concept learning from examples*);
- uczenie się modelu (ang. *model-based learning*);
- inne metody – nie należące do żadnej z powyższych kategorii.

Zgromadzone przez autora prace są omawiane według powyższej klasyfikacji. Najwięcej miejsca poświęcone jest uczeniu się ze wzmocnieniem, gdyż ta grupa metod wyraźnie dominuje wśród opracowanych dotąd rozwiązań adaptacji agentów.

Uczenie się ze wzmocnieniem (ang. *reinforcement learning*, RL)

Jest to w ostatnich latach zdecydowanie najbardziej popularne podejście wśród prac dotyczących uczenia się agentów. Ogólna idea uczenia się ze wzmocnieniem została podana w poprzednim rozdziale, natomiast dogłębna, wyczerpująca analiza tej grupy metod w kontekście systemów agenckich została przedstawiona w pracy Ribeiro [Rib2002]. Tam też można znaleźć ogólny model agenta uczącego się ze wzmocnieniem, przedstawiony na Rys. 2.2.

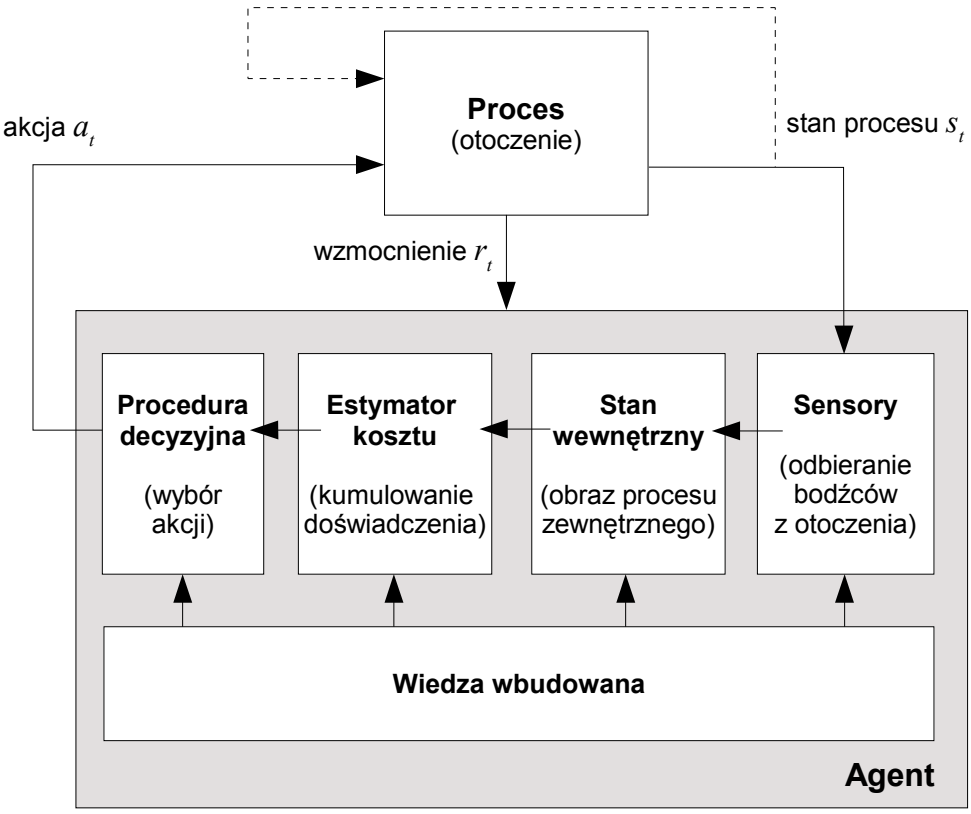
Agent posiada kilka modułów, które są wykorzystywane w procesie uczenia się:

- *wiedza wbudowana* (ang. *built-in knowledge*) – wiedza przekazana agentowi na etapie jego projektowania;
- *sensory* (ang. *sensors*) – mechanizmy prowadzące obserwację stanu procesu zewnętrznego;
- *stan wewnętrzny* (ang. *internal state*) – struktura reprezentująca przekonania agenta o bieżącym stanie procesu zewnętrznego;
- *estymator kosztu* (ang. *cost estimator*) – moduł gromadzący doświadczenie w postaci stanów wewnętrznych oraz związanych z nimi wzmocnień i przypisujący do nich koszt, który odzwierciedla przekonanie agenta, na ile dobry (użyteczny) jest dany stan;
- *procedura decyzyjna* (ang. *action policy*) – strategia wyboru akcji na podstawie wiedzy posiadanej przez agenta.

Agent za pomocą sensorów odbiera informację o bieżącym stanie procesu (otoczenia) i zapisuje ją w strukturze stanu wewnętrznego. Informacja o stanie wewnętrznym może uwzględniać wcześniejsze obserwacje oraz wiedzę wbudowaną. Następnie estymator kosztu przypisuje stanowi wewnętrznemu oraz związanemu z nim wzmocnieniu określony koszt,

który stanowi ocenę stanu, na podstawie zgromadzonego dotąd doświadczenia. Wówczas procedura decyzyjna, na podstawie kosztów obliczonych przez estymator kosztu, dokonuje wyboru akcji, która ma być wykonana przez agenta. Wiedza wbudowana może wpływać na zachowanie agenta w sposób bezpośredni, poprzez zmianę strategii decyzyjnej, lub też pośredni, oddziałując na pracę sensorów lub estymatora kosztu. W opisanym tutaj modelu agenta proces gromadzenia doświadczenia i podejmowania decyzji przebiega cyklicznie, poprzez powtarzanie poniższych etapów:

1. Obserwacja procesu (otoczenia) i dostarczanego przez niego wzmocnienia.
2. Wykonanie na procesie akcji, która jest wybierana przez agenta na podstawie wcześniejszego doświadczenia, ostatniej obserwacji i wzmocnienia.
3. Nowa obserwacja procesu, aktualizacja skumulowanego doświadczenia.



Rys. 2.2. Ogólny model agenta uczącego się ze wzmocnieniem (na podstawie [Rib2002], str. 227).

Innymi słowy, agent przeprowadza eksperymentowanie na środowisku, w którym jest osadzony. Wykonuje on pewną akcję i obserwuje stan otoczenia, który jest wynikiem wykonania akcji. W zależności od zaobserwowanego rezultatu, agent otrzymuje określone wzmocnienie, informujące go o jakości jego działania w kontekście postawionego przed nim zadania. Celem uczenia się agenta jest znalezienie związku pomiędzy obserwowanymi stanami otoczenia i wykonanymi akcjami, a ich wynikiem w postaci wzmocnienia tak, aby procedura wyboru akcji, zwana tutaj strategią, była jak najbardziej zbliżona do optymalnej.

Opisane tutaj uczenie się ze wzmocnieniem (RL) jest przedmiotem wielu prac z zakresu uczenia się agentów. Istnieje także wiele udanych aplikacji algorytmów RL w architekturze

agenckiej, w takich dziedzinach, jak: planowanie i alokacja zasobów, robotyka oraz systemy wieloagenckie.

Cole, Lloyd i Ng [Col2003] [Ng2004] zaproponowali system uczenia się drzew decyzyjnych o nazwie ALKEMY, wykorzystujący odmianę RL – *Q-learning*, w którym zastosowana jest symboliczna reprezentacja stanu, akcji i funkcji *Q*. Autorzy zwracają uwagę na korzyści płynące z symbolicznej reprezentacji wiedzy w algorytmach RL, w stosunku do alternatywnych podejść niesymbolicznych, na przykład sieci neuronowych: (i) reprezentacja symboliczna zapewnia wygodny sposób opisu dziedziny (ang. *domain knowledge*), strategii wyboru akcji (ang. *policy*) oraz funkcji Q; (ii) strategie oraz funkcje Q, zapisane w formie symbolicznej, są bezpośrednio zrozumiałe i dostępne do manipulowania przez człowieka (np. dzięki temu agent może wyjaśnić użytkownikowi, dlaczego podejmuje takie, a nie inne akcje). Autorzy twierdzą, iż symboliczna reprezentacja funkcji strategii wyboru akcji jest kluczowa dla wydajnej integracji wnioskowania i mechanizmu uczenia się w architekturze agenckiej. Ważne jest jednak, aby agentowi przekazywane było jak najwięcej wiedzy wbudowanej, a mechanizm uczenia się służył jedynie jej uzupełnieniu (np. dostosowaniu agenta do konkretnego użytkownika) [Col2003]. U podstaw systemu ALKEMY leży rozbudowany formalizm logiki wyższego rzędu, opisany w pracy [Ng2004].

Ring [Rin1997] zaproponował system o nazwie CHILD, który wykorzystuje dość rzadkie połączenie algorytmu *Q-learning* oraz algorytmu *hierarchii przejść czasowych* (ang. *Temporal Transition Hierarchies*, TTH), opartego na sieciach neuronowych. System CHILD stanowi realizację koncepcji *ustawicznego uczenia się* (ang. *continual learning*), według której autonomiczny agent uczy się w trakcie swojego działania w sposób inkrementacyjny (tzn. równolegle z realizacją swoich normalnych funkcji) i hierarchiczny (raz wyuczone umiejętności mogą być rozbudowywane i zmieniane w późniejszym czasie).

Tesauro i Kephart [Tes2002] przeprowadzili eksperymentalną analizę działania agentów uczących się metodą *Q-learning* w różnych wariantach problemu konkurencyjnego ustalania ceny przez dwóch sprzedawców (ang. *two-seller economy*). Uzyskane wyniki wskazują, że strategie agentów uzyskane z uczenia się *Q-learning* zwiększają zyski agentów i redukują cykliczne wojny cenowe (ang. *cyclic price wars*), w porównaniu z prostszymi strategiami, takimi jak krótkoterminowe planowanie z wyprzedzeniem (ang. *short-term lookahead*).

Zhong, Wu i Kimbrough [Zho2002] wykazali eksperymentalnie pozytywny wpływ algorytmu RL na efektywność działania agenta podczas gry w ultimatum (ang. *ultimatum game*), która jest znanym modelem procesów negocjacji. Şenkul i Polat [Snk2002] zastosowali algorytm *Q-learning* w systemie wieloagenckim, rozwiązującym klasyczny problem *wilków i owcy* (ang. *prey-hunter capture game*) i zaproponowali grupowanie tabel wartości funkcji *Q* (ang. *Q-tables*) jako metodę zwiększenia wydajności uczenia się poszczególnych agentów. Badaniu algorytmów *Q-learning* w aplikacjach z zakresu teorii gier poświęcona została także praca Kimbrougha i Lu [Kmb2005].

Crites i Barto [Cri1998] zastosowali algorytmy RL do implementacji agentów, które sterują systemem wind w symulatorze. Marsella i współpracownicy [Mar2001] zastosowali RL w symulacjach ligi piłkarskiej RoboCup, prowadzonych w architekturze ISIS. Algorytm RL wykorzystywany jest przez agenty do uczenia się wyboru planu przechwytywania zbliżającej się piłki. W tej samej dziedzinie (RoboCup) Thawonmas, Hirayama i Takeda [Tha2002] stworzyli system KUP-RP, w którym zastosowali algorytm *Q-learning* do uczenia agenta

wybranych umiejętności gry w piłkę nożną. Wang i Usher [WgU2005] zastosowali algorytm *Q-learning* do realizacji uczenia się reguł przydziału zadań (ang. *dispatching rules*) przez autonomicznego agenta, który steruje maszyną produkcyjną. Chen i jego współpracownicy [ChY2005] użyli algorytmów uczenia się ze wzmocnieniem do rozwiązania problemu koordynacji agentów w systemie wieloagenckim. Jako model koordynacji wykorzystana została *rozmyta struktura zadań subjektywnych* (ang. *Fuzzy Subjective Task Structure*, FSTS). Kaya i Alhajj [Kay2005b], [Kay2005c] wykorzystali techniki *Q-learning* do realizacji uczenia się współpracujących ze sobą agentów. Autorzy oparli proponowane rozwiązanie na rozmytej strukturze wielowymiarowej OLAP (ang. *on-line analytical processing*) oraz algorytmach znajdowania rozmytych reguł związku (ang. *fuzzy association rules*).

Takadama, Nakasuka i Terano [Tak1998] zastosowali algorytmy RL w dziedzinie projektowania płytek drukowanych (ang. *Printed Circuit Board design*, PCBs). Ich rozwiązanie jest oparte na *systemie uczących się klasyfikatorów* (ang. *Learning Classifier System*, LCS), wykorzystującym reguły produkcyjne oraz algorytmy genetyczne i należącym do omawianych w poprzednim rozdziale *systemów klasyfikatorów* (ang. *classifier systems*, CS).

Van Bragt i La Poutré [Brg2003] zaproponowali rozwiązanie adaptacji negocjujących agentów, które również jest oparte na CS, przy czym klasyfikatorami są tutaj strategie negocjacyjne, reprezentowane w formie *automatów skończonych* (ang. *finite automata*). Algorytmy ewolucyjne (ang. *evolutionary algorithms*, EA) są natomiast wykorzystywane do: (i) eksperymentowania z nowymi strategiami (mutacja), (ii) łączenia poprzednich strategii (krzyżowanie – ang. *cross-over*), (iii) usuwania gorszych strategii (selekcja). Bardzo podobne podejście można zauważyć w pracy Gordon [Gor2001], która opracowała architekturę agencką APT (ang. *Adaptive*, *Predictable and Timely*). W architekturze tej agenty wykazują zdolność adaptacji do nieprzewidzianych okoliczności, przewidywalność zachowania oraz czas reakcji akceptowalny z punktu widzenia stawianych przed nimi zadań. Część adaptacyjna jest oparta na algorytmach ewolucyjnych, które odpowiednio modyfikują plany agenta, reprezentowane w formie automatów skończonych (ang. *Finite State Automata*, FSA). Należy podkreślić, że autorzy prac [Brg2003] i [Gor2001] sami nie lokują ich w obszarze systemów CS, jednak mimo to można je tak zakwalifikować, ponieważ występują w nich klasyfikatory o pewnej reprezentacji (strategie, plany) oraz metoda eksperymentowania w formie algorytmów ewolucyjnych.

Jak widać, szereg omówionych wyżej prac wskazuje na przydatność wykorzystania algorytmów uczenia się ze wzmocnieniem w architekturach agenckich, szczególnie dla zastosowań, w których uczenie się agenta musi przebiegać równocześnie z wykonywaniem przez niego normalnych zadań (ang. *on-line learning*). Mimo to, w kontekście systemów agenckich uczenie się ze wzmocnieniem posiada również pewne ograniczenia, do których można zaliczyć między innymi [Mae1994b], [Rib2002]:

- trudności uczenia się przy celach zmiennych w czasie – zmiana celu agenta może wymuszać jego uczenie się od początku;
- nieakceptowalny czas uczenia się przy bardzo dużych przestrzeniach problemu, występujących w rzeczywistych zastosowaniach;
- zawodność aparatu percepcyjnego agenta – obserwowany przez niego stan otoczenia zazwyczaj tylko w przybliżeniu odpowiada rzeczywistości, przez co efektywność uczenia się może ulegać obniżeniu;

- trudności modelowania złożonych środowisk (co jest częste w rzeczywistych zastosowaniach) za pomocą procesów decyzyjnych Markova (MDP).

Przedstawione problemy wyznaczają bieżący obszar zastosowań technik uczenia się ze wzmocnieniem i wytyczają kierunki ich dalszego rozwoju. Nie ulega natomiast wątpliwości, że dziedzina ta będzie nadal przedmiotem intensywnych badań.

Uczenie się oparte na wyjaśnianiu (ang. *explanation-based learning*, EBL)

Jest to grupa metod należących do kategorii *uczenia się na podstawie analizy* (ang. *analytical learning*) lub inaczej *uczenia się optymalizującego* (ang. *speedup learning*) [Alo2001], [Die2003]. Podczas uczenia się agent nie musi prowadzić interakcji z otoczeniem, lecz analizuje zgromadzone wcześniej doświadczenie, aby wyciągnąć na jego podstawie uogólnione wnioski. Agent wykorzystuje przy tym pewną wbudowaną wiedzę o dziedzinie (ang. *built-in knowledge*, *background knowledge*), aby wyjaśnić (stąd nazwa) przyczyny sukcesu lub porażki zapisanych, wcześniejszych działań. Na tej podstawie agent tworzy uogólnione reguły, które, wykorzystane w toku dalszego działania, mogą zwiększać jego efektywność. Dzięki temu agent, który wielokrotnie wykonuje pewne zadanie, po nauczeniu się odpowiednich reguł w pierwszym przebiegu, może znacznie szybciej dochodzić do celu w kolejnych uruchomieniach.

Najbardziej znaną i rozwiniętą architekturą agencką, wykorzystującą uczenie się oparte na wyjaśnianiu, jest system *Soar* [Lai1987]. Jest to architektura kognitywna, której podstawą teoretyczną jest zunifikowana teoria poznawania (ang. *unified theory od cognition*), wprowadzona przez Newella [New1990]. W architekturze tej agent dąży do celu poprzez *przeszukiwanie przestrzeni problemu* (ang. *problem space search*), będącej przestrzenią wszystkich możliwych stanów, w których może się znajdować agent i jego otoczenie. W danym stanie agent wybiera operator, za pomocą którego może przejść do nowego stanu i zbliżyć się do stanu celu (ang. *goal state*). Przy wyborze operatora agent, zależnie od konkretnej implementacji, może się posługiwać różnymi strategiami heurystycznymi, na przykład *analizą środków prowadzących do celów* (ang. *means-ends analysis*) lub *przeszukiwaniem z wyprzedzeniem* (ang. *look-ahed search*). *Soar* jest systemem produkcyjnym i dlatego całość wiedzy długoterminowej jest zapisywana w formie reguł *jeśli warunek*, *to akcja*. Działanie systemu opiera się na *cyklu decyzyjnym* (ang. *decision cycle*), złożonym z dwóch podstawowych faz: *fazy wykonania* (ang. *elaboration phase*), w której wykonywane są aż do wyczerpania wszystkie produkcje pasujące do bieżącego stanu oraz *fazy decyzyjnej* (ang. *decision phase*), w której system wybiera kolejny operator [Lai1987].

Uczenie się oparte na wyjaśnianiu (EBL) zachodzi w agencie *Soar* wówczas, gdy dochodzi do tak zwanego *impasu* (ang. *impass*), na przykład wtedy, gdy agent nie posiada wystarczającej wiedzy, aby wybrać operator spośród wielu dostępnych. Wówczas agent wstrzymuje rozwiązywanie problemu w głównej przestrzeni, próbuje natomiast rozwiązać impas w podprzestrzeni, utworzonej poprzez mechanizm automatycznego tworzenia podcelów (ang. *automatic subgoaling*). W podprzestrzeni tej agent może przeanalizować dostępne, remisowe operatory posługując się pewną heurystyką (np. *look-ahead search*) i na tej podstawie zwrócić do głównej przestrzeni problemu wynik – brakujące wcześniej preferencje dotyczące wyboru operatora. W ten sposób impas zostaje przezwyciężony, a agent może kontynuować rozwiązywanie problemu. W momencie gdy zwracane są wyniki do przestrzeni, w której doszło

do impasu, system dokonuje *analizy wstecznej* (ang. *backtracking*): łączy warunki (atrybuty stanu) z początku impasu ze zwróconymi wynikami (akcjami), uogólnia je (podstawia zmienne za konkretne wartości atrybutów) i tworzy nową produkcję, zwaną *kawałkiem* (ang. *chunk*). Uogólniona wiedza w postaci nowych produkcji może być wykorzystana w toku dalszego działania agenta, pozwalając na unikanie impasów i w konsekwencji – na zwiększenie efektywności agenta [Lai1987], [New1990], [Dud1999].

Od lat dziewięćdziesiątych ubiegłego wieku architektura Soar jest ciągle rozwijana; powstało szereg prac teoretycznych oraz aplikacji w takich dziedzinach jak: symulatory lotnictwa wojskowego, środowiska treningowe w wirtualnej rzeczywistości, gry komputerowe [URLSoar]. Opracowano także szereg ogólnych rozszerzeń architektury, jak na przykład połączone systemy *InstructoSoar* (EBL) oraz IMPROV (realizujący empiryczne uczenie się), które zapewniają korzystną dla agenta integrację *poziomu działania celowego* (ang. *deliberate level*) oraz *poziomu planowania i dekompozycji problemu* (ang. *reflective level*) [Lai1997]. W ostatnim czasie Nason i Laird [Nas2004], [Nas2005] zaproponowali bardzo ciekawy system o nazwie Soar-RL, który rozszerza architekturę Soar o możliwość uczenia się ze wzmocnieniem. Z kolei Young w pracy [You2005] porusza problem *uczenia się danych* (ang. *data learning*) w kontekście systemu systemu *Soar* i innych architektur kognitywnych. Ten typ uczenia się jest związany z pozyskiwaniem przez agenta deklaratywnej informacji z zewnątrz, na przykład odpowiedzi na pytanie.

Uczenie się pojęć (ang. *concept learning*)

Jest to kolejna, duża grupa technik stosowanych w systemach agenckich, nazywana także *uczeniem się pojęć na podstawie przykładów* (ang. *concept learning from examples*) [Gue2003] i zaliczana do indukcyjnego uczenia się [Cic2000]. Agent uczący się w ten sposób przegląda pewną przestrzeń potencjalnych hipotez, szukając tej, która najlepiej pasuje do przykładów trenujących. W ten sposób uzyskiwana i zapamiętywana jest uogólniona wiedza, nazywana *pojęciem* (ang. *concept*), a przykłady trenujące są usuwane z pamięci [Gue2003]. Istnieje wiele odmian uczenia się pojęć, z których tutaj omówimy dwie najczęściej spotykane w systemach agenckich: *indukcyjne programowanie logiczne* oraz *indukcję drzew decyzyjnych*.

Indukcyjne programowanie logiczne (ang. *inductive logic programming*, ILP) jest to technika uczenia się, która pozwala na wyznaczenie definicji (lub przybliżenia) pewnego predykatu docelowego, na podstawie pozytywnych i negatywnych przykładów trenujących dla tego predykatu [Alo2001]. Definicja ma formę programu logicznego. W odróżnieniu od EBL, metody ILP wyznaczają hipotezę nie tylko na podstawie wiedzy zgromadzonej uprzednio, lecz także w oparciu o nieznane wcześniej informacje, pozyskiwane z otoczenia przez agenta.

Kazakov i Kudenko [Kaz2001] przeprowadzili szczegółową analizę technik ILP i ich wykorzystania w systemach agenckich. Przedstawili oni także ogólne środowisko wieloagenckie, opracowane na Uniwersytecie York, które wykorzystuje implementację języka Prolog i może być wykorzystane w szczególności do badania agentów uczących się poprzez ILP.

Lamma, Riguzzi i Pereira [Lam1999] opracowali metodę uczenia się pojęć opartą na ILP, która pozwala wykorzystywać trzy wartości logiczne wiedzy agenta: *prawda*, *fałsz* i *wartość nieznana*, w odróżnieniu od *założenia świata zamkniętego* (ang. *Closed World Assumption*, CWA). Jako reprezentację wiedzy potrzebną przy uczeniu się, autorzy wykorzystali

rozszerzone programy logiczne (ang. *Extended Logic Programs*, ELP) oraz *rozszerzoną semantykę z jawną negacją* (ang. *Well-Founded Semantics with explicit negation*, WFSX).

Indukcja drzew decyzyjnych (ang. *decision tree induction*) jest metodą uczenia się pojęć, dająca w wyniku strukturę drzewa, w której do węzłów nie będących liśćmi przypisywane są określone atrybuty lub pytania, a do węzłów potomnych – możliwe wartości tych atrybutów lub odpowiedzi na pytania. Węzły – liście odpowiadają klasom, do których można przypisywać przykłady trenujące (ang. *instances*) [Wls2003].

Marsella i współpracownicy [Mar2001] wykorzystali algorytm generowania drzew decyzyjnych C4.5 do uczenia agentów – piłkarzy *RoboCup* – umiejętności strzelania bramek. Uczenie się z nadzorem zostało przeprowadzone na przykładach trenujących w trybie *off-line*, czyli poza czasem normalnego działania agentów, a uzyskane w ten sposób reguły pozwoliły na znaczące zwiększenie ich efektywności.

W tej samej dziedzinie (*RoboCup*) Thawonmas, Hirayama i Takeda [Tha2002] stworzyli system KUP-RP, w którym zastosowali algorytm C4.5 do uczenia agenta podejmowania decyzji na wzór człowieka. Reguły postaci *warunek – akcja*, które służą do trenowania agenta, są znajdowane na podstawie rejestru zachowań człowieka, który gra w piłkę w tym samym środowisku symulacyjnym, co agent. Wyniki eksperymentów wykazały, że jest to efektywna metoda uczenia agenta skomplikowanych reguł działania. Wytrenowany w ten sposób agent działał tylko nieznacznie gorzej od agenta, którego wiedza została w całości przekazana przez programistę (ang. *hand coded agent*), tymczasem czas potrzebny do ręcznego zaimplementowania wiedzy agenta był kilkakrotnie większy od czasu trenowania.

Guerra Hernández [Gue2003] zastosował metodę indukcji logicznych drzew decyzyjnych (ang. *logical decision trees*), opartą na algorytmie ID3, do uczenia się agenta w samodzielnie zaimplementowanej architekturze BDI (ang. *Belief Desire Intention*), zgodnej ze specyfikacją systemu dMARS [DIn1998]. Algorytmy indukcji drzew decyzyjnych wykorzystywane są także w omawianym wcześniej systemie ALKEMY, zaproponowanym przez Ng [NgK2004].

Uczenie się modelu (ang. *model-based learning*)

W tym przypadku agent na podstawie doświadczenia uczy się probabilistycznego modelu przyczynowego (ang. *causal model*), który mówi, jakie są możliwe rezultaty wykonania określonej akcji w danej sytuacji [Mae1994b]. Model ten może być następnie wykorzystany przez agenta w procesie decyzyjnym, przy czym mechanizm wyboru akcji i uczenia się są tutaj znacznie bardziej niezależne od siebie, niż w przypadku np. uczenia się ze wzmocnieniem. Poza tym model, którego uczy się agent, zazwyczaj ma charakter częściowy, łącząc ze sobą tylko sytuacje i akcje o największym znaczeniu z punktu widzenia działania agenta (a więc nie wszystkie możliwe pary sytuacji i akcji). Pozyskiwana wiedza ma postać zestawów, zwanych zachowaniami, schematami lub modułami (ang. *behaviour*, *schema*, *module*), z których każdy zawiera: (i) zbiór warunków, (ii) akcję (prostą lub złożoną), (iii) zbiór spodziewanych wyników wraz ich prawdopodobieństwami. Zbiór schematów jest regularnie aktualizowany przez agenta w miarę zdobywania przez niego nowego doświadczenia, a więc wykonywania akcji i obserwowania ich rezultatów w otoczeniu. Według Maes [Mae1994b] zaletami uczenia się modelu są: (i) ogólność zdobywanej w ten sposób wiedzy i możliwość przenoszenia wyuczonego zachowania z jednego kontekstu do drugiego (np. gdy zmienia się cel agenta); (ii) łatwość przekazania do agenta wiedzy o dziedzinie (ang. *background*

knowledge) na etapie projektowania. Natomiast wadą tego typu uczenia się jest brak połączenia sytuacji z optymalnymi akcjami, przez co proces wyboru akcji przez agenta może ulegać wydłużeniu.

W dostępnych pracach można wyróżnić dwa główne zastosowania uczenia się modelu przez systemy agenckie: *modelowanie użytkownika* (ang. *user modeling*) oraz *modelowanie innego agenta* w systemie wieloagenckim. W pierwszym przypadku agent pełni rolę osobistego asystenta użytkownika i poprzez mechanizm uczenia się dostosowuje się do jego zachowań oraz preferencji, na podstawie doświadczeń zebranych podczas interakcji z użytkownikiem lub obserwacji jego działań. Aby to było możliwe, w szczególności muszą być spełnione dwa warunki: (i) użytkownik wykazuje zachowania, które są wyraźnie powtarzalne; (ii) powtarzalne zachowania są potencjalnie różne dla różnych użytkowników [Mae1994a]. Tego typu uczenie się zostało zrealizowane przez Maes [Mae1994a] w kilku systemach agenckich, wspomagających użytkownika w zarządzaniu pocztą elektroniczną, planowaniu spotkań i filtrowaniu list wiadomości oraz rekomendujących różne formy rozrywki. W podobnej grupie zastosowań mieszczą się projekty agentów LAW, ELVIS i MAVA zaproponowane przez Edwardsa, Greena, Lockiera i Lukinsa [Edw1997]. Systemy te również wykorzystują zapis dotychczasowego zachowania użytkownika, aby zgodnie z jego profilem i preferencjami rekomendować strony WWW, proponować plany podróży lub modyfikować zapytania wprowadzane do formularzy, w celu zwiększenia ich efektywności. Yao, Hamilton i Wang [Yao2002] stworzyli agenta o nazwie *PagePrompter*, który wspomaga obsługę serwisu internetowego. System modeluje użytkowników – zarówno indywidualnie, jak i grupowo – poprzez analizę rejestru zdarzeń w serwisie (ang. *web log*), obejmującego zapis odwiedzin i zachowań użytkowników, oraz informacji o modyfikacjach zawartości serwisu, dokonywanych przez administratorów. Agent uczy się analizując zgromadzoną informację algorytmami odkrywania reguł związku (ang. *association rules*) oraz grupowania (ang. *clustering*). Do klasy agentów modelujących użytkownika możemy zaliczyć także system aktywnego przeglądania bibliotek programistycznych (ang. *active browsing*), opracowany przez Drummonda, Ionescu i Holte'a [Dru2000]. System ten próbuje ustalić cel użytkownika – programisty na podstawie jego dotychczasowych działań, a następnie sugeruje mu obiekty, które są oceniane jako najbardziej przydatne ze względu na rozpoznany cel. Reprezentacja wiedzy oparta jest na predykatach i regułach ze stopniami wiarygodności (ang. *confidence*) z przedziału [0; 1].

Modelowanie innego agenta zazwyczaj stosowane jest w *słabym systemie wieloagenckim* (ang. *weak multi-agent system*), w którym nie występuje zamierzone dążenie agentów do wspólnego celu, lecz poszczególne agenty mają indywidualne cele lub nawet konkurują ze sobą. Zastosowania lub symulacje tego typu rozwiązań dotyczą w większości dziedziny e-biznesu, w szczególności negocjacji [Crm1999], [HuW2001]. Są one omawiane w podrozdziale poświęconym uczeniu się systemu wieloagenckiego.

Inne metody uczenia się agentów

Przedstawione wcześniej, główne grupy metod uczenia się agentów nie wyczerpują całego zakresu prac z tej dziedziny. Poniżej zestawione są propozycje, które wnoszą wyraźny wkład w omawianą dziedzinę, lecz ze względu na swoją oryginalność nie należą do żadnego z podejść „klasycznych" – uznanych, rozpowszechnionych i silnie rozwijanych.

Uczenie się oparte na przykładach (ang. *instance-based learning*)

Jest to jedna z prostszych form uczenia się, stosowana w systemach agenckich. Agent zapamiętuje przykłady trenujące, a następnie przeszukuje je, aby odnaleźć najlepsze dopasowanie do nowego przykładu, który ma być sklasyfikowany [Kaz2001]. Agent uogólnia przykłady w ograniczony sposób – tylko lokalnie, w kontekście konkretnego zadania [Gue2003]. Z tego względu uczenie się oparte na przykładach zaliczane jest, obok najprostszego *uczenia się na pamięć* (ang. *rote learning*), do tak zwanych *metod leniwych* (ang. *lazy learning*) [Gue2003], [Kaz2001]. Przykład tego typu uczenia się został przedstawiony w pracy Garlanda i Altermana [Gar2004], poświęconej koordynacji w systemie wieloagenckim. W proponowanym przez nich rozwiązaniu poszczególne agenty, po zakończeniu wspólnego działania, uczą się skoordynowanych procedur. Ponadto, w oparciu o algorytm klasyfikacji COBWEB, uczą się inkrementacyjnie *drzew prawdopodobieństwa operatorów* (ang. *operator probability trees*), które wyrażają prawdopodobieństwo powodzenia dostępnych akcji i są wykorzystywane do ich planowania. Enembreck i Barthès [Ene2005] opracowali inkrementacyjną metodę uczenia się o nazwie ELA (ang. *entropy-based learning approach*), która może być uznana jako modyfikacja *technik pamięciowych* (ang. *memory-based reasoning*, MBR). Autorzy zastosowali szybki algorytm klasyfikowania przykładów (bez użycia miar podobieństwa), oparty na reprezentacji wiedzy w postaci *grafu pojęć* (ang. *concept graph*, CG).

Proceduralne uczenie się

Jest to grupa metod, w których uczenie się agenta polega na zapamiętywaniu, modyfikowaniu i wykorzystywaniu wiedzy o charakterze proceduralnym: operatorów i planów. Można tutaj zaliczyć część prac omawianych w ramach innych kategorii, na przykład [Lai1997], [Gor2001]. Garcia-Martinez i Borrajo [GrM2000] opracowali oryginalną architekturę agencką ogólnego przeznaczenia (ang. *domain independent*) o nazwie LOPE (ang. *Learning by Observation in Planning Environment*), która pozwala na uczenie się definicji operatorów, planowanie wykorzystujące te operatory, wykonywanie planów i modyfikowanie pozyskanych operatorów. Jest to rozwiązanie nieco podobne do technik uczenia się ze wzmocnieniem (np. w LOPE są również stosowane nagrody), ale autorzy wykazali wyraźne różnice obu podejść.

Trenowanie agentów

Omawiane do tej pory rozwiązania dotyczą sytuacji, w której agent uczy się w sposób autonomiczny. Oznacza to, że po zakończeniu etapu projektowania i implementacji, agentowi nie może być przekazana żadna informacja na zasadzie bezpośredniego manipulowania w jego bazie wiedzy. Zostały jednak opracowane propozycje, w których wymóg ten nie występuje, to znaczy agenty mogą mieć z zewnątrz zmieniany stan bazy wiedzy, także po wdrożeniu w docelowym środowisku pracy.

Kechagias, Mitkas, Symeonidis i inni [Kec2002], [Mit2002], [Sym2002] opracowali heterogeniczny system Agent Academy (AA), który umożliwia automatyczne trenowanie agentów działających w określonym środowisku – zarówno nowych (ang. *training*), jak i tych, które pracują tam już od pewnego czasu (ang. *retraining*). System AA składa się z czterech podstawowych modułów: *fabryki agentów* (ang. *Agent Factory*, AF), *repozytorium* (ang. *Agent Use Repository*, AUR), *eksploratora* (ang. *Data Mining Module*, DMM) oraz *trenera agentów* (ang. *Agent Training Module*, ATM). Moduł AF służy do generowania nowych agentów na żądanie użytkownika. Repozytorium (AUR) ustawicznie rejestruje dane o środowisku zewnętrznym, zachowaniach agentów i interakcji między nimi. Eksplorator (DMM) dokonuje analizy informacji gromadzonej w repozytorium (AUR) i na jej podstawie odkrywa zależności, wykorzystując między innymi algorytmy ID3, C4.5 i Apriori. W końcu moduł ATM przekazuje agentom wiedzę odkrytą przez moduł eksploratora (DMM). Autorzy omówili zastosowanie architektury AA w systemie czasu rzeczywistego O3RTAA, który alarmuje o stanie ozonu na podstawie danych rejestrowanych w atmosferze. Niewątpliwą zaletą systemu AA jest gromadzenie i analizowanie wiedzy pochodzącej z doświadczenia wielu agentów. Przy założeniu, że poszczególne agenty mogą działać w różnych obszarach środowiska i posiadać różne doświadczenia z interakcji z otoczeniem, takie podejście zwiększa wszechstronność i kompletność gromadzonej wiedzy w porównaniu z indywidualnym uczeniem się agentów. Dyskusyjna jest natomiast kwestia, czy tego typu obiekty, poddawane co pewien czas bezwarunkowej ingerencji w wewnętrzne struktury wiedzy (czyli swoistemu „praniu mózgu"), nadal mogą być nazywane agentami, ponieważ kłóci się to z podstawową cechą od nich wymaganą – autonomicznością.

Nieco podobna sytuacja występuje w omawianych wcześniej pracach [Mar2001], [Tha2002], poświęconych trenowaniu agentów – piłkarzy *RoboCup*. Tam jednak uczenie się agentów z nadzorem zachodzi jeszcze przed wdrożeniem w środowisku docelowym.

2.4.4 Integracja technik uczenia się z architekturą agencką

Istnieje wiele kwestii, które muszą być rozstrzygnięte podczas integracji ogólnych algorytmów uczenia się z konkretną architekturą agencką. Poniżej omówione są ważniejsze z nich na podstawie prac [Gue2003], [Kaz2001].

Ograniczenia czasowe

W architekturze agenckiej dużego znaczenia nabiera czasowa efektywność integrowanych z nią algorytmów uczenia się. Agent działający autonomicznie w pewnym środowisku zazwyczaj posiada pewien horyzont czasowy, zależny od dziedziny zastosowania i wymuszający wykonanie zadania w określonym terminie (np. poszukiwane strony WWW powinny być dostarczone użytkownikowi w akceptowanym przez niego czasie). Tymczasem uczenie się jest w agencie, obok percepcji, wnioskowania, wykonywania akcji, tylko jednym z wielu procesów zużywających dostępne zasoby. Z tego względu w architekturze agenckiej mogą być instalowane tylko te algorytmy, które spełniają wymogi efektywności, zależne od danego zastosowania i wyznaczane na przykład w oparciu o dopuszczalny czasu reakcji systemu[3]. Ponadto duże znaczenie ma umieszczenie w algorytmach agenta procedury przydziału zasobów, która uwzględnia wykorzystywane mechanizmy uczenia się i zapewnia równowagę

[3] Wymagania te są szczególnie wysokie w systemach czasu rzeczywistego (ang. *Real Time Systems*, RTS).

pomiędzy zdobywaniem nowej wiedzy, a wykorzystywaniem już posiadanej w trakcie normalnego funkcjonowania. Dopuszczalne może być, co prawda, pewne chwilowe spowolnienie działania agenta z powodu uczenia się, jeśli ma to przynieść korzyść długoterminową, ale tolerancja ta jest ograniczona.

Warunek zatrzymania procesu uczenia się

Istnieją tutaj dwa podstawowe podejścia [Kaz2001]: (i) algorytm posiada wyraźny warunek stopu (ang. *halting condition*); (ii) uczenie się ma charakter ustawiczny (ang. *any-time learning*). Do pierwszej kategorii należą techniki intensywnego przetwarzania danych trenujących lub przestrzeni hipotez (np. algorytmy eksploracji danych). W tym przypadku wspomniana wcześniej procedura przydziału zasobów powinna decydować, czy w danym momencie uczenie się w ogóle może być uruchamiane (po uwzględnieniu szacunkowego czasu trwania procesu uczenia się, dostępnych zasobów i bieżących ograniczeń czasowych). Z kolei przy zastosowaniu drugiej grupy technik – ustawicznego uczenia się (np. uczenia się ze wzmocnieniem, RL), procedura sterująca procesem uczenia się powinna zapewniać zatrzymywanie go, jeśli nie pozwala on na spełnienie bieżących ograniczeń czasowych lub jest zbyt kosztowny w stosunku do przewidywanych korzyści.

Odwoływanie się do pamięci (ang. *recall*)

Uczący się agent, odbierając informację z otoczenia, może natychmiast wykorzystać ją do aktualizacji swojego modelu świata (co potencjalnie będzie miało wpływ na wybór kolejnych akcji), albo też odwołać się (ang. *recall*) do zapamiętanego modelu, zapisując nową informację i odkładając faktyczne uczenie się na później. Kazakov i Kudenko [Kaz2001] podają trzy podstawowe możliwości rozwiązania tego problemu, które można wykorzystać w zależności od stosowanej techniki uczenia się i dziedziny zastosowania.

- *Równoległe uczenie się i odwoływanie do pamięci* – oba procesy przebiegają jednocześnie.
- *Naprzemienne uczenie się i odwoływanie do pamięci* – agent działa w dwóch trybach: (i) w pierwszym aktywnie rozwiązuje problem, wybierając akcje na podstawie posiadanej wiedzy (bez możliwości jej aktualizacji) i zapisując obserwacje w celu ich późniejszego przetworzenia; (ii) w drugim trybie agent wyłącznie uczy się w oparciu o fakty zarejestrowane wcześniej; po przeanalizowaniu obserwacji mogą one być usunięte lub zachowane w zależności o tego, czy uczenie się przebiega inkrementacyjnie, czy wsadowo. Metody inkrementacyjne zazwyczaj są bardziej wskazane ponieważ mają mniejsze wymagania odnośnie mocy obliczeniowej i pamięci.
- *Proste uczenie się natychmiastowe i odłożone uczenie się złożone obliczeniowo*. Opisane wyżej, naprzemienne zapamiętywanie obserwacji i uczenie się może wymagać dużej ilości pamięci do zapisu faktów z otoczenia. W takiej sytuacji można zastosować proste przetwarzanie faktów (np. wstępne klasyfikowanie przypadków) zaraz po ich zarejestrowaniu, bez zatrzymywania normalnego działania agenta, natomiast precyzyjne, a więc i złożone obliczeniowo uczenie się uruchamiać w oddzielnym trybie. Podejście to przypomina nieco kognitywne procesy dotyczące pamięci krótko- i długoterminowej u człowieka.

2.5. Uczenie się systemu wieloagenckiego

Uczenie się w systemie wieloagenckim (ang. *multi-agent learning*, MAL) to grupa mechanizmów bardziej złożonych od uczenia się pojedynczego agenta, ponieważ zarówno przetwarzanie informacji, jak i miary oceny zdobywanej wiedzy dotyczą wielu agentów, działających we wspólnym środowisku. Uczenie się systemu wieloagenckiego zostało bardzo dogłębnie przeanalizowane w pracy Sena i Weissa [Sen1999]. Poniżej omawiamy tylko wybrane zagadnienia i kryteria podziału, przytaczane w pracach [Gue2003], [Kaz2001], [Sen1999], [Wei1996], [Wei1999].

2.5.1 Wspólny albo indywidualny cel uczenia się

Opierając się na wspomnianej już wcześniej definicji Weissa [Wei1996], możemy wyodrębnić dwa podstawowe rodzaje MAL: *silne* i *słabe*, które omawiamy poniżej.

Silne MAL

W systemie wieloagenckim, który wykazuje *silne uczenie się* (ang. *strong multi-agent learning*), agenty dążą do wspólnych celów, poprzez komunikację współpracują ze sobą i wzajemnie dzielą się wiedzą oraz doświadczeniem, przez co wzrastają globalne miary efektywności całego systemu. Realizacja silnego MAL jest zadaniem skomplikowanym, które musi się opierać na modelu wspólnej wiedzy, celu i działania agentów. Ogólne, logiczne modele tego typu omawiane są między innymi w pracy Fagina, Halperna, Mosesa i Vardiego [Fag1995], w której jednak zagadnienie MAL nie jest bezpośrednio poruszane.

Shen, Maturana i Norrie [She2000] przeanalizowali silne uczenie się systemu wieloagenckiego w dziedzinie zarządzania produkcją (ang. *manufacturing*), w szczególności koncepcje: *uczenia się na podstawie przeszłości* (ang. *learning from history*), wykorzystującego doświadczenie zgromadzone w systemie, i *uczenia się na podstawie przyszłości* (ang. *learning from the future*), opartego na mechanizmach prognostycznych.

Takadama, Nakasuka i Terano [Tak1998] opracowali wieloagencki system projektowania płytek drukowanych (ang. *Printed Circuit Board design*, PCBs), który opiera się na idei *organizacyjnego uczenia się* (ang. *organizational learning*).

Williams [Wil2004] zwrócił uwagę na bardzo ważny problem uzgadniania ontologii, który jest podstawą komunikacji i współpracy w MAS. Autor opracował metodologię, która opiera się na bezpośrednim dzieleniu ontologii pomiędzy poszczególnymi agentami, zamiast stosowanego wcześniej, uproszczonego założenia, że agenty mają wspólną ontologię (ang. *common ontology paradigm*). Proponowane podejście zostało wdrożone i zweryfikowane eksperymentalnie w systemie DOGGIE (ang. *Distributed Ontology Gathering Group Integration*).

Słabe MAL

Jeśli system wieloagencki uczy się w sposób *słaby* (ang. *weak multi-agent learning*), poszczególne agenty dążą do indywidualnych celów lub nawet rywalizują ze sobą (np. w negocjacjach cenowych), jednak ucząc się i podejmując decyzje biorą pod uwagę wiedzę i zachowanie innych agentów. Potencjalnie zwiększa to ich indywidualną

efektywność, ale też może w sposób nieplanowany (ang. *emergent*) wpływać korzystnie na działanie całego systemu (np. zmniejszać średni czas negocjacji). W słabym MAL świadomość poszczególnych agentów co do uczenia się całego systemu, z założenia nie istnieje lub jest mocno ograniczona, natomiast zachowanie globalne jest często rozpatrywane z perspektywy pewnego zewnętrznego obserwatora [Gss2004].

Crites i Barto [Cri1998] przeanalizowali eksperymentalnie uczenie się grupy agentów, z których każdy niezależnie steruje jedną windą, należącą do systemu wind. Uzyskane wyniki wskazują, że przy zastosowaniu algorytmów uczenia się ze wzmocnieniem indywidualnie przez każdego agenta, pojawia się nowy, kolektywny algorytm uczenia się całej grupy. Autorzy twierdzą, że tego typu podejście może być zastosowane do rozwiązywania dużych, złożonych obliczeniowo problemów sterowania.

Carmel i Markovitch [Crm1999] zaproponowali metodę uczenia się modeli innych agentów, wykorzystującą strategię eksploracji opartą na przeszukiwaniu z wyprzedzeniem (ang. *lookahead-based exploration strategii*). Metoda pozwala na zachowanie równowagi pomiędzy wykorzystaniem przez agenta posiadanej wiedzy oraz eksperymentowaniem w celu znalezienia nowych rozwiązań. Uwzględnia ona także ryzyko związane z eksperymentowaniem.

Van Bragt i La Poutré [Brg2003] wykazali, że zastosowanie uczenia się CS do adaptacji strategii negocjacyjnej zwiększa efektywność agentów w porównaniu z agentami o strategiach ustalonych i niezmiennych (ang. *fixed*).

Vidal i Durfee [Vid2003] opracowali ogólną, teoretyczną architekturę CLRI (ang. *Change*, *Learning*, *Retention*, *Impact*), przeznaczoną do modelowania i przewidywania globalnego zachowania systemu wieloagenckiego, złożonego z uczących się agentów. W proponowanej architekturze poszczególne agenty nie znają funkcji decyzyjnych innych agentów, lecz opierają strategię swojego działania na obserwacji otoczenia zewnętrznego.

Garland i Alterman [Gar2004] wprowadzili metodę uczenia się agentów, które posiadają indywidualne cele, lecz współpracują ze sobą wówczas, gdy ich cele się pokrywają (całkowicie lub częściowo). Weryfikacja eksperymentalna metody wykazała, że indywidualne uczenie się agentów wyraźnie poprawia wydajność całego systemu poprzez obniżenie kosztów planowania i komunikacji.

2.5.2 Świadomość istnienia innych agentów

Mimo iż uczący się system wieloagencki jest złożony w wielu agentów działających we wspólnym środowisku, nie gwarantuje to automatycznie, że dany agent wie o istnieniu innych agentów [Gue2003]. Vidal i Durfee [Vid1997] zaproponowali trzy klasy agentów w zależności od poziomu ich *świadomości* (ang. *awareness*) istnienia i działania innych agentów.

- *Agent poziomu* 0 (ang. 0-*level agent*) – nie jest w stanie rozpoznać faktu, że w otoczeniu są inne agenty. Agent ten może jedynie zaobserwować zmiany w środowisku, które są wynikiem działania innych agentów.
- *Agent poziomu* 1 (ang. 1-*level agent*) – rozpoznaje fakt istnienia innych agentów i wykonywania przez nie akcji, ale nie wie o nich nic więcej. Zadaniem agenta poziomu 1 jest dążenie do optymalnej strategii, która uwzględnia przewidywanie działań innych

agentów na podstawie posiadanej wiedzy, w tym obserwacji ich wcześniejszego zachowania.

- *Agent poziomu* 2 (ang. 2-*level agent*) – również zauważa istnienie i działanie innych agentów w otoczeniu, ale oprócz tego posiada ich tzw. *model intencjonalny* (ang. *intentional model*), który zawiera pewną informację o ich procesach decyzyjnych i obserwacjach. Najprostszym sposobem przekształcenia agenta poziomu 1 do poziomu 2 jest modelowanie przez niego innych agentów przy założeniu, że są one takie same jak on, czyli że takich samych algorytmów i obserwacji, jak dotychczasowy agent poziomu 1. Istnieje możliwość tworzenia agentów poziomu *n* – jeszcze głębszego od 2, ale nie jest to praktykowane ze względu na zmniejszającą się dokładność modeli i wzrastającą złożoność obliczeniową ich przetwarzania.

Gmytrasiewicz, Noh i Kellogg [Gmy1998] zaproponowali Bayesowską metodę aktualizacji przekonań agenta o intencjonalnych modelach innych agentów w oparciu o ich obserwowane zachowanie. Przedstawiona technika opiera się na formalizmie *metody modelowania rekurencyjnego* (ang. *Recursive Modeling Method*, RMM), w którym modele agentów mogą być zagnieżdżane do określonej liczby poziomów (np. na poziomie trzecim agent A posiada model, który stanowi wyobrażenie tego, jak agent B może modelować agenta A). Autorzy wykazali eksperymentalnie efektywność swojej metody w środowiskach symulacyjnych, realizujących problem obrony przeciwrakietowej i grę w wilki i owcę (ang. *the pursuit game*).

Hu i Wellman [HuW2001] przebadali uczenie się przez agenty modeli innych agentów w środowisku symulacyjnym aukcji elektronicznej. Wykazali oni, że efektywność uczenia się agenta nie zależy tylko od samej metody, ale także od ilości informacji, która jest dla niego dostępna. Jeśli bezpośrednia obserwacja akcji innych agentów jest ograniczona, agent musi się opierać na dowodach pośrednich – skutkach tych działań w otoczeniu, co może utrudniać lub ograniczać zdolności uczenia się. Autorzy uzyskali bardzo ciekawe wyniki efektywności uczenia się agentów na różnych poziomach: najgorsze rezultaty osiągały agenty poziomu 0 (co jest zgodnie z intuicyjnymi oczekiwaniami), ale wśród pozostałych najlepiej radziły sobie agenty o minimalnym poziomie zagnieżdżenia modeli (np. agenty poziomu 1). Można to wyjaśnić w ten sposób, że agenty o wyższym poziomie zagnieżdżenia posiadały więcej wiedzy na temat innych agentów, często jednak wiedza ta była obarczona dużym błędem i dlatego wyższą efektywność uzyskały agenty o mniejszej ilości wiedzy, ale za to bardziej wiarygodnej.

Guerra Hernández [Gue2003] przeprowadził analizę uczenia się agenta na poziomie 0, 1 i 2 (według [Vid1997]) w samodzielnie zaimplementowanej architekturze BDI, wykorzystującej metodę indukcji drzew decyzyjnych. Autor zaproponował hierarchiczną metodę zarządzania procesem uczenia się, w której wyższy poziom jest uruchamiany wówczas, gdy nie powiedzie się plan uczenia się na poziomie niższym.

2.5.3 Inne aspekty uczenia się w systemie wieloagenckim

Uczenie się i komunikacja

Komunikacja odgrywa bardzo ważną rolę w systemie wieloagenckim, warunkując lub znacząco wpływając na możliwość zaistnienia wspólnej wiedzy, celu i działania agentów.

Sen i Weiss [Sen1999] wyodrębniają i opisują dwa rodzaje relacji pomiędzy uczeniem się, a komunikacją w MAS.

- *Uczenie się komunikowania*. Mechanizmy uczenia się są wykorzystywane jako metoda udoskonalenia komunikacji między agentami. Na przykład agent może zdobywać wiedzę o preferencjach lub zdolnościach innych agentów, dzięki czemu, zamiast powiadamiania wszystkich agentów, jest on w stanie kierować komunikaty bezpośrednio do tych, które mogą przyczynić się do osiągnięcia jego bieżącego celu.
- *Komunikowanie się w celu uczenia się*. W tym przypadku komunikacja jest metodą wymiany informacji, która jest warunkiem uczenia się agentów lub ważnym czynnikiem na nie wpływającym. Uczenie się może być wspomagane przez: (i) komunikację *niskiego poziomu* (ang. *low-level communication*), służącą wymianie brakujących faktów i opartą na prostej interakcji typu *zapytanie – odpowiedź*; (ii) komunikację *wysokiego poziomu* (ang. *high-level communication*), w której interakcja jest bardziej złożona (np. negocjacje, wzajemne wyjaśnianie pojęć) i służy łączeniu wiedzy posiadanej przez poszczególne strony.

W obu przypadkach konieczne jest ustalenie wspólnej ontologii przez komunikujące się agenty tak, aby wykorzystywane symbole miały przypisane takie samo znaczenie. Problemowi temu poświęcona jest omawiana wcześniej praca Williamsa [Wil2004].

Uczenie się pojedynczego agenta i MAL

Weiss i Dillenbourg [Wei1999] wyróżniają trzy, częściowo nachodzące na siebie, rodzaje relacji pomiędzy indywidualnym uczeniem się agentów, a uczeniem się MAS.

- *Zwielokrotnienie* (ang. *multiplication*). Każdy agent uczy się samodzielnie i niezależnie od innych agentów, dążąc do indywidualnego celu, wskutek czego uczy się także cały system. Agenty mogą kontaktować się ze sobą, lecz komunikacja ta nie jest warunkiem uczenia się, a jedynie może dostarczać agentowi informacji wejściowej takiej, jak inne zdarzenia obserwowane w otoczeniu. Poszczególne agenty mogą mieć takie same lub różne algorytmy uczenia się. Podejście to jest charakterystyczne dla *słabego* uczenia się MAS i zostało zastosowane na przykład w omawianych wcześniej pracach [Cri1998], [Gar2004].
- *Podział* (ang. *division*). W systemie jest jeden wspólny cel uczenia się, a zadania służące jego osiągnięciu są dzielone pomiędzy poszczególne agenty. Podział może mieć charakter funkcjonalny (np. w systemie zarządzania produkcją opisanym w [She2000]) lub wynikać z fragmentacji danych (np. agenty wyszukujące informację w różnych domenach sieci WWW). W celu połączenia wyników uzyskanych przez poszczególne agenty konieczna jest interakcja między nimi, jednak jest ona określana przez projektanta i dotyczy tylko danych wejściowych i wyjściowych algorytmów uczenia się, nie wpływa natomiast na sam jego przebieg. Korzyści z tego podejścia są bardzo zbieżne do ogólnych zalet systemów wieloagenckich i przetwarzania rozproszonego (ang. *distributed problem solving*, DPS). Obejmują one uproszczenie projektu i zmniejszenie obciążenia poszczególnych agentów oraz ogólne zwiększenie wydajności systemu, dzięki rozproszeniu przetwarzania.

- *Interakcja* (ang. *interaction*). Podobnie jak w przypadku podziału, poszczególne agenty mają wspólny cel uczenia się, tutaj jednak komunikacja między nimi nie ogranicza się tylko do wyspecyfikowanej przez projektanta wymiany danych, lecz wpływa na przebieg pośrednich etapów uczenia się (np. poprzez negocjację lub wyjaśnianie). Jest to mechanizm najbardziej zaawansowany i najtrudniejszy do zaimplementowania, dlatego też obecnie jest mało dostępnych opracowań z tego zakresu [Kaz2001]. Gaines [Gai1997] zaproponował metodę silnego uczenia się systemu wieloagenckiego, w której poszczególnym agentom przydzielane są zadania o trudności rosnącej stopniowo, tak aby uczenie się danego agenta było na stałym poziomie. Autor wykazał, że takie podejście optymalizuje miary efektywności uczenia się w systemie.

3. Metoda APS inkrementacyjnego pozyskiwania reguł

Przedstawiona w niniejszym rozdziale, metoda APS (Analiza Przeszłych Stanów, ang. *Analysis of Past States*), jest oryginalną propozycją rozwiązania problemu pozyskiwania reguł, przedstawionego we Wstępie rozprawy[4]. W języku przedformalnym można ten problem sprecyzować następująco. **Dany jest przyrastający ciąg KB_H obserwacji agenta, uporządkowanych przez czas ich zarejestrowania i opisanych przez atrybuty należące do zbioru U. Przetwarzając pojedynczo k następujących po sobie podciągów $KB^{(1)}_H$, $KB^{(2)}_H$,...,$KB^{(k)}_H$ ciągu KB_H, wyznaczyć zbiór KB^I_R wszystkich reguł związku nad zbiorem atrybutów U w sumie tych podciągów tak, aby zbiór reguł KB^I_R był zbliżony do zbioru reguł KB^B_R, wyznaczonego wsadowo na całym ciągu obserwacji KB_H.**

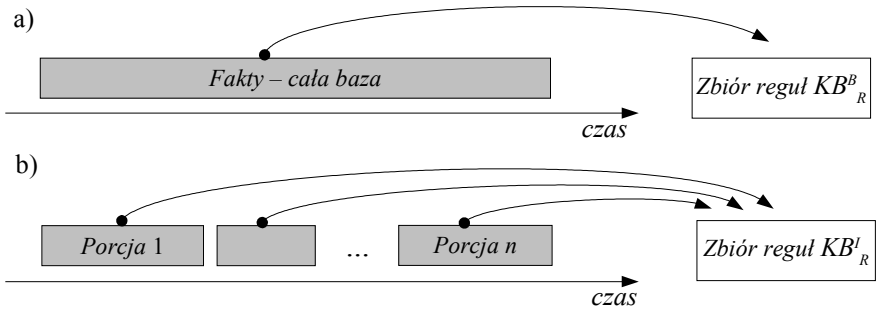
Układ rozdziału jest następujący. Pierwsza część zawiera ogólny, nieformalny przegląd metody. W podrozdziale drugim zestawione są założenia metody. Kolejna część zawiera definicję struktury bazy wiedzy agenta, która jest wykorzystywana podczas procesu pozyskiwania reguł. W czwartym podrozdziale zdefiniowany jest cykl działania metody, a więc poszczególne etapy i ich wzajemne relacje oraz interakcja z modułami bazy wiedzy agenta. Podrozdział piąty zawiera opis algorytmów, które są wykorzystywane w cyklu działania metody. Wraz z algorytmami podana jest także formalna analiza ich własności oraz złożoności obliczeniowej. W szóstym podrozdziale opisana jest eksperymentalna weryfikacja metody, w tym: charakterystyka środowiska i danych testowych, plan eksperymentów, uzyskane wyniki, ich opracowanie i omówienie. W końcowej części przedstawione jest porównanie metody APS z innymi pracami o podobnym zakresie tematycznym.

3.1. Przegląd metody APS

Podstawowym założeniem metody APS jest odkrywanie reguł w sposób inkrementacyjny, to znaczy na podstawie kolejnych przyrostów faktów rejestrowanych w bazie wiedzy agenta, bez konieczności rozpoczynania całej analizy od początku w razie dodania nowych danych trenujących. Przyjmujemy zatem, że agent w trakcie działania zapisuje obserwacje w odpowiednich strukturach swojej bazy wiedzy i analizuje je po zgromadzeniu wystarczająco

[4] Początkowa wersja metody APS opisana została we wcześniejszej pracy autora [Dud2003]. Formalizm i algorytmy prezentowane w niniejszej rozprawie zostały przedstawione w sposób skrócony w pracy [Dud2005b].

dużej porcji faktów, pod warunkiem, że przetwarzanie to jest możliwe przy uwzględnieniu zasobów systemowych[5] i bieżących zadań agenta. Jednocześnie zbiór reguł pozyskiwanych inkrementacyjnie powinien być zbliżony do zbioru reguł, uzyskanego wsadowo na całym zbiorze faktów, zarejestrowanych od początku (zob. Rys. 3.1). Miary podobieństwa zbiorów reguł są przedstawione w podrozdziale poświęconym eksperymentalnej weryfikacji metody.



Rys. 3.1. Odkrywanie reguł w trybie: (a) wsadowym, (b) inkrementacyjnym. Wynikowe zbiory reguł KB^B_R oraz KB^I_R powinny być do siebie zbliżone. Porcje faktów, analizowane w trybie inkrementacyjnym, nie muszą mieć ściśle jednakowego rozmiaru, ale powinny być porównywalnej wielkości. Rzeczywisty rozmiar każdej porcji faktów zależy od sytuacji, gdy: (i) agent zgromadzi reprezentatywną, czyli wystarczająco dużą liczbę obserwacji, zgodnie z programi ustalonymi na etapie projektowania systemu; (ii) istnieją odpowiednie warunki do uruchomienia analizy, tzn. agent nie ma bieżących zadań do wykonania – jest w fazie oczekiwania (ang. *stand-by phase*) oraz zasoby systemowe są wystarczająco mało obciążone, aby obsłużyć proces pozyskiwania reguł.

3.1.1 Struktura bazy wiedzy agenta

W metodzie APS zaproponowana jest odpowiednia struktura bazy wiedzy agenta, która pozwala na pozyskiwania reguł zgodnie z powyższymi założeniami. Baza wiedzy jest podzielona na 4 moduły funkcjonalne:

- KB_H – *historia* jest rejestrem obserwacji przeszłych stanów świata oraz interakcji agenta z otoczeniem zewnętrznym;
- KB_R – *baza reguł* jest zbiorem reguł związku, które są odkrywane na podstawie faktów z historii; reguły są odpowiednio umieszczane i utrzymywane w module KB_R przez algorytmy metody APS;
- KB_T – *wiedza chwilowa* (*pamięć krótkoterminowa*) opisuje bieżący stan agenta, otoczenia i przetwarzania rozwiązywanego problemu;
- KB_G – *wiedza ogólna* (*pamięć długoterminowa*) zawiera informacje o dziedzinie zastosowania, otoczeniu agenta, metodach rozwiązywania problemu i innych aspektach potrzebnych do działania agenta; tego typu wiedza pochodzi przede wszystkim (choć nie tylko) z fazy projektowania i implementacji agenta, jest ona zatem w dużej mierze *wiedzą odziedziczoną* (ang. *inherited knowledge*).

Rejestr stanów świata (czyli historia), prowadzony przez agenta może, w zależności od konkretnej aplikacji, obejmować ciąg zaobserwowanych stanów świata, czyli konfiguracji atrybutów świata w określonych momentach (np. jakie dokumenty w sieci WWW przegląda

[5] W pracy [Dud2002] została zarysowana koncepcja tzw. *funkcji oceny zajętości zasobów*, która jest zintegrowana z architekturą agencką i cyklicznie zwraca bieżący stopień wykorzystania zasobów systemowych (np. procesora, pamięci operacyjnej).

użytkownik o określonej porze dnia) lub zapis interakcji interakcji agenta z otoczeniem. W tym drugim przypadku jako pojedynczy fakt (obserwacja) może być zapisana transakcja (czyli zamknięty, niepodzielny ciąg operacji) postaci: *warunki – akcja – rezultat*. Rejestrowana interakcja składa się zatem z następujących zapisów.

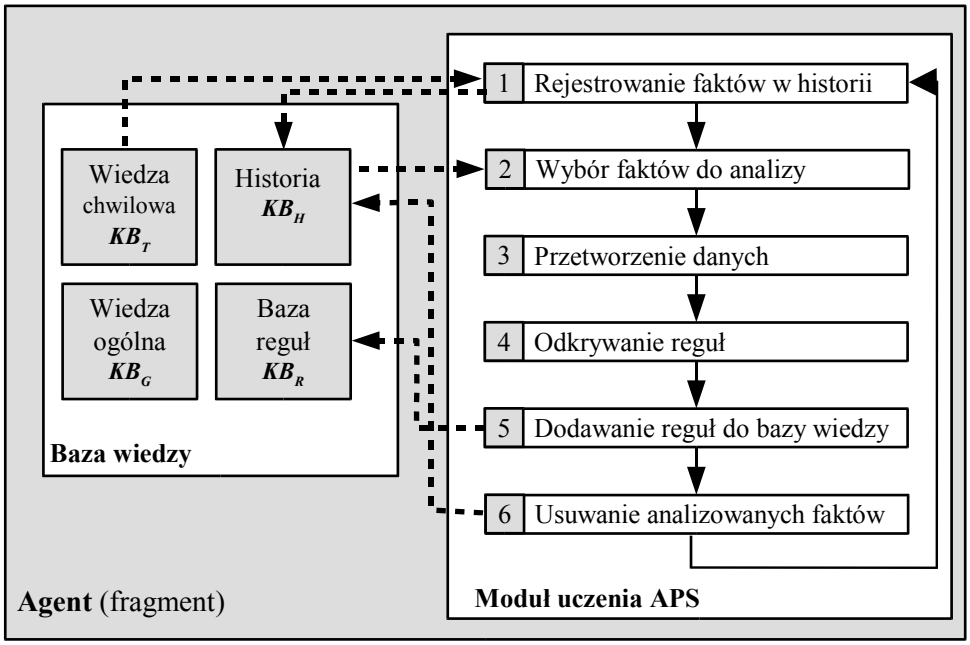
W danych *warunkach* (stanie świata) została przez agenta wykonana określona *akcja* i doprowadziła ona do określonego *rezultatu* w otoczeniu.

Na przykład: *gdy użytkownik zadał pytanie q (warunki)*, *agent zwrócił mu dokument d (akcja), który został przez użytkownika oceniony jako relewantny (rezultat).*

W pewnych zastosowaniach pojedyncze obserwacje atrybutów otoczenia lub faktów postaci *warunki – akcja – rezultat* nie mogą być podstawą do wyciągnięcia przez agenta wiarygodnych wniosków co do zależności dotyczących środowiska agenta lub jego interakcji z otoczeniem. Ewentualne reguły pozyskane na podstawie niewielkiej liczby faktów mogą być niewiarygodne statystycznie lub nawzajem ze sobą sprzeczne. Stąd też chcemy, aby agent przeprowadzał analizę dopiero po zgromadzeniu odpowiednio dużego zbioru faktów. Określenie minimalnego, wystarczającego rozmiaru zbioru obserwacji zależy od dziedziny zastosowania i powinno być wykonywane przez eksperta w fazie projektowania i implementacji systemu agenckiego.

3.1.2 Etapy metody APS

Cykl działania metody APS jest przedstawiony na Rys. 3.2. Podczas normalnego działania (ang. *performance phase*) agent realizuje swoje podstawowe cele projektowe, jednocześnie rejestrując w historii fakty – to znaczy zdefiniowane przez projektanta informacje o stanie świata (etap 1). Jeżeli: (i) w historii zgromadzona jest wystarczająca, reprezentatywna liczba faktów (zgodnie z miarami ustalonymi w fazie projektowania i zależnymi od konkretnej dziedziny), (ii) agent nie ma do wykonania żadnych bieżących zadań, czyli jest w fazie czuwania (ang. *stand-by phase*) oraz (iii) zasoby systemu nie wystarczająco mało obciążone,



Rys. 3.2. Etapy metody APS i ich relacje z modułami bazy wiedzy agenta.

wówczas wyzwalany jest przebieg pozyskiwania reguł (ang. *run*), który składa się z 5 kroków. Przebieg rozpoczyna się od wyboru faktów do analizy (etap 2). Następnie następuje przetworzenie wybranych danych do odpowiedniego formatu, wymaganego przez algorytm odkrywania reguł związku (etap 3). Przygotowane fakty są danymi wejściowymi algorytmu eksploracji danych (ang. *data mining*), który odkrywa i zwraca na wyjściu reguły związku (etap 4). Nowe reguły są następnie dodawane do bazy wiedzy z uwzględnieniem reguł, które były tam wcześniej (etap 5), przeanalizowane fakty są zaś trwale usuwane z historii (etap 6), po czym agent powraca do swojej normalnej aktywności.

Zakładamy, że w czasie trwania przebiegu pozyskiwania reguł (etapy 2 – 6) agent nie wykonuje zadań związanych z normalnym działaniem, lecz przeznacza wszystkie zasoby na jak najszybsze pozyskanie reguł z przetwarzanej porcji faktów. W cyklu działania agenta mamy więc do czynienia z przeplatającymi się fazami normalnej aktywności i uczenia się.

3.1.3 Przekształcanie danych

Format danych przekazywanych algorytmowi odkrywania reguł związku (w etapie 4.) wymaga, aby atrybuty opisujące fakty przyjmowały wyłącznie wartość 1 albo 0, bez wartości nieznanych. W metodzie APS zakładamy, że obserwacje (fakty) zapisywane w historii mogą być opisywane atrybutami nominalnymi, czyli atrybutami, których dziedziny są skończonymi zbiorami dyskretnych wartości. Niedopuszczalne są atrybuty o dziedzinach ciągłych (np. takich, jak zbiór liczb rzeczywistych), jeśli zaś użycie takich atrybutów wynika ze specyfiki dziedziny zastosowania systemu, zakładamy, że przed zapisaniem faktu w historii ich wartości są odpowiednio dyskretyzowane. Sama technika dyskretyzacji nie wchodzi jednak w zakres metody APS. W trakcie etapu 3 następuje przygotowanie faktów do formatu, który jest odpowiedni dla algorytmu znajdowania reguł związku. Dzieje się to w trzech krokach:

1) przekształcenie schematu historii;
2) wypełnienie nowego schematu danymi;
3) eliminacja wartości nieznanych.

W pierwszym kroku tworzony jest nowy schemat historii, w którym atrybutami stają się wartości atrybutów w schemacie pierwotnym. Nie są to jednak wszystkie możliwe wartości, wynikające z dziedzin atrybutów, ale tylko te wartości, które rzeczywiście wystąpiły w przetwarzanej porcji faktów. Dzięki temu nowy schemat zawiera minimalną liczbę atrybutów.

Drugi krok polega na wypełnieniu nowego schematu historii danymi na podstawie faktów zgromadzonych w bieżącej porcji pobranej z historii. Atrybuty w przekształconym schemacie przyjmują wartości 1, 0 lub N (wartość nieznana).

W ostatnim kroku przygotowania faktów usuwane są nieznane wartości atrybutów. W metodzie APS przyjęta została technika generowania nowych faktów, które zamiast wartości nieznanej danego atrybutu zawierają kolejno jego wszystkie możliwe wartości. W języku logicznych modeli wiedzy agentów, nowe fakty, powstające na podstawie nieznanych wartości atrybutów, mogą być traktowane jako *światy możliwe* (ang. *possible worlds*) [Fag1995] w danym stanie, którym jest pierwotny fakt (z nieznanymi wartościami niektórych atrybutów). Rozwiązanie to pozwala na szybkie wyeliminowanie wartości nieznanych bez straty wartości informacyjnej zbioru przetwarzanych faktów. Z drugiej strony generowanie wszystkich

możliwości powoduje gwałtowny, wykładniczy wzrost liczby faktów do przetworzenia i dlatego może ono być stosowane z ograniczeniami, to znaczy tylko dla pewnej przyjętej, maksymalnej liczby wartości nieznanych w pojedynczym fakcie. Jeśli natomiast liczba nieznanych wartości przekracza przyjęty próg, fakt jest usuwany bez zastępowania go wszystkimi możliwościami. Jest to uzasadnione nie tylko ze względów wydajnościowych, ale też dlatego, że fakty z dużą liczbą nieznanych wartości atrybutów niosą ze sobą małą wartość informacyjną z punktu widzenia opisu przeszłego stanu i odkrywanych na tej podstawie reguł.

Istotną cechą, przyjętego w metodzie APS, algorytmu eliminacji wartości nieznanych jest założenie *światów dowolnych* (ang. *random worlds*) [Bac1996], zgodnie z którym wszystkie fakty generowane na podstawie faktu z nieznanymi wartościami atrybutów, a więc wszystkie *światy możliwe* w danym stanie, są traktowane jako jednakowo prawdopodobne, bez ustalonego porządku. Dzięki temu nowe, możliwe fakty mogą być szybko wygenerowane bez konieczności wcześniejszego analizowania historii w celu obliczenia rozkładu prawdopodobieństwa poszczególnych wartości atrybutów, a w konsekwencji – prawdopodobieństwa poszczególnych światów możliwych. Autorzy koncepcji światów dowolnych wykazali, że ich podejście pozwala na efektywne wnioskowanie w bogatych informacyjnie bazach wiedzy, zawierających dużo danych statystycznych [Bac1996]. Jako takie podejście to zostało zatem zaadaptowane w metodzie APS. Zauważmy jednak, że rozwiązanie to przynosi zadowalające wyniki tylko przy spełnieniu odpowiednich warunków: (i) liczba wartości nieznanych w faktach jest stosunkowo niska (w przeciwnym razie wysoka, wykładnicza złożoność procesu generowania światów możliwych silnie obniży ogólna wydajność metody); (ii) jest bardzo mało (a najlepiej nie ma w ogóle) wartości *N* w kolumnach predykcyjnych, to znaczy w atrybutach, które zgodnie z zadanymi ograniczeniami semantycznymi mają wystąpić w następnikach reguł. W tym drugim przypadku wygenerowane fakty ze wszystkimi możliwościami wartości kolumny predykcyjnej i tak nie mają większej wartości informacyjnej, gdyż reprezentują one jedynie jednorodny rozkład prawdopodobieństwa wystąpienia poszczególnych wartości – zgodnie z założeniem światów dowolnych.

3.1.4 Odkrywanie reguł związku

Znajdowanie reguł związku na podstawie przetworzonych faktów (etap 4) odbywa się zgodnie z wybranym algorytmem, np. Apriori. Metoda APS jest całkowicie zewnętrzna w stosunku do algorytmu odkrywania reguł związku, to znaczy traktuje go jako niezależny moduł, do którego przekazywane są dane wejściowe (odpowiednio przygotowane fakty) oraz parametry (np. progi minimalnego poparcia i pewności) i z którego pobierane są dane wyjściowe: reguły związku wraz z towarzyszącymi miarami statystycznymi (poparcie, pewność). A zatem metoda APS względem samego algorytmu eksploracji danych zachowuje się jak odrębny proces zarządzający. Zaletą przyjętej metody jest ogólność i elastyczność, która pozwala na wykorzystanie różnych algorytmów odkrywania reguł związku, w zależności od wymogów danego zastosowania i środowiska, w którym pracuje agent. Inną korzyścią jest możliwość łączenia ze sobą reguł związku, które są odkrywane przez różne algorytmy, co z kolei pozwala na współdzielenie wiedzy w systemach wieloagenckich.

Zapewnienie pełnej niezależności procesu APS od algorytmu odkrywania reguł wiąże się z poniesieniem pewnych kosztów. Bez możliwości ingerencji w algorytm eksploracji danych nie można uzyskać pewnych danych, których znajomość zwiększa dokładność wynikowego

zbioru reguł. Na przykład bez dodatkowych przebiegów analizy zbioru faktów nie można ustalić częstości niektórych zbiorów atrybutów, ani średniego poparcia reguł, które są odrzucane (jako zbyt mało wiarygodne). Aby zniwelować wspomniane braki danych, w metodzie APS przyjęte zostały pewne oszacowania, które, choć nie pozwalają na uzyskanie doskonałej dokładności uzyskiwanych reguł (w stosunku do metod wsadowych), minimalizują błędy obliczanych wartości.

3.1.5 Założenie niedoskonałej pamięci

Kasowanie obserwacji z historii po ich przetworzeniu i odkryciu reguł związku (etap 6), jest oparte na założeniu *niedoskonałej pamięci* (ang. *imperfect recall*) [Fag1995], w myśl którego agent nie przechowuje w bazie wiedzy wszystkich obserwacji przeszłych stanów przez cały czas swojego działania. A zatem agent o niedoskonałej pamięci z założenia nie może odtworzyć każdego stanu zaobserwowanego w przeszłości. Zgodnie z tym, co zostało zauważone wcześniej, w wielu zastosowaniach elementarne fakty i tak nie stanowią żadnej wartości dla agenta, ponieważ są zbyt szczegółowe, za mało wiarygodne statystycznie lub wzajemnie sprzeczne. W takiej postaci zatem nie mogą one być wykorzystane w procesie decyzyjnym agenta. Dodatkową, poważną wadą *doskonałej pamięci* jest także sam rozmiar historii, który ciągle się zwiększa i w pewnym momencie może nadmiernie obciążać zasoby systemu. Z tego względu w metodzie APS pojedyncze fakty są gromadzone tylko do momentu ich przetworzenia przez algorytmy pozyskiwania reguł. Przeanalizowana porcja obserwacji jest następnie usuwana, dzięki czemu rozmiar historii nie jest zwiększany bez ograniczeń, lecz stale utrzymywany na akceptowalnym poziomie. W bazie wiedzy, zamiast pojedynczych faktów, agent trwale przechowuje uogólnione, wiarygodne statystycznie reguły, które z tych faktów wynikają. Reguły mogą być wykorzystane w procesie decyzyjnym agenta (np. przy tworzeniu lub wybieraniu planu) i potencjalnie stopniowo polepszać jego skuteczność lub efektywność poprzez adaptację agenta do otoczenia, w którym się znajduje.

Odnosząc się do pojęć *wiedzy jawnej* (ang. *explicit knowledge*) i *wiedzy niejawnej* (ang. *implicit knowledge*), wprowadzonych przez Levesque [Lev1984], można stwierdzić, że fakty zapisywane w historii stanowią wiedzę jawną, na podstawie której, w trakcie procesu APS, jest wywnioskowywana indukcyjnie wiedza niejawna – reguły, które pierwotnie są *ukryte* w faktach. Choć z formalnego punktu widzenia dodawanie do bazy agenta wiedzy niejawnej (reguł), a więc wywnioskowanej na podstawie wiedzy jawnej (faktów), prowadzi do nadmiarowości wiedzy, jest to w pełni uzasadnione wspomnianą wyżej, ograniczoną przydatnością elementarnych faktów, przy potencjalnie dużej użyteczności i wartości informacyjnej ich konsekwencji – reguł związku.

3.1.6 Utrzymanie bazy reguł

Niezwykle newralgiczną częścią procesu APS jest etap 5., w którym baza reguł KB_R jest modyfikowana na podstawie reguł odkrytych w ostatnim przebiegu oraz danych opisujących ten przebieg. Stosowany tutaj algorytm utrzymania bazy reguł KB_R jest najważniejszą częścią metody APS, ponieważ decyduje on o tym, jak zbliżony będzie wynik inkrementacyjnego pozyskiwania reguł do wyniku przetwarzania wsadowego (na całym zbiorze) (zob. Rys. 3.1), który jest ostatecznym punktem odniesienia dla dokładności metod przyrostowych. Algorytm

ten opiera się na obserwacji, że przy odkrywaniu reguł związku przetwarzane fakty mogą być traktowane jako zbiór, a nie ciąg o ściśle ustalonej sekwencji czasowej. Własność ta jest spełniona także po zastąpieniu faktów z nieznanymi wartościami wszystkimi możliwościami, zgodnie ze wspomnianym wyżej założeniem światów dowolnych. W tej sytuacji inkrementacyjne dodawanie reguł może być uproszczone do obliczania statystycznych miar wiarygodności reguł (poparcia i pewności) w oparciu o wzory proporcji częstościowych. Przypuśćmy, że do pewnego momentu proces APS, po przeanalizowaniu b_1 faktów z porcji h_1 (tzn. z części historii), odkrył regułę r z wartością poparcia i pewności odpowiednio $sup_1(r)$ oraz $con_1(r)$. Po kolejnym przebiegu, w którym przetworzone zostało b_2 faktów, znaleziona została ta sama reguła r, ale z nowymi wartościami $sup_2(r)$ oraz $con_2(r)$. Można wykazać, że wartości poparcia i pewności reguły r w połączonym zbiorze faktów h_1 i h_2 są dane wzorami (odpowiednie dowody są zawarte w dalszej części rozdziału):

$$sup(r, h_1 \cup h_2) := \frac{b_1\, sup_1(r) + b_2\, sup_2(r)}{b_1 + b_2}, \tag{3.1}$$

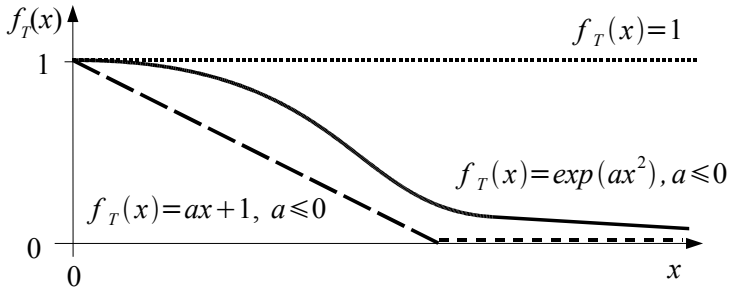
$$con(r, h_1 \cup h_2) := \frac{con_1(r)\, con_2(r)\left(b_1\, sup_1(r) + b_2\, sup_2(r)\right)}{b_1\, sup_1(r)\, con_2(r) + b_2\, sup_2(r)\, con_1(r)}. \tag{3.2}$$

Powyższe wzory gwarantują, że baza reguł uzyskana inkrementacyjnie jest identyczna z bazą pochodzącą z przetwarzania wsadowego, o ile wszystkie reguły występują zarówno w h_1 jak i h_2. Tymczasem założenie to jest mocno wyidealizowane i może nie być spełnione w rzeczywistych aplikacjach. Przede wszystkim nie są tutaj wzięte pod uwagę ograniczenia, np. progi minimalnego poparcia i pewności, które są powszechnie stosowane w algorytmach eksploracji danych. Zastosowanie ograniczeń komplikuje inkrementacyjne dodawanie reguł z wykorzystaniem powyższych wzorów proporcji częstościowych. Rozważmy następujący przykład. Przypuśćmy, że przed rozpoczęciem odkrywania reguł ustalone zostały progi minimalnego poparcia i pewności, odpowiednio σ i γ. Następnie, tak jak poprzednio, uruchamiane są dwa kolejne przebiegi odkrywania reguł: najpierw na porcji faktów h_1, a następnie na porcji h_2. Reguła r jest odkrywana w obu przebiegach, ale w drugim przebiegu jest ona odrzucana z powodu nie osiągnięcia wymaganych progów σ lub γ. Wówczas wzór (3.1) zwraca nieprawidłowy wynik, ponieważ wartość sup_2 jest przyjmowana jako 0 (zero), podczas gdy w rzeczywistości musiała ona się mieścić w przedziale obustronnie otwartym $(0; \sigma)$, skoro reguła r została znaleziona w tym przebiegu. Z kolei wyrażenie (3.2) w ogóle przyjmuje wartość nieokreśloną (kwestia ta jest poruszana w dalszej części rozdziału), nie pozwalając na obliczenie pewności. W celu wyeliminowania tych anomalii w metodzie APS zastosowano dwa estymatory $\hat{\sigma}$ oraz $\hat{\gamma}$, które reprezentują odpowiednio oczekiwane poparcie i pewność losowo wybranej reguły, która jest odrzucana. A zatem, w opisanej wyżej sytuacji (gdy reguła r nie występuje w zbiorze reguł odkrytych w porcji h_2), zamiast zerowych wartości $sup_2(r)$ i $con_2(r)$, do wzorów podstawiane są odpowiednio wartości estymatorów, które mogą być ustalone arbitralnie przez projektanta, albo przyjęte domyślnie jako środek przedziału, to znaczy: $\hat{\sigma} = \frac{\sigma}{2}$, $\hat{\gamma} = \frac{\gamma}{2}$. Istnieje także możliwość dokładnego obliczenia $\hat{\sigma}$ oraz $\hat{\gamma}$ jako średnich wartości poparcia i pewności reguł rzeczywiście odrzucanych. To rozwiązanie wymagałoby jednak modyfikacji algorytmów metody APS, polegającej albo na ingerencji w algorytm odkrywania reguł (przez co w konsekwencji utracona byłaby niezależność metody

od algorytmu eksploracji danych), albo na dodaniu algorytmu filtrowania reguł na podstawie progów σ i γ, zewnętrznego w stosunku do algorytmu odkrywania reguł związku. W tym drugim przypadku proces znajdowania reguł uruchamiany byłby przy zerowych progach minimalnego poparcia i pewności, co miałoby niekorzystny wpływ na jego wydajność, z powodu potencjalnie bardzo dużej liczby zwracanych reguł (których duża część i tak byłaby potem odrzucana).

Przy inkrementacyjnym odkrywaniu i dodawaniu reguł do bazy KB_R, w oparciu o opisane wyżej wzory częstościowe, może się pojawić pewne niekorzystne zjawisko, które można określić mianem *oporności bazy reguł na zmiany*. Otóż skoro poparcie i pewność reguł są ustalane proporcjonalnie do liczby faktów, na podstawie których reguły te zostały odkryte, po długim czasie działania metody APS zbiór reguł KB_R jest coraz mniej podatny na jakiekolwiek zmiany na podstawie wyników kolejnych przebiegów analizy. Dzieje się tak dlatego, że bazowa liczba faktów dotychczasowych reguł w KB_R jest niewspółmiernie duża w stosunku do liczby faktów przetwarzanych w pojedynczym przebiegu (czyli pojedynczej porcji faktów z historii). W tej sytuacji, po upływie pewnego czasu baza KB_R będzie praktycznie niezmienna, nawet jeśli w rzeczywistości, w obserwacjach rejestrowanych przez agenta zachodzą zmiany, które powinny być uwzględnione. W celu wyeliminowania tego efektu, do wzorów częstościowych została wprowadzona *funkcja wpływu czasowego* f_T, która zmniejsza znaczenie reguł wraz ze zwiększającym się upływem czasu od ich odkrycia. Zakładamy, że kształt funkcji zależy od konkretnej dziedziny zastosowania i jest ustalany przez projektanta systemu. W metodzie APS definiowane są jedynie ogóle wymogi względem tej funkcji, które mówią, że ma to być funkcja nierosnąca, określona na przedziale [0; +∞), z wartościami należącymi do przedziału [0; 1], przy czym wartość dla 0 (zera) musi być równa 1 (jeden). Przykłady funkcji, spełniających te postulaty, są przedstawione na Rys. 3.3.

Zastosowanie funkcji wpływu czasowego w, przytoczonych wyżej, wzorach częstościowych (3.1) i (3.2), polega na mnożeniu poparcia reguły przez wartość f_T dla różnicy czasu teraźniejszego t_{now} i średniego czasu tej reguły. W ten sposób reguły nowe zyskują swoistą przewagę nad regułami starymi o tych samych początkowych miarach poparcia i pewności, gdyż miary te są odpowiednio zmniejszane wraz z upływem czasu. Własności wzorów proporcji częstościowych z funkcją f_T są przedstawione i dowodzone formalnie w dalszej części rozdziału. W danym przebiegu analizy faktów funkcja f_T nie jest stosowana indywidualnie do czasu każdego faktu, ale jednorazowo do średniego czasu wszystkich faktów w przetwarzanej porcji danych. Choć rozwiązanie to czyni obliczenia przybliżonymi, jest ono uzasadnione względami wydajności przetwarzania (średni czas wszystkich faktów można szybko obliczyć już podczas ich wyboru w etapie 2.) oraz utrzymaniem niezależności metody APS od algorytmu odkrywania reguł związku.



Rys. 3.3. Przykłady funkcji wpływu czasowego f_T. Argumentem x jest czas, który upłynął od danego momentu w przeszłości do chwili obecnej.

Przytoczony powyżej, ogólny i nieformalny opis metody APS, jest precyzowany w kolejnych podrozdziałach.

3.1.7 Klasyfikacja metody APS

Klasyfikacja metody APS zgodnie z kryteriami podziału metod uczenia się, podanymi w Rozdziale 1., jest następująca. Pod względem reprezentacji wiedzy [Cic2000] APS jest metodą symboliczną, ponieważ wszystkie struktury wiedzy agenta, biorące udział w cyklu pozyskiwania reguł, przechowują informację w postaci symboli. Dotyczy to w szczególności wykorzystywanej reprezentacji odkrywanych reguł związku.

W odniesieniu do źródła i postaci informacji trenującej [Cic2000] APS należy umiejscowić w klasie metod uczenia się bez nadzoru [Has2001], gdyż bazuje ona na regułach związku, które powstają w wyniku analizy i przekształcania przestrzeni zmiennych wejściowych, bez zewnętrznej informacji o prawidłowych odpowiedziach wyjściowych.

Posługując się terminologią Cichosza [Cic2000], można stwierdzić, że metoda APS pracuje w trybie epokowym, jako że proces pozyskiwania reguł odbywa się w cyklach, w których kolejno przetwarzane są porcje przykładów trenujących (faktów historii). Do czasu przetworzenia kolejnej epoki, system uczący się może korzystać z wyników uzyskanych we wcześniejszych cyklach. Warto tutaj wyjaśnić, dlaczego w wielu miejscach rozprawy używane jest określenie metody APS jako *inkrementacyjnej*, skoro aktualizacja reguł nie następuje po każdym dodaniu pojedynczego faktu do historii, ale co pewien czas, po zgromadzeniu określonej liczby obserwacji. Otóż, w literaturze międzynarodowej określenie *epokowy* w odniesieniu do metod znajdowania reguł związku praktycznie nie jest stosowane. Natomiast powszechnie używane są terminy *metoda wsadowa* oraz *metoda inkrementacyjna*, także dla algorytmów, które dopuszczają retencję przykładów trenujących (jak np. FUP_2 [Che1997], DELI [LeS1998]). Dlatego też autor podjął decyzję o zastosowaniu terminologii, która będzie zrozumiała dla szerokiego grona odbiorców.

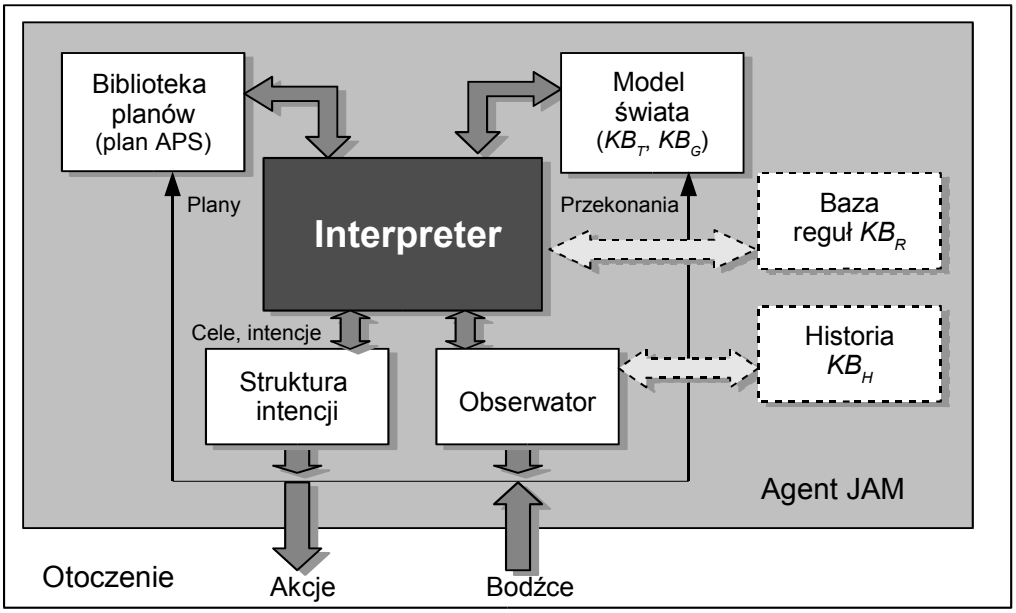
W końcu, w odniesieniu do klasyfikacji omawianej przez Maloofa i Michalskiego [Mal2000], [Mal2004], metoda APS jest metodą o częściowej pamięci przykładów (ang. *partial instance memory*). W historii przechowywane są wyłącznie nowe fakty, które nie zostały jeszcze przetworzone. Obserwacje te są zaś trwale usuwane (zapominane) po przeanalizowaniu w przebiegu cyklu pozyskiwania reguł.

3.1.8 Umiejscowienie cyklu metody APS w architekturze agenckiej

W świetle definicji przytoczonych w Rozdziale 1. system uczący się powinien charakteryzować się udoskonalaniem swojego działania wraz z rosnącym doświadczeniem i nabywaną wiedzą. To samo odnosi się w szczególności do uczącego się agenta (zob. Rozdział 2.), którego zdolności adaptacyjne wiążą się ściśle z umiejętnością zastosowania gromadzonej wiedzy do odpowiedniego modyfikowania własnej, indywidualnej procedury decyzyjnej. W tym kontekście model pozyskiwania wiedzy, wyznaczony przez metodę APS, nie może być traktowany jako pełny proces uczenia się agenta, ponieważ nie zawiera on bezpośredniego odniesienia do procesu decyzyjnego. Zakładamy jednak, że w omawianej wcześniej, ogólnej architekturze uczącego się agenta według Russela i Norviga [Rus1995], cykl metody APS wraz z jego poszczególnymi etapami i algorytmami, jest umiejscowiony w wyodrębnionym

module uczenia się (ang. *learning element*), który współpracuje z innymi modułami systemu (zob. Rys. 2.1. w rozdziale 2.4.). Z kolei realizacja wymaganej przez metodę APS struktury bazy wiedzy jest zależna od konkretnej architektury agenckiej, w której osadzony jest cykl APS. A zatem, mimo iż sama metoda APS nie definiuje pełnego procesu uczenia się, stanowi ona z założenia istotną jego część.

Jako przykład rozważmy możliwości osadzenia cyklu metody APS we wspomnianej wcześniej architekturze JAM [Hub1999], która łączy w sobie elementy *systemów wnioskowania proceduralnego* (ang. *Procedural Reasoning Systems*, PRS) [Ing1992], w tym głównie UMPRS [Lee1994b], a także rozszerzenia proceduralne zaczerpnięte z *semantyki obwodów strukturalnych* (ang. *Structured Circuit Semantics*, SCS) [Lee1994a] oraz języka *Act* [Mye1997].



Rys. 3.4. Architektura agencka *JAM* (na podstawie [Hub1999], str. 237) wraz z osadzonymi elementami metody APS.

Agent *JAM* składa się z (zob. Rys. 3.4.): *modelu świata* (ang. *world model*), *biblioteki planów*, *interpretera*, *struktury intencji* (ang. *intention structure*) oraz *obserwatora* (ang. *observer*). *Model świata* zawiera fakty w postaci *relacja-argument* (np. zmienne stanu, wyniki wnioskowania, dane sensoryczne, komunikaty), reprezentujące bieżący stan świata, znany agentowi. *Plany* agenta zawierają proceduralną specyfikację sposobu osiągnięcia określonych celów, reagowania na określone zdarzenia lub wykazywania przez agenta pewnego zachowania. Pojedynczy plan definiowany jest m.in. przez cel, warunki początkowe, kontekst, ciało i użyteczność. *Interpreter* – główny mechanizm wnioskujący – jest działającą w pętli procedurą, odpowiedzialną za wybór i wykonywanie planów w oparciu o intencje, plany, cele oraz przekonania dotyczące bieżącego stanu świata. Sprzężona z interpreterem *struktura intencji* jest stosem gromadzącym cele (z przypisanymi planami i bez). Wszystkie plany, które pasują do danego celu i aktualnego stanu świata umieszczane są na *liście stosowalnych planów* (ang. *Applicable Plan List*, APL) i mają nadawane wartości *użyteczności* (ang. *utility*). Plan o największej użyteczności wybierany jest przez interpreter jako *intencja* na rzecz danego celu, i wykonywany. *Cele* agenta *JAM* mogą być trojakiego rodzaju: *cel do osiągnięcia* (ang.

ACHIEVE goal), *cel do wykonywania* (ang. *PERFORM goal*), *cel do utrzymania* (ang. *MAINTAIN goal*). *Obserwator* jest składnikiem odpowiedzialnym głównie za cykliczne śledzenie i odbieranie asynchronicznych zdarzeń w otoczeniu (np. przesłanych komunikatów).

Przy osadzeniu w architekturze JAM struktur wymaganych przez metodę APS, wiedza chwilowa KB_T i wiedza ogólna KB_G umieszczone byłyby w obrębie modelu świata. Baza reguł KB_R także może być częścią modelu świata, jednak przy dużej liczbie reguł rozwiązanie takie mogłoby cechować się niską wydajnością (powolne wyszukiwanie i modyfikowanie wobec braku struktur indeksowych). Wówczas baza reguł powinna być wyodrębnioną, dodatkową strukturą, najlepiej przechowywaną w postaci tabeli relacyjnej bazy danych, co pozwalałoby na jej szybkie przeszukiwanie i aktualizację. Podobnie, ze względów wydajnościowych, historia KB_H również musiałaby być zaimplementowana jako dodatkowa struktura danych (tabela), cyklicznie uzupełniana przez procedurę obserwatora.

Procedura `Pozyskuj_Reguły_APS` (omawiana szczegółowo w rozdziale 3.4.), realizująca cykl metody APS, w architekturze JAM może być zaimplementowana jako plan, który jest przechowywany w bibliotece planów. Jego warunki początkowe (ang. *precondition*), od których zależy uruchomienie, obejmowałyby minimalną liczbę faktów w historii (ustaloną przez projektanta) oraz niską zajętość zasobów systemowych (zwracaną jako wartość pewnej funkcji [Dud2002], zależnej od aplikacji i platformy sprzętowo-programowej). Dodatkowo potrzebna jest tutaj procedura *metawnioskowania* (ang. *metareasoning*) w formie odpowiedniego planu, która zadecyduje w kontekście wszystkich planów na liście APL, czy plan przebiegu APS może być uruchomiony w danym momencie. Jest to konieczne, ponieważ zgodnie z założeniami metody APS, pozyskiwanie reguł nie powinno zaburzać bieżącego działania agenta, lecz powinno być uruchamiane w okresach bezczynności (ang. *stand-by phase*).

Jak widać, osadzenie cyklu APS w architekturze JAM nie wymaga modyfikacji jej podstawowych komponentów, takich jak interpreter, model świata, biblioteka planów i struktura intencji. Sposób implementacji procedury obserwatora, aktualizującej historię zależy od konkretnej dziedziny zastosowania, co jest zresztą zgodne z założeniami architektury JAM. Natomiast ze względu na szybkość przetwarzania danych, wskazane byłoby wyposażenie agenta w dodatkowe, wydajne struktury historii i bazy reguł, które byłyby odpowiednio wykorzystywane w planach, realizujących cykl APS.

3.2. Założenia metody APS

W metodzie Analizy Przeszłych Stanów (APS) przyjęto poniższe założenia, wyrażone w języku przedformalnym.

(R1) Agent rejestruje obserwacje stanów świata w zbiorze nazywanym *historią*. Każdy zapis w historii, nazywany *faktem*, jest identyfikowany za pomocą unikalnego klucza i posiada atrybut opisujący unikalny moment jego zarejestrowania (ang. *time stamp*). Fakty zgromadzone w historii są uporządkowane rosnąco względem momentu zarejestrowania.

(R2) Wszystkie fakty zapisane w historii są opisywane za pomocą wartości tych samych atrybutów. Dopuszczalne są wyłącznie dyskretne (przeliczalne) zbiory wartości atrybutów. Atrybuty, wraz z unikalnym kluczem i momentem zarejestrowania faktu, tworzą *schemat historii*.

(R3) Nieznana wartość danego atrybutu w określonym fakcie jest jawnie zapisywana jako wartość *N*.

(R4) W ramach pojedynczego przebiegu pozyskiwania reguł znajdowane są reguły związku na podstawie jednego podciągu faktów wybranych z historii, nazywanego *porcją*.

(R5) Przebieg pozyskiwania reguł jest uruchamiany po zaistnieniu zdarzenia wyzwalającego (ang. *triggering event*), zewnętrznego w stosunku do metody APS.

(R6) Porcja, w której czasy zarejestrowania najwcześniejszego i najpóźniejszego faktu wynoszą odpowiednio t_1 i t_2, musi zawierać wszystkie fakty należące do historii, który czas zarejestrowania mieści się w przedziale $[t_1; t_2]$. Jeśli bieżący przebieg nie jest pierwszym przebiegiem w cyklu APS, najwcześniejszy fakt bieżącej porcji musi być bezpośrednio następnym, zarejestrowanym faktem po najpóźniejszym fakcie w porcji przetworzonej w poprzednim przebiegu.

(R7) W każdym przebiegu fakty należące do bieżącej porcji są przekształcane do postaci wymaganej przez algorytm odkrywania reguł związku.

(R8) Eliminacja nieznanej wartości atrybutu A w określonym fakcie s_i odbywa się po przekształceniu faktów, zgodnie z (R7), poprzez zastąpienie tego faktu nowymi faktami $\{s_i^{(1)}, s_i^{(2)}, ..., s_i^{(n)}\}$, których wartości czasu zarejestrowania są równe czasowi zarejestrowania s_i, a wartości atrybutu A odpowiadają wszystkim dopuszczalnym wartościom atrybutu A. Wszystkie nowo wygenerowane fakty są wzajemnie równorzędne i traktowane są jako jednakowo prawdopodobne.

(R9) Dane wejściowe przekazywane do algorytmu odkrywania reguł związku obejmują fakty przekształcone zgodnie z (R7) i (R8) oraz ograniczenia (ang. *constraints*): minimalne poparcie, minimalną pewność i atrybuty, które mogą wystąpić w poprzedniku albo następniku reguły.

(R10) Dane wyjściowe zwracane przez algorytm odkrywania reguł związku obejmują reguły związku wraz z wartościami ich poparcia i pewności w przetworzonej porcji faktów.

(R11) Reguła związku, odkryta w danym przebiegu, jest zapisywana w bazie reguł agenta wraz następującymi danymi: poparcie, pewność, liczba bazowa i średni czas (odpowiednio liczba i średni czas zarejestrowania przeanalizowanych faktów, stanowiących bazę reguły).

(R12) Przed dodaniem określonej reguły do bazy reguł, obliczane są dla niej zaktualizowane wartości: poparcia, pewności, liczby bazowej i średniego czasu, z uwzględnieniem reguł przechowywanych dotychczas w bazie wiedzy. Reguła jest dodawana do bazy reguł, jeżeli: (i) nie ma jej w bazie reguł; (ii) zaktualizowane wartości poparcia i pewności spełniają aktualne, globalne wymagania minimalnego poparcia i pewności.

(R13) Po odkryciu nowych reguł przez algorytm eksploracji danych, dla każdej reguły przechowywanej dotychczas w bazie reguł, obliczane są zaktualizowane wartości: poparcia, pewności, liczby bazowej i średniego czasu, z uwzględnieniem reguł nowo odkrytych w ostatnim przebiegu. Jeżeli zaktualizowane wartości poparcia i pewności danej reguły nie spełniają aktualnych, globalnych wymagań minimalnego poparcia i pewności, reguła ta jest trwale usuwana z bazy reguł.

(R14) Dla dowolnych dwóch reguł p i r o jednakowych liczbach bazowych, takich, że bezpośrednio po ich odkryciu $sup(p) = sup(r)$ oraz $con(p) = con(r)$, jeżeli czas p jest mniejszy lub równy czasowi r (reguła p nie jest późniejsza, niż reguła r), to po aktualizacji poparcia i pewności reguł zgodnie z (R12) i (R13), $sup'(p) \leq sup'(r)$ oraz $con'(p) \leq con'(r)$.

(R15) Wszystkie fakty, składające się na porcję przetwarzaną w danym przebiegu, po przeanalizowaniu i odkryciu nowych reguł, są trwale usuwane z historii.

(R16) Agent posiada pamięć długoterminową, w której są trwale przechowywane (także pomiędzy przebiegami) informacje potrzebne w cyklu pozyskiwania. Informacje te są aktualizowane w każdym przebiegu.

(R17) Agent posiada pamięć krótkoterminową, która zawiera informacje potrzebne do przeprowadzenia pojedynczego przebiegu pozyskiwania. Informacje te są aktualizowane i przechowywane tylko przez czas trwania danego przebiegu.

3.3. Reprezentacja wiedzy agenta

Poniżej podane są definicje formalne struktur bazy wiedzy agenta. Częściowo wykorzystana jest w nich notacja relacyjnego modelu danych, podana przez Pankowskiego w pracy [Pan1992].

Definicja 1 *Fakt*

Dane są: symbol K, nazywany *kluczem* i symbol T, nazywany *czasem* oraz skończony zbiór $U = \{A^S_1, ..., A^S_n, A^M_1, ..., A^M_k\}$, którego elementy A^S_i dla $i = 1, ..., n$ są nazywane *atrybutami jednowartościowymi*, natomiast elementy A^M_j dla $j = 1, ..., k$ są nazywane *atrybutami wielowartościowymi*. Niech symbolom K oraz T będą przyporządkowane odpowiednio: zbiór $D_K \subseteq \mathbb{N}$ nazywany *dziedziną klucza* oraz przeliczalny zbiór punktów czasowych D_T, który jest uporządkowany przez relację silnego porządku liniowego $<$. Niech każdemu atrybutowi jednowartościowemu A^S_i, dla $i = 1, ..., n$, będzie przyporządkowany skończony zbiór wartości D^S_i, nazywany *dziedziną atrybutu* A^S_i. Niech każdemu atrybutowi wielowartościowemu A^M_j, dla $j = 1, ..., k$, będzie przyporządkowany zbiór $D^M_j = 2^{V_j}$ wszystkich podzbiorów skończonego zbioru wartości V_j, nazywany *dziedziną atrybutu* A^M_j.

Faktem o schemacie $S_H = \{K, T, U\}$, nazywamy dowolną funkcję s, taką, że:

$$s\colon \{K, T, U\} \rightarrow D_K \cup D_T \cup \bigcup\{D^S_i\colon A^S_i \in U, \text{dla } i = 1,..., n\} \cup \bigcup\{D^M_i\colon A^M_j \in U, \text{dla } j = 1,...,k\}$$

Komentarz

Każdy fakt odzwierciedla obserwację stanu świata dokonaną przez agenta. Fakt jest funkcją, która symbolom atrybutów ze schematu S_H jednoznacznie przyporządkowuje wartości tych atrybutów, zgodnie z ich dziedzinami. Symbole K i T są traktowane jako atrybuty specjalne, celowo oddzielone od właściwych atrybutów opisujących stan, należących do zbioru U. Atrybut specjalny K służy do jednoznacznego identyfikowania faktów, natomiast atrybut T pozwala na ich liniowe uszeregowanie względem czasu zarejestrowania. Zbiór D_T wraz z relacją $<$ można traktować jako prostą strukturę czasu punktowego. Przykładowym zbiorem, który może być zastosowany jako D_T, jest zbiór liczb naturalnych $\mathbb{N}$ wraz z relacją mniejszości $<$. (Dogłębny przegląd różnych struktur czasu został przedstawiony w pracy Hajnicz [Haj1996]).

Zwróćmy uwagę, że przyjęta, punktowa reprezentacja czasu jest zgodna z praktyką rzeczywistych systemów informatycznych, gdzie czas jest często mierzony punktami liczbowymi (ang. *time ticks*), oddalonymi od siebie o bardzo krótkie interwały czasowe, a naliczanymi począwszy o pewnego, ustalonego momentu w przeszłości. Na przykład typ daty i czasu *datetime* w systemie bazodanowym *MS SQL Server 2000* reprezentowany jest liczbą punktów odległych od siebie o 0,03 sekundy, naliczanych od 1 stycznia 1753 roku, maksymalnie do 31 grudnia 9999 roku [Ran2003].

Oznaczenia

Fakt o schemacie $S_H = \{K, T, U\}$ zapisujemy jako zbiór par atrybutów i ich wartości, to znaczy: $s = \{(K, k), (T, t), (A^S_1, a^S_1), ..., (A^S_n, a^S_n), (A^M_1, a^M_1), ..., (A^M_k, a^M_k)\}$, gdzie $A^S_i, A^M_j \in U$, dla $i = 1, ..., n$, $j = 1, ..., k$. Wówczas wartości: klucza K, czasu T, atrybutu jednowartościowego A^S_i oraz atrybutu wielowartościowego A^M_j w fakcie s o schemacie $S_H = \{K, T, U\}$ oznaczamy jako odpowiednio:

$K(s) = k$,

$T(s) = t$,

$A^S_i(s) = a^S_i$,

$A^M_j(s) = a^M_j$.

Wartość nieznaną atrybutu jednowartościowego lub wielowartościowego, należącego do zbioru U, oznaczamy symbolem N.

Zbiór wszystkich atrybutów jednowartościowych A^S_i dla $i = 1, ..., n$, należących do zbioru U w schemacie $S_H = \{K, T, U\}$ oznaczamy symbolem S^S_H.

Zbiór wszystkich atrybutów wielowartościowych A^M_j dla $j = 1, ..., k$, należących do zbioru U w schemacie $S_H = \{K, T, U\}$ oznaczamy symbolem S^M_H.

Zachodzą przy tym zależności:

$S^S_H \subseteq S_H \setminus \{K, T\}$,

$S^M_H \subseteq S_H \setminus \{K, T\}$,

$S^S_H \cap S^M_H \equiv \varnothing$.

Zbiór wszystkich faktów o schemacie S_H oznaczamy symbolem $FAKT(S_H)$.

Definicja 2 *Historia*

Dany jest schemat $S_H = \{K, T, U\}$. *Historią* KB_H *o schemacie* S_H nazywamy dowolny podzbiór zbioru wszystkich faktów o schemacie S_H, to znaczy: $KB_H \subseteq FAKT(S_H)$.

Komentarz

Historia może być traktowana jak zdenormalizowana tabela relacyjnej bazy danych. Dopuszczalność atrybutów wielowartościowych sprawia, że w ogólnym przypadku tabela ta nie jest w pierwszej postaci normalnej (ang. *first normal form*, 1NF) [Pan1992]. Jest to jednak zabieg celowy, który zwiększa ekspresję proponowanego modelu, zbliżając go nieco do modeli obiektowych (kolekcje wartości danego typu) i potencjalnie ułatwiając projektowanie konkretnego systemu, jak to zostało omówione wcześniej. Jednocześnie z punktu widzenia cyklu metody APS, atrybuty wielowartościowe nie obniżają znacząco wydajności przetwarzania danych.

Oznaczenia

K-ty punkt czasowy, należący do zbioru D_T, oznaczamy symbolem t_k.

Punkt czasowy pierwszego faktu w historii oznaczamy jako t_0.

Punkt czasowy odpowiadający chwili obecnej (teraźniejszości) oznaczamy symbolem t_{now}.

Przedział czasowy (interwał) od chwili t_i do chwili t_j, gdzie $t_i \le t_j$ oznaczamy jako $[t_i; t_j]$.

Własność 1

Dana jest historia KB_H o schemacie $S_H = \{K, T, U\}$. Dla każdego faktu $s \in KB_H$ zachodzi zależność: $t_0 \le T(s) \le t_{now}$.

Definicja 3 *Porcja faktów*

Dana jest historia KB_H o schemacie $S_H = \{K, T, U\}$. *Porcją faktów* z historii KB_H dla interwału czasowego $[t_1; t_2]$, gdzie $t_1, t_2 \in D_T \wedge t_1 \le t_2$, jest zbiór:

$$KB_H(t_1, t_2) = \{s \in KB_H: t_1 \le T(s) \le t_2\}.$$

Definicja 4 *Dziedzina właściwa atrybutu jednowartościowego*

Dana jest historia KB_H o schemacie $S_H = \{K, T, U\}$. *Dziedziną właściwą atrybutu jednowartościowego* $A^S_i \in S^S_H$ w historii KB_H, dla interwału czasowego $[t_1; t_2]$, gdzie $t_1, t_2 \in D_T \wedge t_1 \le t_2$, jest zbiór:

$$D^W_i (KB_H(t_1, t_2)) = \{A^S_i(s) \ne N: s \in KB_H \wedge (t_1 \le T(s) \le t_2)\}.$$

Definicja 5 *Dziedzina właściwa atrybutu wielowartościowego*

Dana jest historia KB_H o schemacie $S_H = \{K, T, U\}$. *Dziedziną właściwą atrybutu wielowartościowego* $A^M_j \in S^M_H$ w historii KB_H, dla interwału czasowego $[t_1; t_2]$, gdzie $t_1, t_2 \in D_T \wedge t_1 \le t_2$, jest zbiór:

$$D^W_j (KB_H(t_1, t_2)) = \bigcup\{A^M_j(s) \ne \emptyset. s \in KB_H \wedge (t_1 \le T(s) \le t_2)\} \setminus \{N\}.$$

Komentarz

Dziedzina właściwa atrybutu jest zbiorem wszystkich jego wartości (z wyjątkiem wartości nieznanej N), które rzeczywiście wystąpiły w określonym podzbiorze historii. W metodzie APS zawężenie zbiorów wartości atrybutów do ich dziedzin właściwych w badanej porcji danych pozwala na przyśpieszenie przekształcania historii do postaci z atrybutami binarnymi $\{0, 1, N\}$.

Definicja 6 *Reguła*

Dany jest zbiór $U' = \{A_1, A_2, ..., A_m\}$, którego elementy są symbolami, nazywanymi *atrybutami binarnymi*. Każdemu atrybutowi binarnemu A_i dla $i = 1,2,...,m$ przypisany jest zbiór wartości $D_b = \{0, 1, N\}$. Dany jest przeliczalny zbiór punktów czasowych D_T, który jest uporządkowany przez relację silnego porządku liniowego $<$.

Regułą r nazywamy siódemkę uporządkowaną:

$$r = (X, Y, sup, con, b, t_m) \in 2^{U'} \times 2^{U'} \times [0; 1] \times [0; 1] \times \mathbb{N} \times D_T.$$

Poszczególne symbole mają następujące znaczenie:

X – poprzednik,

Y – następnik,

sup – poparcie,

con – pewność,

b – liczba bazowa,

t_m – średni czas.

Komentarz

Zwróćmy uwagę, że powyższa definicja reguły stanowi rozszerzenie definicji reguły związku, podanej wcześniej, w rozdziale *Eksploracja danych jako metoda maszynowego uczenia się*. Rozszerzenie to obejmuje elementy wymagane w metodzie APS, a więc: liczbę bazową b oraz średni czas reguły t_m. Wielkości te oznaczają odpowiednio: liczbę i średni czas faktów z historii, na podstawie których została odkryta reguła r.

Definicja 7 *Semantyczna równość reguł*

Mówimy, że dwie reguły $r_1 = (X_1, Y_1, sup_1, con_1, b_1, t_{m1}) \in 2^{U'} \times 2^{U'} \times [0; 1] \times [0; 1] \times \mathbb{N} \times D_T$ oraz $r_2 = (X_2, Y_2, sup_2, con_2, b_2, t_{m2}) \in 2^{U'} \times 2^{U'} \times [0; 1] \times [0; 1] \times \mathbb{N} \times D_T$ są *semantycznie równe*, co oznaczamy przez $r_1 \equiv r_2$, wtedy i tylko wtedy, gdy: $X_1 \equiv X_2 \wedge Y_1 \equiv Y_2$.

Definicja 8 *Baza reguł*

Dany jest zbiór $U' = \{A_1, A_2, ..., A_m\}$, którego elementy są symbolami, nazywanymi *atrybutami binarnymi*. Każdemu atrybutowi binarnemu A_i dla $i = 1,2,...,m$ przypisany jest zbiór wartości $D_b = \{0, 1, N\}$. Dany jest przeliczalny zbiór punktów czasowych D_T.

Baza reguł KB_R jest zbiorem określonym następująco:

$$KB_R \subseteq \{r = (X, Y, sup, con, b, t_m) \in 2^{U'} \times 2^{U'} \times [0; 1] \times [0; 1] \times \mathbb{N} \times D_T\},$$

przy czym $\neg\exists\ r_1, r_2 \in KB_R.\ r_1 \equiv r_2$.

Komentarz

Zgodnie z Definicją 8 do bazy KB_R nie mogą należeć dwie reguły semantycznie równe, czyli o takich samych poprzednikach X i następnikach Y. Jest to bardzo istotne ograniczenie, zabezpieczające przez wystąpieniem sprzeczności w bazie reguł.

Definicja 9 *Wektor danych przebiegu*

Dany jest zbiór $U = \{A^S_1, ..., A^S_n, A^M_1, ..., A^M_k\}$, którego elementy A^S_i dla $i = 1, ..., n$ są nazywane *atrybutami jednowartościowymi*, natomiast elementy A^M_j dla $j = 1, ..., k$ są nazywane *atrybutami wielowartościowymi*. Dany jest przeliczalny zbiór punktów czasowych D_T, który jest uporządkowany przez relację silnego porządku liniowego $<$.

Wektorem danych przebiegu nazywamy krotkę:

$$\vec{v}_c = \left(id_c, X_c, Y_c, b_{max}, b_c, \eta_c, \sigma_c, \gamma_c, \hat{\sigma}_c, \hat{\gamma}_c, m_x, m_y, t_{rc}, t_{mc}, t_{sc}, t_{ec}, k_{ec}\right) \in$$
$$\mathbb{N} \times 2^U \times 2^U \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times [0;1] \times [0;1] \times [0;1] \times [0;1] \times \mathbb{N} \times \mathbb{N} \times D_T \times D_T \times D_T \times D_T \times \mathbb{N}$$

przy czym: $\hat{\sigma}_c \in [0;\sigma_c] \wedge \hat{\gamma}_c \in [0;\gamma_c] \wedge t_{sc} \leqslant t_{mc} \leqslant t_{ec}$.

Poszczególne symbole mają następujące znaczenie:

id_c – unikalny identyfikator przebiegu;

X_c – zbiór atrybutów, które mogą wystąpić w poprzedniku reguły;

Y_c – zbiór atrybutów, które mogą wystąpić w następniku reguły;

b_{max} – maksymalna liczba faktów, które mogą być przeanalizowane w bieżącym przebiegu;

b_c – liczba faktów przeanalizowanych w bieżącym przebiegu;

η_c – maksymalna, dopuszczalna liczba wartości N w pojedynczym fakcie;

σ_c – próg minimalnego poparcia reguł;

γ_c – próg minimalnej pewności reguł;

$\hat{\sigma}_c$ – oczekiwane poparcie reguł odrzucanych;

$\hat{\gamma}_c$ – oczekiwana pewność reguł odrzucanych;

m_x – maksymalna liczba atrybutów, które mogą wystąpić w poprzedniku reguły;

m_y – maksymalna liczba atrybutów, które mogą wystąpić w następniku reguły;

t_{rc} – czas rozpoczęcia bieżącego przebiegu;

t_{mc} – średni czas faktów przetwarzanych w bieżącym przebiegu;

t_{sc} – czas najwcześniejszego faktu, analizowanego w bieżącym przebiegu;

t_{ec} – czas najpóźniejszego faktu, analizowanego w bieżącym przebiegu;

k_{ec} – wartość klucza najpóźniejszego faktu, analizowanego w bieżącym przebiegu.

Oznaczenia

Przez KB_T oznaczamy *moduł wiedzy chwilowej* (pamięci krótkoterminowej), do którego należy wektor danych przebiegu, to znaczy $\vec{v}_c \in KB_T$. Nie podajemy formalnej definicji modułu KB_T, ponieważ zakładamy, że oprócz powyższego wektora może on zawierać także inne rodzaje wiedzy chwilowej agenta, zależne od dziedziny zastosowania.

Definicja 10 *Funkcja wpływu czasowego*

Funkcją wpływu czasowego nazywamy funkcję f_T: [0; +∞) → [0; 1], taką, że:

(W1) $f_T(0) = 1$ (wartość 1 dla $x = 0$);

(W2) $\forall x_1,x_2 \in [0; +\infty).\ x_1 < x_2 \Rightarrow f_T(x_1) \geq f_T(x_2)$ (funkcja nierosnąca).

Oznaczenia

Przez $\mathscr{F}$ oznaczamy rodzinę wszystkich funkcji wpływu czasowego f_T.

Komentarz

W metodzie APS argumentem funkcji wpływu czasowego f_T jest czas, który upłynął od określonego momentu t do chwili obecnej t_{now}, to znaczy: $(t_{now} - t)$, dla punktów t, $t_{now} \in D_T$, zgodnie z oznaczeniami wprowadzonymi wcześniej. Jeśli $t = t_{now}$, to wartość funkcji f_T jest maksymalna i wynosi 1. W zastosowaniach praktycznych wzór i współczynniki funkcji $f_T \in \mathscr{F}$ są zależne od konkretnej dziedziny.

Definicja 11 *Wektor parametrów globalnych*

Dana jest rodzina funkcji wpływu czasowego $\mathscr{F}$ oraz przeliczalny zbiór punktów czasowych D_T, który jest uporządkowany przez relację silnego porządku liniowego $<$.

Wektorem parametrów globalnych nazywamy krotkę:

$$\vec{v}_g = \left(b_g, \sigma_g, \gamma_g, \hat{\sigma}_g, \hat{\gamma}_g, t_{mg}, t_{sg}, t_{eg}, k_{eg} f_T\right) \in$$
$$\mathbb{N} \times [0;1] \times [0;1] \times [0;1] \times [0;1] \times D_T \times D_T \times D_T \times D_T \times \mathbb{N} \times \mathscr{F}$$

przy czym: $\hat{\sigma}_g \in [0; \sigma_g] \wedge \hat{\gamma}_g \in [0; \gamma_g] \wedge t_{sg} \leqslant t_{mg} \leqslant t_{eg}$.

Poszczególne symbole mają następujące znaczenie:

b_g – liczba faktów przeanalizowanych we wszystkich dotychczasowych przebiegach;

σ_g – próg minimalnego poparcia reguł dla wszystkich przebiegów;

γ_g – próg minimalnej pewności reguł dla wszystkich przebiegów;

$\hat{\sigma}_g$ – oczekiwane poparcie reguł odrzucanych;

$\hat{\gamma}_g$ – oczekiwana pewność reguł odrzucanych;

t_{mg} – średni czas faktów przetwarzanych we wszystkich przebiegach;

t_{sg} – czas najwcześniejszego faktu, analizowanego we wszystkich przebiegach;

t_{eg} – czas najpóźniejszego faktu, analizowanego we wszystkich przebiegach;

k_{eg} – klucz najpóźniejszego faktu, analizowanego we wszystkich przebiegach;

f_T – funkcja wpływu czasowego.

Oznaczenia

Przez KB_G oznaczamy *moduł wiedzy ogólnej* (pamięci długoterminowej), do którego należy wektor parametrów globalnych i schemat historii, to znaczy $\vec{v}_g \in KB_G \wedge S_H \in KB_G$.

Komentarz

Sposób opisu funkcji f_T w wektorze parametrów globalnych nie jest ściśle określony. Zakładamy, że funkcja ta jest charakteryzowana za pomocą wyrażenia regularnego, które zawiera symbole i współczynniki, pozwalające na obliczenie wartości funkcji przy podaniu wartości argumentu.

Podobnie, jak w przypadku KB_T nie podajemy formalnej definicji modułu KB_G, ponieważ zakładamy, że oprócz wektora parametrów globalnych i informacji o schemacie historii, może on zawierać także inne rodzaje wiedzy ogólnej agenta, zależne od dziedziny zastosowania.

3.4. Cykl pozyskiwania reguł w metodzie APS

Poniżej zdefiniowany jest cykl metody APS w formie procedury opartej na pseudokodzie. W procedurze są wykorzystane oznaczenia, które zostały wprowadzone w poprzednim podrozdziale.

```
Procedura: Pozyskuj_Reguły_APS.
1. BEGIN
2. WHILE (NOT Event_Zakończ_Proces_Agenta)
3. BEGIN
4.    IF (Event_Nowe_Fakty)
5.         Zapisz_Fakty(S, KB_H, OUT KB_H);
6.    IF (Event_Analizuj_Fakty)
7.    BEGIN
8.         Pobierz_Parametry(OUT v_c);
9.         Wybierz_Fakty(KB_H, v_c, OUT KB_H(t_sc; t_ec), OUT v_c);
10.        Przekształć_Schemat(KB_H(t_sc; t_ec), S_H, OUT S#);
11.        Wypełnij_Nowy_Schemat(KB_H(t_sc; t_ec), S^S_H, S^M_H, S#,
                                 OUT KB#_H(t_sc; t_ec));
12.        Usuń_Wartości_N(KB#_H(t_sc; t_ec), v_c, S^S_H, S^M_H,
                           OUT KB#_H(t_sc; t_ec), OUT v_c);
13.        Znajdź_Reguły(KB#_H(t_sc; t_ec), v_c, OUT R);
14.        Aktualizuj_Bazę_Reguł(KB_R, R, v_c, v_g, t_now, OUT KB_R, OUT v_g);
15.        Usuń_Fakty(KB_H, v_c, OUT KB_H);
16.   END // IF (Event_Analizuj_Fakty)
17. END //WHILE
18. END
```

Omówienie procedury cyklu metody APS

Zakładamy, że procedura `Pozyskuj_Reguły_APS` jest częścią ogólnej pętli działania agenta: rozpoczyna się ona wraz z wywołaniem procesu agenta i kończy się wraz z jego zamknięciem (`Event_Zakończ_Proces_Agenta`). Procedury, które realizują poszczególne etapy metody APS, są wywoływane w odpowiedzi na zdarzenia (ang. *events*), przekazywane przez proces agenta. Samo generowanie zdarzeń nie należy zatem do metody APS, lecz jest dokonywane zewnętrznie – w innych modułach architektury agenta. I tak, podczas normalnego działania agenta, co pewien czas może być zgłaszane zdarzenie `Event_Nowe_Fakty` (wiersz 4.), które oznacza, że pojawiły się nowe fakty, które powinny zostać zapisane w historii, zgodnie z założeniami projektowymi (np. użytkownik przejrzał nowe strony WWW). Wywoływana jest wówczas procedura `Zapisz_Fakty` (wiersz 5.), której danymi (parametrami) wejściowymi jest zbiór nowych faktów S oraz historia KB_H. Procedura ta dopisuje nowe fakty, nadając każdemu z nich wartość klucza K oraz czasu zarejestrowania T, i zwraca uzupełnioną historię (parametry wyjściowe są poprzedzone słowem `OUT`).

Właściwe pozyskiwanie reguł rozpoczyna się wtedy, gdy architektura agencka przekazuje zdarzenie `Event_Analizuj_Fakty` (wiersz 6.). Wówczas kolejno uruchamianych jest 8 procedur, realizujących główne etapy przebiegu pozyskiwania reguł metody APS (wiersze 8–15.). Najpierw wywoływana jest procedura `Pobierz_Parametry` (wiersz 8.), której zadaniem jest pobranie odpowiednich parametrów z architektury agenckiej do wektora danych przebiegu v_c. Sposób określenia wartości tych parametrów jest zależny od konkretnej aplikacji i nie należy do metody APS. Przykładowo, mogą to być dane pochodzące z innego modułu zarządzającego agenta, wartości domyślne, zdefiniowane na etapie projektowania, albo ustawienia wykorzystane podczas poprzedniego przebiegu APS. Wektor v_c z odpowiednimi wartościami jest zapisywany w module KB_T.

W kolejnym kroku procedura `Wybierz_Fakty` (wiersz 9.) pobiera odpowiednią porcję faktów z historii KB_H (t_{sc}; t_{ec}), na podstawie bieżącej wartości parametru b_{max} w wektorze v_c. Oprócz samej porcji, procedura ta zwraca wektor v_c ze zaktualizowanymi wartościami parametrów: b_c, t_{sc}, t_{mc}, t_{ec}, k_e.

Pobrana porcja faktów oraz schemat historii (informacja z KB_G) są przekazywane jako dane wejściowe do procedury `Przekształć_Schemat` (wiersz 10.), transformującej schemat historii do nowej postaci $S^{\#}$, w której wszystkie atrybuty, z wyjątkiem atrybutów specjalnych K i T, mają dziedzinę $\{0, 1, N\}$. Liczba atrybutów schematu $S^{\#}$ zależy od mocy dziedzin właściwych (czyli zbiorów różnych wartości) atrybutów w porcji KB_H (t_{sc}; t_{ec}). Informacja o nowym schemacie $S^{\#}$ jest zapisywana w module KB_T.

Procedura `Wypełnij_Nowy_Schemat` (wiersz 11.), na podstawie porcji KB_H (t_{sc}; t_{ec}), tworzy tymczasową strukturę danych – porcję faktów $KB^{\#}{}_H$(t_{sc}; t_{ec}) o nowym schemacie $S^{\#}$.

Porcja ta jest następnie przetwarzana przez procedurę `Usuń_Wartości_N` (wiersz 12.), która zwraca porcję $KB^{\#}{}_H$(t_{sc}; t_{ec}) bez wartości nieznanych N. W trakcie tego procesu brany jest pod uwagę parametr η_c (maksymalna, dopuszczalna liczba wartości N w jednym fakcie) z wektora v_c. Ponieważ w związku z eliminacją wartości N liczba faktów może ulec zmianie, procedura `Usuń_Wartości_N` zwraca wektor v_c ze zaktualizowanymi wartościami parametrów b_c, t_{sc}, t_{mc}, t_{ec}.

W kolejnym kroku uruchamiana jest procedura `Znajdź_Reguły` (wiersz 13.), która stanowi implementację wybranego algorytmu odkrywania reguł związku (np. Apriori). Danymi wejściowymi procedury są przekształcona i pozbawiona wartości N porcja $KB^{\#}_H(t_{sc}; t_{ec})$ oraz parametry wektora v_c, definiujące odpowiednie ograniczenia dla algorytmu eksploracji danych: X_c, Y_c, σ_c, γ_c, m_x, m_y. W wyniku procedura `Znajdź_Reguły` zwraca zbiór R, który zawiera reguły postaci $r = (X, Y, sup, con, b, t_m)$, zgodnie z omówioną wcześniej notacją, przy czym sam algorytm eksploracji danych nie nadaje wartości liczby bazowej b i średniego czasu reguły t_m. Wszystkim odkrytym regułom $r \in R$ nadawane są takie same wartości b oraz t_m, równe parametrom wektora v_c, odpowiednio: b_c oraz t_{mc}. Następnie przetworzona porcja faktów $KB^{\#}_H(t_{sc}; t_{ec})$ jest usuwana.

Po odkryciu nowych reguł R uruchamiana jest procedura `Aktualizuj_Bazę_Reguł` (wiersz 14.), która odpowiednio modyfikuje reguły w bazie KB_R na podstawie nowego zbioru reguł R, bieżącego czasu t_{now}, pobranego z systemu oraz parametrów wektorów v_c: $b_c, \sigma_c, \gamma_c, \hat{\sigma}_c, \hat{\gamma}_c, t_{mc}$ i v_g: $b_g, \sigma_g, \gamma_g, \hat{\sigma}_g, \hat{\gamma}_g, t_{mg}, f_T$. W wyniku zwracana jest zaktualizowana baza KB_R oraz wektor v_g. Zbiór reguł R jest usuwany.

Cykl kończy procedura `Usuń_Fakty` (wiersz 15.), która w historii KB_H kasuje wszystkie fakty od początku do faktu o wartości klucza k_e, pobranej z wektora v_c.

Następnie proces `Pozyskuj_Reguły_APS` kontynuuje nasłuchiwanie zdarzeń w głównej pętli `WHILE`, co jest jednoznaczne z powrotem procesu agenta do normalnej aktywności. Omówione wyżej procedury są sprecyzowane w następnym podrozdziale, w formie odpowiednich algorytmów przetwarzania danych.

3.5. Algorytmy przetwarzania danych

W niniejszej części wprowadzane są szczegółowo oryginalne algorytmy przetwarzania danych w cyklu pozyskiwania reguł APS. Pominięte są tutaj, omówione w poprzednim podrozdziale, procedury: `Zapisz_fakty` i `Pobierz_Parametry`. Pierwsza z nich jest prostą operacją dodania faktów do historii, która w języku SQL może być zrealizowana na przykład zwykłym poleceniem `INSERT` (oczywiście przy założeniu, że historia jest przechowywana w strukturze relacyjnej bazy danych). Z kolei implementacja procedury `Pobierz_Parametry` jest ściśle zależna od dziedziny zastosowania oraz wybranej architektury agenckiej, w której jest osadzony proces APS. Zakładamy zatem, że przed uruchamianiem poniższych algorytmów odpowiednie parametry są już pobrane z innych części systemu.

Dla każdego, opisywanego poniżej etapu zdefiniowany jest formalnie odpowiedni problem przetwarzania danych. Następnie podawany jest proponowany algorytm rozwiązania tego problemu i komentarz, w którym zawarta jest jego dodatkowa charakterystyka. Dla każdego algorytmu przeprowadzona jest także analiza jego złożoności obliczeniowej z wykorzystaniem notacji i metod zaczerpniętych z pracy [Mos1986].

3.5.1 Wybór faktów do analizy

Realizowana przez procedurę `Wybierz_Fakty` (wiersz 9.), selekcja faktów, które mają być przetworzone w bieżącym przebiegu cyklu, może być scharakteryzowana, jak poniżej.

Problem: *Wybór faktów do analizy*

Dane: historia KB_H o schemacie $S_H = \{K, T, U\}$; maksymalna liczba faktów do pobrania b_{max}.

Wyznaczyć: porcję faktów $KB_H(t_{sc}; t_{ec})$, taką, że: $card\ (KB_H(t_{sc}; t_{ec})) \leq b_{max}$
oraz $\forall s \in KB_H(t_{sc}; t_{ec})\ \neg\exists s' \in KB_H \setminus KB_H(t_{sc}; t_{ec}).\ T(s') < T(s)$.

Algorytm rozwiązania problemu

Algorytm FSEL *Wybór faktów do analizy*

Wejście: historia KB_H; maksymalna liczba faktów do pobrania $b_{max}(v_c)$, gdzie $v_c \in KB_T$.

Wyjście: porcja faktów do analizy $KB_H(t_{sc}, t_{ec})$; zaktualizowane wartości parametrów $b_c(v_c)$, $t_{sc}(v_c)$, $t_{mc}(v_c)$, $t_{ec}(v_c)$, $k_e(v_c)$, gdzie $v_c \in KB_T$.

1. $KB_H(t_{sc}, t_{ec}) := \emptyset$.
2. Jeżeli $card\ KB_H \leq b_{max}$, to $KB_H(t_{sc}, t_{ec}) := KB_H$.
3. Jeżeli $card\ KB_H > b_{max}$, to $KB_H(t_{sc}, t_{ec}) :=$
   $\{s_1, s_2, ..., s_n \in KB_H: n = b_{max} \wedge \forall i,j \in \{1,2,...,n\}.(i < j \Leftrightarrow T(s_i) < T(s_j))$
   $\wedge\ \forall s \in KB_H.T(s_1) < T(s)\}$.
4. $b_c(v_c) := card\ KB_H(t_{sc}, t_{ec})$.
5. $t_{sc}(v_c) := \text{MIN}\{T(s): s \in KB_H(t_{sc}, t_{ec})\}$.
6. $t_{ec}(v_c) := \text{MAX}\{T(s): s \in KB_H(t_{sc}, t_{ec})\}$.
7. $t_{mc}(v_c) := \frac{1}{n} \sum_{s \in KB_H(t_{sc}, t_{ec})} T(s)$.
8. $k_e(v_c) := K(s)$, gdzie $T(s) = t_{ec}(v_c)$.
9. Zwróć: $KB_H(t_{sc}, t_{ec})$, $b_c(v_c)$, $t_{sc}(v_c)$, $t_{mc}(v_c)$, $t_{ec}(v_c)$, $k_e(v_c)$, gdzie $v_c \in KB_T$.

Komentarz

Wybieranie faktów do analizy jest operacją stosunkowo prostą i mało złożoną obliczeniowo. Jeśli, na przykład, historia jest zapisana w strukturze tabeli relacyjnej bazy danych, operację tę można efektywnie zaimplementować w pojedynczym ciągu poleceń SQL, takich jak: `SELECT TOP N...FROM...ORDER BY`, wraz z funkcjami agregującymi: `MIN`, `MAX`,

AVERAGE. W wydajnych systemach zarządzania bazą danych (ang. *database management system*, DBMS), takich jak *MS SQL Server*, tego typu zapytania mogą być skutecznie optymalizowane (przede wszystkim przez sam wbudowany optymalizator zapytań, ale też na przykład przez nałożenie odpowiednich indeksów na pola tabel), dzięki czemu uzyskuje się krótki czas ich wykonywania [Ran2003].

Nazwa algorytmu FSEL jest utworzona na podstawie pierwszych liter angielskiego opisu realizowanej operacji *wybieranie faktów* (ang. *Fact SELection*, FSEL).

Analiza złożoności obliczeniowej algorytmu FSEL

Rozpatrujemy liczbę operacji koniecznych do wykonania przez algorytm, takich, jak: odczyt, porównanie i zapis wartości w strukturach danych. Przypuśćmy, że historia KB_H zawiera n faktów. W pesymistycznym wariancie (gdy $card\ KB_H \leq b_{max}$) wymagane są następujące operacje: zliczenie n faktów KB_H (w celu obliczenia mocy zbioru), odczyt n faktów z KB_H, zapis n faktów do $KB_H\,(t_{sc},\ t_{ec})$, trzykrotny odczyt n faktów z $KB_H\,(t_{sc},\ t_{ec})$ w celu obliczenia wartości minimalnej, maksymalnej i średniej czasu. Stąd też sumaryczną liczbę rozważanych operacji możemy oszacować za pomocą poniższej funkcji $T_z\,(n)$.

$|T_z\,(n)| = |n + n + n + 3n| = |6n| \leq 6|n| = c|F(n)|,\ c = 6,\ n \geq 0.$

A zatem algorytm FSEL ma wielomianową złożoność obliczeniową rzędu $O\,(n)$.

3.5.2 Przekształcanie schematu historii

Problem transformacji schematu historii, realizowanej w cyklu APS przez procedurę Przekształć_Schemat (wiersz 10.), można opisać następująco.

Problem: *Przekształcanie schematu historii*

Dane: schemat historii $S_H = \{K, T, U\}$, porcja faktów $KB_H(t_1, t_2)$.

Wyznaczyć: schemat histori $S^{\#} = \{K, T, U'\}$, taki, że $\forall A_i \in U'.\ D_i = \{0, 1, N\}$.

Algorytm rozwiązania problemu

Algorytm HTRANS *Przekształcanie schematu historii*

Wejście: $S_H = \{K, T, U\}$ – schemat historii, $KB_H(t_{sc}, t_{ec})$ – porcja faktów do analizy.

Wyjście: $S^{\#}$ – nowy schemat historii, w którym dziedziną wszystkich atrybutów oprócz K i T jest zbiór $\{1, 0, N\}$.

1. $S^{\#} := \varnothing$

2. $S^{\#} := S^{\#} \cup \{K, T\}$

3. Dla każdego $A_i \in U$, gdzie $i \in [1; \text{card } U]$, powtarzaj kroki 4–5.

4. Wyznacz $D^{W_i}(KB_H(t_{sc}, t_{ec}))$.

5. $S^{\#} := S^{\#} \cup \{A_i^{(v)}: A_i \in S_H \wedge v \in D^{W_i}(KB_H(t_{sc}, t_{ec}))\}$.

6. Koniec dla.

7. Zwróć $S^{\#}$.

Komentarz

Atrybuty nowego schematu $S^{\#}$ są znajdowane na podstawie dziedzin właściwych atrybutów pierwotnego schematu S_H, to znaczy zbiorów wszystkich wartości (z wyjątkiem wartości nieznanej *N*), które poszczególne atrybuty przyjmowały w analizowanej porcji faktów. Tak więc w nowym schemacie znajdzie się tyle nowych atrybutów $A_i^{(v)}$, ile różnych wartości v przyjął atrybut A_i w danej porcji. Zwróćmy uwagę, że w algorytmie nie ma rozróżnienia na atrybuty jedno- i wielowartościowe, ponieważ wyznaczenie dziedziny właściwej dla atrybutu tak jednej, jak i drugiej grupy daje identyczny wynik – skończony zbiór wartości. Odpowiednie rozróżnienie powinno być dokonane dopiero na poziomie implementacyjnym, zgodnie z podanymi wcześniej Definicjami 4 i 5.

Nazwa algorytmu jest utworzona na podstawie pierwszych liter opisu działania w języku angielskim: *przekształcanie historii* (ang. *History TRANSformation*, HTRANS).

Przykład obliczeniowy, ilustrujący działanie algorytmu HTRANS, jest przedstawiony w Dodatku A.

Analiza złożoności obliczeniowej algorytmu HTRANS

Rozważamy liczbę operacji koniecznych do wykonania przez algorytm, takich, jak: odczyt, porównanie i zapis wartości w strukturach danych. Przypuśćmy, że porcja $KB_H(t_{sc}, t_{ec})$ zawiera n faktów, zbiór U zawiera m atrybutów (łącznie jedno- i wielowartościowych), z których każdy przyjmuje w porcji $KB_H(t_{sc}, t_{ec})$ średnio k różnych wartości oraz średnio l różnych wartości w pojedynczym fakcie, przy czym $n, m, k, l \in \mathbb{N}$. Wówczas liczbę odczytów w kroku 4. możemy oszacować jako iloczyn lmn, natomiast liczbę porównań wartości atrybutów (w celu ustalenia zbiorów wszystkich różnych wartości) – jako iloczyn $klmn$. Z kolei liczbę nowych atrybutów dodawanych do nowego schematu możemy ocenić na km. Zauważmy, że dla każdych l, k zachodzi $l \le k$. Stąd też sumaryczną liczbę rozważanych operacji możemy oszacować za pomocą poniższej funkcji $T_z(k, m, n)$.

$|T_z(k, m, n)| = |kmn + klmn + km| \le |kmn + k^2mn + km| \le 3|k^2mn| = c|F(k, m, n)|, c = 3,$
dla $k \ge 0, m \ge 0, n > 0$.

A zatem złożoność algorytmu HTRANS jest rzędu $O(k^2mn)$.

3.5.3 Wypełnianie danymi nowego schematu

Problem wstawiania porcji faktów do przekształconego schematu historii, realizowanego w cyklu APS przez procedurę `Wypełnij_Nowy_Schemat` (wiersz 11.), można formalnie opisać następująco.

Problem: *Wypełnianie danymi nowego schematu historii*

Dane: porcja faktów $KB_H(t_1, t_2)$, nowy schemat historii $S^{\#}$.

Wyznaczyć: zbiór faktów $KB^{\#}_H(t_1, t_2)$ o schemacie $S^{\#}$.

Algorytm rozwiązania problemu

Algorytm HFILL *Wypełnianie danymi nowego schematu historii*

Wejście: $KB_H(t_{sc}; t_{ec})$ – porcja faktów; zbiory $S^S{}_H \subseteq S_H \setminus \{K, T\}$, $S^M{}_H \subseteq S_H \setminus \{K, T\}$, gdzie $S^S{}_H \cap S^M{}_H \equiv \varnothing$; $S^{\#}$ – nowy schemat historii.

Wyjście: $KB^{\#}_H(t_{sc}; t_{ec})$ – zbiór faktów o schemacie $S^{\#}$.

1. $KB^{\#}_H(t_{sc}, t_{ec}) := \varnothing$.
2. Dla każdego faktu $s \in KB_H(t_{sc}, t_{ec})$ powtarzaj kroki 3–14.
3. Utwórz nowy fakt s^* o schemacie $S^{\#}$, którego wartość klucza K i czasu T są równe odpowiednim wartościom faktu s, pozostałe zaś atrybuty mają wartości N:
   $s^* = \{(K, K(s)), (T, T(s)), (A_1{}^{(v1)}, N), (A_1{}^{(v2)}, N), ..., (A_1{}^{(vk)}, N), ...,$
   $(A_n{}^{(vl)}, N), (A_n{}^{(v\_l+1)}, N), ..., (A_n{}^{(vm)}, N)\}$, gdzie $A_i{}^{(vj)} \in S^{\#}$, $i \in \{1,2,\ldots,n\}$, $j \in \{1,2,\ldots,m\}$.
4. Dla każdego $A_i{}^{(vj)} \in S^{\#}$, takiego, że $A_i \in S^S{}_H$, $i \in \{1,2,\ldots,n\}$, $j \in \{1,2,\ldots,m\}$ powtarzaj kroki 5–7.
5. Jeżeli $A_i(s) = v_j$, to $A_i{}^{(vj)}(s^*) := 1$.
6. Jeżeli $A_i(s) \neq v_j$, to $A_i{}^{(vj)}(s^*) := 0$.
7. Jeżeli $A_i(s) = N$, to $A_i{}^{(vj)}(s^*) := N$.
8. Koniec Dla.
9. Dla każdego $A_i{}^{(vj)} \in S^{\#}$, takiego, że $A_i \in S^M{}_H$, $i \in \{1,2,\ldots,n\}$ powtarzaj kroki 10–12.
10. Jeżeli $v_j \in A_i(s)$, to $A_i{}^{(vj)}(s^*) := 1$.
11. Jeżeli $v_j \notin A_i(s)$, to $A_i{}^{(vj)}(s^*) := 0$.
12. Jeżeli $A_i(s) = \{N\}$, to $A_i{}^{(vj)}(s^*) := N$.
13. Koniec Dla.
14. $KB^{\#}_H(t_{sc}, t_{ec}) := KB^{\#}_H(t_{sc}, t_{ec}) \cup \{s^*\}$.
15. Koniec Dla.
16. Zwróć $KB^{\#}_H(t_{sc}, t_{ec})$.

Komentarz

Algorytm HFILL daje w wyniku porcję $KB^{\#}_H$ (t_{sc}, t_{ec}) o takiej samej liczbie faktów, jak pierwotna porcja KB_H (t_{sc}, t_{ec}). Istotna jest różnica w przetwarzaniu atrybutów jedno- i wielowartościowych. W pierwszym przypadku może dokonywane jest bezpośrednie porównywanie wartości (kroki 5 –7), natomiast w drugim – badanie przynależności danej wartości do zbioru wartości, jakie dany atrybut wielowartościowy przyjął w danym fakcie.

Nazwa algorytmu jest utworzona na podstawie pierwszych liter opisu działania w języku angielskim: *wypełnianie historii* (ang. *History FILLing*, HFILL).

Przykład obliczeniowy, ilustrujący działanie algorytmu HFILL, jest przedstawiony w Dodatku A.

Analiza złożoności obliczeniowej algorytmu HFILL

Rozpatrujemy liczbę operacji koniecznych do wykonania przez algorytm, takich, jak: odczyt, porównanie i zapis wartości w strukturach danych. Przypuśćmy, że porcja KB_H (t_{sc}, t_{ec}) zawiera n faktów, natomiast w schemacie $S^{\#} = \{K, T, U'\}$ zbiór U' zawiera m atrybutów, przy czym $n, m \in \mathbb{N}$. (Zwróćmy uwagę, że tutaj n i m to nie są te same wielkości, które występują jako oznaczenia indeksów atrybutów w Algorytmie HFILL). Wówczas liczba odczytów faktów z porcji KB_H (t_{sc}, t_{ec}) wynosi n, liczbę porównań wartości atrybutów w krokach 4–12 możemy oszacować jako iloczyn mn, natomiast liczba nowych faktów, zapisanych w $KB^{\#}_H$ (t_{sc}, t_{ec}) wynosi n. Stąd też sumaryczną liczbę rozważanych operacji możemy oszacować za pomocą poniższej funkcji T_z (m, n).

$$|T_z(m, n)| = |n + mn + n| = |n(2 + m)| \le |n(m + m)| = 2|mn| = c|F(m,n)|,\ c = 2,\ \text{dla } n \ge 0,\ m \ge 2.$$

A zatem algorytm HFILL ma wielomianową złożoność obliczeniową rzędu $O(mn)$.

3.5.4 Eliminacja wartości nieznanych

Problem usuwania wartości N w przekształconej porcji faktów, realizowanego w cyklu APS przez procedurę `Usuń_Wartości_N` (wiersz 12.), można sformułować w poniższy sposób.

Problem: *Usuwanie wartości nieznanych w porcji faktów*

Dane: porcja faktów $KB^{\#}_H$ (t_1, t_2), schemat historii $S^{\#}$, maksymalna dopuszczalna liczba atrybutów z nieznaną wartością w jednym fakcie η_c.

Wyznaczyć: porcję faktów $KB^{\#}_H$ (t_1, t_2) bez wartości N.

Algorytm rozwiązania problemu

Algorytm ENV *Usuwanie nieznanych wartości atrybutów*

Wejście: porcja faktów $KB^{\#}_H$ (t_{sc}, t_{ec}), maksymalna dopuszczalna liczba atrybutów z nieznaną wartością w jednym fakcie $\eta_c(v_c)$, gdzie $v_c \in KB_T$; zbiory $S^S_H \subseteq S_H \setminus \{K, T\}$, $S^M_H \subseteq S_H \setminus \{K, T\}$, gdzie $S^S_H \cap S^M_H \equiv \varnothing$; schemat $S^{\#} = \{K, T, U'\}$.

Wyjście: porcja faktów $KB^{\#}_H(t_{sc}, t_{ec})$ bez wartości N; zaktualizowane wartości $t_{sc}(v_c)$ i $t_{ec}(v_c)$.

1. Dla każdego faktu $s^* \in KB^{\#}_H(t_{sc}, t_{ec})$ powtarzaj kroki 2–4.

2. $m := card\ \{A_i^{(v)} \in S^{\#}: A_i^{(v)}(s^*) = N\}$

3. Jeżeli $0 < m \leq \eta_c$, to $KB^{\#}_H(t_{sc}, t_{ec}) := KB^{\#}_H(t_{sc}, t_{ec}) \cup$
$\{s' = \{(K, K(s^*)), (T, T(s^*)), (A_1^{(v1)}, a_1^{(v1)}), (A_1^{(v2)}, a_1^{(v2)}), ..., (A_1^{(vk)}, a_1^{(vk)}), ..., (A_n^{(vl)}, a_n^{(vl)}),$
$(A_n^{(v\_l+1)}, a_n^{(v\_l+1)}), ..., (A_n^{(vp)}, a_n^{(vp)})\}: A_i^{(vj)} \in S^{\#} \wedge i \in \{1,2, \ldots, n\} \wedge j \in \{1,2, \ldots, p\}$
$\wedge (\forall A_i^{(vj)} \in S^{\#}$, gdzie $A_i \in S^M_H . (A_i^{(vj)}(s') \in \{0, 1\}) \wedge (A_i^{(vj)}(s^*) \neq \mathrm{N} \Leftrightarrow A_i^{(vj)}(s^*) = A_i^{(vj)}(s')))$
$\wedge (\forall A_i^{(vj)} \in S^{\#}$, gdzie $A_i \in S^S_H . (A_i^{(vj)}(s') \in \{0, 1\}) \wedge (A_i^{(vj)}(s^*) \neq \mathrm{N} \Leftrightarrow A_i^{(vj)}(s^*) = A_i^{(vj)}(s'))$
$\wedge (\forall A_i \in S^S_H . card\ \{A_i^{(vj)} \in S^{\#}: A_i^{(vj)}(s') = 1\} = 1))\}$

4. Jeżeli $m > 0$, to $KB^{\#}_H(t_{sc}, t_{ec}) := KB^{\#}_H(t_{sc}, t_{ec}) \setminus \{s^*\}$.

5. Koniec Dla.

6. $t_{sc}(v_c) := \mathrm{MIN}\{T(s'): s' \in KB^{\#}_H(t_{sc}, t_{ec})\}$.

7. $t_{ec}(v_c) := \mathrm{MAX}\{T(s'): s' \in KB^{\#}_H(t_{sc}, t_{ec})\}$.

8. Zwróć $KB^{\#}_H(t_{sc}, t_{ec})$.

Komentarz

Algorytm ENV zastępuje wszystkie fakty z co najmniej 1 i co najwyżej η_c wartościami N, nowymi faktami, w których atrybuty o dotychczas nieznanych wartościach przyjmują wszystkie dopuszczalne wartości. Złożony warunek w kroku 3. zapewnia podstawową spójność danych, gwarantując, że w podzbiorze atrybutów $A_i^{(vj)} \in S^{\#}$, pochodzących od atrybutu jednowartościowego $A_i \in S^S_H$, tylko jeden element może przyjąć wartość 1. Z kolei w podzbiorze atrybutów $A_w^{(vj)} \in S^{\#}$, pochodzących od atrybutu wielowartościowego $A_w \in S^M_H$, wartość 1 może przyjąć więcej, niż jeden element.

Zaktualizowanie w wektorze danych przebiegu v_c wartości czasu pierwszego faktu t_{sc} i czasu ostatniego faktu t_{ec} w badanej porcji jest konieczne, ponieważ w kroku 4. tak najwcześniejszy, jak i najpóźniejszy fakt może być usunięty bez zastąpienia go innymi faktami (w kroku 3.), jeśli zawiera on więcej, niż η_c wartości nieznanych.

Warto jeszcze zauważyć, że w wynikowej porcji $KB^{\#}_H(t_{sc}, t_{ec})$ oba atrybuty specjalne, to znaczy zarówno klucz K, jak i czas T, mogą utracić własność unikalności, ponieważ zgodnie z warunkiem w kroku 3. wszystkie nowe fakty s', generowane z faktu s^*, zawierającego wartości N, mają takie same wartości K i T, jak bazowy fakt s^*. Brak unikalności atrybutów K i T nie powinien stanowić problemu dla algorytmu eksploracji danych, który ma przetwarzać porcję $KB^{\#}_H(t_{sc}, t_{ec})$. Gdyby jednak było inaczej (np. przy zastosowaniu nowego algorytmu, który wymaga unikalnych kluczy przetwarzanych faktów), należałoby nieco zmodyfikować algorytm ENV, na przykład dodając na końcu pętlę, w której wszystkie fakty nowej porcji $KB^{\#}_H(t_{sc}, t_{ec})$ miałyby nadane nowe, unikalne wartości kluczy K. Nie należy natomiast modyfikować powtarzających się wartości czasu T, gdyż przy obliczaniu średniego czasu

reguły opartej na porcji $KB^{\#}_H$ (t_{sc}, t_{ec}) zwielokrotnione wartości T działają jak wagi, czyli w pełni zgodnie z intuicją i założeniami metody APS.

Nazwa algorytmu jest utworzona na podstawie pierwszych liter opisu działania w języku angielskim: *usunięcie nieznanych wartości* (ang. *Elimination of N-Values*, ENV). Przykład obliczeniowy, ilustrujący działanie algorytmu ENV, jest zamieszczony w Dodatku A.

Analiza złożoności obliczeniowej algorytmu ENV

Szacujemy liczbę operacji wykonywanych przez algorytm, takich, jak: odczyt, porównanie i zapis wartości w strukturach danych. Przypuśćmy, że porcja $KB^{\#}_H$ (t_{sc}, t_{ec}) zawiera n faktów o schemacie $S^{\#} = \{K, T, U'\}$, z czego k faktów zawiera przynajmniej jedną wartość nieznaną, a maksymalna liczba wartości nieznanych w pojedynczym fakcie wynosi m; przy czym $k, m, n \in \mathbb{N}$ i $k \leq n$. (Symbole k, m i n nie oznaczają tutaj tych wielkości, które występują w Algorytmie ENV). Wówczas liczba odczytów faktów s^* z porcji $KB^{\#}_H$ (t_{sc}, t_{ec}) w kroku 2. wynosi n. Tyle samo, to znaczy n, jest operacji obliczania liczby wartości N w wierszu (krok 3.). Liczbę nowych faktów s', dodawanych do $KB^{\#}_H$ (t_{sc}, t_{ec}) w kroku 3. można oszacować z góry za pomocą iloczynu $k\ 2^m$. Jest to najbardziej pesymistyczna ocena, która zakłada, że we wszystkich k wierszach liczba atrybutów z wartością N jest maksymalna i wynosi m, oraz że żadne dwa z nich nie pochodzą od pojedynczego atrybutu jednowartościowego, przez co liczba generowanych faktów jest również zawsze maksymalna i wynosi 2^m. Następnie w kroku 4. wykonywanych jest k operacji usuwania starych faktów z wartościami N. Dodatkowo, w krokach 6 i 7 obliczane są nowe wartości parametrów t_{sc} *i* t_{ec}, co oznacza w najgorszym przypadku około $2(n + k2^m)$ operacji odczytu i porównań. Stąd też sumaryczną liczbę rozważanych operacji możemy oszacować za pomocą poniższej funkcji T_z (k, m, n).

$$|T_z(k, m, n)| = |n + n + k2^m + 2n + k2^m| = |4n + 2k2^m| = |4n + 4k2^{m-1}| \leq |4(n + n2^m)| =$$
$$|4n(1 + 2^m)| \leq 8|n\ 2^m| = c|F(m, n)|,\ c = 8,\ \text{dla}\ n \geq 0,\ m \geq 0.$$

A zatem algorytm ENV ma wysoką, wykładniczą złożoność obliczeniową rzędu $O(n2^m)$. Z tego względu może on być wydajnie stosowany tylko dla niskich wartości progu η_c.

3.5.5 Odkrywanie reguł związku

Charakterystyka procesu odkrywania reguł związku, realizowanego przez procedurę `Znajdź_Reguły` (wiersz 13.), jest zgodna z definicją problemu, podaną w Rozdziale 1. *Eksploracja danych jako metoda maszynowego uczenia się*. Różnica pomiędzy oboma sformułowaniami problemu polega na zmodyfikowanej reprezentacji reguł $r = (X, Y, sup, con, b, t_m)$, stosowanej w metodzie APS, która wymaga opisania reguł za pomocą dodatkowych parametrów: liczby bazowej b oraz czasu t_m. Proponowany poniżej algorytm ARM zawiera w sobie wywołanie właściwego algorytmu odkrywania reguł związku, którego wybór zależy od konkretnej aplikacji.

Problem: *Znajdowanie reguł związku*

Dane są: zbiór atrybutów U', porcja faktów $B = KB^{\#}_H$ (t_{sc}, t_{ec}) o schemacie $S^{\#} = \{K, T, U'\}$ oraz progi minimalnego poparcia σ_c i pewności γ_c, dla bieżącego przebiegu.

Wyznaczyć: zbiór reguł R (B, σ_c, γ_c).

Algorytm rozwiązania problemu

Algorytm ARM *Znajdowanie reguł związku*

Wejście: porcja $KB^{\#}_{H}(t_{sc}; t_{ec})$; ograniczenia $X_c(v_c)$, $Y_c(v_c)$, $\sigma_c(v_c)$, $\gamma_c(v_c)$, $m_x(v_c)$, $m_y(v_c)$, $b_c(v_c)$, $t_{mc}(v_c)$, gdzie $v_c \in KB_T$.

Wyjście: zbiór nowych reguł R.

1. $R := \varnothing$.
2. Jeżeli algorytm odkrywania reguł związku przyjmuje ograniczenia atrybutów poprzednika X i następnika Y reguły, to:
   $R :=$ *Algorytm_Odkrywania_Reguł* $(\sigma_c, \gamma_c, X_c, Y_c, m_x, m_y)$.
3. Jeżeli algorytm odkrywania reguł związku nie przyjmuje ograniczeń atrybutów poprzednika X i następnika Y reguły, to wykonaj kroki 4–5.
4. $R :=$ *Algorytm_Odkrywania_Reguł* (σ_c, γ_c).
5. $R := \{r \in R: X(r) \subseteq X_c \wedge Y(r) \subseteq Y_c \wedge card\, X(r) \le m_x \wedge card\, Y(r) \le m_y\}$.
6. Koniec Jeżeli.
7. Jeżeli $\neg R \equiv \varnothing$, to wykonaj krok 8.
8. Dla każdej reguły $r_i \in R$, $i \in [1; card\, R]$ powtarzaj kroki 9–10.
9. $b(r_i) := b_c(v_c)$.
10. $t_m(r_i) := t_{mc}(v_c)$.
11. Koniec Dla.
12. Koniec Jeżeli.
13. Zwróć: R.

Komentarz

Algorytm ARM stanowi jedynie swoistą *obudowę* właściwego algorytmu odkrywania reguł związku, która umożliwia jego dopasowanie do wymogów metody APS. W algorytmie rozważane są dwa przypadki. W pierwszym z nich algorytm *Algorytm_Odkrywania_Reguł*, obok standardowych progów minimalnego poparcia i pewności (odpowiednio σ_c i γ_c), pozwala także na przekazanie ograniczeń semantycznych dotyczących odkrywanych reguł, a więc: zbioru atrybutów, które mogą wystąpić (w sensie *są dozwolone*) w poprzedniku i następniku reguły – odpowiednio X_c i Y_c, oraz maksymalnej liczby atrybutów w poprzedniku i następniku – odpowiednio m_x i m_y. Wówczas wszystkie ograniczenia są przekazywane do tego algorytmu, a on daje w wyniku zbiór reguł R. W drugim przypadku zakładamy, że wykorzystany *Algorytm_Odkrywania_Reguł* akceptuje wyłącznie progi σ_c i γ_c. W takiej sytuacji algorytm ten zwraca zbiór reguł, który musi być następnie odpowiednio zawężony,

zgodnie z dodatkowymi ograniczeniami X_c, Y_c, m_x, m_y (krok 5.). Rozróżnienie na te dwa przypadki zwiększa elastyczność algorytmu ARM, pozwalając na dopasowanie do różnych, gotowych algorytmów odkrywania reguł związku.

Niezależnie od tego, który z powyższych przypadków zachodzi, wszystkim nowo odkrytym regułom nadawane są takie same wartości miar $b(r)$ i $t_m(r)$, czyli wspólne dla wszystkich reguł znajdowanych w pojedynczym przebiegu.

Nazwa algorytmu ARM jest utworzona na podstawie pierwszych liter angielskiego opisu realizowanej operacji *odkrywanie reguł związku* (ang. *Association Rule Mining*, ARM).

Analiza złożoności obliczeniowej algorytmu ARM

Złożoność obliczeniowa algorytmu ARM jest silnie zależna od złożoności wywoływanego w nim, oddzielnego *Algorytmu_Odkrywania_Reguł*. Dodatkowe operacje, realizowane w krokach 9. i 10. dają złożoność rzędu $O(n)$, przy liczbie n reguł $r \in R$, a zatem nie stanowią one poważnego obciążenia obliczeniowego.

3.5.6 Aktualizacja bazy reguł

Kluczowy dla metody APS problem modyfikowania bazy reguł KB_R, na podstawie zbioru reguł nowo odkrytych w bieżącym przebiegu, można opisać tak, jak poniżej.

Problem: *Aktualizowanie bazy reguł*

Dane są: zbiór reguł KB_R wyznaczony na zbiorze faktów h_1;
zbiór reguł R wyznaczony na zbiorze faktów h_2,
gdzie $h_1 \cap h_2 \equiv \varnothing \wedge \forall s \in h_1, \forall s^* \in h_2 \,.\, T(s) < T(s^*)$.

Wyznaczyć: zbiór reguł KB'_R taki, $KB'_R \cong KB^S_R$, gdzie KB^S_R jest zbiorem reguł wyznaczonych na zbiorze faktów $h_1 \cup h_2$.

Algorytm rozwiązania problemu

Algorytm RMAIN *Aktualizowanie bazy reguł*

Wejście: baza reguł KB_R, zbiór nowo odkrytych reguł R; parametry bieżącego przebiegu, zapisane w wektorze $v_c \in KB_T$: próg minimalnego poparcia σ_c, próg minimalnej pewności γ_c, oczekiwane poparcie odrzucanych reguł $\hat{\sigma}_c \in [0\,;\sigma_c]$, oczekiwana pewność odrzucanych reguł $\hat{\gamma}_c \in [0\,;\gamma_c]$, średni czas faktów t_{mc}, liczba faktów w porcji b_c; parametry globalne, zapisane w wektorze $v_g \in KB_G$: próg minimalnego poparcia σ_g, próg minimalnej pewności γ_g, oczekiwane poparcie odrzucanych reguł $\hat{\sigma}_g \in [0\,;\sigma_g]$, oczekiwana pewność odrzucanych reguł $\hat{\gamma}_g \in [0\,;\gamma_g]$, średni czas regu t_{mg}, liczba przeanalizowanych faktów b_g, funkcja wpływu czasowego f_T; parametr pobrany z systemu operacyjnego: bieżący czas t_{now}.

Wyjście: zaktualizowane: KB_R, $\sigma_g(v_g)$, $\gamma_g(v_g)$, $\hat{\sigma}_g(v_g)$, $\hat{\gamma}_g(v_g)$, $t_{mg}(v_g)$, $b_g(v_g)$.

1. Zaktualizuj parametry wektora v_g:

$$\sigma_g(v_g) := \frac{\sigma_g b_g + \sigma_c b_c}{b_g + b_c}, \qquad \gamma_g(v_g) := \frac{\gamma_g b_g + \gamma_c b_c}{b_g + b_c}.$$

2. Dla każdej reguły $r_i \in R$, $i \in [1; card\ R]$, powtarzaj kroki 3–6.

3. Jeżeli $\exists p_j \in KB_R\ .\ p_j \equiv r_i$, to zaktualizuj parametry p_j:

$$sup(p_j) := \frac{b(p_j) f_T(t_{now} - t_m(p_j)) sup(p_j) + b(r_i) f_T(t_{now} - t_m(r_i)) sup(r_i)}{b(p_j) f_T(t_{now} - t_m(p_j)) + b(r_i) f_T(t_{now} - t_m(r_i))},$$

$$con(p_j) := \frac{con(p_j) con(r_i) \left(b(p_j) sup(p_j) f_T(t_{now} - t_m(p_j)) + b(r_i) sup(r_i) f_T(t_{now} - t_m(r_i)) \right)}{b(p_j) sup(p_j) f_T(t_{now} - t_m(p_j)) con(r_i) + b(r_i) sup(r_i) f_T(t_{now} - t_m(r_i)) con(p_j)},$$

$$t_m(p_j) := \frac{b(p_j) t_m(p_j) + b(r_i) t_m(r_i)}{b(p_j) + b(r_i)},$$

$$b(p_j) := b(p_j) + b(r_i).$$

4. Jeżeli po aktualizacji $sup(p_j) < \sigma_g$ lub $con(p_j) < \gamma_g$, to $KB_R := KB_R \setminus \{p_j\}$.

5. Jeżeli $\neg\exists p_j \in KB_R\ .\ p_j \equiv r_i$, to zaktualizuj parametry r_i:

$$sup(r_i) := \frac{b_g f_T(t_{now} - t_{mg}) \hat{\sigma}_g + b(r_i) f_T(t_{now} - t_m(r_i)) sup(r_i)}{b_g f_T(t_{now} - t_{mg}) + b(r_i) f_T(t_{now} - t_m(r_i))},$$

$$con(r_i) := \frac{\hat{\gamma}_g con(r_i) \left(b_g \hat{\sigma}_g f_T(t_{now} - t_{mg}) + b(r_i) sup(r_i) f_T(t_{now} - t_m(r_i)) \right)}{b_g \hat{\sigma}_g f_T(t_{now} - t_{mg}) con(r_i) + b(r_i) sup(r_i) f_T(t_{now} - t_m(r_i)) \hat{\gamma}_g}, \text{ jeśli } 0 < \hat{\gamma}_g \leqslant 1,$$

$$con(r_i) := con(r_i), \text{ jeśli } \hat{\gamma}_g = 0,$$

$$t_m(r_i) := \frac{b_g \cdot t_{mg} + b(r_i) \cdot t_m(r_i)}{b_g + b(r_i)},$$

$$b(r_i) := b_g + b(r_i).$$

6. Jeżeli po aktualizacji $sup(r_i) \geq \sigma_g \wedge con(r_i) \geq \gamma_g$, to $KB_R := KB_R \cup \{r_i\}$.

7. Koniec Dla.

8. Dla każdej reguły $p_j \in KB_R$, takiej, że $\neg\exists r_i \in R\ .\ p_j \equiv r_i$, powtarzaj kroki 9–10.

9. Zaktualizuj parametry reguły p_j:

$$sup(p_j) := \frac{b(p_j) f_T(t_{now} - t_m(p_j)) sup(p_j) + b_c f_T(t_{now} - t_{mc}) \hat{\sigma}_c}{b(p_j) f_T(t_{now} - t_m(p_j)) + b_c f_T(t_{now} - t_{mc})},$$

$$con(p_j) := \frac{con(p_j) \hat{\gamma}_c \left(b(p_j) sup(p_j) f_T(t_{now} - t_m(p_j)) + b_c \hat{\sigma}_c f_T(t_{now} - t_{mc}) \right)}{b(p_j) sup(p_j) f_T(t_{now} - t_m(p_j)) \hat{\gamma}_c + b_c \hat{\sigma}_c f_T(t_{now} - t_{mc}) con(p_j)}, \text{ jeśli } 0 < \hat{\gamma}_c \leqslant 1,$$

$$con(p_j) := con(p_j), \text{ jeśli } \hat{\gamma}_c = 0,$$

$$t_m(p_j) := \frac{b(p_j) t_m(p_j) + b_c t_{mc}}{b(p_j) + b_c},$$

$$b(p_j) := b(p_j) + b_c.$$

10. Jeżeli po aktualizacji $sup(p_j) < \sigma_g$ lub $con(p_j) < \gamma_g$, to $KB_R := KB_R \setminus \{p_j\}$.

11. Koniec Dla.

12. Zaktualizuj parametry wektora v_g:

$$\hat{\sigma}_g(v_g) := \frac{\hat{\sigma}_g b_g + \hat{\sigma}_c b_c}{b_g + b_c}, \qquad \hat{\gamma}_g(v_g) := \frac{\hat{\gamma}_g b_g + \hat{\gamma}_c b_c}{b_g + b_c}, \qquad t_{mg}(v_g) := \frac{b_g t_{mg} + b_c t_{mc}}{b_g + b_c}.$$

13. Zaktualizuj parametr wektora v_g: $b_g(v_g) := b_g + b_c$.

14. Zwróć zaktualizowane: KB_R, $\sigma_g(v_g)$, $\gamma_g(v_g)$, $\hat{\sigma}_g(v_g)$, $\hat{\gamma}_g(v_g)$, $t_{mg}(v_g)$, $b_g(v_g)$.

Komentarz

W głównych częściach algorytmu aktualizacji bazy reguł rozpatrywane są: (i) reguły, które występują jednocześnie w zbiorach R i KB_R (pod uwagę brana jest równość semantyczna, natomiast różnice innych parametrów są dopuszczalne) (kroki 2–4); (ii) reguły, które występują w zbiorze R, a nie występują w zbiorze KB_R (kroki 2, 5–6); (iii) reguły, które występują w zbiorze KB_R, a nie występują w zbiorze R (kroki 8–10). W każdym z tych przypadków dla reguł aktualizowane są miary poparcia, pewności, liczby bazowej i czasu, na podstawie odpowiednich wzorów. W przypadku (ii) nieznane wartości parametrów reguł, które nie występują w KB_R, są szacowane za pomocą estymatorów $\hat{\sigma}_g \in [0; \sigma_g]$ oraz $\hat{\gamma}_g \in [0; \gamma_g]$. W przypadku (iii) nieznane wartości parametrów reguł, które nie występują w R, są szacowane za pomocą estymatorów $\hat{\sigma}_c \in [0; \sigma_c]$ oraz $\hat{\gamma}_c \in [0; \gamma_c]$. Do bazy KB_R dodawane są reguły ze zbioru R, których nowe wartości poparcia i pewności spełniają wymaganych progi, natomiast te reguły, które progów tych nie spełniają – są usuwane KB_R.

Bardzo ważna jest kolejność aktualizowania poszczególnych parametrów w krokach 3, 5 i 9, która powinna przebiegać tak, jak jest to podane w algorytmie – z góry do dołu. Zamiana kolejności aktualizacji (np. obliczenie najpierw liczby bazowej b reguły, a potem jej czasu t_m) spowodowałaby wystąpienie błędnych wyników.

Nazwa algorytmu jest utworzona na podstawie pierwszych liter opisu działania w języku angielskim: *utrzymanie reguł* (ang. *Rule MAINtenance*, RMAIN). Prezentowany tutaj algorytm RMAIN stanowi poprawioną i rozszerzoną wersję algorytmu utrzymania bazy reguł związku, przedstawionego we wcześniejszej pracy autora [Dud2005a]. Przykład obliczeniowy dla przebiegu algorytmu RMAIN jest przedstawiony w Dodatku A.

Analiza złożoności obliczeniowej algorytmu RMAIN

Szacujemy z góry liczbę operacji wykonywanych przez algorytm, takich, jak: odczyt, porównanie i zapis wartości w strukturach danych. Przypuśćmy, że zbiór R zawiera n reguł, a zbiór KB_R zawiera m reguł, przy czym $m, n \in \mathbb{N}$. Wykonywane jest n odczytów reguł ze zbioru R (w kroku 2.). W krokach 3 i 4 wykonywanych jest: mn odczytów reguł ze zbioru KB_R, mn porównań reguł z obu zbiorów, $4mn$ obliczeń nowych wartości parametrów (krok 3.) i mn operacji kasowania reguł ze zbioru KB_R (krok 4.). W kroku 5. wykonywanych jest n obliczeń nowych wartości parametrów, w kroku 6. zaś wykonywanych jest n operacji kasowania reguł ze zbioru R. W kroku 8. przeprowadzanych jest m odczytów reguł z bazy KB_R.

W krokach 9. i 10. wykonywane jest odpowiednio: $4m$ obliczeń nowych parametrów reguł oraz m operacji usuwania reguł z KB_R. Stąd też sumaryczną liczbę rozważanych operacji możemy oszacować za pomocą poniższej funkcji $T_z(m, n)$.

$|T_z(m, n)| = |n + mn + mn + 4mn + mn + n + n + m + 4m + m| = |3n + 7mn + 6m|$
$\leq |3mn + 7mn + 6mn| = 16\,|mn| = c|F(m, n)|$, $c = 16$, dla $n > 0$, $m > 0$.

A zatem algorytm RMAIN ma wielomianową złożoność obliczeniową rzędu $O(mn)$.

Własności algorytmu RMAIN

W bieżącej sekcji szczegółowo omawiane i dowodzone są kluczowe własności algorytmu aktualizacji bazy reguł RMAIN. Stosowane oznaczenia są zgodne z notacją reguł związku wprowadzoną w Rozdziale 1. oraz z definicjami przedstawionymi w podrozdziale 3.3 *Reprezentacja wiedzy agenta*.

W pierwszej kolejności wykażemy prawidłowość stosowanych wzorów proporcji częstościowych, które są podstawą obliczania zaktualizowanych miar poparcia i pewności reguł w algorytmie RMAIN.

Lemat 1

Dane są: zbiór atrybutów U oraz dwa niepuste, rozłączne zbiory faktów h_1 i h_2.
Dla każdego $X \subseteq U$, częstość X w połączonych zbiorach h_1 i h_2 jest dana równaniem:

$$freq(X, h_1 \cup h_2) = \frac{card(h_1)\, freq(X, h_1) + card(h_2)\, freq(X, h_2)}{card(h_1) + card(h_2)}.$$

Dowód

Częstość podzbioru atrybutów $Y \subseteq U$ w bazie faktów B jest zdefiniowana następująco:

$$freq(Y, B) = \frac{card\{s \in B : s|-Y\}}{card\, B}.$$

A zatem:

$$freq(X, h_1 \cup h_2) = \frac{card\{s \in h_1 \cup h_2 : s|-X\}}{card(h_1 \cup h_2)} =$$

$$= \frac{card\{s \in h_1 : s|-X\} + card\{s \in h_2 : s|-X\}}{card(h_1) + card(h_2)} =$$

$$= \frac{\frac{card(h_1)\, card\{s \in h_1 : s|-X\}}{card(h_1)} + \frac{card(h_2)\, card\{s \in h_2 : s|-X\}}{card(h_2)}}{card(h_1) + card(h_2)} =$$

$$= \frac{card(h_1)\, freq(X, h_1) + card(h_2)\, freq(X, h_2)}{card(h_1) + card(h_2)}. \blacksquare$$

Wniosek 1

Dane są: zbiór atrybutów U oraz dwa niepuste, rozłączne zbiory faktów h_1 i h_2. Dla każdej reguły r: $X \Rightarrow Y$, gdzie $X \subset U$, $Y \subset U$ oraz $X \cap Y = \varnothing$, poparcie reguły r w połączonych zbiorach h_1 i h_2 jest dane równaniem:

$$sup(r, h_1 \cup h_2) = \frac{card(h_1)\, sup(r, h_1) + card(h_2)\, sup(r, h_2)}{card(h_1) + card(h_2)}.$$

Dowód

Poparcie reguły r: $X \Rightarrow Y$, gdzie $X \subset U$, $Y \subset U$ oraz $X \cap Y = \varnothing$ w zbiorze faktów B jest równe częstości zbioru $X \cup Y$ w zbiorze faktów B, to znaczy $sup(r, B) = freq(X \cup Y, B)$ (1^*). Dowodzone równanie jest oczywistą konsekwencją równania (1^*) oraz *Lematu 1*. ■

Obserwacja 1

Dla każdej reguły r: $X \Rightarrow Y$, gdzie $X \subset U$, $Y \subset U$ oraz $X \cap Y = \varnothing$, jeżeli $freq(X, B) > 0$, to: $sup(r, B) = 0$ wtedy i tylko wtedy, gdy $con(r, B) = 0$.

Dowód

Pewność reguły r w zbiorze faktów B jest dana wzorem:

$$con(r, B) = \frac{sup(r, B)}{freq(X, B)}, \quad \text{co jest równoważne:} \quad sup(r, B) = con(r, B)\, freq(X, B).$$

Dowodzona zależność jest zatem oczywistą konsekwencją powyższych wzorów, dla częstości $freq(X, B) > 0$. ■

Twierdzenie 1

Dane są: zbiór atrybutów U oraz dwa niepuste, rozłączne zbiory faktów h_1 i h_2. Dla każdej reguły r: $X \Rightarrow Y$, gdzie $X \subset U$, $Y \subset U$ oraz $X \cap Y = \varnothing$, jeżeli $con(r, h_1) > 0$ i $con(r, h_2) > 0$, to pewność reguły r w połączonych zbiorach h_1 i h_2 jest dana równaniem:

$$con(r, h_1 \cup h_2) = \frac{con(r, h_1)\, con(r, h_2)\left(card(h_1)\, sup(r, h_1) + card(h_2)\, sup(r, h_2)\right)}{card(r, h_1)\, sup(r, h_1)\, con(r, h_2) + card(h_2)\, sup(r, h_2)\, con(r, h_1)}.$$

Dowód

Korzystamy z *Lematu 1*, *Wniosku 1* oraz ogólnej definicji pewności. Pewność reguły r: $X \Rightarrow Y$, gdzie $X \subset U$, $Y \subset U$ oraz $X \cap Y = \varnothing$ w zbiorze faktów B jest równa poparciu tej reguły w zbiorze B podzielonemu przez większą od zera, częstość zbioru X w zbiorze faktów B, to znaczy:

$$con(r, B) = \frac{sup(r, B)}{freq(X, B)}, \quad \text{co jest równoważne:} \quad sup(r, B) = con(r, B)\, freq(X, B).$$

A zatem:

$$con(r, h_1 \cup h_2) = \frac{sup(r, h_1 \cup h_2)}{freq(X, h_1 \cup h_2)} =$$

$$= \frac{\frac{1}{card(h_1)+card(h_2)}\left(card(h_1)\,sup(r, h_1)+card(h_2)\,sup(r, h_2)\right)}{\frac{1}{card(h_1)+card(h_2)}\left(card(h_1)\,freq(X, h_1)+card(h_2)\,freq(X, h_2)\right)} =$$

$$= \frac{\boldsymbol{con(r, h_1)\,con(r, h_2)}\left(card(h_1)\,sup(r, h_1)+card(h_2)\,sup(r, h_2)\right)}{\boldsymbol{con(r, h_1)\,con(r, h_2)}\left(card(h_1)\,freq(X, h_1)+card(h_2)\,freq(X, h_2)\right)} =$$

$$= \frac{con(r, h_1)\,con(r, h_2)\left(card(h_1)\,sup(r, h_1)+card(h_2)\,sup(r, h_2)\right)}{card(h_1)\,\boldsymbol{con(r, h_1)\,freq(X, h_1)}\,con(r, h_2)+card(h_2)\,\boldsymbol{con(r, h_2)\,freq(X, h_2)}\,con(r, h_1)} =$$

$$= \frac{con(r, h_1)\,con(r, h_2)\left(card(h_1)\,sup(r, h_1)+card(h_2)\,sup(r, h_2)\right)}{card(r, h_1)\,sup(r, h_1)\,con(r, h_2)+card(h_2)\,sup(r, h_2)\,con(r, h_1)}. \blacksquare$$

Powyższe zależności wskazują, że wzory proporcji częstościowych, stosowane w algorytmie RMAIN, zapewniają inkrementacyjne uzyskanie identycznego zbioru reguł w stosunku do zbioru reguł otrzymanego wsadowo na całym zbiorze faktów. Zależności bazują jednak na wyidealizowanym założeniu, że dokładnie te same reguły (semantycznie równe) występują zarówno w porcji faktów dotychczas przetworzonej, jak w porcji bieżąco analizowanej. W rzeczywistym przypadku zbiory te mogą się różnić, szczególnie, jeśli: (i) stosowane są niezerowe progi minimalnego poparcia i pewności reguł, (ii) fakty są na tyle zróżnicowane, że w obu porcjach odkrywane są różne reguły. Stąd też w dalszej części rozważany jest przypadek, w którym dana reguła jest odkrywana tylko w jednej porcji faktów, dlatego, że w drugiej nie osiągnęła ona wymaganego progu minimalnego poparcia.

Twierdzenie 2

Dane są: zbiór atrybutów U oraz dwa niepuste, rozłączne zbiory faktów h_1 i h_2. Dla każdej reguły r: $X \Rightarrow Y$, gdzie $X \subset U$, $Y \subset U$ oraz $X \cap Y = \emptyset$, jeżeli $sup(r, h_1) \geq \sigma$ i $sup(r, h_2) \in (0; \sigma)$, dla $\sigma \in (0; 1]$, to dla każdego $\hat{\sigma} \in (0; \sigma)$ maksymalny błąd oceny poparcia reguły r w połączonych zbiorach h_1 i h_2 przez wyrażenie (2^*) jest mniejszy, niż maksymalny błąd oceny tego poparcia przez wyrażenie (3^*):

$$sup(r, h_1 \cup h_2) = \frac{card(h_1)\,sup(r, h_1)+card(h_2)\,\hat{\sigma}}{card(h_1)+card(h_2)} \quad (2^*)$$

$$sup(r, h_1 \cup h_2) = \frac{card(h_1)\,sup(r, h_1)}{card(h_1)+card(h_2)} \quad (3^*).$$

Dowód

Wyznaczymy najpierw oddzielnie maksymalne błędy oceny wyrażeń (2^*) i (3^*), a następnie obliczymy ich różnicę.

Rzeczywista wartość poparcia reguły r w połączonych zbiorach h_1 i h_2 jest dana równaniem:

$$sup(r, h_1 \cup h_2) = \frac{card(h_1)\,sup(r, h_1) + card(h_2)\,sup(r, h_2)}{card(h_1) + card(h_2)} \quad (4^*).$$

Stąd też maksymalny błąd ε_1 oceny poparcia reguły r w połączonych zbiorach h_1 i h_2 przez wyrażenie (2^*) jest równy granicy funkcji, będącej wartością bezwzględną różnicy wyrażeń (2^*) i (4^*), przy $sup(r, h_2)$ dążącym do σ:

$$\varepsilon_1 = \lim_{sup(r,h_2)\to\sigma} \left| \frac{card(h_1)\,sup(r, h_1) + card(h_2)\,sup(r, h_2)}{card(h_1) + card(h_2)} - \frac{card(h_1)\,sup(r, h_1) + card(h_2)\,\hat{\sigma}}{card(h_1) + card(h_2)} \right|$$

$$\varepsilon_1 = \lim_{sup(r,h_2)\to\sigma} \left| \frac{card(h_1)\,sup(r, h_1) + card(h_2)\,sup(r, h_2) - card(h_1)\,sup(r, h_1) - card(h_2)\,\hat{\sigma}}{card(h_1) + card(h_2)} \right|$$

$$\varepsilon_1 = \lim_{sup(r,h_2)\to\sigma} \left| \frac{card(h_2)\,sup(r, h_2) - card(h_2)\,\hat{\sigma}}{card(h_1) + card(h_2)} \right| = \frac{(\sigma - \hat{\sigma})\,card(h_2)}{card(h_1) + card(h_2)}.$$

Z kolei dla wyrażenia (3^*) obliczamy odpowiednio błąd ε_2:

$$\varepsilon_2 = \lim_{sup(r,h_2)\to\sigma} \left| \frac{card(h_1)\,sup(r, h_1) + card(h_2)\,sup(r, h_2)}{card(h_1) + card(h_2)} - \frac{card(h_1)\,sup(r, h_1)}{card(h_1) + card(h_2)} \right|$$

$$\varepsilon_2 = \lim_{sup(r,h_2)\to\sigma} \left| \frac{card(h_1)\,sup(r, h_1) + card(h_2)\,sup(r, h_2) - card(h_1)\,sup(r, h_1)}{card(h_1) + card(h_2)} \right|$$

$$\varepsilon_2 = \lim_{sup(r,h_2)\to\sigma} \left| \frac{card(h_2)\,sup(r, h_2)}{card(h_1) + card(h_2)} \right| = \frac{\sigma\, card(h_2)}{card(h_1) + card(h_2)}.$$

Ponieważ $0 < \hat{\sigma} < \sigma$, obliczając różnicę ε_1 i ε_2 otrzymujemy:

$$\varepsilon_2 - \varepsilon_1 = \frac{\sigma\, card(h_2)}{card(h_1) + card(h_2)} - \frac{(\sigma - \hat{\sigma})\,card(h_2)}{card(h_1) + card(h_2)} = \frac{\hat{\sigma}\, card(h_2)}{card(h_1) + card(h_2)}$$

$$\varepsilon_2 - \varepsilon_1 > 0 \Leftrightarrow \varepsilon_1 < \varepsilon_2 . \blacksquare$$

Z Twierdzenia 2 wynika, że stosowany w algorytmie RMAIN estymator $\hat{\sigma}$ oczekiwanego poparcia reguły zmniejsza maksymalny błąd oszacowania poparcia reguły w połączonych zbiorach faktów, jeśli przy aktualizacji dana reguła występuje tylko w bazie reguł KB_R albo tylko w zbiorze nowych reguł R.

W zastosowaniach praktycznych, zakładając równomierny rozkład poparcia reguł

odrzucanych z powodu nie osiągnięcia minimalnego poparcia, można domyślnie przyjmować jako wartość oczekiwanego poparcia $\hat{\sigma} = \frac{1}{2}\sigma$.

W dalszej części rozważań badany jest wpływ funkcji f_T na uzyskiwane inkrementacyjnie wartości poparcia i pewności reguł. Intuicyjnie funkcja f_T, zastosowana we wzorach algorytmu RMAIN, ma *promować* nowe reguły, a *deprecjonować* reguły stare. Działanie to ma polegać na odpowiednim zmniejszaniu, wraz ze zwiększającym się czasem zarejestrowania faktów, wiarygodności reguł, które na podstawie tych faktów są odkrywane. Wiarygodność ta wyrażana jest standardowo przez ich poparcie i pewność. Opisane tutaj pożądane własności funkcji f_T, zastosowanej we wzorach algorytmu RMAIN, są dowodzone w Twierdzeniach 3 i 4.

Twierdzenie 3

Dane są: zbiór atrybutów U oraz dwa niepuste, rozłączne zbiory faktów h_1 i h_2, przy czym $card(h_1) = card(h_2) = b$, $(\forall s_1 \in h_1 \, \forall s_2 \in h_2 \, . \, T(s_1) < T(s_2))$, średni czas faktów należących do h_1 wynosi t_{mg}, średni czas faktów należących do h_1 wynosi t_{mc}, gdzie $t_{mg} < t_{mc}$. Dana jest funkcja f_T: $[0; +\infty) \rightarrow [0; 1]$, taka, że: $f_T(0) = 1; \forall x_1, x_2 \in [0; +\infty)$. $x_1 < x_2 \Rightarrow f_T(x_1) \geq f_T(x_2)$.

Dla każdych dwóch reguł r i p, jeżeli $sup(p, h_1) = sup(r, h_2) = s \wedge sup(p, h_2) = sup(r, h_1) = \hat{\sigma}$ dla $\hat{\sigma} \in [0; \sigma)$, $s \geq \sigma$, to

$$\boldsymbol{sup(p, h_1 \cup h_2) \leq sup(r, h_1 \cup h_2)}, \text{ dla } sup(q, h_i \cup h_j), \text{ liczonego według wzoru:}$$

$$sup(q, h_j \cup h_k) = \frac{card(h_j) f_T(t_{now} - t_m(q, h_j)) sup(r, h_j) + card(h_k) f_T(t_{now} - t_m(q, h_k)) sup(r, h_k)}{card(h_j) + card(h_k)} \quad (5^*),$$

gdzie $t_m(q, h_j) < t_{now}$ oraz $t_m(q, h_k) < t_{now}$.

Dowód

Zauważmy, że $t_m(p, h_1) = t_{mg}$ i $t_m(r, h_2) = t_{mc}$. Ponieważ $t_{mg} < t_{mc}$, to $t_{now} - t_{mg} > t_{now} - t_{mc}$, i w rezultacie $f_T(t_{now} - t_{mg}) \leq f_T(t_{now} - t_{mc})$. Ponadto zachodzi $\hat{\sigma} < \sigma \leqslant s$. Stąd też, zgodnie z równaniem (5^*), otrzymujemy:

$$sup(p, h_1 \cup h_2) - sup(r, h_1 \cup h_2) =$$

$$= \frac{b f_T(t_{now} - t_{mg}) sup(p, h_1) + b f_T(t_{now} - t_{mc}) sup(p, h_2) - b f_T(t_{now} - t_{mg}) sup(r, h_1) - b f_T(t_{now} - t_{mc}) sup(r, h_2)}{b + b} =$$

$$= \frac{b f_T(t_{now} - t_{mg}) s + b f_T(t_{now} - t_{mc}) \hat{\sigma} - b f_T(t_{now} - t_{mg}) \hat{\sigma} - b f_T(t_{now} - t_{mc}) s}{2b} =$$

$$= \frac{(s - \hat{\sigma})(f_T(t_{now} - t_{mg}) - f_T(t_{now} - t_{mc}))}{2} \leqslant 0$$

$$sup(p, h_1 \cup h_2) - sup(r, h_1 \cup h_2) \leqslant 0 \Leftrightarrow sup(p, h_1 \cup h_2) \leqslant sup(r, h_1 \cup h_2). \blacksquare$$

Twierdzenie 4

Dane są: zbiór atrybutów U oraz dwa niepuste, rozłączne zbiory faktów h_1 i h_2, przy czym $card\,(h_1) = card\,(h_2) = b$, $(\forall s_1 \in h_1\, \forall s_2 \in h_2\,.\, T(s_1) < T(s_2))$, średni czas faktów należących do h_1 wynosi t_{mg}, średni czas faktów należących do h_1 wynosi t_{mc}, gdzie $t_{mg} < t_{mc}$. Dana jest funkcja f_T: $[0; +\infty) \rightarrow [0; 1]$, taka, że: $f_T(0) = 1; \forall x_1,x_2 \in [0; +\infty).\ x_1 < x_2 \Rightarrow f_T(x_1) \geq f_T(x_2)$.

Dla każdych dwóch reguł r i p, jeżeli $sup(p, h_1) = sup(r, h_2) = s \wedge con(p, h_1) = con(r, h_2) = c \wedge sup(p, h_2) = sup(r, h_1) = \hat{\sigma} \wedge con(p, h_2) = con(r, h_1) = \hat{\gamma}$ dla $\hat{\sigma} \in [0; \sigma)$, $s \geq \sigma$, dla $\hat{\gamma} \in [0; \gamma)$, $c \geq \gamma$, to

$\boldsymbol{con(p, h_1 \cup h_2) \leq con(r, h_1 \cup h_2)}$, dla $con(q, h_i \cup h_j)$, liczonego według wzoru (6*):

$$con(q, h_j \cup h_k) =$$

$$= \frac{con(q,h_j)\,con(q,h_k)\left(card\,(h_j)\,f_T(t_{now} - t_m(q,h_j))\,sup(r,h_j) + card\,(h_k)\,f_T(t_{now} - t_m(q,h_k))\,sup(r,h_k)\right)}{card\,(h_j)\,sup(q,h_j)\,f_T(t_{now} - t_m(q,h_j))\,con(q,h_k) + card\,(h_k)\,sup(q,h_k)\,f_T(t_{now} - t_m(q,h_k))\,con(q,h_j)},$$

gdzie $t_m(q, h_j) < t_{now}$ oraz $t_m(q, h_k) < t_{now}$.

Dowód

Zauważmy, że $t_m(p, h_1) = t_{mg}$ i $t_m(r, h_2) = t_{mc}$. Ponieważ $t_{mg} < t_{mc}$, to $t_{now} - t_{mg} > t_{now} - t_{mc}$, i w rezultacie $f_T(t_{now} - t_{mg}) \leq f_T(t_{now} - t_{mc})$. Ponadto zachodzi $\hat{\sigma} < \sigma \leqslant s \wedge \hat{\gamma} < \gamma \leqslant c$. W celu skrócenia zapisu, stosujemy symbole: $f_T(t_{now} - t_{mg}) = f_1$ oraz $f_T(t_{now} - t_{mc}) = f_2$. Zgodnie z równaniem (6*), otrzymujemy:

$$con(p, h_1 \cup h_2) - con(r, h_1 \cup h_2) =$$

$$= \frac{c\,\hat{\gamma}\left(bsf_1 + b\,\hat{\sigma}\,f_2\right)}{bsf_1\,\hat{\gamma} + b\,\hat{\sigma}\,f_2\,c} - \frac{\hat{\gamma}\,c\left(b\,\hat{\sigma}\,f_1 + b\,s\,f_2\right)}{b\,\hat{\sigma}\,f_1 c + b\,s\,f_2\,\hat{\gamma}} = \frac{c\,\hat{\gamma}\left(sf_1 + \hat{\sigma}\,f_2\right)}{sf_1\,\hat{\gamma} + \hat{\sigma}\,f_2\,c} - \frac{\hat{\gamma}\,c\left(\hat{\sigma}\,f_1 + s\,f_2\right)}{\hat{\sigma}\,f_1 c + s\,f_2\,\hat{\gamma}} =$$

$$= \frac{\left(c\,\hat{\gamma}\,sf_1 + c\,\hat{\gamma}\,\hat{\sigma}\,f_2\right)\left(\hat{\sigma}\,f_1 c + s\,f_2\,\hat{\gamma}\right) - \left(\hat{\gamma}\,c\,\hat{\sigma}\,f_1 + \hat{\gamma}\,c\,s\,f_2\right)\left(sf_1\,\hat{\gamma} + \hat{\sigma}\,f_2\,c\right)}{\left(sf_1\,\hat{\gamma} + \hat{\sigma}\,f_2\,c\right)\left(\hat{\sigma}\,f_1 c + s\,f_2\,\hat{\gamma}\right)} =$$

$$= \frac{c^2\,s\,\hat{\gamma}\,\hat{\sigma}\,f_1^2 + cs^2\,\hat{\gamma}^2\,f_1\,f_2 + c^2\,\hat{\gamma}\,\hat{\sigma}^2\,f_1\,f_2 + c\,s\,\hat{\gamma}^2\,\hat{\sigma}\,f_2^2 - cs\,\hat{\gamma}^2\,\hat{\sigma}\,f_1^2 - c^2\,\hat{\gamma}\,\hat{\sigma}^2\,f_1\,f_2 - cs^2\,\hat{\gamma}^2\,f_1\,f_2 - c^2\,s\,\hat{\gamma}\,\hat{\sigma}\,f_2^2}{\left(sf_1\,\hat{\gamma} + \hat{\sigma}\,f_2\,c\right)\left(\hat{\sigma}\,f_1 c + s\,f_2\,\hat{\gamma}\right)} =$$

$$= \frac{c^2\,s\,\hat{\gamma}\,\hat{\sigma}\left(f_1^2 - f_2^2\right) + c\,s\,\hat{\gamma}^2\,\hat{\sigma}\left(f_2^2 - f_1^2\right)}{\left(sf_1\,\hat{\gamma} + \hat{\sigma}\,f_2\,c\right)\left(\hat{\sigma}\,f_1 c + s\,f_2\,\hat{\gamma}\right)} = \frac{\left(f_2^2 - f_1^2\right)\left(c\,s\,\hat{\gamma}^2\,\hat{\sigma} - c^2\,s\,\hat{\gamma}\,\hat{\sigma}\right)}{\left(sf_1\,\hat{\gamma} + \hat{\sigma}\,f_2\,c\right)\left(\hat{\sigma}\,f_1 c + s\,f_2\,\hat{\gamma}\right)} = \frac{c\,s\,\hat{\gamma}\,\hat{\sigma}\left(f_2^2 - f_1^2\right)\left(\hat{\gamma} - c\right)}{\left(sf_1\,\hat{\gamma} + \hat{\sigma}\,f_2\,c\right)\left(\hat{\sigma}\,f_1 c + s\,f_2\,\hat{\gamma}\right)}.$$

Mianownik powyższego wyrażenia jest zawsze większy od zera. Różnica $f_2^2 - f_1^2 \geqslant 0$, ponieważ $f_1 \leqslant f_2$. Z kolei $\hat{\gamma} - c \leqslant 0$, dlatego, że $\hat{\gamma} < \gamma \leqslant c$. Stąd też otrzymujemy dowodzoną zależność:

$$con(p, h_1 \cup h_2) - con(r, h_1 \cup h_2) \leqslant 0 \Leftrightarrow con(p, h_1 \cup h_2) \leqslant con(r, h_1 \cup h_2).\blacksquare$$

3.5.7 Usuwanie przetworzonych faktów

Usuwanie przetworzonych faktów, realizowane przez procedurę `Usuń_Fakty` (wiersz 15.), jest problemem trywialnym, który nie wymaga szerszego omówienia. Jeśli historia KB_H ma postać tabeli relacyjnej bazy danych, procedurę `Usuń_Fakty` można efektywnie zaimplementować jako pojedynczy ciąg poleceń SQL, opartych na komendzie `DELETE... WHERE`. Pierwotna porcja faktów, pobrana z bazy przez procedurę `Wybierz_Fakty`, jest tutaj identyfikowana za pomocą wartości klucza ostatniego wybranego faktu $k_e(v_c)$, gdzie $v_c \in KB_T$. Jest to konieczne, ponieważ przetworzona porcja $KB^{\#}_H$ (t_{sc}, t_{ec}) może nie zawierać najwcześniejszego i najpóźniejszego faktu (mogły zostać usunięte z powodu zbyt dużej liczby wartości nieznanych), przez co wartości czasu t_{sc}, t_{ec} mogły ulec zmianie i nie mogą być traktowane jako wiarygodny wyznacznik zakresu faktów do usunięcia.

Złożoność obliczeniowa operacji usuwania faktów jest liniowa, to znaczy rzędu $O(n)$, gdzie n jest liczbą faktów do usunięcia.

3.6. Weryfikacja eksperymentalna metody APS

W celu oceny zaproponowanej metody APS, oprócz przytoczonej wyżej, weryfikacji formalnej, przeprowadzone zostały jej badania eksperymentalne. W niniejszym podrozdziale opisane są kolejno: cel i plan eksperymentu, środowisko testowe, dane wykorzystane w badaniach, wyniki eksperymentów, ich omówienie i wnioski.

3.6.1 Plan eksperymentu

Celem opisanej tutaj weryfikacji technicznej było sprawdzenie, czy metoda APS jest właściwym rozwiązaniem problemu naukowego, zdefiniowanego we Wprowadzeniu do pracy, a przytoczonego dokładniej na początku tego rozdziału. Główne postulaty, dotyczące rozwiązania tego problemu, wyrażone w celu pracy, można podsumować następująco.

(P1) Przetwarzana może być duża liczba obserwacji, przy zachowaniu ograniczonego rozmiaru przechowywanych danych historycznych, który jest mniejszy, niż w przypadku metody wsadowej.

(P2) Zbiór reguł, uzyskanych w sposób inkrementacyjny, powinien być porównywalny ze zbiorem reguł otrzymanych przez wsadowe przetwarzanie całego zbioru danych.

Dodatkowy postulat, który co prawda nie wynika bezpośrednio z celu pracy, ale jest pożądany z punktu widzenia wykorzystania metody APS w rozwiązaniach technicznych, dotyczy aspektu wydajności.

(P3) Czas przetwarzania faktów przy odkrywaniu reguł powinien być mniejszy dla metody inkrementacyjnej, niż dla metody wsadowej.

Postulat (P1) jest w oczywisty sposób spełniony przez to, że w cyklu metody APS wszystkie fakty, po przetworzeniu w danym przebiegu, są bezpowrotnie usuwane z historii. Tak więc rozmiar historii jest mniejszy, niż dla metody wsadowej, w której przechowywane są wszystkie fakty od początku ich rejestrowania.

Spełnianie postulatu (P2) zostało częściowo udowodnione formalnie w poprzednim podrozdziale, przez zestawione tam własności algorytmu aktualizacji bazy reguł RMAIN.

Przedstawiona niżej, weryfikacja eksperymentalna metody APS, miała na celu dodatkowe wykazanie spełniania postulatu (P2) oraz zbadanie, czy i w jakich warunkach spełniany jest postulat (P3).

Miary oceny zbioru reguł

Poniżej zdefiniowane są miary oceny i porównywania zbiorów reguł, które do tej pory były opisywane na poziomie intuicyjnym, w sposób niezobiektywizowany.

Oznaczenia

Przez $KB^B_R\,(h)$ oznaczamy zbiór reguł odkrytych wsadowo w zbiorze faktów h, to znaczy w pojedynczym przebiegu analizy przez cały badany zbiór h.

Przez $KB^I_R\,(h)$ oznaczamy zbiór reguł odkrytych inkrementacyjnie w zbiorze faktów h, to znaczy w przynajmniej dwóch przebiegach, na dwóch różnych porcjach faktów h_1 i h_2, takich, że $h_1 \cap h_2 \equiv \varnothing \wedge h_1 \cup h_2 \equiv h$.

Definicja 12 *Przecięcie semantyczne zbioru reguł*

Przecięciem semantycznym dwóch zbiorów reguł R_1 i R_2 nazywamy zbiór:

$$KB^O_R\left(R_1, R_2\right) \equiv \left\{r \in R_1 : \exists\, p \in R_2 . r \equiv p\right\}.$$

Definicja 13 *Współczynnik zgodności semantycznej*

Współczynnikiem zgodności semantycznej dwóch zbiorów reguł R_1 i R_2, oznaczanym przez $rule_{overlap}\,(R_1, R_2)$, nazywamy ułamek:

$$rule_{overlap}\left(R_1\,,\,R_2\right) = \frac{card\ KB^O_R\left(R_1,\ R_2\right)}{card\ R_1 + card\ R_2 - card\ KB^O_R\left(R_1,\ R_2\right)}.$$

Definicja 14 *Współczynnik zgodności poparcia*

Współczynnikiem zgodności poparcia dwóch zbiorów reguł R_1 i R_2, oznaczanym przez $sup_{overlap}\,(R_1, R_2)$, nazywamy wyrażenie:

$$sup_{overlap}\left(R_1, R_2\right) = \frac{1}{n}\sum_{i=1}^{n}\left(1 - \left|sup(p_i) - sup(r_i)\right|\right),$$

gdzie: $n = card\ KB^O_R\,(R_1, R_2) \wedge r_i \equiv p_i \wedge p_i \in KB^O_R\,(R_1, R_2) \wedge p_i \in R_1 \wedge r_i \in R_2 \wedge i = \{1, ..., n\}$.

Definicja 15 *Współczynnik zgodności pewności*

Współczynnikiem zgodności pewności dwóch zbiorów reguł R_1 i R_2, oznaczanym przez $con_{overlap}\,(R_1, R_2)$, nazywamy wyrażenie:

$$con_{overlap}(R_1, R_2) = \frac{1}{n}\sum_{i=1}^{n}\left(1 - \left|con(p_i) - con(r_i)\right|\right),$$

gdzie: $n = card\, KB^O_R(R_1, R_2) \wedge r_i \equiv p_i \wedge p_i \in KB^O_R(R_1, R_2) \wedge p_i \in R_1 \wedge r_i \in R_2 \wedge i = \{1, ..., n\}$.

Definicja 15 *Średnie odchylenie czasowe*

Średnim odchyleniem czasowym dwóch zbiorów reguł R_1 i R_2, oznaczanym przez $time_{dev}(R_1, R_2)$, nazywamy wyrażenie:

$$time_{dev}(R_1, R_2) = \frac{1}{nT_{ref}}\sum_{i=1}^{n}\left|t_m(p_i) - t_m(r_i)\right|,$$

gdzie: $n = card\, KB^O_R(R_1, R_2) \wedge r_i \equiv p_i \wedge p_i \in KB^O_R(R_1, R_2) \wedge p_i \in R_1 \wedge r_i \in R_2 \wedge i = \{1, ..., n\} \wedge T_{ref} \in D_T$ nazywamy *jednostką odniesienia*, przy czym D_T jest omawianym wcześniej, przeliczalnym zbiorem punktów czasowych, który jest uporządkowany przez relację silnego porządku liniowego $<$.

Komentarz

Wprowadzone wyżej miary (z wyjątkiem $time_{dev}$) pozwalają na procentowe porównywanie dwóch zbiorów reguł. I tak, współczynnik zgodności semantycznej wskazuje odsetek reguł, które są jednakowe semantycznie (bez badania innych parametrów) w obu zbiorach. Z kolei współczynniki zgodności poparcia i pewności pozwalają na obliczenie średniej, procentowej zgodności reguł, które występują w obu porównywanych zbiorach reguł i są sobie równe semantycznie. Każdy z trzech omówionych tutaj współczynników przyjmuje wartości z przedziału [0; 1], przy czym wartości 0 (zero) i 1 (jeden) oznaczają odpowiednio najmniejszą i największą zgodność badanych zbiorów. Nieco inaczej zdefiniowana jest miara średniego odchylenia czasowego, która umożliwia sprawdzanie zgodności czasu reguł równych semantycznie w dwóch zbiorach. Ponieważ w metodzie APS czas jest reprezentowany za pomocą struktury punktowej $(D_T, <)$, zgodność porównywanych zbiorów reguł jest mierzona w podanych jednostkach odniesienia T_{ref} (np. mogą to być minuty, godziny, doby – w zależności od dziedziny zastosowania i reprezentacji czasu w danym systemie informatycznym). Tutaj zatem największej zgodności zbiorów odpowiada wartość 0 (zero), a wartość najmniejszej zgodności nie jest określona.

Badania przy jednorodnym rozkładzie reguł

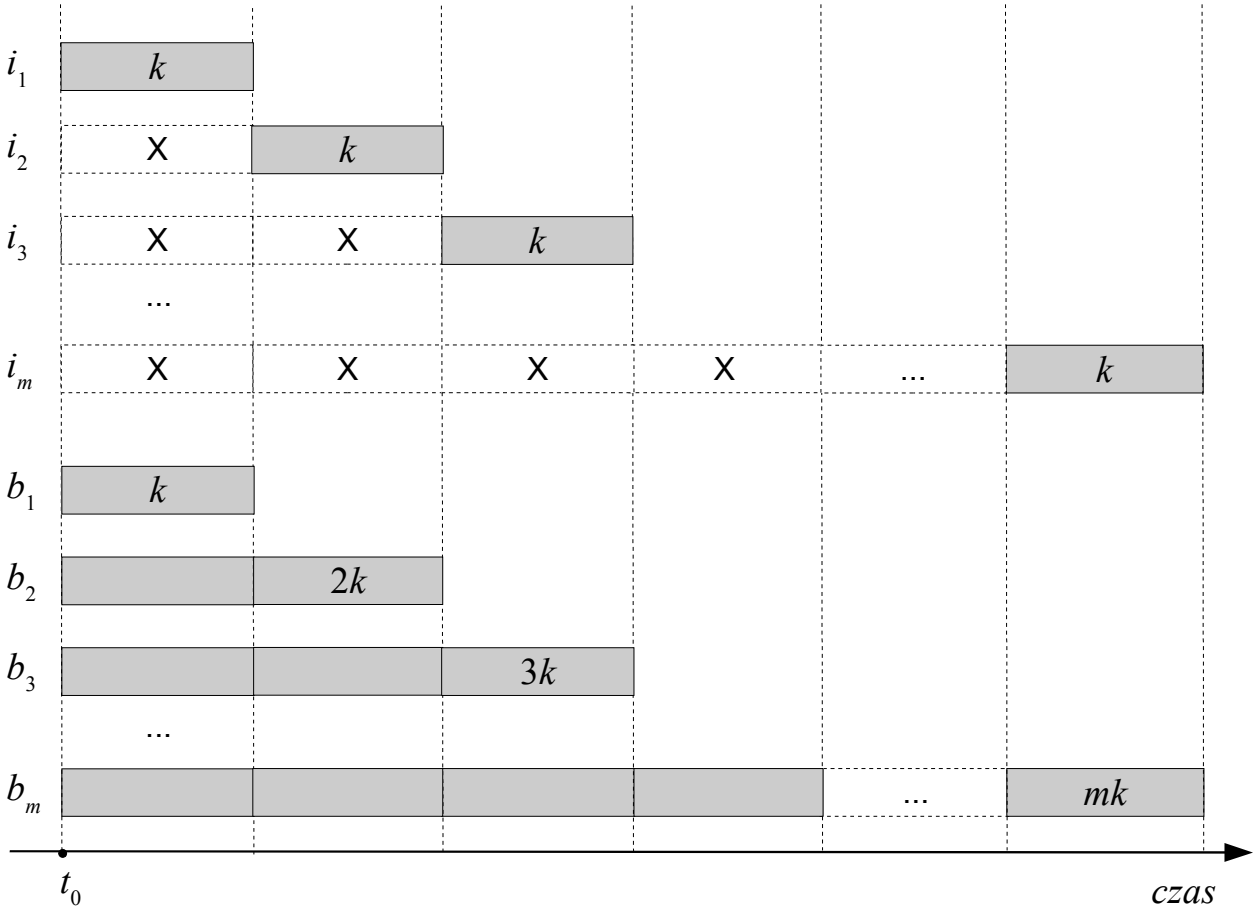
Eksperyment ma na celu sprawdzenie podstawowych własności metody APS i jej zgodności z przyjętymi postulatami (P2) oraz (P3). Przy jednorodnym rozkładzie w historii KB_H faktów spełniających reguły, w każdym przebiegu cyklu APS zbiór odkrywanych reguł R powinien być podobny. Oczekujemy, że będzie to optymalna sytuacja dla metody APS, gdyż w algorytmie aktualizacji bazy reguł RMAIN najczęściej powinno być wykonywane bezpośrednie porównywanie reguł z dotychczasowej bazy KB_R i nowego zbioru R, bez konieczności częstego korzystania z estymatorów oczekiwanego poparcia i pewności (dla brakujących reguł w jednym lub drugim zbiorze).

Strategia eksperymentu jest następująca.

1. Na początku baza reguł KB_R jest pusta, a historia KB_H jest zbiorem n faktów (rzędu kilkunastu – kilkudziesięciu tysięcy), wygenerowanych tak, iż fakty spełniające wygenerowane reguły są losowo rozłożone w całym zbiorze.
2. Aż do wyczerpania faktów z historii KB_H kolejno uruchamiane jest na niej m inkrementacyjnych przebiegów metody APS. W każdym przebiegu pobierana, przetwarzana i kasowana jest porcja k faktów (od kilkuset do jednego tysiąca), dając w wyniku zaktualizowaną bazę reguł KB_R (zob. Rys. 3.5.).
3. W każdym przebiegu jest mierzony i trwale zapisywany (w pliku) sumaryczny czas jego wykonania, a także czasy trwania poszczególnych etapów (w nawiasach podane są odpowiednie procedury cyklu APS):
   a) t_{sum} – całkowity czas trwania przebiegu;
   b) t_{select} – czas wybierania faktów z historii (procedura `Wybierz_Fakty`);
   c) $t_{convert}$ – czas przekształcania schematu historii (procedura `Przekształć_Schemat`);
   d) t_{fill} – czas wypełniania faktami przekształconego schematu historii (procedura `Wypełnij_Nowy_Schemat`);
   e) t_{elim} – czas eliminowania wartości N (procedura `Usuń_Wartości_N`);
   f) t_{mine} – czas odkrywania reguł związku przez algorytm eksploracji danych (procedura `Znajdź_Reguły`);
   g) t_{add} – czas aktualizacji bazy reguł KB_R (procedura `Aktualizuj_Bazę_Reguł`);
   h) t_{del} – czas usuwania faktów z historii KB_H (procedura `Usuń_Fakty`).
4. Po każdym i-tym przebiegu wynikowa baza reguł KB^I_R ($KB^{(i)}_H$) jest trwale zapisywana w oddzielnym pliku.
5. Wszystkie reguły z bazy KB_R są usuwane.
6. Wykonywanych jest kolejno m wsadowych przebiegów odkrywania reguł. W każdym przebiegu analizowane jest pierwszych k, $2k$, ..., n faktów z historii KB_H (zob. Rys. 3.5), której zawartość jest za każdym razem identyczna z zawartością bazy faktów, opisanej w punkcie 1.
7. W każdym przebiegu jest mierzony i trwale zapisywany (w pliku) sumaryczny czas jego wykonania, a także czasy trwania poszczególnych etapów – identycznie, jak w punkcie 3.
8. Po każdym przebiegu na pierwszych $i\,k$ faktach z historii KB_H, baza reguł KB^B_R ($KB^{(i)}_H$) jest trwale zapisywana w oddzielnym pliku i porównywana z odpowiednim zbiorem reguł KB^I_R ($KB^{(i)}_H$), na podstawie poniższych miar (wyniki porównania są zapisywane):
   a) $rule_{overlap}$ – współczynnik zgodności semantycznej;
   b) $sup_{overlap}$ – współczynnik zgodności poparcia;
   c) $con_{overlap}$ – współczynnik zgodności pewności;
   d) $time_{dev}$ – średnie odchylenie czasowe.

9. Wyniki pomiarów (czasu i miar porównania), przeprowadzonych dla poszczególnych przebiegów inkrementacyjnych i wsadowych, są porównywane ze sobą.

10. Dla poszczególnych przebiegów inkrementacyjnych i wsadowych obliczany jest procentowy udział czasu trwania poszczególnych etapów (zgodnie z listą w punkcie 3.).



Rys. 3.5. Porównanie porcji danych przetwarzanych w przebiegach inkrementacyjnych (i_1, i_2, ...,i_m) oraz w przebiegach wsadowych (b_1, b_2, ...,b_m). W trybie inkrementacyjnym przeanalizowane porcje faktó są usuwane (znak X), natomiast w trybie wsadowym pozostają one w historii.

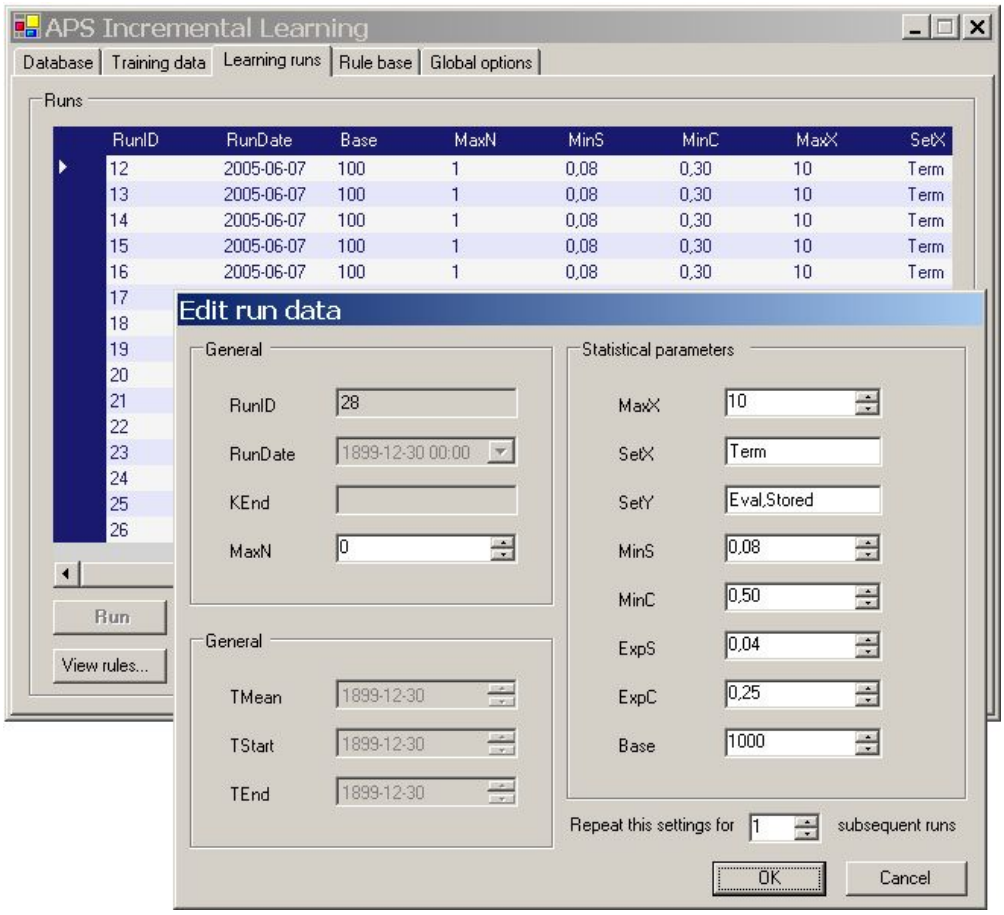
Badania przy niejednorodnym rozkładzie reguł

Przy jednorodnym rozkładzie w historii KB_H faktów spełniających reguły, w każdym przebiegu cyklu APS zbiór odkrywanych reguł R powinien się znacznie różnić od zbioru w przebiegu poprzednim, albo następnym. Oczekujemy, że będzie to dla metody APS gorsza sytuacja, niż przy rozkładzie jednorodnym, gdyż w algorytmie aktualizacji bazy reguł RMAIN często wykorzystywane będą estymatory oczekiwanego poparcia i pewności (dla brakujących reguł w jednym lub drugim zbiorze). Powinno to powodować pogorszenie jakości zbioru reguł, uzyskiwanych inkrementacyjnie, wyrażanej opisanymi wcześniej miarami.

Sam przebieg eksperymentu jest niemal identyczny, jak dla jednorodnego rozkładu reguł, z tą różnicą, że historia KB_H, choć o takim samym rozmiarze, jak poprzednio (rzędu kilkunastu – kilkudziesięciu tysięcy faktów), zawiera tym razem fakty nie rozłożone losowo, ale grupowane w gęste regiony z poparciem dla wybranego podzbioru reguł.

3.6.2 Opis środowiska testowego

Na potrzeby weryfikacji eksperymentalnej zostało przygotowane środowisko testowe. Opiera się ono na programie o nazwie *APS Incremental Learning*, zaimplementowanym w języku *MS Visual C++ .NET*, w narzędziu programistycznym *MS Visual Studio .NET* [Dud2005b]. Program stanowi implementację etapów cyklu APS wraz z funkcjami rejestrowania i zapisywania odpowiednich miar, zgodnie z przyjętą strategią eksperymentu).



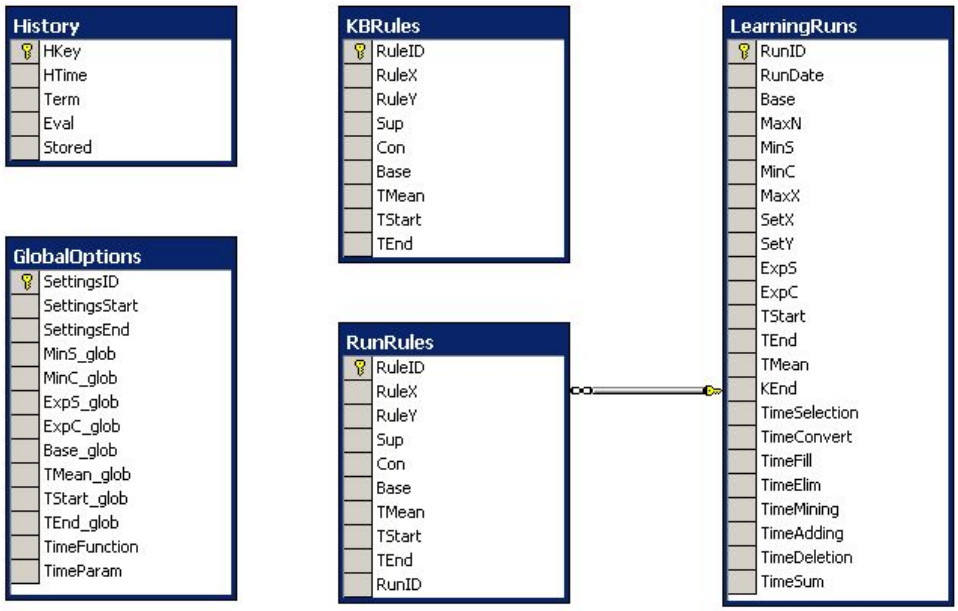
Rys. 3.6. Edycja danych przebiegu, które odpowiadają parametrom wektora v_c.

Aplikacja *APS Incremental Learning* umożliwia między innymi:

- łączenie z wybraną bazą danych na podanym serwerze bazodanowym (*MS SQL Server*);
- przeglądanie faktów historii (zob. Rys. 3.10.);
- wybór atrybutów historii, na podstawie których mają być odkrywane reguły związku; istnieje możliwość m.in. definiowania atrybutów jedno- i wielowartościowych oraz określania dopuszczalności występowania danego atrybutu w poprzedniku i następniku reguły (zob. Rys. 3.10.);
- zarządzanie przebiegami APS: dodawanie, edycja parametrów, uruchamianie, zapis danych przebiegu do pliku *.CSV – ustawień i wyników pomiarów (zob. Rys. 3.6);
- zarządzanie bazą reguł: przeglądanie, edycja, zapis bazy reguł do pliku *.CSV, porównanie bieżącej bazy z innym, zewnętrznym zbiorem reguł w oddzielnym pliku *.CSV (zob. Rys. 3.8.);

- zarządzanie parametrami globalnymi metody APS (zob. Rys. 3.8.).

Do odkrywania reguł związku wykorzystana została implementacja Goethalsa [Goe2003] algorytmu Apriori [Agr1994]. Z kolei struktury danych wykorzystywane w cyklu APS: historia, baza reguł, zbiór reguł z ostatniego przebiegu, dane przebiegu i parametry globalne są przechowywane w oddzielnych tabelach relacyjnej bazy danych, w systemie *MS SQL Server* 2000 *Enterprise Edition* wersja 8.00.760 (Service Pack 3) [Ran2003]. Struktura bazy danych jest przedstawiona na Rys. 3.7. Dostęp do bazy danych z aplikacji jest realizowany za pomocą obiektów biblioteki *ADO.NET* (np. *sqlConnection*, *sqlDataAdapter*), optymalizowanych dla produktu *MS SQL Server*.



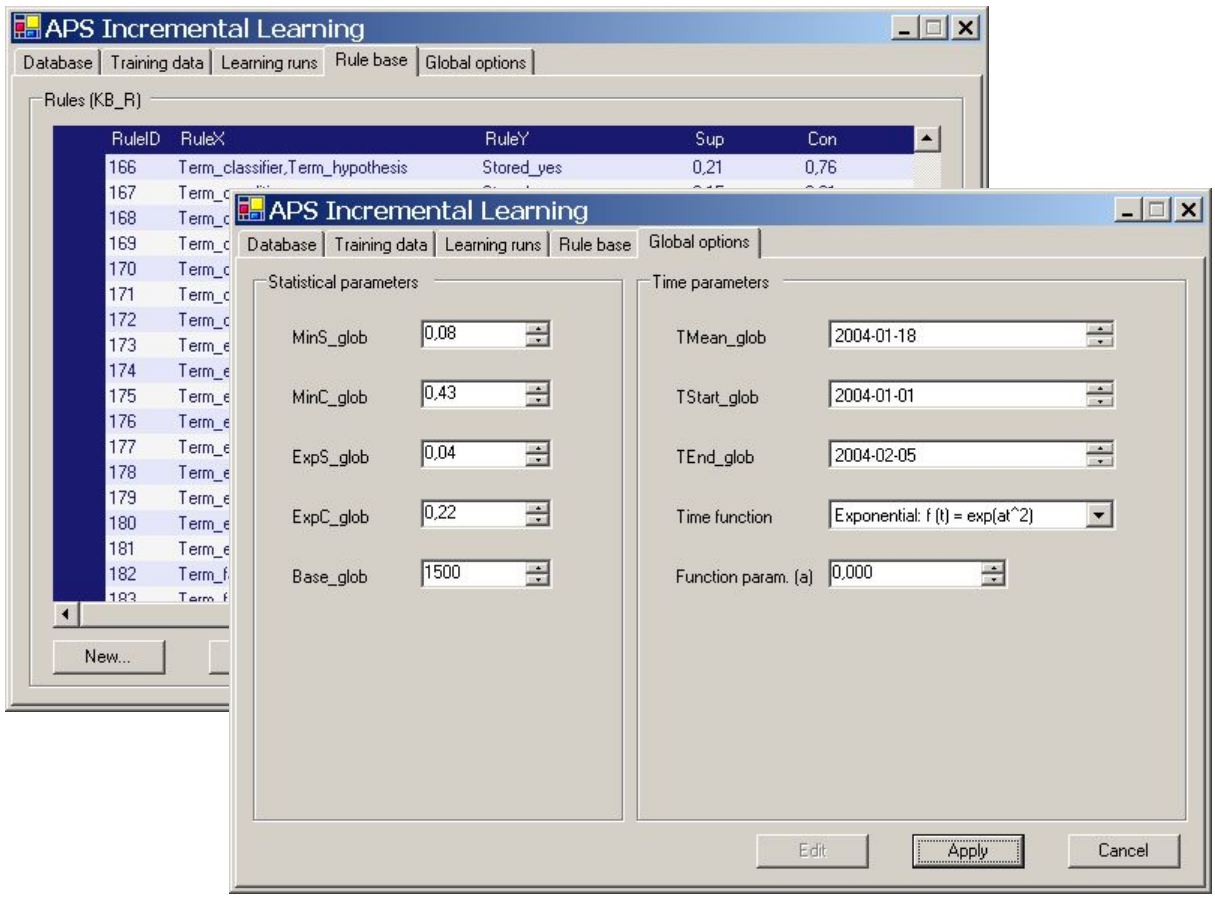
Rys. 3.7. Struktura bazy danych programu (wykorzystano zrzut ekranowy diagramu bazy z funkcji *Diagrams* konsoli *MS Enterprise Manager*).

Platforma sprzętowo-programowa, na której przeprowadzone zostały eksperymenty, ma poniższe parametry. Wskaźniki prędkości składników sprzętowych są mierzone w programie *SiSoft Sandra* wersja 2001.3.7.50 (www.sisoftware.co.uk/sandra).

- **CPU**: *AMD Duron* (model 3), rzeczywista częstotliwość taktowania 892,50 MHz; (prędkość: *Dhrystone* ALU 2466 MIPS, *Whetstone* FPU 1221 MFLOPS).
- **Płyta główna**: *Silicon Integrated Systems* (SiS) 730, M810.
- **Pamięć RAM**: 752 MB SDRAM (ALU/RAM 194 MS/s; FPU/RAM 218 MB/s).
- **Dysk twardy**: *Maxtor* 6E040L0, 40 GB, aktywna partycja 7,6 GB, system plików NTFS (czas dostępu: 11 ms; *Drive Index* 9311; odczyt buforowany 67 MB/s; odczyt sekwencyjny 12 MB/s; odczyt losowy 4 MB/s; zapis buforowany 54 MB/s; zapis sekwencyjny 14 MB/s; zapis losowy 5 MB/s).
- **System operacyjny**: *MS Windows 2000 Server* wersja 5.00.2195 + Service Pack 4.

3.6.3 Charakterystyka danych testowych

Do eksperymentów zostały wykorzystane dane syntetyczne, wygenerowane w programie *DataGen* (zob. Rys. 3.9.), który został zaimplementowany w języku *MS Visual C# .NET*, w narzędziu programistycznym *MS Visual Studio .NET* [Dud2005b]. Program korzysta z relacyjnej bazy danych w systemie *MS SQL Server* 2000 *Enterprise Edition* wersja 8.00.760 (Service Pack 3). Dostęp do bazy danych z aplikacji jest realizowany za pomocą obiektów biblioteki *ADO.NET* (np. *sqlConnection*, *sqlDataAdapter*), optymalizowanych dla produktu *MS SQL Server*.



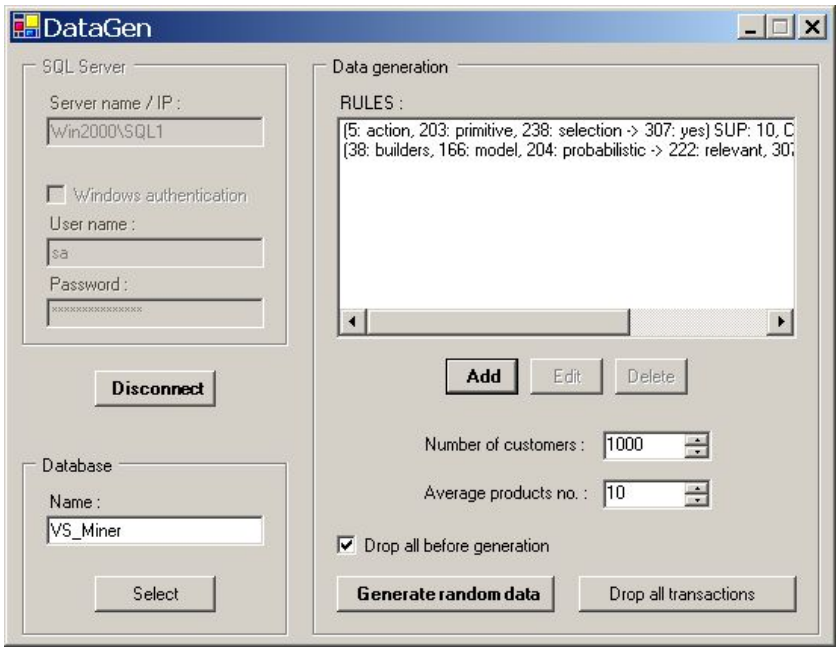
Rys. 3.8. Widok ekranu ustawień parametrów globalnych wektora v_g oraz bazy reguł KB_R (pod spodem).

Dane testowe zostały przygotowane dla dziedziny wspomagania wyszukiwania informacji w sieci WWW. Zakładamy następujący scenariusz ich powstawania. Podczas przeglądania stron sieci Internet, użytkownik korzysta z pomocy osobistego agenta rekomendującego, który rejestruje dane wyświetlanych dokumentów, ich jawną ocenę przez użytkownika i zdarzenia zapisu strony lokalnie na dysk, sugerujące zainteresowanie nią użytkownika. Agent analizuje witrynę, która jest oceniana lub zapisywana przez użytkownika, i zapamiętuje dla niej (oprócz oceny i faktu zapisu) *n* terminów indeksowych, które najczęściej na niej występują. Celem pozyskiwania reguł związku przez agenta jest zdobycie wiedzy, od jakich terminów na stronie zależy to, że jest ona zapisywana lub pozytywnie oceniana przez użytkownika. Reguły mogą być wykorzystywane przez agenta do przewidywania (predykcji) zainteresowania użytkownika kolejnymi stronami, na podstawie ich zawartości. Zastosowanie reguł związku do modelowania intencji i zainteresowań użytkownika sieci Internet (np. w systemach

rekomendujących) zostało przedstawione między innymi w pracach [ChL2002], [Fac2005], [WgS2004], [Yan2004].

Fakty, wchodzące w skład danych testowych, mają następujące atrybuty:

- *HKey* – unikalny klucz (odpowiednik atrybutu specjalnego *K* w modelu formalnym); dziedzina: typ całkowitoliczbowy *bigint*;
- *HTime* – czas zarejestrowania faktu (odpowiednik atrybutu specjalnego *T*); dziedzina: typ daty i czasu *datetime*;
- *Term* – kolekcja terminów indeksowych, opisujących stronę (atrybut wielowartościowy); dziedzina: zbiór 303 słów języka angielskiego (słownictwo ogólne i informatyczne); zakładamy binarny sposób indeksowania (system Boolowski): dana strona albo zawiera określony termin, albo go nie zawiera (bez wag, wartości rozmytych itp.);
- *Eval* – ocena strony przez użytkownika; dziedzina: ciąg znaków *varchar*, dopuszczalne wartości *relevant*, *irrelevant* (atrybut jednowartościowy);
- *Stored* – zdarzenie zapisu strony na dysk; dziedzina: ciąg znaków *varchar*, dopuszczalne wartości *yes*, *no* (atrybut jednowartościowy).



Rys. 3.9. Główne okno programu do generowania reguł związku *DataGen*.

Fakty z jednorodnym rozkładem reguł

W programie *DataGen* zostało wygenerowanych 20 000 faktów, z których każdy zawiera średnio 9 różnych wartości atrybutów *Term*, *Eval* i *Stored*. Fakty te odzwierciedlają losową (a więc w przybliżeniu jednorodną) dystrybucję poniższych, predefiniowanych reguł związku.

r_1: $action \wedge primitive \wedge selection \Rightarrow stored\_yes$ $sup(r_1) = 0{,}10$; $con(r_1) = 0{,}50$

r_2: $builders \wedge model \wedge probabilistic \Rightarrow relevant \wedge stored\_yes$ $sup(r_2) = 0{,}10$; $con(r_2) = 0{,}70$

r_3: $agent \wedge experience \wedge goals \Rightarrow relevant$ $sup(r_3) = 0{,}15$; $con(r_3) = 0{,}60$

r_4:	*fact* ∧ *learn* ∧ *rule*	⇒	*relevant* ∧ *stored_yes*	$sup(r_4) = 0{,}10$;	$con(r_4) = 0{,}50$
r_5:	*classifies* ∧ *evaluating* ∧ *hypothesis*	⇒	*relevant*	$sup(r_5) = 0{,}20$;	$con(r_5) = 0{,}80$
r_6:	*exploration* ∧ *levels* ∧ *mapping*	⇒	*stored_yes*	$sup(r_6) = 0{,}15$;	$con(r_6) = 0{,}70$
r_7:	*change* ∧ *condition* ∧ *random*	⇒	*relevant* ∧ *stored_yes*	$sup(r_7) = 0{,}10$;	$con(r_7) = 0{,}60$
r_8:	*decide* ∧ *multiple* ∧ *process*	⇒	*relevant*	$sup(r_8) = 0{,}20$;	$con(r_8) = 0{,}70$
r_9:	*algorithm* ∧ *class* ∧ *external*	⇒	*stored_yes*	$sup(r_9) = 0{,}10$;	$con(r_9) = 0{,}50$
r_{10}:	*idea* ∧ *limitations* ∧ *sample*	⇒	*relevant* ∧ *stored_yes*	$sup(r_{10}) = 0{,}15$;	$con(r_{10}) = 0{,}80$

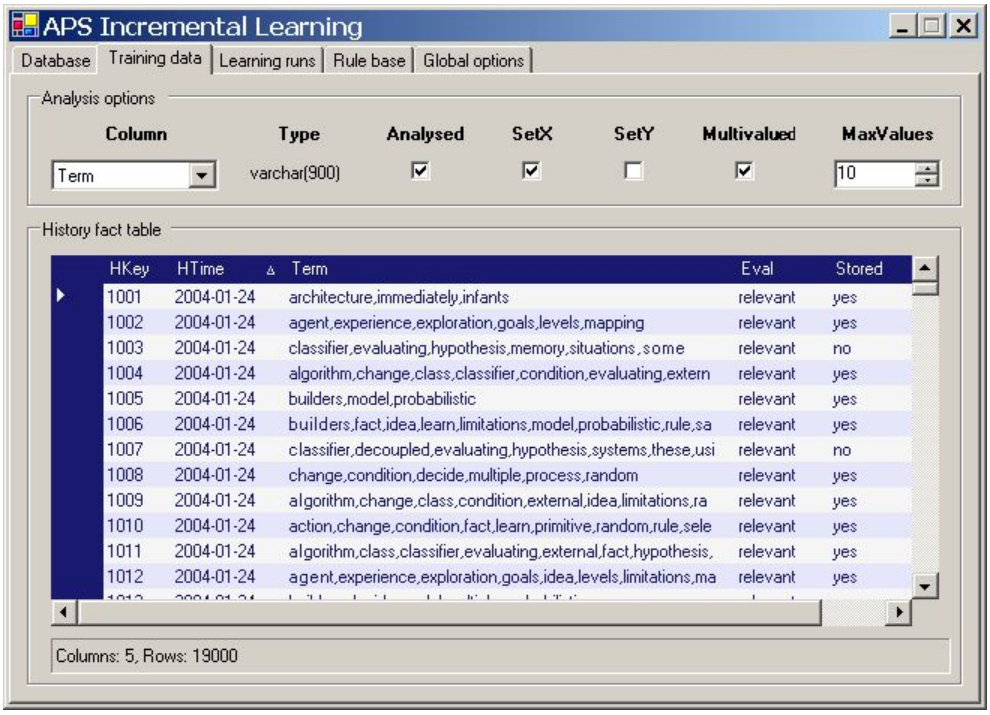
Wszystkie fakty są równomiernie rozłożone w czasie od „2004-01-01 00:09:45.000" do „2005-04-10 23:25:23.000". Można je zatem traktować jako hipotetyczny zapis aktywności użytkownika Internetu, który przez 465 dni (około 1 rok i 3 miesiące) przejrzał 20 000 stron WWW (czyli średnio 43 strony dziennie).

Posługując się tradycyjnym językiem, stosowanym do opisu reguł związku [Agr1993], [Agr1994], [Has2001], można odnieść te dane do 20 000 koszyków (ang. *shopping carts*), z których każdy zawiera średnio 9 produktów (ang. *items*). Średni rozmiar częstego zbioru atrybutów (ang. *frequent itemset*, *large itemset*) wynosi 4. W literaturze spotykane są poniższe oznaczenia parametrów zbiorów testowych [Agr1994]:

- $|D|$ – liczba transakcji (faktów);
- $|T|$ – średni rozmiar transakcji (faktu);
- $|I|$ – średni rozmiar maksymalnych, potencjalnie częstych zbiorów atrybutów;
- $|L|$ – liczba maksymalnych, potencjalnie częstych zbiorów atrybutów;
- N – liczba atrybutów (binarnych).

Dla wygenerowanego zbioru faktów określone są następujące wartości parametrów: $|D| = 20\,000$; $|T| = 9$; $|I| = 4$; $N = 307$. Stąd też, przyjmując konwencję stosowaną w wielu pracach z dziedziny eksploracji danych, można przypisać temu zbiorowi zakodowaną nazwę: **T9.I4.D20K**. Przykładowe fakty z tego zbioru zostały przedstawione na Rys. 3.10.

Wygenerowane dane nie zawierają wartości nieznanych *N*. Są dwa powody podjęcia takiej decyzji: (i) w opisywanej dziedzinie zastosowania wartości nieznane nie mają większego sensu (to znaczy mogłyby one występować jedynie dla atrybutu predykcyjnego *Eval*, ponieważ kolekcje terminów indeksowych w polu *Term* oraz opisy zdarzenia zapisu strony *Stored* są zawsze jednoznacznie określone); (ii) wykładnicza złożoność obliczeniowa algorytmu ENV do eliminacji wartości *N* (zgodnie z przytoczonym wcześniej oszacowaniem formalnym), podczas eksperymentów objawia się bardzo dużą zmiennością liczby przetwarzanych faktów w porcji; efekt ten może bardzo komplikować badanie głównych własności metody APS (w konsekwencji może to utrudniać weryfikację podstawowych postulatów). Stąd też podjęta została decyzja o przeniesieniu testów działania metody APS dla danych z wartościami *N* do propozycji dalszych badań.



Rys. 3.10. Fragment zbioru faktów T9.I4.D20K, zaimportowanego jako historia do programu *APS Incremental Learning*.

W analogiczny sposób, jak wyżej, przygotowany został drugi zestaw faktów z jednorodnym rozkładem reguł, o wartościach parametrów: $|D| = 20\,000$; $|T| = 10$; $|I| = 5$; $N = 307$. Tak więc zestawowi temu nadana została zakodowana nazwa: **T10.I5.D20K**, zgodnie z opisaną wcześniej konwencją. Fakty odzwierciedlają losową dystrybucję poniższych reguł.

r_{11}: $abstract \wedge complete \wedge include \wedge results \Rightarrow stored\_yes$ $sup(r_{11}) = 0{,}15$; $con(r_{11}) = 0{,}84$

r_{12}: $active \wedge map \wedge use \Rightarrow relevant \wedge stored\_yes$ $sup(r_{12}) = 0{,}10$; $con(r_{12}) = 0{,}53$

r_{13}: $behavior \wedge necessary \wedge piagets \wedge theories \Rightarrow relevant$ $sup(r_{13}) = 0{,}20$; $con(r_{13}) = 0{,}68$

r_{14}: $architecture \wedge development \wedge schema \Rightarrow relevant \wedge stored\_yes$ $sup(r_{14}) = 0{,}10$; $con(r_{14}) = 0{,}63$

r_{15}: $category \wedge example \wedge matching \wedge possible \Rightarrow relevant$ $sup(r_{15}) = 0{,}12$; $con(r_{15}) = 0{,}72$

r_{16}: $defined \wedge method \wedge optimize \wedge system \Rightarrow stored\_yes$ $sup(r_{16}) = 0{,}18$; $con(r_{16}) = 0{,}80$

r_{17}: $adaptive \wedge autonomous \wedge learning \Rightarrow relevant \wedge stored\_yes$ $sup(r_{17}) = 0{,}11$; $con(r_{17}) = 0{,}51$

r_{18}: $component \wedge mechanism \wedge parallel \wedge share \Rightarrow relevant$ $sup(r_{18}) = 0{,}16$; $con(r_{18}) = 0{,}64$

r_{19}: $increase \wedge space \wedge symbol \wedge time \Rightarrow stored\_yes$ $sup(r_{19}) = 0{,}13$; $con(r_{19}) = 0{,}70$

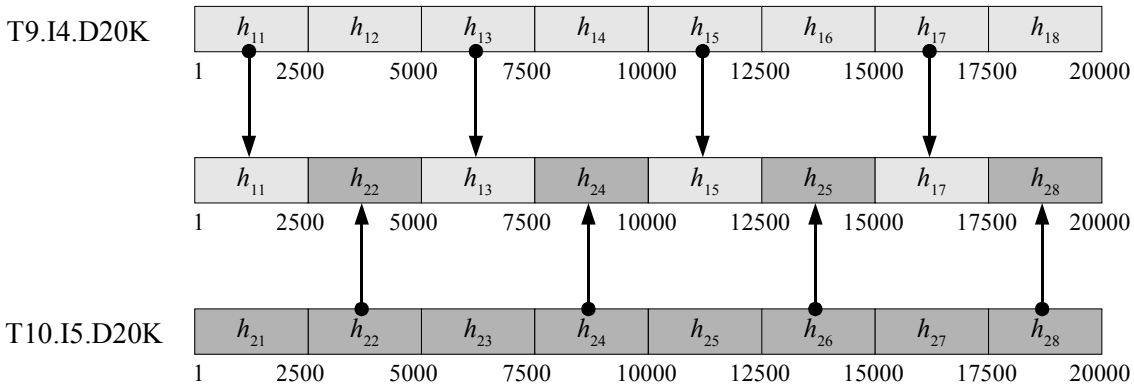
r_{20}: $data \wedge generalization \wedge suggested \Rightarrow relevant \wedge stored\_yes$ $sup(r_{20}) = 0{,}11$; $con(r_{20}) = 0{,}59$

Wszystkie fakty są równomiernie rozłożone w czasie od „2004-01-01 00:27:45.000” do „2005-04-10 23:32:57.000”. Jest to zatem, podobnie jak w przypadku zbioru T9.I4.D20K, hipotetyczny zapis aktywności użytkownika Internetu, który przez 465 dni (około 1 rok i 3 miesiące) przejrzał 20 000 stron WWW (średnio 43 strony dziennie). Zauważmy jednak, że zbiory reguł, będących podstawą wygenerowania obu zestawów (T9.I4.D20K i T10.I5.D20K), są całkowicie różne. Zwróćmy także uwagę, że w zbiorze T10.I5.D20K średni rozmiar częstego zbioru atrybutów (ang. *frequent itemset*) jest o jeden większy, niż w zbiorze T9.I4.D20K.

Podczas przetwarzania zbioru T10.I5.D20K przez algorytmy typu *Apriori*, powinna być zatem zwracana większa liczba reguł, niż dla drugiego zbioru (reguły podstawowe wraz z regułami pochodnymi – zawierającymi się semantycznie w regułach podstawowych).

Fakty z niejednorodnym rozkładem reguł

Jako dane źródłowe dla eksperymentu przy niejednorodnym rozkładzie reguł, przygotowany został zbiór T9.I4.D20K + T10.I5.D20K, który powstał w wyniku połączenia omówionych wcześniej zestawów jednorodnych. Każdy z tych zestawów został podzielony na 8 porcji po 2500 faktów. Następnie porcje z obu zestawów zostały ze sobą połączone naprzemiennie, dając wyniku zbiór, który przypomina wynik *skrzyżowania* (ang. *cross-over*) zbiorów T9.I4.D20K i T10.I5.D20K (zob. Rys. 3.11.). Wartości pola *HTime* (czas zarejestrowania faktu) w całym połączonym zbiorze zostały przepisane ze zbioru T10.I5.D20K, aby zachować jednolity ciąg chronologiczny wszystkich faktów. W ten sposób uzyskany został zestaw danych testowych, zawierający naprzemiennie rozmieszczone podzbiory faktów, które zostały wygenerowane na podstawie całkowicie rozłącznych zbiorów reguł.



Rys. 3.11. Sposób łączenia krzyżowego jednorodnych zbiorów testowych T9.I4.D20K i T10.I5.D20K. W zestawie wynikowym przeplatają się ze sobą regiony (o rozmiarze 2500 faktów), w których fakty wspierają całkowicie różne reguły.

3.6.4 Wyniki eksperymentów

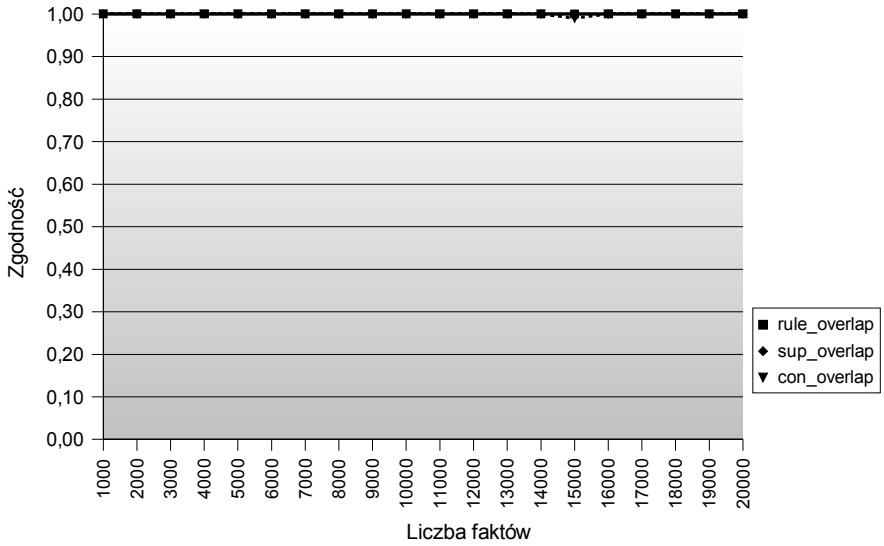
Uzyskane wyniki liczbowe eksperymentów zostały zestawione w Dodatku B. Poniżej przedstawione jest ich omówienie wraz z opracowaniem w formie graficznej.

Jednorodny rozkład reguł w analizowanych faktach

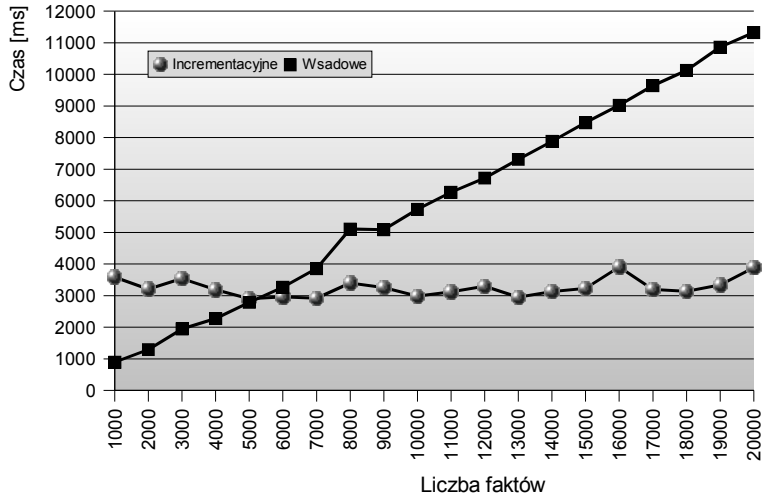
Testy metody APS przy jednorodnym rozkładzie reguł w historii zostały przeprowadzone zgodnie z opisaną wcześniej strategią na zbiorze T9.I4.D20K. W pełnym zestawie testowym (to znaczy w 20000 faktów) znajdowane jest 210 reguł związku przy zadanych parametrach (zob. Dodatek B), między innymi przy progu minimalnego poparcia σ_c = 0,08 i progu minimalnej pewności γ_c = 0,50. Rozmiar porcji przetwarzanej w pojedynczym przebiegu inkrementacyjnym został ustalony na 1000 faktów.

Wyniki porównania jakościowego z wykorzystaniem miar $rule_{overlap}$, $sup_{overlap}$ i $con_{overlap}$, wskazują, iż zbiory reguł KB^I_R i KB^B_R (otrzymane odpowiednio inkrementacyjnie i wsadowo), są niemal identyczne (zob. Rys. 3.12.), gdyż dla wszystkich przebiegów omawiane miary

przyjmowały wartość rzędu 99 – 100%. Jest to dowód poprawności podstawowych wzorów proporcji częstościowych, które są stosowane w algorytmie RMAIN do naliczania zaktualizowanych wartości poparcia i pewności. Również odchylenie czasowe $time_{dev}$, które wynosiło średnio zaledwie kilkanaście minut (0,01 doby, a więc niewspółmiernie mało w stosunku do piętnastomiesięcznej rozpiętości czasowej zbioru faktów), wskazuje na prawidłowość odpowiednich wzorów do aktualizacji czasu w algorytmie RMAIN.



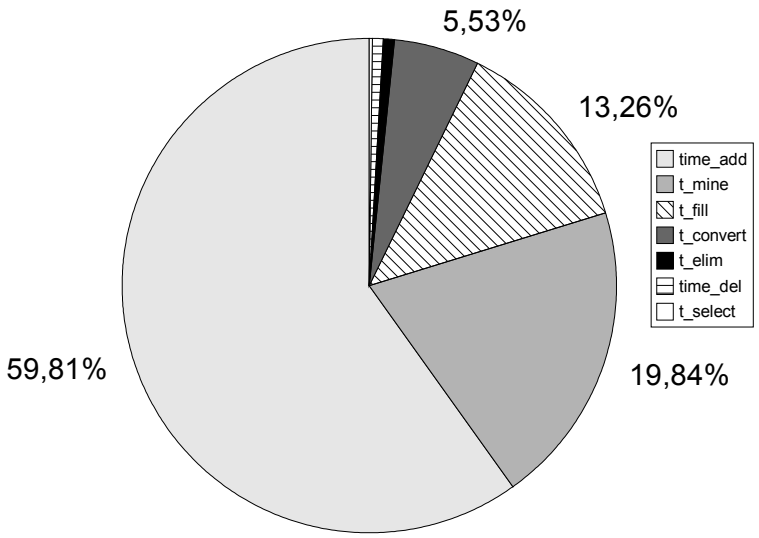
Rys. 3.12. Zgodność zbiorów reguł uzyskanych inkrementacyjnie i wsadowo dla jednorodnego zestawu danych testowych T9.I4.D20K.



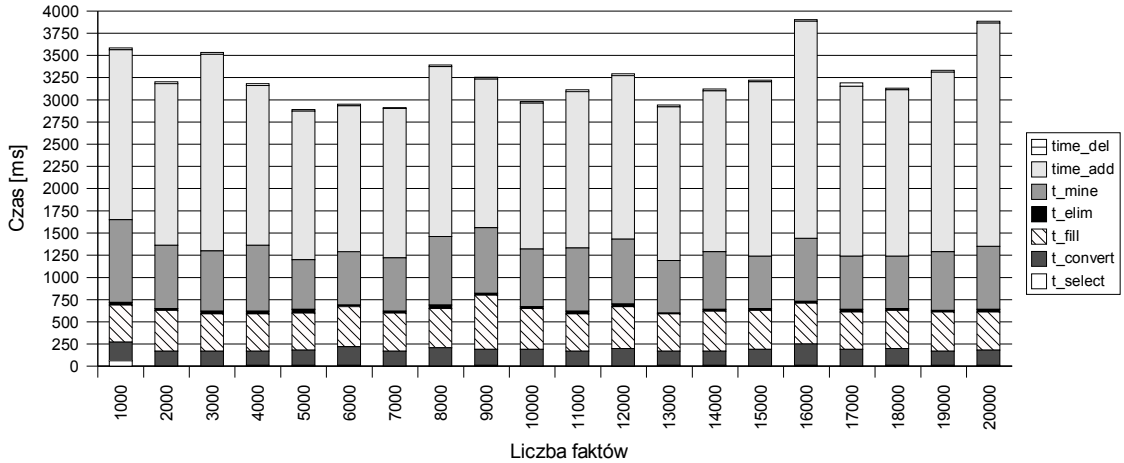
Rys. 3.13. Porównanie długości trwania przebiegów inkrementacyjnych APS i czasu samego odkrywania reguł w trybie wsadowym.

W celu oceny wydajności proponowanej metody, porównany został całkowity czas przebiegów inkrementacyjnych z wyodrębnionym (izolowanym) czasem samego odkrywania reguł przez algorytm Apriori w przebiegach wsadowych. Zwróćmy uwagę, że jest to porównanie maksymalnie niekorzystne dla metody APS, ponieważ zakłada ono, że w przebiegu wsadowym nie ma żadnych dodatkowych operacji poza samym wykonaniem algorytmu eksploracji danych (czyli nie ma dodatkowych odczytów, zapisów, przekształceń danych). Uzyskane

wyniki wskazują, że wsadowy algorytm odkrywania reguł związku jest szybszy do momentu, gdy rozmiar przetwarzanej porcji osiąga pewien próg ψ, mieszczący się w przedziale 5000 – 6000 faktów (zob. Rys. 3.13.). Później, dla większych porcji wyższą wydajność ma metoda APS, ponieważ czas trwania przebiegów inkrementacyjnych pozostaje w przybliżeniu na tym samym poziomie, podczas gdy długość trwania przebiegów wsadowych rośnie liniowo.



Rys. 3.14. Procentowy udział średniej długości trwania poszczególnych etapów metody APS w przebiegach inkrementacyjnych na zbiorze T9.I4.D20K. Przeważa etap aktualizowania bazy reguł (algorytm RMAIN), na kolejnych miejscach plasują się: odkrywanie reguł związku, wypełnianie i przekształcanie schematu historii. Pozostałe etapy mają znaczenie marginalne.



Rys. 3.15. Udział procentowy długości trwania etapów metody APS w poszczególnych przebiegach inkrementacyjnych na zbiorze T9.I4.D20K.

Przeanalizowane zostały także wyniki pomiarów długości trwania etapów metody APS. Procentowy udział czasu wykonywania poszczególnych kroków został zobrazowany na Rys. 3.14. i 3.15. Widać wyraźnie, że najbardziej kosztowym krokiem jest aktualizacja bazy reguł, przeprowadzana przez algorytm RMAIN (przypomnijmy, o złożoności rzędu $O(nm)$, gdzie n jest liczbą reguł w bazie KB_R, a m jest liczbą reguł nowo znalezionych w ostatnim przebiegu). Jednakże pomimo tego swoistego *narzutu* dodatkowych kroków w przebiegach

inkrementacyjnych w porównaniu ze zwykłym przetwarzaniem wsadowym, metoda APS uzyskuje przewagę w wydajności dla pewnego, wystarczająco dużego zbioru analizowanych faktów, jak to zostało omówione wyżej. Zwraca uwagę stosunkowo niewielki koszt czasowy każdorazowego przekształcania i wypełniania schematu historii (realizowanego odpowiednio przez algorytmy: HTRANS i HFILL), które, przypomnijmy, zwiększa elastyczność metody, ponieważ zbiory atrybutów opisujących fakty mogą być różne w poszczególnych przebiegach.

Niejednorodny rozkład reguł w analizowanych faktach

Testy metody APS przy niejednorodnym rozkładzie reguł w historii zostały przeprowadzone w dwóch oddzielnych seriach na połączonym zbiorze T9.I4.D20K + T10.I5.D20K.

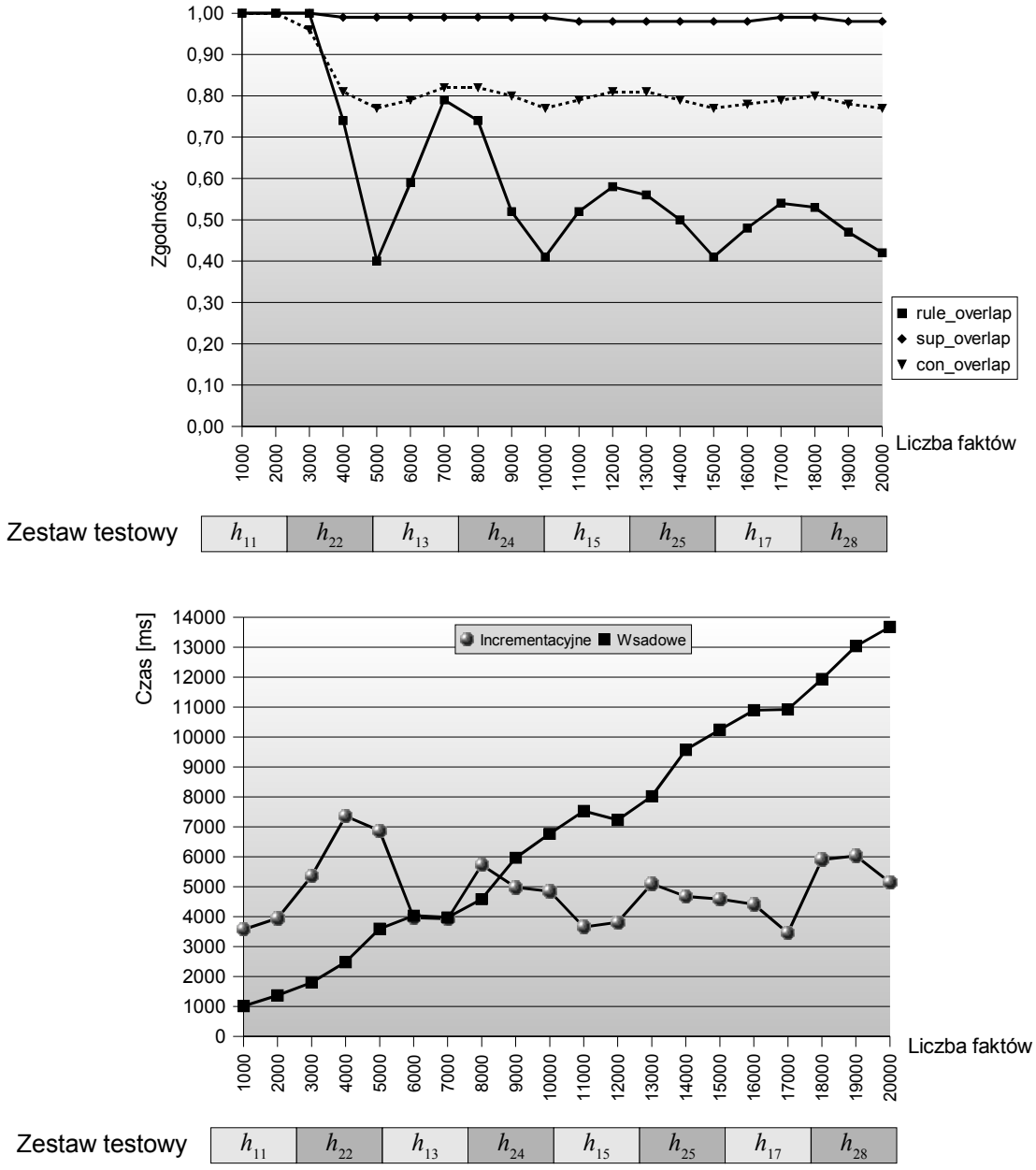
Seria I

W serii pierwszej zastosowane zostały identyczne parametry, jak w eksperymencie dla zestawu jednorodnego (m.in. minimalne poparcie $\sigma_c = 0{,}08$ i minimalna pewność $\gamma_c = 0{,}50$), przy których w pełnym zestawie testowym (to znaczy w 20000 faktów) znajdowane jest 427 reguł związku. Rozmiar porcji przetwarzanej w pojedynczym przebiegu inkrementacyjnym został ustalony tak, jak poprzednio na 1000 faktów.

Zestawienie miar $rule_{overlap}$, $sup_{overlap}$ i $con_{overlap}$ wskazuje, że na niejednorodnym zestawie faktów jakość zbioru reguł uzyskiwanych inkrementacyjnie jest znacząco niższa w stosunku od przetwarzania wsadowego, przy zachowaniu podobnej charakterystyki wydajnościowej, jak poprzednio (zob. Rys. 3.16.). Mimo iż wartości miary zgodności poparcia osiągały blisko 100%, a zgodności pewności – średnio ponad 80%, niskie wyniki najważniejszej miary zgodności semantycznej zbiorów KB^I_R i KB^B_R (tzn. $rule_{overlap}$), rzędu 50–60% są niezadowalające. Uzyskane wyniki można wyjaśnić następująco. Podczas trzeciego przebiegu inkrementacyjnego APS (po przetworzeniu pierwszych 2500 faktów) następuje gwałtowna zmiana zbioru odkrywanych reguł (to znaczy z zestawu T9.I4.D20K na zestaw T10.I5.D20K). Zgodnie z wcześniejszymi przewidywaniami, jest to bardzo niekorzystna sytuacja dla algorytmu RMAIN, ponieważ we wzorach aktualizacji poparcia i pewności reguł, zamiast wartości dokładnych, musi on stosować estymatory oczekiwanego poparcia i pewności. Ponieważ zaś wartości tych estymatorów, ustalone na ½ odpowiednich wartości progów σ i γ, są stosunkowo niskie, po zaktualizowaniu wartości poparcia i pewności nowych reguł (ze zbioru T10.I5.D20K) w większości nie osiągają bieżących progów (zwłaszcza minimalnego poparcia), które są stosunkowo wysokie. A zatem spora część nowych reguł w ogóle nie jest dodawana do bazy KB^I_R, co powoduje dużą rozbieżność tego zbioru z jego odpowiednikiem KB^B_R, otrzymanym wsadowo. Sytuacja ta powtarza się wielokrotnie (po przetworzeniu 2500, 7500, 12500 i 17500 faktów), sprawiając, że reguły odkryte w pierwszych dwóch przebiegach (pochodzące ze zbioru T9.I4.D20K) już do końca *wygrywają* z równie częstymi regułami ze zbioru T10.I5.D20K. To tłumaczy niskie wartości współczynnika $rule_{overlap}$.

W metodzie APS istnieją przynajmniej dwie metody polepszenia jakości zbioru reguł, otrzymywanych na mocno niejednorodnym zestawie faktów. Po pierwsze, można zastosować odpowiednią funkcję wpływu czasowego f_T tak, aby najnowsze fakty mocniej wpływały na zawartość bazy reguł, aniżeli fakty stare. O tym jednak, czy funkcja f_T może być użyta i jaki ma być jej kształt oraz parametry, decydują własności konkretnej dziedziny zastosowania. Tymczasem druga metoda poprawy jakości nawiązuje do ogólnych metod konfiguracji algorytmów eksploracji danych i polega na odpowiednim obniżeniu progów minimalnego poparcia

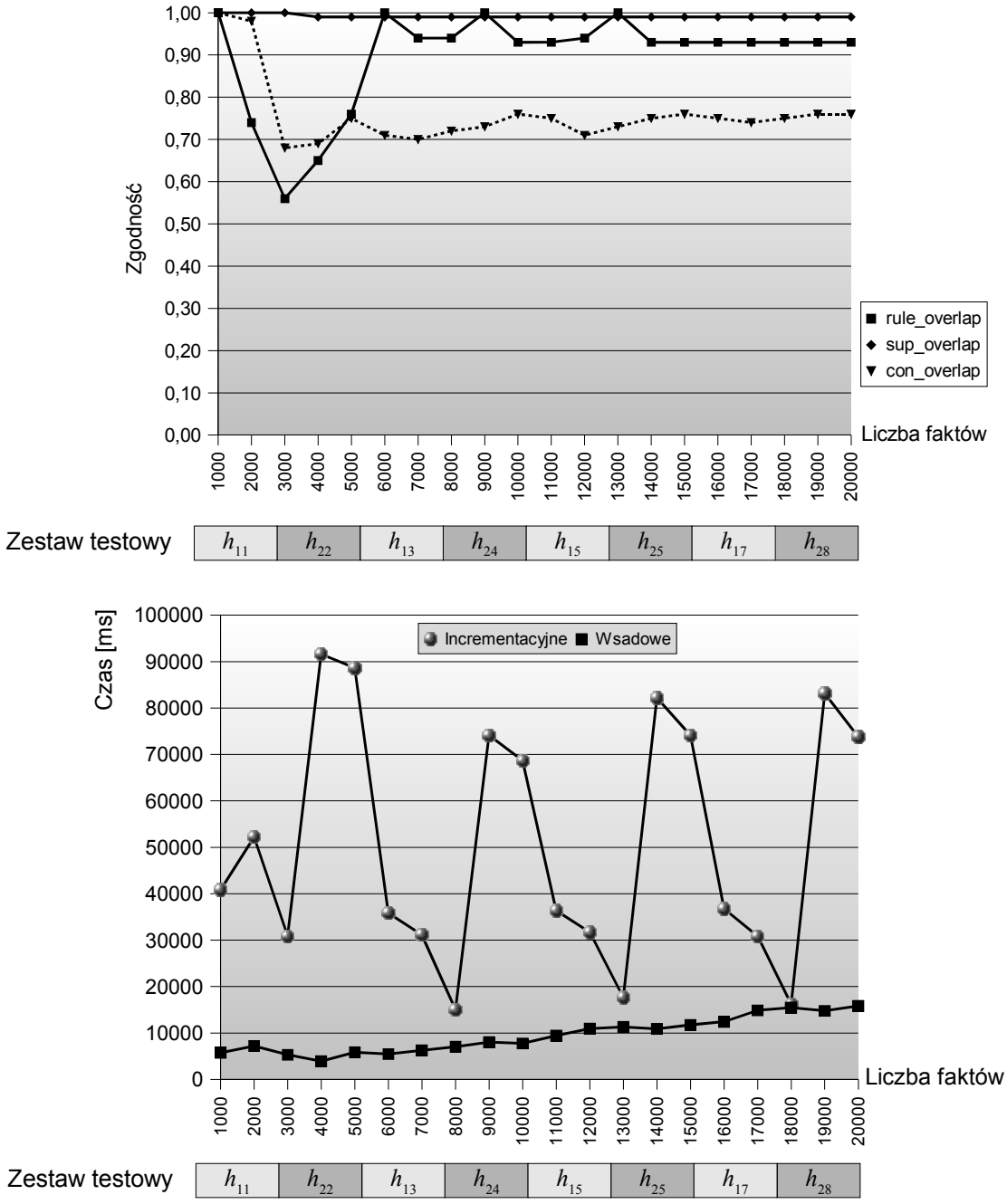
i pewności tak, aby w tym samym zbiorze testowym znajdowanych mogło być więcej reguł. To rozwiązanie zostało zbadane w drugiej serii eksperymentów na niejednorodnym zestawie.



Rys. 3.16. Zgodność zbiorów reguł i czas przetwarzania dla metody APS i przebiegów wsadowych (Seria I). Pod wykresami umieszczone są schematy łączonego zbioru testowego T9.I4.D20K + T10.I5.D20K.

Seria II

W drugiej serii obniżone zostały progi minimalnego poparcia do $\sigma_c = 0{,}04$ i minimalnej pewności do $\gamma_c = 0{,}30$. Odpowiednio zmniejszone zostały także estymatory oczekiwanego poparcia i pewności, tak jak poprzednio ustalone na ½ wartości progów σ_c oraz γ_c. Inne ustawienia pozostały niezmienione. Z pozoru niewielka modyfikacja parametrów ma znaczący wpływ na liczbę znajdowanych reguł, których liczba wzrosła z 427 do 564 (w pełnym zestawie testowym), przekraczając w pierwszym przebiegu liczbę 3000.



Rys. 3.17. Zgodność zbiorów reguł i czas przetwarzania dla metody APS i przebiegów wsadowych (Seria II). Pod wykresami umieszczone są schematy łączonego zbioru testowego T9.I4.D20K + T10.I5.D20K.

Zestawienie miar $rule_{overlap}$, $sup_{overlap}$ i $con_{overlap}$ dla przebiegów ze zmienionymi parametrami wskazuje na znaczącą poprawę jakości zbioru reguł uzyskiwanych inkrementacyjnie (zob. Rys. 3.17). Zgodność semantyczna $rule_{overlap}$ wzrosła średnio do 90%, natomiast miary zgodności poparcia i pewności utrzymały się mniej więcej na dotychczasowym, dobrym poziomie, odpowiednio 99% i 76%. Miara $con_{overlap}$ uzyskała średnio niższe wartości od $sup_{overlap}$, ponieważ wzory do inkrementacyjnego obliczania pewności reguł (w algorytmie RMAIN) są wzorami przybliżonymi (z powodu nieznanej wartości poparcia poprzednika reguły w jednym ze zbiorów faktów).

Odmiennie do poprzednich serii eksperymentów prezentują się wyniki wydajnościowe (zob. Rys. 3.17.), tym razem bowiem porównanie długości trwania przebiegów inkrementacyjnych i wsadowych wychodzi wyraźnie na niekorzyść metody APS (przynajmniej dla badanego zbioru 20000 niejednorodnych faktów). Poza tym sumaryczny czas trwania przebiegów APS wykazuje bardzo silne wahania (charakterystyczny efekt *piły* na wykresie), które wynikają z różnej liczby reguł znajdowanych naprzemiennie w faktach pochodzących z zestawu T9.I4.D20K oraz T10.I5.D20K. Jest to bezpośrednią konsekwencją większego o jeden, średniego rozmiaru zbioru częstego w zbiorze T10.I5.D20K, ponieważ złożoność obliczeniowa algorytmu RMAIN zależy w przybliżeniu kwadratowo, to znaczy $O(nm)$, od liczby reguł, które są w nim porównywane. Można jedynie przypuszczać, że oscylacje te będą słabły wraz z rosnącą liczba przetwarzanych faktów, co przy liniowym wzroście długości trwania przebiegów wsadowych może doprowadzić w końcu do pożądanego przecięcia obu wykresów, obserwowanego w poprzednich seriach testów.

Na podstawie wyników eksperymentów widać, że przy nieoptymalnym, to znaczy niejednorodnym rozkładzie reguł w analizowanym zestawie testowym, należy poszukiwać konfiguracji parametrów metody APS, która jest optymalna z punktu widzenia konkretnej dziedziny zastosowania. Szukane optimum z założenia stanowi *złoty środek* między jakością zbioru reguł otrzymywanych inkrementacyjnie, a długością procesu ich znajdowania w poszczególnych przebiegach analizy.

3.6.5 Omówienie wyników i wnioski

Badania eksperymentalne miały na celu wykazanie, że zaproponowana metoda APS spełnia postulat (P2), przedstawiony w planie eksperymentu, zgodnie z celem pracy. Eksperymenty miały także udzielić odpowiedzi na pytanie, czy i w jakich warunkach spełniony jest dodatkowy postulat (P3).

Rozważając postulat (P2), w świetle uzyskanych wyników można stwierdzić, że zbiór reguł, uzyskanych w sposób inkrementacyjny, jest porównywalny ze zbiorem reguł otrzymanych wsadowo, zgodnie z przyjętymi miarami $rule_{overlap}$, $sup_{overlap}$ i $con_{overlap}$. Podobieństwo obu zbiorów jest największe, jeśli zestaw danych testowych jest jednorodny pod względem rozkładu reguł w faktach, zmniejsza się ono zaś wówczas, gdy dane testowe są niejednorodne.

Dodatkowy postulat (P3) jest spełniony, a więc czas przetwarzania faktów przy inkrementacyjnym odkrywaniu reguł jest mniejszy od długości analizy tych samych zbiorów faktów metodą wsadową, jeżeli sumaryczna liczba przetworzonych faktów przekracza pewien próg ψ, którego wartość zależy od własności danych i ustawień parametrów metody APS.

Można stąd wywnioskować, że metoda APS potencjalnie przynosi oczekiwane efekty wówczas, gdy dane wejściowe wykazują wyraźną powtarzalność. Jeśli natomiast każda kolejna porcja faktów zawiera reguły mocno różniące się od poprzednich, przydatność metody APS jest zmniejszona. Zwróćmy jednak uwagę, że w tym drugim przypadku ogólnie obniżona jest przydatność reguł związku jako narzędzia analizy danych, gdyż przy braku powtarzalności wzorców w badanej dziedzinie, wartość predykcyjna znajdowanych reguł jest niewielka.

3.7. Porównanie metody APS z innymi pracami

Zaproponowana w niniejszej pracy metoda APS może być odniesiona do innych prac na dwóch płaszczyznach, które obejmują: (i) inkrementacyjne metody znajdowania reguł związku oraz (ii) zastosowanie reguł związku do pozyskiwania wiedzy w systemie agenckim.

3.7.1 Inne metody inkrementacyjne znajdowania reguł związku

Większość metod inkrementacyjnych, omawianych w Rozdziale 1., to metody dokładne, które zwracają precyzyjny zbiór reguł (w pełni porównywalny z metodą wsadową). Zakładają one przechowywanie całej aktualizowanej bazy transakcji i wymagają powrotów do przeanalizowanych faktów, co gwarantuje wysoką dokładność zbioru reguł pozyskiwanych inkrementacyjnie. Oczywiście, w metodach tych dąży się do minimalizacji powtórnych przebiegów. Do nielicznych rozwiązań całkowicie eliminujących takie powroty należy algorytm grafowy *DB-Tree* [Eze2002], który zapisuje w formie drzewa *FP-tree* częstości wszystkich atrybutów w zbiorze testowym. Ceną za to jest jednak potencjalnie bardzo duży rozmiar drzewa. Z kolei metoda APS całkowicie eliminuje ponowne przetwarzanie danych, kosztem dokładności wynikowego zbioru reguł (przede wszystkim pewności).

Jeśli chodzi o metodę aktualizacji bazy reguł zgodnie z modyfikacjami w bazie transakcji, zbliżone do metody APS są na przykład algorytmy FUP_2 [Che1997] i [Tsa1999], które wykorzystują wyniki poprzedniego przebiegu odkrywania reguł. Rozpatrywane są w nich różne przypadki, kiedy dany zbiór atrybutów: (i) jest częsty zarówno w bazie poprzednio analizowanej jak i zmienionej (natychmiastowa aktualizacja); (ii) jest częsty w części przeanalizowanej, ale nie jest częsty w zmienionej (wystarczy przeanalizować tylko część zmienioną); (iii) jest częsty tylko w części zmienionej, ale nie był częsty w części przeanalizowanej poprzednio (jest to przypadek potencjalnie najbardziej kosztowny obliczeniowo, gdyż może wymagać powtórnej analizy poprzedniego zbioru); (iv) nie jest częsty w żadnym fragmencie bazy (jest odrzucany). Podobne warianty rozważane są także w algorytmie RMAIN, należącym do cyklu metody APS. Tam jednak przypadki (ii) i (iii) rozwiązywane są przez aproksymację nieznanych wartości poparcia i pewności reguł za pomocą odpowiednich estymatorów, nie zaś przez dodatkowe przetwarzanie bazy transakcji.

Algorytmy takie, jak FUP_2 [Che1997], DELI [LeS1998], DLG* i DUP [LeG2001], wymagają rejestrowania, które transakcje w bazie zostały dodane, usunięte i pozostały bez zmian. Dzięki temu możliwe jest uwzględnienie modyfikacji w zbiorze, który był już wcześniej analizowany. Śledzenie zmian może być jednak pewnym obciążeniem dla systemu. Z kolei metoda APS w ogóle nie rejestruje zmian w danych już przetworzonych, ponieważ uwzględnia ona wyłącznie nowo dodawane fakty. Jest to jednak uzasadnione z punktu widzenia przeznaczenia metody, ponieważ historia agenta (a także innego systemu opartego na wiedzy) nie funkcjonuje na tych samych zasadach, co baza transakcyjna (ang. *On-Line Transactional Processing*, OLTP), gdzie ciągle są dokonywane zmiany danych. Baza obserwacji agenta (np. pochodzących z percepcji) praktycznie nie ulega modyfikacji, z wyjątkiem ustawicznego rejestrowania i dodawania nowych faktów. A zatem obserwacje raz zapisane w historii z założenia są zmieniane bardzo rzadko. Zauważmy, że ewentualna zmiana obserwacji (faktów) w historii oznaczałaby w istocie weryfikację mechanizmu percepcyjnego agenta, to znaczy np. uznanie, że to, co zarejestrował on z otoczenia nie jest zgodne ze stanem

faktycznym i wymaga modyfikacji. Tego typu rewizja obserwacji przez agenta jest czynnością rzadką i nieopłacalną zwłaszcza w sytuacji, gdy historia ma duży rozmiar, a pojedyncze obserwacje nie mają dużej wagi (w przeciwieństwie do odpowiednio dużego ich zbioru, który posiada wymaganą wiarygodność statystyczną).

Algorytm DELI [LeS1998] dopuszcza pewne przybliżenie zbioru reguł w stosunku do zbioru rzeczywiście uzyskanego na całej zmienionej bazie, które stanowi tolerancję dla zmian, aby nie było konieczne uruchamianie analizy danych, jeśli zmiany są niewielkie. Jest to podobne podejście do APS, gdzie nowe fakty są również kumulowane i analizowane po zgromadzeniu określonej ich liczby (porcji). Jednakże w odróżnieniu od APS, po zajściu zmian w bazie transakcji algorytm DELI szacuje, czy aktualizacja reguł jest konieczna. Tymczasem APS dokonuje aktualizacji bezwarunkowo po zgromadzeniu dostatecznie dużej liczby nowych faktów i uzyskaniu wolnych zasobów systemowych.

Wiele z omówionych wcześniej algorytmów inkrementacyjnych wykorzystuje reprezentację pośrednią, to znaczy struktury danych (np. grafy), w których przechowywanie są informacje o częstościach zbiorów atrybutów. Choć jest to wydajne podejście przetwarzania zbiorów częstych, struktury te muszą być odpowiednio przetworzone w celu uzyskania docelowych reguł związku. Tymczasem APS operuje na gotowych regułach, czyli takiej reprezentacji, która jest możliwa do przechowywania i bezpośredniego wykorzystania przez agenta w jego procesie decyzyjnym (bez poważnych opóźnień).

Żadna z przeanalizowanych metod inkrementacyjnych nie uwzględnia wpływu czasu zarejestrowania faktów na miary wiarygodności reguł związku, które na ich podstawie są odkrywane. W metodzie APS wprowadzona została funkcja wpływu czasowego f_T, pozwalająca na nadawanie największej istotności faktom najnowszym. Może to mieć szczególnie duże znaczenie, jeśli agent pozyskuje reguły na podstawie obserwacji silnie zmieniającego się środowiska.

W końcu zwróćmy uwagę, że w innych rozwiązaniach inkrementacyjnych zdolność przyrostowej aktualizacji bazy reguł jest własnością samego algorytmu znajdowania reguł. Tymczasem APS jest metodą wysoce niezależną od algorytmu odkrywania reguł związku, dzięki czemu z założenia można w niej zastosować różne algorytmy, o ile mogą one być dopasowane do cyklu metody pod względem struktur danych i parametrów wejściowych oraz wyjściowych[6]. Jako algorytm znajdowania reguł mogą być użyte algorytmy wsadowe (zarówno sekwencyjne, jak i równoległe, zob. [Zak1999]). Natomiast nie można raczej zastosować innych algorytmów inkrementacyjnych, ponieważ wymagają one innych struktur danych i mają często sprzeczne założenia w stosunku do metody APS.

3.7.2 Zastosowanie reguł związku w architekturach agenckich

Korzystamy tutaj częściowo z przeglądu metod uczenia się agentów, omówionego szczegółowo w Rozdziale 2.

Yao, Hamilton i Wang [Yao2002] zastosowali reguły związku odkrywane algorytmem Apriori do realizacji uczenia się rekomendującego agenta *PagePrompter*, który wspomaga

[6] W cyklu APS służy do tego algorytm ARM, który stanowi „obudowę" algorytmu odkrywania reguł związku. Algorytm ARM może być modyfikowany stosownie do różnych algorytmów DM, pod warunkiem, że nie naruszy to jego współpracy z innymi algorytmami cyklu.

obsługę serwisu internetowego. Dane wejściowe dla algorytmu pobierane są z rejestru zdarzeń w serwisie (ang. *web log*). Symeonidis, Mitkas i Kechagias [Sym2002] wykorzystali algorytmy eksploracji danych (w tym ID3, C4.5, Apriori) w systemie *Agent Academy*, realizującym trenowanie agentów na podstawie informacji zgromadzonej we wspólnym repozytorium wiedzy. Omawiane rozwiązania nie mają jednak charakteru ogólnych teorii, lecz prac aplikacyjnych. Natomiast metoda APS ma charakter ogólny i nie jest zawężona do algorytmu Apriori, ani żadnego konkretnego zastosowania.

W ostatnim czasie Kaya i Alhajj [Kay2005b], [Kay2005c] opracowali, przeznaczony dla systemu wieloagenckiego, model uczenia się ze wzmocnieniem (wykorzystujący algorytmy *Q-learning*), który opiera się na rozmytej strukturze wielowymiarowej OLAP (ang. *on-line analytical processing*) [Koh1998], przetwarzanej przez algorytmy znajdowania rozmytych reguł związku. Praca ta różni się pod wieloma względami od metody APS, w której wykorzystywane są zarówno inne struktury danych (tabele relacyjne), jak i inny model reguł związku (model Agrawala, Imielinskiego, Swamiego i Srikanta [Agr1993], [Agr1994], odpowiednio rozszerzony zgodnie z wymogami algorytmów cyklu APS).

4. Podsumowanie

W pracy doktorskiej zaproponowana została oryginalna metoda APS inkrementacyjnego pozyskiwania reguł związku, która stanowi rozwiązanie problemu przedstawionego we wstępie rozprawy. Do szczegółowych wyników pracy należą:

1) opracowanie założeń metody;
2) zdefiniowanie struktury bazy wiedzy agenta;
3) zdefiniowanie cyklu pozyskiwania reguł metody APS;
4) zaproponowanie algorytmów: FSEL, HTRANS, HFILL, ENV, ARM i RMAIN, które służą do przetwarzania danych w ramach cyklu APS;
5) analiza własności algorytmów i ich złożoności obliczeniowej;
6) weryfikacja formalna własności algorytmu aktualizacji bazy reguł RMAIN;
7) weryfikacja eksperymentalna metody APS i porównanie jakościowych oraz wydajnościowych wyników jej działania z metodą wsadową.

W metodzie APS schemat historii i przechowywane tam dane są w każdym przebiegu transformowane do postaci wymaganej przez algorytm eksploracji danych. Rozwiązanie takie umożliwia inkrementacyjne łączenie reguł, które są znajdowane w porcjach historii o różniących się od siebie schematach, przy założeniu, że zachowana jest ciągłość ontologii, to znaczy utrzymane jest znaczenie poszczególnych atrybutów i ich wartości.

Oryginalność proponowanej metody polega na założeniu *niedoskonałej pamięci* (ang. *imperfect recall*), zgodnie z którym raz przetworzone fakty są trwale usuwane z historii. Jest to podejście różniące się od innych, inkrementacyjnych metod znajdowania reguł związku, które w zdecydowanej większości nie wykluczają całkowicie powtórnych przebiegów na przeanalizowanej bazie faktów, lecz minimalizują ich liczbę.

Kolejną cechą odróżniającą metodę APS od innych prac, jest jej zewnętrzny charakter w stosunku do wykorzystywanego algorytmu eksploracji danych. Pozwala to zastosowanie różnych algorytmów odkrywania reguł związku, o ile spełniają one wymagania odnośnie danych wejściowych i wyjściowych.

W końcu, metoda APS operuje na regułach związku w postaci docelowej, w której są one gotowe do bezpośredniego wykorzystania w procesie decyzyjnym agenta. Tymczasem inne

metody inkrementacyjne bazują na pośrednich reprezentacjach (takich, jak zbiory częste), które muszą przetworzone w celu uzyskania reguł związku. Wymienione właściwości są istotne z punktu widzenia dopasowania metody APS do potrzeb systemów agenckich lub innych systemów opartych na wiedzy.

Wyniki badań eksperymentalnych wskazują, że metoda APS pozwala na przetwarzanie przez agenta dużej liczby obserwacji przy zachowaniu ograniczonego rozmiaru historii. Jednocześnie zbiór reguł, będących wynikiem działania inkrementacyjnych przebiegów metody, jest porównywalny ze zbiorem reguł uzyskanych wsadowo na tym samym zbiorze faktów. Szczególnie duże podobieństwo obu zbiorów reguł zachodzi przy przetwarzaniu zbioru danych, który jest jednorodny pod względem rozkładu reguł w faktach. Ponadto, długość trwania inkrementacyjnego przetwarzania faktów w kolejnych przebiegach metody APS jest mniejsza od długości trwania przebiegów wsadowych, jeśli liczba faktów przeanalizowanych od początku przekracza pewną wielkość ψ, zależną od własności danych i ustawień parametrów metody.

Metoda APS może stanowić część szerszego modelu statystycznego uczenia się bez nadzoru w architekturze agenckiej. Potencjalnymi dziedzinami zastosowań proponowanej metody są systemy agenckie, rejestrujące z otoczenia dużą liczbę obserwacji, które bezpośrednio nie mogą być wykorzystane w procesie decyzyjnym, lecz wymagają przetworzenia w celu odkrycia uogólnionych wzorców. Do aplikacji takich należą systemy tworzące model użytkownika, na przykład agenty asystujące przy wyszukiwaniu i filtrowaniu informacji, które wykorzystują zapis aktywności użytkownika do automatycznego budowania wiedzy o jego preferencjach.

Propozycje dalszych badań obejmują osadzenie metody APS w architekturze agenckiej i pełnym cyklu uczenia się agenta. Wymaga to odniesienia cyklu APS i wyników jego działania do procesu decyzyjnego systemu agenckiego. Z tym wiąże się ciekawe zagadnienie opracowania architektury meta-uczenia się (ang. *metalearning*), zawierającej algorytmy autonomicznej konfiguracji parametrów metody APS przez uczącego się agenta. W końcu, naturalną kontynuacją rozwiązań prezentowanych w niniejszej pracy jest zastosowanie metody APS w rzeczywistych aplikacjach i środowiskach wdrożeniowych.

5. Dodatek A: Przykład obliczeniowy

Niniejszy dodatek zawiera przykłady przetwarzania danych w wybranych etapach cyklu APS: przekształcaniu schematu historii, wypełnianiu historii faktami, eliminacji wartości nieznanych i aktualizacji bazy reguł. Przedstawione dane dotyczą dziedziny wyszukiwania informacji w sieci WWW – tej samej zatem, do której odnoszą się zbiory testowe T9.I4.D20K i T10.I5.D20K, wykorzystywane w badaniach eksperymentalnych metody APS. Przypomnijmy założenia agenta, którego dotyczą przykładowe dane. System ten można określić jako osobistego agenta (ang. *personal agent*), wspomagającego użytkownika przy przeglądaniu stron internetowych. Jeśli użytkownik podczas tej aktywności jawnie ocenia lub zapisuje lokalnie daną stronę, działający w tle agent analizuje ją pełnotekstowo i zapisuje w historii n terminów indeksowych o największej częstości (z wyłączeniem typowych słów pomijanych, np. *a*, *and*, *for*, *he*, *she*, *to*), ocenę oraz fakt zapisu. Celem działania agenta jest odkrycie reguł, które pozwalają określić, od jakich terminów na stronie zależy to, że jest ona interesująca dla użytkownika.

5.1. Zawartość historii

Na etapie projektowania systemu historia KB_H została zdefiniowana jako tabela relacyjnej bazy danych z poniższymi polami (przykład wypełnionej historii jest na Rys. A.1.):

- *HKey* – klucz (odpowiednik atrybutu specjalnego *K* w modelu formalnym);
- *HTime* – data i czas zarejestrowania faktu (odpowiednik atrybutu specjalnego *T*);
- *Term* – kolekcja terminów indeksowych, opisujących stronę (atrybut wielowartościowy);
- *Eval* – ocena strony: *relevant* albo *irrelevant* (atrybut jednowartościowy);
- *Stored* – zdarzenie zapisu strony na dysk: *yes* albo *no* (atrybut jednowartościowy).

HKey	HTime	Term	Eval	Stored
1	2005-04-19 16:30	*base*, *disc*, *computer*, ..., *storage*	*relevant*	*yes*
2	2005-04-20 15:35	*base*, *matrix*, *page*, ..., *server*	*relevant*	*no*
3	2005-04-20 18:12	*disc*, *Internet*, *scanner*, ..., *storage*	*N*	*N*
...	...	...	...	...
k	2005-05-21 17:38	*base*, *copy*, *pattern*, ..., *server*	*N*	*yes*

Rys. A.1. Przykładowa historia agenta rekomendującego strony WWW.

Jak widać, takie same terminy (np. *disc*) mogą występować w wielowartościowym polu *Term* na różnych pozycjach (np. raz jako pierwszy, inny raz jako *i*-ty termin), ponieważ w różnych dokumentach (stronach) zazwyczaj jest inny zbiór *n* najczęściej występujących terminów indeksowych. W fakcie o wartości klucza [HKey] = 3 atrybuty *Eval* i *Stored* mają jednocześnie wartość nieznaną *N*. Jest to zabieg świadomy, który służy ilustracji jednego z przypadków, omawianego dalej, algorytmu ENV. W rzeczywistym przypadku taki fakt nie byłby w ogóle dodany przez agenta do historii, ponieważ zapis ma dotyczyć wyłącznie stron jawnie ocenianych (atrybut *Eval*) lub zapisywanych (atrybut *Stored*) przez użytkownika.

5.2. Przekształcanie schematu historii

Poniżej podany jest przykładowy przebieg algorytmu HTRANS, który wykonuje transformację schematu historii (procedura `Przekształć_Schemat` w cyklu APS). Na wejściu do algorytmu przekazywana jest porcja faktów o pierwotnym schemacie S_H, pobrana z historii przez algorytm FSEL.

1. (Krok 1) Nowy schemat jest zbiorem pustym.
2. (Krok 2) Do nowego schematu wchodzą bez zmian atrybuty specjalne (odpowiedniki atrybutów *K* i *T* w modelu formalnym):
   $S^{\#} := S^{\#} \cup \{HKey, HTime\}$.
3. (Kroki 3 – 5, pierwsza iteracja) Wyznaczana jest dziedzina właściwa D^{W}_{1} atrybutu $A_1 = Term$. Ponieważ jest to atrybut wielowartościowy, brane są wszystkie wartości atomowe, które pojawiły się w kolumnie *Term*:

$\{base, computer, copy, disc, Internet, matrix, page, pattern, scanner, ..., server, storage\}$.

Na podstawie tych wartości tworzone są nowe atrybuty $A_1^{(v)}$, dodawane do schematu $S^{\#}$:

$S^{\#} := S^{\#} \cup \{Term^{(base)}, Term^{(computer)}, Term^{(copy)}, Term^{(disc)}, Term^{(Internet)}, Term^{(matrix)}, Term^{(page)}, Term^{(pattern)}, Term^{(scanner)}, ..., Term^{(server)}, Term^{(storage)}\}$.

4. (Kroki 3 – 5, druga iteracja) Wyznaczana jest dziedzina właściwa D^{W}_{2} atrybutu jednowartościowego $A_2 = Eval$. Atrybut ten przyjął wartości: $\{relevant, irrelevant\}$.

Na podstawie tych wartości tworzone są nowe atrybuty $A_2^{(v)}$, dodawane do schematu $S^{\#}$:

$S^{\#} := S^{\#} \cup \{Eval^{(relevant)}, Eval^{(irrelevant)}\}$.

5. (Kroki 3 – 5, trzecia iteracja) Wyznaczana jest dziedzina właściwa D^{W}_{3} atrybutu jednowartościowego $A_3 = Stored$. Atrybut ten przyjął wartości: $\{yes, no\}$.

Na podstawie tych wartości tworzone są nowe atrybuty $A_3^{(v)}$, dodawane do schematu $S^{\#}$:

$S^{\#} := S^{\#} \cup \{Stored^{(yes)}, Stored^{(no)}\}$

6. (Krok 7) W wyniku zwracany jest nowy schemat:
   $S^{\#} = \{HKey, HTime, Term^{(base)}, Term^{(computer)}, Term^{(copy)}, Term^{(disc)}, Term^{(Internet)}, Term^{(matrix)}, Term^{(page)}, Term^{(pattern)}, Term^{(scanner)}, ..., Term^{(server)}, Term^{(storage)}, Eval^{(relevant)}, Eval^{(irrelevant)}, Stored^{(yes)}, Stored^{(no)}\}$.

5.3. Wypełnianie danymi nowego schematu

Algorytm HFILL, którego przykładowy przebieg prezentowany jest poniżej, przepisuje fakty z porcji $KB_H(t_{sc}; t_{ec})$ do tabeli o nowym schemacie $S^{\#}$. Jest to etap realizowany w cyklu APS przez procedurę `Wypełnij_Nowy_Schemat`.

1. (Krok 1) Nowy zbiór faktów $KB^{\#}_H(t_{sc}; t_{ec})$ jest pusty (Rys. A.2).

HKey	HTime	*Term* (*base*)	*Term* (*disc*)	...	*Eval* (*relevant*)	*Eval* (*irrelevant*)	*Stored* (*yes*)	*Stored* (*no*)
				...				

Rys. A.2. Pusty zbiór faktów o przekształconym schemacie $S^{\#}$.

2. (Kroki 2 – 14, *m* iteracji) Do zbioru $KB^{\#}_H(t_{sc}; t_{ec})$ dodawane są kolejno fakty z porcji.

3. (Krok 16) W wyniku zwracany jest zbiór przekształconych faktów (Rys. A.3).

HKey	HTime	*Term* (*base*)	*Term* (*disc*)	...	*Eval* (*relevant*)	*Eval* (*irrelevant*)	*Stored* (*yes*)	*Stored* (*no*)
1	2005-04-19 16:30	1	1	...	1	0	1	0
2	2005-04-20 15:35	1	0	...	1	0	0	1
3	2005-04-20 18:12	0	1	...	*N*	*N*	*N*	*N*
...	...	...	...	...	...	...	...	...
k	2005-05-21 17:38	1	0	...	*N*	*N*	1	0

Rys. A.3. Zbiór faktów o nowym schemacie $S^{\#}$.

5.4. Eliminacja wartości nieznanych

Kolejnym etapem jest usunięcie wartości *N* z przekształconej porcji faktów, które jest realizowane przez algorytm ENV (procedura `Usuń_Wartości_N` w cyklu APS).

W omawianym przykładzie danymi wejściowymi algorytmu są: przekształcona porcja faktów $KB^{\#}_H(t_{sc}, t_{ec})$ na Rys. A.3. oraz maksymalna dopuszczalna liczba atrybutów z nieznaną wartością w jednym fakcie $\eta_c(v_c) = 3$, gdzie $v_c \in KB_T$.

1. (Kroki 1 – 4, pierwsza iteracja) Analizowany jest fakt o kluczu [HKey] = 1. Liczba *m* wartości *N* jest równa zero, więc fakt jest pozostawiany bez zmian (zob. Rys. A.4).

HKey	HTime	*Term* (*base*)	*Term* (*disc*)	...	*Eval* (*relevant*)	*Eval* (*irrelevant*)	*Stored* (*yes*)	*Stored* (*no*)
1	2005-04-19 16:30	1	1	...	1	0	1	0
2	2005-04-20 15:35	1	0	...	1	0	0	1
3	2005-04-20 18:12	0	1	...	*N*	*N*	*N*	*N*
...	...	...	...	...	...	...	...	...
k	2005-05-21 17:38	1	0	...	*N*	*N*	1	0

Rys. A.4. W pierwszej iteracji algorytmu ENV pierwszy fakt pobranej porcji jest pozostawiany bez zmian.

2. (Kroki 1 – 4, druga iteracja) Przetwarzany jest fakt o kluczu [HKey] = 2. Liczba *m* wartości *N* jest równa zero, więc fakt jest pozostawiany bez zmian.

3. (Kroki 1 – 4, trzecia iteracja) Analizowany jest fakt o kluczu [HKey] = 3. Liczba *m* wartości *N* jest równa 4. Ponieważ $m > \eta_c(v_c)$, krok 3. jest pomijany i w kroku 4. fakt ten jest usuwany z historii (zob. Rys. A.5.).

HKey	HTime	*Term* (*base*)	*Term* (*disc*)	...	*Eval* (*relevant*)	*Eval* (*irrelevant*)	*Stored* (*yes*)	*Stored* (*no*)
1	2005-04-19 16:30	1	1	...	1	0	1	0
2	2005-04-20 15:35	1	0	...	1	0	0	1
~~3~~	~~2005-04-20 18:12~~	~~0~~	~~1~~	~~...~~	***N***	***N***	***N***	***N***
...	...	...	...	...	...	...	...	...
k	2005-05-21 17:38	1	0	...	*N*	*N*	1	0

Rys. A.5. W trzeciej iteracji fakt o kluczu [HKey] = 3 jest usuwany z historii bez zamiany na inne fakty, ponieważ liczba *m* nieznanych wartości przekracza przyjęty próg $\eta_c(v_c) = 3$.

HKey	HTime	*Term* (*base*)	*Term* (*disc*)	...	*Eval* (*relevant*)	*Eval* (*irrelevant*)	*Stored* (*yes*)	*Stored* (*no*)
1	2005-04-19 16:30	1	1	...	1	0	1	0
2	2005-04-20 15:35	1	0	...	1	0	0	1
...	...	...	...	...	...	...	...	...
~~*k*~~	~~2005-05-21 17:38~~	~~1~~	~~0~~	~~...~~	***N***	***N***	~~1~~	~~0~~
k	2005-05-21 17:38	1	0	...	**1**	**0**	1	0
k	2005-05-21 17:38	1	0	...	**0**	**1**	1	0

Rys. A.6. Tabela faktów w trakcie eliminacji wartości *N*. Wiersz numer *k* został zastąpiony dwoma wierszami, które nie zawierają wartości *N*. Możliwe są tutaj tylko dwa wiersze, ponieważ *Eval* jest atrybutem jednowartościowym i w jednym fakcie może mieć dokładnie jedną wartość.

4. (Kroki 1 – 4, *k*-ta iteracja) Sprawdzany jest fakt o kluczu [HKey] = *k*. Liczba *m* wartości *N* jest równa 2, a więc $0 < m \leq \eta_c(v_c) = 3$. Dlatego też w kroku 3. fakt ten jest zastępowany dwoma wierszami, które odpowiadają wszystkim możliwym wartościom atrybutów $A_i^{(v)}$, pochodzących od atrybutu A_i = *Eval*. Zwróćmy uwagę, że ponieważ jest to atrybut jednowartościowy, w jednym fakcie tylko jeden atrybut $A_i^{(v)}$ może przyjąć wartość 1 (zob. Rys. A.6.).

5.5. Aktualizacja bazy reguł

Przypuśćmy, że zawartość bazy reguł KB_R jest taka, jak na Rys. A.7.

	X	*Y*	*sup*	*con*	*b*	t_m
p_1	*Term_base*, *Term_disc*	*Eval_relevant*	0,12	0,73	3000	2005-01-16 10:45
p_2	*Term_base*, *Term_copy*	*Stored_yes*	0,21	0,69	3000	2005-01-16 10:45

Rys. A.7. Zawartość bazy reguł KB_R przed aktualizacją.

Załóżmy, że po wykonaniu algorytmu ARM w ramach procedury `Znajdź_Reguły` znaleziony został zbiór nowych reguł *R*, przedstawiony na Rys. A.8.

	X	*Y*	*sup*	*con*	*b*	t_m
r_1	*Term_base*, *Term_disc*	*Eval_relevant*	0,15	0,90	500	2005-05-05 17:04
r_2	*Term_base*, *Term_matrix*	*Eval_relevant*	0,28	0,85	500	2005-05-05 17:04

Rys. A.8. Zbiór nowych reguł *R*, odkrytych podczas bieżącego przebiegu cyklu APS.

Parametry

Przyjmujemy poniższe wartości parametrów (wykorzystywanych przez algorytm RMAIN).

- Wektor bieżącego przebiegu v_c:
  - $b_c = 500$ (liczba faktów przeanalizowanych w bieżącym przebiegu);
  - $\sigma_c = 0{,}06$ (próg minimalnego poparcia reguł);
  - $\gamma_c = 0{,}70$ (próg minimalnej pewności reguł);
  - $\hat{\sigma}_c = 0{,}03$ (oczekiwane poparcie reguł odrzucanych, przyjęte jako $\sigma_c/2$);
  - $\hat{\gamma}_c = 0{,}35$ (oczekiwana pewność reguł odrzucanych, przyjęta jako $\gamma_c/2$);
  - t_{mc} = „2005-05-05 17:04” (średni czas faktów analizowanych w bieżącym przebiegu);
  - t_{sc} = „2005-04-19 16:30” (czas najwcześniejszego faktu w bieżącym przebiegu);
  - t_{ec} = „2005-05-21 17:38” (czas najpóźniejszego faktu w bieżącym przebiegu).
- Wektor parametrów globalnych v_g:
  - $b_g = 3000$ (liczba faktów analizowanych w dotychczasowych przebiegach);
  - $\sigma_g = 0{,}10$ (próg minimalnego poparcia reguł dla wszystkich przebiegów);
  - $\gamma_g = 0{,}60$ (próg minimalnej pewności reguł dla wszystkich przebiegów);
  - $\hat{\sigma}_g = 0{,}05$ (oczekiwane poparcie reguł odrzucanych);
  - $\hat{\gamma}_g = 0{,}30$ (oczekiwana pewność reguł odrzucanych);
  - t_{mg} = „2005-01-16 10:45” (średni czas faktów w dotychczasowych przebiegach);

- t_{sg} = „2004-10-15 7:24” (czas najwcześniejszego faktu we wszystkich przebiegach);
- t_{eg} = „2005-04-19 14:07” (czas najpóźniejszego faktu we wszystkich przebiegach);
- $f_T(x) = exp(ax^2)$, gdzie $a = -0{,}0001$ (łagodna, eksponencjalna funkcja wpływu czasowego dla czasu x, odmierzanego w dniach; przykładowe wartości: $f_T(30) \approx 0{,}91$; $f_T(60) \approx 0{,}70$; $f_T(90) \approx 0{,}44$; $f_T(180) \approx 0{,}04$; $f_T(365) \approx 0{,}00$).

• Parametry pobrane z systemu operacyjnego:

- t_{now} = „2005-05-21 18:05” (bieżący czas, traktowany jako moment analizy).

Przebieg algorytmu

Wartości parametrów z przedziału [0; 1] są zaokrąglane do drugiego miejsca po przecinku, różnice dat, podawane jako argumenty funkcji f_T, są liczone w pełnych dniach. Daty są reprezentowane jako typ *datetime*, to znaczy liczba interwałów o długości 0,03 sekundy, które upłynęły od 1 stycznia 1753 roku. Na poziomie technicznym, podczas liczenia średnich ważonych dat, należy wykonać szereg odpowiednich konwersji (np. daty → liczby; obliczenie średniej ważonej na liczbach; średnia → data).

---

1. (Krok 1) Aktualizowane są parametry wektora v_g:

$$\sigma_g(v_g) := \frac{0{,}10 \cdot 3000 + 0{,}06 \cdot 500}{3000 + 500} \approx 0{,}09,$$

$$\gamma_g(v_g) := \frac{0{,}60 \cdot 3000 + 0{,}70 \cdot 500}{3000 + 500} \approx 0{,}61.$$

2. (Krok 2, pierwsza iteracja) Kroki 3–6 są powtarzane dla reguły $r_1 \in R$.

3. (Krok 3, pierwsza iteracja) Ponieważ $p_1 \equiv r_1$, gdzie $p_1 \in KB_R$, to aktualizowane są parametry reguły p_1:

$$sup(p_1) := \frac{3000 \cdot exp(-0{,}0001 \cdot 125^2) \cdot 0{,}12 + 500 \cdot exp(-0{,}0001 \cdot 16^2) \cdot 0{,}15}{3000 \cdot exp(-0{,}0001 \cdot 125^2) + 500 \cdot exp(-0{,}0001 \cdot 16^2)} \approx 0{,}13,$$

$$con(p_1) := \frac{0{,}73 \cdot 0{,}90 \left(3000 \cdot exp(-0{,}0001 \cdot 125^2) \cdot 0{,}12 + 500 \cdot exp(-0{,}0001 \cdot 16^2) \cdot 0{,}15\right)}{3000 \cdot 0{,}12 \cdot exp(-0{,}0001 \cdot 125^2) \cdot 0{,}90 + 500 \cdot 0{,}15 \cdot exp(-0{,}0001 \cdot 16^2) \cdot 0{,}73} \approx 0{,}80,$$

$$t_m(p_1) := datetime\left(\frac{3000 \cdot int(2005\text{-}01\text{-}16\ 10{:}45) + 500 \cdot int(2005\text{-}05\text{-}05\ 17{:}04)}{3000 + 500}\right) \approx 2005\text{-}02\text{-}01\ 01{:}22,$$

$$b(p_1) := 3000 + 500 = 3500.$$

4. (Krok 4, pierwsza iteracja) Warunki nie są spełnione, a więc p_1 pozostaje w bazie KB_R.

5. (Kroki 5 i 6, pierwsza iteracja) Kroki nie są wykonywane, ponieważ nie są spełnione odpowiednie warunki.

6. (Krok 2, druga iteracja) Kroki 3–6 są powtarzane dla reguły $r_2 \in R$.

7. (Kroki 3 i 4, druga iteracja) Kroki nie są wykonywane, ponieważ nie są spełnione odpowiednie warunki.

8. (Krok 5, druga iteracja) Ponieważ $\neg\exists p_j \in KB_R \,.\, p_j \equiv r_2$, aktualizowane są parametry reguły r_2:

$$sup(r_2) := \frac{3000 \cdot exp(-0{,}0001 \cdot 125^2) \cdot 0{,}05 + 500 \cdot exp(-0{,}0001 \cdot 16^2) \cdot 0{,}28}{3000 \cdot exp(-0{,}0001 \cdot 125^2) + 500 \cdot exp(-0{,}0001 \cdot 16^2)} \approx 0{,}15,$$

$$con(r_2) := \frac{0{,}30 \cdot 0{,}85 \left(3000 \cdot 0{,}05 \cdot exp(-0{,}0001 \cdot 125^2) + 500 \cdot 0{,}28 \cdot exp(-0{,}0001 \cdot 16^2)\right)}{3000 \cdot 0{,}05 \cdot exp(-0{,}0001 \cdot 125^2) \cdot 0{,}85 + 500 \cdot 0{,}28 \cdot exp(-0{,}0001 \cdot 16^2) \cdot 0{,}30} \approx 0{,}63,$$

$$t_m(r_2) := datetime\left(\frac{3000 \cdot int(2005\text{-}01\text{-}16\ 10{:}45) + 500 \cdot int(2005\text{-}05\text{-}05\ 17{:}04)}{3000 + 500}\right) \approx 2005\text{-}02\text{-}01\ 01{:}22,$$

$$b(r_2) := 3000 + 500 = 3500.$$

9. (Krok 6, druga iteracja) Ponieważ po aktualizacji $sup(r_2) \geq \sigma_g \wedge con(r_2) \geq \gamma_g$, to $KB_R := KB_R \cup \{r_2\}$.

10. (Krok 8) Ponieważ $\neg\exists r_i \in R \,.\, p_2 \equiv r_i$, dla reguły $p_2 \in KB_R$ powtarzane są kroki 9–10.

11. (Krok 9) Aktualizowane są parametry reguły p_2:

$$sup(p_2) := \frac{3000 \cdot exp(-0{,}0001 \cdot 125^2) \cdot 0{,}21 + 500 \cdot exp(-0{,}0001 \cdot 16^2) \cdot 0{,}03}{3000 \cdot exp(-0{,}0001 \cdot 125^2) + 500 \cdot exp(-0{,}0001 \cdot 16^2)} \approx 0{,}13,$$

$$con(p_2) := \frac{0{,}69 \cdot 0{,}35 \left(3000 \cdot exp(-0{,}0001 \cdot 125^2) \cdot 0{,}21 + 500 \cdot exp(-0{,}0001 \cdot 16^2) \cdot 0{,}03\right)}{3000 \cdot 0{,}21 \cdot exp(-0{,}0001 \cdot 125^2) \cdot 0{,}35 + 500 \cdot 0{,}03 \cdot exp(-0{,}0001 \cdot 16^2) \cdot 0{,}69} \approx 0{,}63,$$

$$t_m(p_2) := datetime\left(\frac{3000 \cdot int(2005\text{-}01\text{-}16\ 10{:}45) + 500 \cdot int(2005\text{-}05\text{-}05\ 17{:}04)}{3000 + 500}\right) \approx 2005\text{-}02\text{-}01\ 01{:}22,$$

$$b(p_2) := 3000 + 500 = 3500.$$

10. (Krok 10) Ponieważ po aktualizacji $sup(p_j) > \sigma_g$ oraz $con(p_j) > \gamma_g$, to reguła p_2 jest pozostawiana w bazie KB_R.

11. (Krok 12) Aktualizowane są parametry wektora v_g:

$$\hat{\sigma}_g(v_g) := \frac{0{,}05 \cdot 3000 + 0{,}03 \cdot 500}{3000 + 500} \approx 0{,}05,$$

$$\hat{\gamma}_g(v_g) := \frac{0{,}30 \cdot 3000 + 0{,}35 \cdot 500}{3000 + 500} \approx 0{,}31,$$

$$t_{mg}(v_g) := datetime\left(\frac{3000 \cdot int(2005\text{-}01\text{-}16\ 10{:}45) + 500 \cdot int(2005\text{-}05\text{-}05\ 17{:}04)}{3000 + 500}\right) \approx 2005\text{-}02\text{-}01\ 01{:}22.$$

12. (Krok 13) Aktualizowany jest parametr wektora v_g:

$$b_g(v_g) := 3000 + 500 = 3500.$$

13. (Krok 14) Zwracane są: KB_R, $\sigma_g(v_g)$, $\gamma_g(v_g)$, $\hat{\sigma}_g(v_g)$, $\hat{\gamma}_g(v_g)$, $t_{mg}(v_g)$, $b_g(v_g)$.

Nowe wartości wektora parametrów globalnych v_g, zwrócone przez algorytm RMAIN:

- $b_g = 3500$;
- $\sigma_g = 0{,}09$;

- $\gamma_g = 0{,}61$;
- $\hat{\sigma}_g = 0{,}05$;
- $\hat{y}_g = 0{,}31$;
- t_{mg} = „2005-02-01 01:22”.

Baza reguł KB_R, zaktualizowana przez algorytm RMAIN, jest przedstawiona na Rys. A.9.

	X	*Y*	*sup*	*con*	*b*	t_m
p_1	*Term_base*, *Term_disc*	*Eval_relevant*	0,13	0,80	3500	2005-02-01 01:22
p_2	*Term_base*, *Term_copy*	*Stored_yes*	0,13	0,63	3500	2005-02-01 01:22
r_2	*Term_base*, *Term_matrix*	*Eval_relevant*	0,15	0,63	3500	2005-02-01 01:22

Rys. A.9. Zawartość bazy reguł KB_R po aktualizacji przez algorytm RMAIN.

6. Dodatek B: Wyniki eksperymentów

W niniejszym dodatku zestawione są liczbowe wyniki eksperymentów, których opracowanie i omówienie jest zawarte w Rozdziale 3. rozprawy. Tam umieszczony jest także szczegółowy opis parametrów metody, strategii eksperymentu i wykorzystywanych zbiorów testowych. Dla jednorodnego rozkładu reguł prowadzona była jedna seria eksperymentów na zbiorze T9.I4.D20K, opisanym w Rozdziale 3. Eksperymenty dla niejednorodnego rozkładu reguł prowadzone były w dwóch odrębnych seriach na skrzyżowanym zbiorze testowym T9.I4.D20K + T10.I5.D20K (zob. Rozdział 3). Ustawienia wartości parametrów globalnych wektora v_g we wszystkich seriach eksperymentu były identyczne, zgodnie z wykazem przedstawionym niżej.

6.1. Parametry globalne

Poniżej zestawione są początkowe wartości parametrów globalnych wektora v_g, ustawione przed rozpoczęciem przebiegów cyklu APS. Takie same wartości były ustawiane przed rozpoczęciem każdego przebiegu wsadowego.

- $b_g = 0$ (liczba faktów przeanalizowanych we wszystkich dotychczasowych przebiegach);
- $\sigma_g = 0{,}00$ (próg minimalnego poparcia reguł dla wszystkich przebiegów);
- $\gamma_g = 0{,}00$ (próg minimalnej pewności reguł dla wszystkich przebiegów);
- $\hat{\sigma}_g = 0{,}00$ (oczekiwane poparcie reguł odrzucanych);
- $\hat{\gamma}_g = 0{,}00$ (oczekiwana pewność reguł odrzucanych);
- t_{mg} – *nieokreślony* (średni czas faktów przetwarzanych we wszystkich przebiegach);
- t_{sg} – *nieokreślony* (czas najwcześniejszego faktu we wszystkich przebiegach);
- t_{eg} – *nieokreślony* (czas najpóźniejszego faktu we wszystkich przebiegach);
- k_{eg} – *nieokreślony* (klucz najpóźniejszego faktu we wszystkich przebiegach);
- $f_T(x) = ax + 1$, gdzie $a = 0$ (funkcja wpływu czasowego; tutaj ustawiona jako funkcja stała o wartości 1).

6.2. Badania przy jednorodnym rozkładzie reguł

6.2.1 Parametry przebiegów inkrementacyjnych

W Tab. B.1. umieszczone są parametry przebiegów inkrementacyjnych na zbiorze T9.I4.D20K.

id_c	b_c	η_c	σ_c	γ_c	m_x	m_y	X_c	Y_c	$\hat{\sigma}_c$	$\hat{\gamma}_c$	t_{sc}	t_{ec}	t_{mc}	k_{ec}
1	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2004-01-24 10:34:26	2004-01-12 19:51:28	1000
2	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-24 12:11:04	2004-02-17 06:41:47	2004-02-05 07:33:43	2000
3	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-02-17 08:52:04	2004-03-11 23:47:16	2004-02-29 02:47:33	3000
4	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-03-12 00:09:39	2004-04-03 11:13:35	2004-03-23 06:53:40	4000
5	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-04-03 11:26:43	2004-04-26 23:19:51	2004-04-14 19:58:24	5000
6	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-04-27 00:19:46	2004-05-20 09:41:16	2004-05-08 15:17:01	6000
7	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-05-20 10:22:38	2004-06-11 14:26:45	2004-05-31 07:06:43	7000
8	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-06-11 14:32:42	2004-07-04 17:44:48	2004-06-23 13:37:35	8000
9	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-07-04 18:54:32	2004-07-28 00:04:49	2004-07-16 08:42:36	9000
10	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-07-28 00:04:55	2004-08-19 21:01:17	2004-08-08 10:24:21	10000
11	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-08-19 21:33:34	2004-09-11 10:43:40	2004-08-30 23:16:15	11000
12	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-09-11 10:56:57	2004-10-05 00:04:24	2004-09-22 18:19:05	12000
13	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-10-05 00:04:48	2004-10-28 09:19:39	2004-10-16 15:44:52	13000
14	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-10-28 10:43:31	2004-11-20 13:05:30	2004-11-08 23:41:23	14000
15	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-11-20 13:15:27	2004-12-14 10:05:24	2004-12-01 22:23:26	15000
16	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-12-14 10:14:36	2005-01-06 07:56:08	2004-12-25 21:09:29	16000
17	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2005-01-06 10:02:31	2005-01-29 23:22:40	2005-01-18 02:55:30	17000
18	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2005-01-29 23:39:13	2005-02-21 19:27:25	2005-02-10 08:11:09	18000
19	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2005-02-21 22:00:33	2005-03-17 10:09:06	2005-03-05 21:15:28	19000
20	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2005-03-17 10:22:30	2005-04-10 23:25:23	2005-03-30 05:35:56	20000

Tab. B.1. Parametry wektora v_c dla przebiegów inkrementacyjnych metody APS dla jednorodnego rozkładu reguł w zbiorze testowym.

6.2.2 Pomiary czasu trwania przebiegów inkrementacyjnych

W Tab. B.2. umieszczone są wyniki pomiarów czasu trwania (w milisekundach) poszczególnych etapów przebiegów inkrementacyjnych na zbiorze T9.I4.D20K.

id_c	t_{select}	$t_{convert}$	t_{fill}	t_{elim}	t_{mine}	t_{add}	t_{del}	t_{sum}
1	60	210	420	30	931	1912	20	3585
2	0	170	460	20	711	1822	20	3204
3	0	170	420	30	680	2213	20	3535
4	10	160	420	30	741	1802	20	3184
5	10	170	420	40	560	1672	20	2894
6	10	210	450	20	600	1642	20	2954
7	0	170	430	20	600	1682	10	2914
8	0	210	440	40	771	1912	20	3394
9	10	180	610	20	741	1672	20	3254
10	0	190	460	20	650	1642	20	2984
11	0	170	420	30	711	1762	20	3114
12	0	200	470	30	731	1842	20	3294
13	10	160	420	10	590	1732	20	2944
14	10	160	450	20	650	1812	20	3124
15	0	190	440	20	590	1962	20	3224
16	10	240	460	20	711	2443	20	3905
17	0	190	420	30	600	1912	40	3194
18	10	190	430	20	590	1872	20	3134
19	10	160	440	20	660	2022	20	3334
20	10	170	430	30	711	2513	20	3885

Tab. B.2. Pomiary czasu trwania poszczególnych etapów przebiegów inkrementacyjnych metody APS dla jednorodnego rozkładu reguł w zbiorze testowym.

6.2.3 Parametry przebiegów wsadowych

W Tab. B.3. umieszczone są parametry przebiegów wsadowych na zbiorze T9.I4.D20K.

id_c	b_c	η_c	σ_c	γ_c	m_x	m_y	X_c	Y_c	$\hat{\sigma}_c$	$\hat{\gamma}_c$	t_{sc}	t_{ec}	t_{mc}	k_{ec}
1	1000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2004-01-24 10:34:26	2004-01-12 19:51:28	1000
2	2000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2004-02-17 06:41:47	2004-01-24 13:42:36	2000
3	3000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2004-03-11 23:47:16	2004-02-05 10:04:15	3000
4	4000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2004-04-03 11:13:35	2004-02-17 03:16:36	4000
5	5000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2004-04-26 23:19:51	2004-02-28 16:12:58	5000
6	6000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2004-05-20 09:41:16	2004-03-11 08:03:38	6000
7	7000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2004-06-11 14:26:45	2004-03-22 21:38:22	7000
8	8000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2004-07-04 17:44:48	2004-04-03 11:38:16	8000
9	9000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2004-07-28 00:04:49	2004-04-15 00:38:45	9000
10	10000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2004-08-19 21:01:17	2004-04-26 13:37:18	10000
11	11000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2004-09-11 10:43:40	2004-05-08 01:24:29	11000
12	12000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2004-10-05 00:04:24	2004-05-19 12:49:02	12000
13	13000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2004-10-28 09:19:39	2004-05-31 01:57:57	13000
14	14000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2004-11-20 13:05:30	2004-06-11 15:31:03	14000
15	15000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2004-12-14 10:05:24	2004-06-23 04:46:32	15000
16	16000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2005-01-06 07:56:08	2004-07-04 19:17:58	16000
17	17000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2005-01-29 23:22:40	2004-07-16 09:51:57	17000
18	18000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2005-02-21 19:27:25	2004-07-28 00:26:21	18000
19	19000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2005-03-17 10:09:06	2004-08-08 15:25:46	19000
20	20000	1	0,08	0,50	3	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:09:45	2005-04-10 23:25:23	2004-08-20 07:44:17	20000

Tab. B.3. Parametry wektora v_c dla przebiegów wsadowych metody APS dla jednorodnego rozkładu reguł w zbiorze testowym.

6.2.4 Pomiary czasu trwania przebiegów wsadowych

W Tab. B.4. umieszczone są wyniki pomiarów czasu trwania (w milisekundach) poszczególnych etapów przebiegów wsadowych na zbiorze T9.I4.D20K.

id_c	t_{select}	$t_{convert}$	t_{fill}	t_{elim}	t_{mine}	t_{add}	t_{del}	t_{sum}
1	70	210	400	10	891	1892	20	3495
2	60	310	1041	30	1291	2303	30	5067
3	60	480	1752	50	1942	2082	50	6419
4	60	690	2203	60	2273	2213	90	7590
5	70	831	2834	80	2784	1972	110	8682
6	60	971	3485	90	3264	1992	110	9974
7	60	1171	4095	120	3855	1792	210	11306
8	60	1341	4716	120	5107	2223	160	13729
9	60	1482	5197	140	5087	2042	140	14150
10	70	1652	5648	150	5728	1732	210	15191
11	50	1812	6279	160	6269	1642	751	16964
12	100	2002	6960	200	6719	1882	640	18506
13	60	2103	7500	200	7310	1832	260	19267
14	60	2313	7951	230	7881	1732	290	20459
15	60	2503	8642	230	8472	2473	1341	23724
16	60	2623	10274	270	9022	2784	510	25546
17	60	2763	10595	260	9643	2153	1582	27058
18	60	3034	11656	270	10124	1722	460	27329
19	60	3074	12247	310	10865	1812	2673	31044
20	60	3474	12147	390	11326	1972	430	29802

Tab. B.4. Pomiary czasu trwania poszczególnych etapów przebiegów wsadowych metody APS dla jednorodnego rozkładu reguł w zbiorze testowym.

6.2.5 Porównanie zbiorów reguł uzyskanych inkrementacyjnie i wsadowo

W Tab. B.5. zestawione są obliczenia liczbowych miar porównania zbiorów reguł, które zostały otrzymane inkrementacyjnie i wsadowo po każdym przebiegu, na zbiorze T9.I4.D20K. Jednostką odniesienia dla średniego odchylenia czasowego $time_{dev}$ jest 1 doba (24 godziny).

id_c	b_g	$rule_{overlap}$	$sup_{overlap}$	$con_{overlap}$	$time_{dev}$
1	1000	1,00	1,00	1,00	0,00
2	2000	1,00	1,00	1,00	0,00
3	3000	1,00	1,00	1,00	0,00
4	4000	1,00	1,00	1,00	0,00
5	5000	1,00	1,00	1,00	0,00
6	6000	1,00	1,00	1,00	0,00
7	7000	1,00	1,00	1,00	0,00
8	8000	1,00	1,00	1,00	0,01
9	9000	1,00	1,00	1,00	0,01
10	10000	1,00	1,00	1,00	0,01
11	11000	1,00	1,00	1,00	0,01
12	12000	1,00	1,00	1,00	0,01
13	13000	1,00	1,00	1,00	0,01
14	14000	1,00	1,00	1,00	0,01
15	15000	1,00	1,00	0,99	0,01
16	16000	1,00	1,00	1,00	0,01
17	17000	1,00	1,00	1,00	0,01
18	18000	1,00	1,00	1,00	0,01
19	19000	1,00	1,00	1,00	0,01
20	20000	1,00	1,00	1,00	0,02
	Średnio:	**1,00**	**1,00**	**1,00**	**0,01**

Tab. B.5. Wyniki porównania zbiorów reguł uzyskanych inkrementacyjnie i wsadowo przy jednorodnym rozkładzie reguł w zbiorze testowym.

6.3. Badania przy niejednorodnym rozkładzie reguł – Seria I

6.3.1 Parametry przebiegów inkrementacyjnych

W Tab. B.6. umieszczone są parametry przebiegów inkrementacyjnych na połączonym zbiorze T9.I4.D20K + T10.I5.D20K (Seria I).

id_c	b_c	η_c	σ_c	γ_c	m_x	m_y	X_c	Y_c	$\hat{\sigma}_c$	$\hat{\gamma}_c$	t_{sc}	t_{ec}	t_{mc}	k_{ec}
1	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2004-01-24 15:34:57	2004-01-12 23:12:27	1000
2	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-24 16:23:59	2004-02-18 14:49:09	2004-02-05 22:02:39	2000
3	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-02-18 15:20:48	2004-03-14 00:13:00	2004-03-01 19:33:29	3000
4	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-03-14 00:28:43	2004-04-06 02:09:12	2004-03-26 02:24:52	4000
5	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-04-06 04:07:30	2004-04-26 23:29:11	2004-04-16 15:21:48	5000
6	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-04-27 00:23:20	2004-05-19 23:07:01	2004-05-08 15:30:41	6000
7	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-05-19 23:35:25	2004-06-12 10:25:03	2004-05-31 19:21:54	7000
8	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-06-12 10:31:15	2004-07-05 21:44:13	2004-06-24 09:25:26	8000
9	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-07-05 21:47:23	2004-07-27 18:26:08	2004-07-16 22:06:48	9000
10	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-07-27 19:30:30	2004-08-20 08:47:21	2004-08-08 22:11:55	10000
11	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-08-20 10:09:40	2004-09-12 23:02:20	2004-09-01 04:21:07	11000
12	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-09-12 23:02:59	2004-10-07 01:13:47	2004-09-25 00:49:05	12000
13	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-10-07 01:53:56	2004-10-30 14:06:30	2004-10-18 19:44:40	13000
14	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-10-30 15:27:07	2004-11-22 06:12:02	2004-11-11 06:55:43	14000
15	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-11-22 07:27:35	2004-12-16 02:37:17	2004-12-04 12:30:17	15000
16	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-12-16 03:24:18	2005-01-08 06:53:51	2004-12-27 23:26:21	16000
17	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2005-01-08 06:57:47	2005-01-31 21:08:02	2005-01-19 12:51:18	17000
18	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2005-01-31 22:33:23	2005-02-24 09:02:52	2005-02-12 13:53:18	18000
19	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2005-02-24 10:27:36	2005-03-19 14:53:11	2005-03-08 03:02:06	19000
20	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2005-03-19 14:54:36	2005-04-10 23:32:57	2005-03-30 17:10:45	20000

Tab. B.6. Parametry wektora v_c dla przebiegów inkrementacyjnych metody APS dla niejednorodnego rozkładu reguł w zbiorze testowym (Seria I).

6.3.2 Pomiary czasu trwania przebiegów inkrementacyjnych

W Tab. B.7. umieszczone są wyniki pomiarów czasu trwania (w milisekundach) poszczególnych etapów przebiegów inkrementacyjnych na połączonym zbiorze T9.I4.D20K + T10.I5.D20K (Seria I).

id_c	t_{select}	$t_{convert}$	t_{fill}	t_{elim}	t_{mine}	t_{add}	t_{del}	t_{sum}
1	60	190	490	20	771	2012	30	3575
2	0	180	751	20	781	2193	20	3945
3	0	240	520	10	861	3705	20	5357
4	10	180	500	20	1422	5177	50	7360
5	10	190	520	10	1422	4696	20	6869
6	10	170	580	20	640	2523	20	3965
7	0	180	490	20	640	2573	20	3925
8	0	240	500	10	781	4176	20	5728
9	10	200	540	20	871	3304	30	4977
10	10	180	741	20	761	3114	20	4846
11	0	180	520	10	650	2233	60	3655
12	10	210	500	10	781	2283	20	3815
13	0	200	811	10	771	3284	20	5097
14	0	160	580	20	801	3094	20	4676
15	30	170	650	10	811	2894	20	4586
16	0	240	490	20	781	2864	20	4416
17	10	180	490	20	670	2062	20	3454
18	0	220	670	20	841	4135	20	5908
19	10	160	781	20	1432	3605	20	6028
20	10	180	520	10	1361	3044	20	5147

Tab. B.7. Pomiary czasu trwania poszczególnych etapów przebiegów inkrementacyjnych metody APS dla niejednorodnego rozkładu reguł w zbiorze testowym (Seria I).

6.3.3 Parametry przebiegów wsadowych

W Tab. B.8. umieszczone są parametry przebiegów wsadowych na połączonym zbiorze T9.I4.D20K + T10.I5.D20K (Seria I).

id_c	b_c	η_c	σ_c	γ_c	m_x	m_y	X_c	Y_c	$\hat{\sigma}_c$	$\hat{\gamma}_c$	t_{sc}	t_{ec}	t_{mc}	k_{ec}
1	1000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2004-01-24 15:34:57	2004-01-12 23:12:27	1000
2	2000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2004-02-18 14:49:09	2004-01-24 22:37:33	2000
3	3000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2004-03-14 00:13:00	2004-02-06 05:36:11	3000
4	4000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2004-04-06 02:09:12	2004-02-18 10:48:22	4000
5	5000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2004-04-26 23:29:11	2004-03-01 02:07:03	5000
6	6000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2004-05-19 23:07:01	2004-03-12 12:20:59	6000
7	7000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2004-06-12 10:25:03	2004-03-23 23:38:16	7000
8	8000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2004-07-05 21:44:13	2004-04-04 12:51:39	8000
9	9000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2004-07-27 18:26:08	2004-04-16 00:33:20	9000
10	10000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2004-08-20 08:47:21	2004-04-27 12:19:12	10000
11	11000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2004-09-12 23:02:20	2004-05-09 00:41:11	11000
12	12000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2004-10-07 01:13:47	2004-05-20 14:41:51	12000
13	13000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2004-10-30 14:06:30	2004-06-01 05:51:18	13000
14	14000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2004-11-22 06:12:02	2004-06-12 21:21:37	14000
15	15000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2004-12-16 02:37:17	2004-06-24 12:46:11	15000
16	16000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2005-01-08 06:53:51	2004-07-06 04:26:12	16000
17	17000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2005-01-31 21:08:02	2004-07-17 19:02:58	17000
18	18000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2005-02-24 09:02:52	2004-07-29 10:45:46	18000
19	19000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2005-03-19 14:53:11	2004-08-10 02:46:38	19000
20	20000	1	0,08	0,50	4	2	Term	Eval,Stored	0,04	0,25	2004-01-01 00:27:45	2005-04-10 23:32:57	2004-08-21 17:53:50	20000

Tab. B.8. Parametry wektora v_c dla przebiegów wsadowych metody APS dla niejednorodnego rozkładu reguł w zbiorze testowym (Seria I).

6.3.4 Pomiary czasu trwania przebiegów wsadowych

W Tab. B.9. umieszczone są wyniki pomiarów czasu trwania (w milisekundach) poszczególnych etapów przebiegów wsadowych na połączonym zbiorze T9.I4.D20K + T10.I5.D20K (Seria I).

id_c	t_{select}	$t_{convert}$	t_{fill}	t_{elim}	t_{mine}	t_{add}	t_{del}	t_{sum}
1	60	180	550	20	1011	2113	10	3945
2	60	330	1321	40	1371	2343	40	5507
3	100	600	1902	60	1802	2263	60	6789
4	60	861	2633	60	2483	3965	80	10144
5	60	1071	3184	130	3595	4656	130	12828
6	80	1371	4075	90	4035	5467	110	15231
7	60	1492	5097	150	3975	2964	130	13869
8	60	1792	5367	130	4586	3745	140	15822
9	60	1922	6138	180	5968	4065	160	18496
10	50	2183	6719	160	6769	4756	170	20809
11	60	2393	7200	200	7530	3705	340	21430
12	50	2733	8171	190	7230	3344	370	22091
13	60	2794	8862	240	8021	3274	390	23643
14	90	3124	9383	210	9573	3875	370	26628
15	90	3264	10845	240	10244	4246	490	29422
16	60	3424	11616	290	10895	5017	470	31775
17	60	3615	12287	260	10925	4216	330	31695
18	60	4075	13889	400	11937	3565	440	34369
19	60	4155	14641	330	13038	4105	470	36802
20	80	4376	14580	340	13679	4786	420	38265

Tab. B.9. Pomiary czasu trwania poszczególnych etapów przebiegów wsadowych metody APS dla niejednorodnego rozkładu reguł w zbiorze testowym (Seria I).

6.3.5 Porównanie zbiorów reguł uzyskanych inkrementacyjnie i wsadowo

W Tab. B.10. zestawione są obliczenia liczbowych miar porównania zbiorów reguł, które zostały otrzymane inkrementacyjnie i wsadowo po każdym przebiegu, na połączonym zbiorze T9.I4.D20K + T10.I5.D20K (Seria I). Jednostką odniesienia średniego odchylenia czasowego $time_{dev}$ jest 1 doba (24 godziny).

id_c	b_g	$rule_{overlap}$	$sup_{overlap}$	$con_{overlap}$	$time_{dev}$
1	1000	1,00	1,00	1,00	0,00
2	2000	1,00	1,00	1,00	0,00
3	3000	1,00	1,00	0,96	0,00
4	4000	0,74	0,99	0,81	0,00
5	5000	0,40	0,99	0,77	0,00
6	6000	0,59	0,99	0,79	0,00
7	7000	0,79	0,99	0,82	0,00
8	8000	0,74	0,99	0,82	0,01
9	9000	0,52	0,99	0,80	0,01
10	10000	0,41	0,99	0,77	0,01
11	11000	0,52	0,98	0,79	0,01
12	12000	0,58	0,98	0,81	0,01
13	13000	0,56	0,98	0,81	0,01
14	14000	0,50	0,98	0,79	0,01
15	15000	0,41	0,98	0,77	0,01
16	16000	0,48	0,98	0,78	0,01
17	17000	0,54	0,99	0,79	0,01
18	18000	0,53	0,99	0,80	0,01
19	19000	0,47	0,98	0,78	0,01
20	20000	0,42	0,98	0,77	0,01
	Średnio:	**0,61**	**0,99**	**0,82**	**0,01**

Tab. B.10. Wyniki porównania zbiorów reguł uzyskanych inkrementacyjnie i wsadowo przy niejednorodnym rozkładzie reguł w zbiorze testowym (Seria I).

6.4. Badania przy niejednorodnym rozkładzie reguł – Seria II

6.4.1 Parametry przebiegów inkrementacyjnych

W Tab. B.11. umieszczone są parametry przebiegów inkrementacyjnych na połączonym zbiorze T9.I4.D20K + T10.I5.D20K (Seria II).

id_c	b_c	η_c	σ_c	γ_c	m_x	m_y	X_c	Y_c	$\hat{\sigma}_c$	$\hat{\gamma}_c$	t_{sc}	t_{ec}	t_{mc}	k_{ec}
1	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2004-01-24 15:34:57	2004-01-12 23:12:27	1000
2	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-24 16:23:59	2004-02-18 14:49:09	2004-02-05 22:02:39	2000
3	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-02-18 15:20:48	2004-03-14 00:13:00	2004-03-01 19:33:29	3000
4	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-03-14 00:28:43	2004-04-06 02:09:12	2004-03-26 02:24:52	4000
5	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-04-06 04:07:30	2004-04-26 23:29:11	2004-04-16 15:21:48	5000
6	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-04-27 00:23:20	2004-05-19 23:07:01	2004-05-08 15:30:41	6000
7	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-05-19 23:35:25	2004-06-12 10:25:03	2004-05-31 19:21:54	7000
8	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-06-12 10:31:15	2004-07-05 21:44:13	2004-06-24 09:25:26	8000
9	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-07-05 21:47:23	2004-07-27 18:26:08	2004-07-16 22:06:48	9000
10	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-07-27 19:30:30	2004-08-20 08:47:21	2004-08-08 22:11:55	10000
11	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-08-20 10:09:40	2004-09-12 23:02:20	2004-09-01 04:21:07	11000
12	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-09-12 23:02:59	2004-10-07 01:13:47	2004-09-25 00:49:05	12000
13	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-10-07 01:53:56	2004-10-30 14:06:30	2004-10-18 19:44:40	13000
14	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-10-30 15:27:07	2004-11-22 06:12:02	2004-11-11 06:55:43	14000
15	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-11-22 07:27:35	2004-12-16 02:37:17	2004-12-04 12:30:17	15000
16	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-12-16 03:24:18	2005-01-08 06:53:51	2004-12-27 23:26:21	16000
17	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2005-01-08 06:57:47	2005-01-31 21:08:02	2005-01-19 12:51:18	17000
18	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2005-01-31 22:33:23	2005-02-24 09:02:52	2005-02-12 13:53:18	18000
19	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2005-02-24 10:27:36	2005-03-19 14:53:11	2005-03-08 03:02:06	19000
20	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2005-03-19 14:54:36	2005-04-10 23:32:57	2005-03-30 17:10:45	20000

Tab. B.11. Parametry wektora v_c dla przebiegów inkrementacyjnych metody APS dla niejednorodnego rozkładu reguł w zbiorze testowym (Seria II).

6.4.2 Pomiary czasu trwania przebiegów inkrementacyjnych

W Tab. B.12. umieszczone są wyniki pomiarów czasu trwania (w milisekundach) poszczególnych etapów przebiegów inkrementacyjnych na połączonym zbiorze T9.I4.D20K + T10.I5.D20K (Seria II).

id_c	t_{select}	$t_{convert}$	t_{fill}	t_{elim}	t_{mine}	t_{add}	t_{del}	t_{sum}
1	60	140	650	20	5517	34409	20	40818
2	10	160	731	20	5507	45725	50	52205
3	10	290	510	20	3114	26788	110	30844
4	0	190	550	10	20910	69900	20	91581
5	10	260	520	20	21460	66355	20	88647
6	10	160	510	30	5988	29091	20	35811
7	10	210	510	30	5718	24625	20	31124
8	10	230	520	20	3394	10785	20	14981
9	0	180	520	10	20669	52725	20	74126
10	10	170	550	30	21480	46416	20	68678
11	10	200	510	30	6569	29011	10	36342
12	0	170	530	20	5958	25005	20	31705
13	10	190	500	30	3444	13459	10	17645
14	10	180	550	30	23273	58093	20	82158
15	0	210	550	20	21781	51514	20	74096
16	10	170	500	20	5978	30063	20	36762
17	10	210	480	30	5547	24555	20	30854
18	0	270	510	60	3444	11937	30	16253
19	0	220	540	20	23794	58484	60	83119
20	0	220	550	30	19978	52946	60	73786

Tab. B.12. Pomiary czasu trwania poszczególnych etapów przebiegów inkrementacyjnych metody APS dla niejednorodnego rozkładu reguł w zbiorze testowym (Seria II).

6.4.3 Parametry przebiegów wsadowych

W Tab. B.13. umieszczone są parametry przebiegów wsadowych na połączonym zbiorze T9.I4.D20K + T10.I5.D20K (Seria II).

id_c	b_c	η_c	σ_c	γ_c	m_x	m_y	X_c	Y_c	$\hat{\sigma}_c$	$\hat{\gamma}_c$	t_{sc}	t_{ec}	t_{mc}	k_{ec}
1	1000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2004-01-24 15:34:57	2004-01-12 23:12:27	1000
2	2000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2004-02-18 14:49:09	2004-01-24 22:37:33	2000
3	3000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2004-03-14 00:13:00	2004-02-06 05:36:11	3000
4	4000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2004-04-06 02:09:12	2004-02-18 10:48:22	4000
5	5000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2004-04-26 23:29:11	2004-03-01 02:07:03	5000
6	6000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2004-05-19 23:07:01	2004-03-12 12:20:59	6000
7	7000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2004-06-12 10:25:03	2004-03-23 23:38:16	7000
8	8000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2004-07-05 21:44:13	2004-04-04 12:51:39	8000
9	9000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2004-07-27 18:26:08	2004-04-16 00:33:20	9000
10	10000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2004-08-20 08:47:21	2004-04-27 12:19:12	10000
11	11000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2004-09-12 23:02:20	2004-05-09 00:41:11	11000
12	12000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2004-10-07 01:13:47	2004-05-20 14:41:51	12000
13	13000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2004-10-30 14:06:30	2004-06-01 05:51:18	13000
14	14000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2004-11-22 06:12:02	2004-06-12 21:21:37	14000
15	15000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2004-12-16 02:37:17	2004-06-24 12:46:11	15000
16	16000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2005-01-08 06:53:51	2004-07-06 04:26:12	16000
17	17000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2005-01-31 21:08:02	2004-07-17 19:02:58	17000
18	18000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2005-02-24 09:02:52	2004-07-29 10:45:46	18000
19	19000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2005-03-19 14:53:11	2004-08-10 02:46:38	19000
20	20000	1	0,04	0,30	4	2	Term	Eval,Stored	0,02	0,15	2004-01-01 00:27:45	2005-04-10 23:32:57	2004-08-21 17:53:50	20000

Tab. B.13. Parametry wektora v_c dla przebiegów wsadowych metody APS dla niejednorodnego rozkładu reguł w zbiorze testowym (Seria II).

6.4.4 Pomiary czasu trwania przebiegów wsadowych

W Tab. B.14. umieszczone są wyniki pomiarów czasu trwania (w milisekundach) poszczególnych etapów przebiegów wsadowych na połączonym zbiorze T9.I4.D20K + T10.I5.D20K (Seria II).

id_c	t_{select}	$t_{convert}$	t_{fill}	t_{elim}	t_{mine}	t_{add}	t_{del}	t_{sum}
1	80	260	500	30	5748	33568	20	40207
2	70	340	1301	50	7220	35010	50	44043
3	60	590	2032	50	5297	15251	60	23343
4	60	831	2653	90	3925	9403	70	17034
5	60	1111	3234	130	5858	7450	100	17945
6	60	1261	3955	110	5457	6509	170	17525
7	60	1452	4566	180	6269	6529	160	19217
8	80	1662	5357	140	7020	7651	180	22091
9	60	1942	6088	290	8021	6048	160	22612
10	60	2113	6499	200	7751	5708	330	22662
11	50	2373	7290	410	9443	5648	310	25526
12	60	2573	8071	240	10965	7550	330	29792
13	60	2994	8992	300	11306	6168	320	30143
14	60	3034	9523	320	10905	5688	380	29913
15	50	3434	10094	310	11746	5938	410	31985
16	60	3354	11997	310	12417	7260	430	35831
17	60	3615	13269	270	14871	6299	460	38845
18	70	3835	13930	410	15462	5778	450	39937
19	70	4055	14661	370	14761	5758	450	40127
20	60	4336	15141	360	15822	7140	430	43292

Tab. B.14. Pomiary czasu trwania poszczególnych etapów przebiegów wsadowych metody APS dla niejednorodnego rozkładu reguł w zbiorze testowym (Seria II).

6.4.5 Porównanie zbiorów reguł uzyskanych inkrementacyjnie i wsadowo

W Tab. B.15. zestawione są obliczenia liczbowych miar porównania zbiorów reguł, które zostały otrzymane inkrementacyjnie i wsadowo po każdym przebiegu, na połączonym zbiorze T9.I4.D20K + T10.I5.D20K (Seria II). Jednostką odniesienia średniego odchylenia czasowego $time_{dev}$ jest 1 doba (24 godziny).

id_c	b_g	$rule_{overlap}$	$sup_{overlap}$	$con_{overlap}$	$time_{dev}$
1	1000	1,00	1,00	1,00	0,00
2	2000	0,74	1,00	0,98	0,00
3	3000	0,56	1,00	0,68	0,00
4	4000	0,65	0,99	0,69	0,00
5	5000	0,76	0,99	0,75	0,00
6	6000	1,00	0,99	0,71	0,00
7	7000	0,94	0,99	0,70	0,00
8	8000	0,94	0,99	0,72	0,01
9	9000	1,00	0,99	0,73	0,01
10	10000	0,93	0,99	0,76	0,01
11	11000	0,93	0,99	0,75	0,01
12	12000	0,94	0,99	0,71	0,01
13	13000	1,00	0,99	0,73	0,01
14	14000	0,93	0,99	0,75	0,01
15	15000	0,93	0,99	0,76	0,01
16	16000	0,93	0,99	0,75	0,01
17	17000	0,93	0,99	0,74	0,01
18	18000	0,93	0,99	0,75	0,01
19	19000	0,93	0,99	0,76	0,01
20	20000	0,93	0,99	0,76	0,01
	Średnio:	**0,90**	**0,99**	**0,76**	**0,01**

Tab. B.15. Wyniki porównania zbiorów reguł uzyskanych inkrementacyjnie i wsadowo przy niejednorodnym rozkładzie reguł w zbiorze testowym (Seria II).

7. Literatura

[Aba1994] ABADI M., HALPERN J.Y., *Decidability and expressiveness for first-order logics of probability*, Information and Computation, Vol. 112, No. 1, 1994, pp. 1–36.

[Agg2001] AGGARWAL C.C., YU P.S., *A New Approach to Online Generation of Association Rules*, IEEE Transactions On Knowledge And Data Engineering, Vol. 13, No. 4, 2001, pp. 527–540.

[Agr1993] AGRAWAL R., IMIELINSKI T., SWAMI A., *Mining association rules between sets of items in large databases*, Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93), May 1993, pp. 207–216.

[Agr1994] AGRAWAL R., SRIKANT R., *Fast Algorithms for Mining Association Rules*, Proceedings of the Twentieth International Conference on Very Large Databases, Santiago, Chile, September 1994.

[Agr1995] AGRAWAL R., PSAILA G., *Active Data Mining*, Proceedings of the First International Conference on Knowledge Discovery and Data Mining, Montreal, August 1995.

[Alo2001] ALONSO E., KUDENKO D., *Machine Learning for Logic-Based Multi-Agent Systems*, [in:] RASH J.L. et al. (Eds.), FAABS 2000, Lecture Notes in Artificial Intelligence, Vol. 1871, Springer-Verlag, Berlin Heidelberg 2001, pp. 306–307.

[AuC2005] AU W.-H., CHAN K.C.C., *Mining changes in association rules: a fuzzy approach*, Fuzzy Sets and Systems, Vol. 149, 2005, pp. 87–104.

[Aum1999] AUMANN Y., FELDMAN R., LIPSHTAT O., MANILLA H., *Borders: An Efficient Algorithm for Association Generation in Dynamic Databases*, Journal of Intelligent Information Systems, Vol. 21, 1999, pp. 61–73.

[Aum2003] AUMANN Y., LINDELL Y., *A Statistical Theory for Quantitative Association Rules*, Journal of Intelligent Information Systems, Vol. 20, No. 3, 2003, pp. 255–283.

[App2000] APPLEBY S., STEWARD S., *Mobile software agents for control in telecommunications network*, BT Technology Journal, Vol. 18, No. 1, January 2000, pp. 68–70.

[Bac1990] BACCHUS F., *Lp, A Logic for Representing and Reasoning with Statistical Knowledge*, Computational Intelligence, Vol. 6, 1990, pp. 209–231.

[Bac1996] BACCHUS F., GROVE A., HALPERN J., KOLLER D., *From statistical knowledge bases to degrees of belief*, Artificial Intelligence, Vol. 87, No. 1–2, 1996, pp. 75–143.

[Bac1999] BACCHUS F., HALPERN J., LEVESQUE H., *Reasoning about noisy sensors and effectors in the situation calculus*, Artificial Intelligence, Vol. 111, 1999, pp. 171–208.

[Bay1999] BAYARDO JR. R. J., AGRAWAL R., *Mining the Most Interesting Rules*, Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 1999.

[Bay2000] BAYARDO JR. R.J., AGRAWAL R., GUNOPULOS D., *Constraint-Based Rule Mining in Large, Dense Databases*, Data Mining and Knowledge Discovery Journal, Vol. 4, No. 2/3, 2000, pp. 217–240.

[Ber1999] BERGADANO F., PULIAFITO A., RICCOBENE S., RUFFO G., VITA L., *Java-based and secure learning agents for information retrieval in distributed systems*, Information Sciences, Vol. 113, 1999, pp. 55–84.

[Bol1991] BOLC L., BORODZIEWICZ W., WÓJCIK M., *Podstawy przetwarzania informacji niepewnej i niepełnej*, Państwowe Wydawnictwo Naukowe, Warszawa 1991.

[Bra1997] BRADSHAW J., *An introduction to software agents*, [in:] BRADSHAW J. (ed.) *Software agents*, MIT Press, 1997, pp. 3–46.

[Brg2003] BRAGT VAN D.D., LA POUTRÉ J.A., *Why agents for automated negotiations should be adaptive*, Netnomics, Vol. 5, 2003, pp. 101–118.

[Brt1987] BRATMAN M., *Intentions, Plans, and Practical Reason*, Harvard University Press, Cambridge, MA, 1987.

[Bub1993] BUBNICKI Z., *Podstawy informatycznych systemów zarządzania*, Wydawnictwo Politechniki Wrocławskiej, Wrocław 1993.

[Car2000] CARDOSO H. L., OLIVEIRA E., *Using and Evaluating Adaptive Agents for Electronic Commerce Negotiation*, [in:] MONARD M.C., SICHMAN J.S. (Eds.), *Lecture Notes in Artificial Intelligence*, Vol. 1952, Springer-Verlag, Berlin Heidelberg 2000, pp. 96–105.

[Crm1999] CARMEL D., MARKOVITCH S., *Exploration Strategies for Model-based Learning in Multi-agent Systems*, Autonomous Agents and Multi-Agent Systems, Vol. 2, 1999, pp. 141–172.

[Ces1999] CESTA A., D'ALOISI D., *Mixed-Initiative Issues in an Agent-Based Meeting Scheduler*, User Modeling and User-Adapted Interaction, Vol. 9, 1999, pp. 45–78.

[ChC2002] CHEN C. C., CHEN M. C., SUN Y., *PVA: A Self-Adaptive Personal View Agent*, Journal of Intelligent Information Systems, Vol. 18, No. 2/3, 2002, pp. 173–194.

[Che1996] CHEUNG D.W., NG V.T., TAM B.W., *Maintenance of Discovered Knowledge: A Case in Multi-level Association Rules*, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996, pp. 307–310.

[Che1997] CHEUNG D.W., LEE S.D., KAO B., *A general incremental technique for maintaining discovered association rules*, Proceedings of the Fifth International Conference on Database Systems for Advanced Applications, Melbourne, Australia, April 1997, pp. 185–194.

[ChL2002] CHEN Z., LIN F., LIU H., LIU Y., MA W.-Y., WENYIN L., *User Intention Modeling in Web Applications Using Data Mining*, World Wide Web: Internet and Web Information Systems, Vol. 5, 2002, pp. 181–191.

[ChW2002] CHEN G., WEI Q., LIU D., WETS G., *Simple association rules (SAR) and the SAR-based rule discovery*, Computers & Industrial Engineering, Vol. 43, 2002, pp. 721–733.

[ChY2005] CHEN G., YANG Z., HE H., GOH K.M., *Coordinating Multiple Agents via Reinforcement Learning*, Autonomous Agents and Multi-Agent Systems, Vol. 10, 2005, pp. 273–328.

[Cic2000] CICHOSZ P., *Systemy uczące się*, Wydawnictwa Naukowo-Techniczne, Warszawa 2000.

[Coe2004] COENEN F., GOULBOURNE G., LENG P., *Tree Structures for Mining Association Rules*, Data Mining and Knowledge Discovery, Vol. 8, 2004, pp. 25–51.

[Coh1990] COHEN P. R., LEVESQUE H. J., *Intention Is Choice with Commitment*, Artificial Intelligence, 1990, Vol. 42, No. 3, pp. 213–261.

[Col2003] COLE J., LLOYD J., NG K.S., *Symbolic Learning for Adaptive Agents*, Technical Note, Computer Sciences Laboratory, The Australian National University, Canberra, Australia, 2003.

[Cri1998] CRITES R.H., BARTO A.G., *Elevator Group Control Using Mutiple Reinforcement Learning Agents*, Machine Learning, Vol. 33, 1998, pp. 235–262.

[Dan2002] DANIŁOWICZ C., NGUYEN N.T., JANKOWSKI Ł., *Metody wyboru reprezentacji stanu wiedzy agentów w systemach multiagenckich*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2002.

[Dav1998] DAVISON R. G., HARDWICKE J. J., COX M. D. J., *Applying the agent paradigm to network menagement*, BT Technology Journal, Vol. 16, No. 3, July 1998, pp. 86–93.

[Dec1997] DECKER K. S., SYCARA K., *Intelligent Adaptive Information Agents*, Journal of Intelligent Information Systems, Vol. 9, 1997, pp. 239–260.

[Del2001] DELOACH S. A., *Analysis and Design using MaSE and agentTool*, Proceedings of the Twelth Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001), Miami University, Oxford, Ohio, March – April 2001.

[Dem2001] DEMOLOMBE R., LIAU C. J., *A logic of graded trust and belief fusion*, Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies, 2001, pp. 13–25.

[Deo1997] DEOGUN J.S., RAGHAVAN V.V., SARKAR A., SEVER H., *Data mining: Research trends, challenges, and applications*, [in:] Lin T. Y. , Cercone N., (Eds.) *Roughs Sets and Data Mining: Analysis of Imprecise Data*, Kluwer Academic Publishers, Boston 1997, pp. 9–45.

[Die1997] DIETTERICH T.G., *Machine Learning Research: Four Current Directions*, AI Magazine, Vol. 18, No. 4, 1997, pp. 97-136.

[Die2003] DIETTERICH T.G., *Machine Learning* [in:] Nature Encyclopedia of Cognitive Science, Macmillan, London 2003 (manuscript to appear).

[DIn1998] D'INVERNO M., KINNY D., LUCK M., WOOLDRIDGE M., *A Formal Specification of dMARS*, [in:] SINGH M., RAO A., WOOLDRIDGE M. (Eds.) *Intelligent Agents IV Springer-Verlag Lecture Notes in AI*, Vol. 1365, February 1998.

[Dor1994] DORIGO M., BERSINI H., *A Comparison of Q-Learning and Classifier Systems.*, [in:] CLIFF D., HUSBANDS P., MEYER J.-A., WILSON S.W. (Eds.), Proceedings of From Animals to Animats, Third International Conference on Simulation of Adaptive Behavior (SAB94), Brighton, UK, MIT Press, 1994, pp. 248–255.

[Dru2000] DRUMMOND C., IONESCU D., HOLTE R., *A Learning Agent that Assists the Browsing of Software Libraries*, IEEE Transactions on Software Engineering, Vol. 26, No. 12 , 2000, pp. 1179–1196.

[Dud1999] DUDEK D., KATARZYNIAK R., *Implementing a Prototype of an Intelligent Time Organizer Using The SOAR Cognitive Architecture*, Proceedings of the Twenty First International Conference on Information Systems Architecture and Technology (ISAT 1999), Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 1999, pp. 261–267.

[Dud2000] DUDEK D., ZGRZYWA A., *Modelowanie programów agenckich z wykorzystaniem architektury Belief-Desire-Intention*, [in:] DANIŁOWICZ C. (Ed.), Multimedialne i Sieciowe Systemy Informacyjne (MiSSI 2000), Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2000, pp. 49–58.

[Dud2001] DUDEK D., ZGRZYWA A., *Uncertain knowledge representation within the Belief-Desire-Intention agent system*, [in:] GRZECH A., WILIMOWSKA Z. (Eds.), Proceedings of the Twenty Third International Scientific School on Information Systems Architecture and Technology (ISAT 2001), Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2001, pp. 147–154.

[Dud2002] DUDEK D., ZGRZYWA A., *Pozyskiwanie wiedzy w systemie agenckim BDI na podstawie analizy przeszłych stanów świata*, [in:] DANIŁOWICZ C. (Ed.), Multimedialne i Sieciowe Systemy Informacyjne (MiSSI 2002), Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2002, pp. 79–88.

[Dud2003] DUDEK D., ZGRZYWA A., *Uczenie się systemu agenckiego Belief-Desire-Intention metodą APS*, [in:] BUBNICKI Z., GRZECH A. (Eds.), *Inżynieria Wiedzy i Systemy Ekspertowe*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2003, Tom 2, pp. 237–244.

[Dud2005a] DUDEK D., ZGRZYWA A., *The Incremental Method for Discovery of Association Rules*, (in:) KURZYNSKI M., PUCHALA E., WOZNIAK M., ZOLNIEREK A. (Eds.), *Proceedings of the Fourth International Conference on Computer Recognition Systems* (CORES'05), Advances in Soft Computing, Springer-Verlag, Berlin Heidelberg 2005, pp. 153–160.

[Dud2005b] DUDEK D., KUBISZ M., ZGRZYWA A., *APS: Agent's Learning With Imperfect Recall*, Proceedings of the Fifth International Conference on Intelligent Systems Design and Applications (ISDA 2005), IEEE Computer Society Press, 2005 (to appear).

[Edw1997] EDWARDS P., GREEN C.L., LOCKIER P.C., LUKINS T.C., *Exploiting Learning Technologies for World Wide Web Agents*, IEE Colloquium on Intelligent World Wide Web Agents (Digest No: 1997/118), 17 March 1997, pp. 3/1 – 3/7.

[Eli2003] ELIASSI-RAD T., SHAVLIK J., *A System for Building Intelligent Agents that Learn to Retrieve and Extract Information*, User Modeling and User-Adapted Interaction, Vol. 13, 2003, pp. 35–88.

[Eme1990] EMERSON E.A., *Temporal and modal logic*, [in:] VAN LEEUWEN J., (Ed.) *Handbook of Theoretical Computer Science*, North-Holland Pub. Co./MIT Press, 1990, Vol. B, pp. 995–1072.

[Ene2005] ENEMBRECK F., BARTHÈS J.-P., *ELA – A New Approach for Learning Agents*, Autonomous Agents and Multi-Agent Systems, Vol. 10, 2005, pp. 215–248.

[Etz1994] ETZIONI O., WELD D., *A Softbot-Based Interface to the Internet*, Communications of the ACM, Vol. 37, No. 7, July 1994, pp. 72–76.

[Eze2002] EZEIFE C.I., SU Y., *Mining Incremental Association Rules with Generalized FP-tree*, Proceedings of the Fifhteenth Canadian Conference on Artificial Intelligence (AI 2002), Calgary, Canada (May 25-29, 2002), Lecture Notes in Computer Science (LNCS) , Springer-Verlag, Berlin Heidelberg 2002.

[Ezz2005] EZZEDINE H., KOLSKI C., PÉNINOU A., *Agent-oriented design of human-computer interface: application to supervision of an urban transport network*, Engineering Applications of Artificial Intelligence, Vol. 18, 2005, pp. 255–270.

[Fac2005] FACCA F.M., LANZI P.L., *Mining interesting knowledge from weblogs: a survey*, Data & Knowledge Engineering, Vol. 53, 2005, pp. 225–241.

[Fag1988] FAGIN R., HALPERN J.Y., *Belief, awareness, and limited reasoning*, Artificial Intelligence, Vol. 34, 1988, pp. 39–76.

[Fag1994] FAGIN R., HALPERN J.Y., *Reasoning about knowledge and probability*, Journal of the ACM, Vol. 41, No. 2, 1994, pp. 340–367.

[Fag1995] FAGIN R., HALPERN J.Y., MOSES Y., VARDI M.Y., *Reasoning About Knowledge*, Cambridge, MA, USA, MIT Press, 1995.

[Fay1996a] FAYYAD U., PIATETSKY-SHAPIRO G., SMYTH P., *From data mining to knowledge discovery: An overview*., [in:] FAYYAD U., PIATETSKY-SHAPIRO G., SMYTH P., UTHURUSAMY R. (Eds.), *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, Cambridge, Massachusets 1996.

[Fay1996b] FAYYAD U., PIATETSKY-SHAPIRO G., SMYTH P., *The KDD Process for Extracting Useful Knowledge from Volumes of Data*, Communications of the ACM, Vol. 39, No. 11, November 1996, pp. 27–34.

[Fon2003] FONG J., WONG H.K., HUANG S.M., *Continuous and incremental data mining association rules using frame metadata model*, Knowledge-Based Systems, Vol. 16, 2003, pp. 91–100.

[Gai1997] GAINES B.R., *Knowledge Management in Societies of Intelligent Adaptive Agents*, Journal of Intelligent Information Systems, Vol. 9, 1997, pp. 277–298.

[GaL2004] GARCIA A.F., DE LUCENA C.J.P., COWAN D.D., *Agents in object-oriented software engineering*, Software Practice and Experience, Vol. 34, 2004, pp. 489–521.

[GrM2000] GARCÍA-MARTÍNEZ R., BORRAJO D., *An Integrated Approach of Learning, Planning, and Execution*, Journal of Intelligent and Robotic Systems, Vol. 29, 2000, pp. 47–78.

[GaP1998] GARDARIN G., PUCHERAL P., WU F., *Bitmap Based Algorithms For Mining Association Rules*, Technical Report No. 1998/18, University of Versailles, Versailles Cedex, France, 1998.

[Gar2004] GARLAND A., ALTERMAN R., *Autonomous Agents that Learn to Better Coordinate*, Autonomous Agents and Multi-Agent Systems, Vol. 8, 2004, pp. 267–301.

[Geo1999] GEORGEFF M., PELL B., POLLACK M., TAMBE M., WOOLDRIDGE M., *The Belief-Desire-Intention Model of Agency* [in:] MÜLLER J. P. et al. (Eds.) *Intelligent Agents V Springer-Verlag Lecture Notes in AI*, Vol. 1365, March 1999.

[Gmy1998] GMYTRASIEWICZ P.J., NOH S., KELLOGG T., *Bayesian Update of Recursive Agent Models*, User Modeling and User-Adapted Interaction, Vol. 8, 1998. pp. 49–69.

[God2004] GODOY D., SCHIAFFINO S., AMANDI A., *Interface agents personalizing Web-based tasks*, Cognitive Systems Research, Vol. 5, 2004, pp. 207–222.

[Goe2002] GOETHALS B., *Efficient Frequent Pattern Mining*, PhD thesis, Transnational University of Limburg, Diepenbeek, Belgium, 2002.

[Goe2003] GOETHALS B., *Implementation of the Apriori Algorithm*, (http://www.adrem.ua.ac.be/~goethals/), University of Helsinki, 2003.

[Gor2001] GORDON D., *APT Agents: Agents That Are Adaptive, Predictable, and Timely*, Lecture Notes in Artificial Intelligence, Vol. 1871, Springer-Verlag, Berlin Heidelberg 2001.

[Gss2004] GUESSOUM Z., *Adaptive Agents and Multiagent Systems*, IEEE Distributed Systems Online, Vol. 5, No. 7, 2004.

[Gue2003] GUERRA HERNÁNDEZ A., *Learning in Intentional BDI Multi-Agent Systems*, PhD thesis, Université de Paris13, Institut Galilée, LIPN - Laboratoire d'Informatique de Paris Nord. Villetaneuse, France, 2003.

[Hag2002] HAGRAS H., SOBH T., *Intelligent learning and control of autonomous robotic agents operating in unstructured environments*, Information Sciences, Vol. 145, 2002, pp. 1–12.

[Haj1996] HAJNICZ E., *Reprezentacja logiczna wiedzy zmieniającej się w czasie*, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1996.

[Hal1990] HALPERN J.Y., *An Analysis of First-Order Logics of Probability*, Artificial Intelligence, Vol. 46, 1990, pp. 311–350.

[Hal1998] HALPERN J.Y., *A logical approach to reasoning about uncertainty: a tutorial,* [in:] ARRAZOLA X., KORTA K., PELLETIER F.J. (Eds.), *Discourse, Interaction, and Communication*, Kluwer, 1998, pp. 141–155.

[Hal2002] HALPERN J.Y., PUCELLA R., *Reasoning about expectation*, Proceedings of the Eighteenth Conference on Uncertainty in AI, 2002, pp. 207–215.

[Har2004] HARMS S.K., DEOGUN J.S., *Sequential Association Rule Mining with Time Lags*, Journal of Intelligent Information Systems, Vol. 22, No. 1, 2004, pp. 7–22.

[Has2001] HASTIE T., TIBSHIRANI R., FRIEDMAN J., *The Elements of Statistical Learning. Data Mining, Inference, and Prediction.*, Springer-Verlag, New York–Berlin–Heidelberg 2001.

[Hee1997] HEESEN C., HOMBURG V., OFFEREINS M., *An Agent View on Law*, Artificial Intelligence and Law, Vol. 5, 1997, pp. 323–340.

[Hua1991] HUANG Z., KWAST K.L., *Awareness, Negation and Logical Omniscience*, [in:] VAN EIJCK J. (Ed.), *Logics in AI, Proceedings of JELIA'90*, Lecture Notes in Computer Science, Vol. 478, Springer-Verlag, Berlin New York 1991, pp. 282–300.

[Hub1999] HUBER M., *JAM: A BDI-theoretic Mobile Agent Architecture*, Proceedings of the Third Annual Conference on Autonomous Agents, 1999, pp. 236–243.

[HuL1999] HU K., LU Y., SHI C.I., *Incremental Discovering Association Rules: A Concept Lattice Approach*, [in:] ZHONG N., ZHOU L. (Eds.), PAKDD'99, Lecture Notes in Artificial Intelligence (LNAI), Vol. 1571, Springer-Verlag, Berlin Heidelberg 1999, pp. 109–113.

[HuW2001] HU J., WELLMAN M.P., *Learning about other agents in a dynamic multiagent system*, Journal of Cognitive Systems Research, Vol. 2, 2001, pp. 67–79.

[Ing1992] INGRAND F., GEORGEFF M., RAO A., *An Architecture for Real-Time Reasoning and System Control*, IEEE Expert, Vol. 7, No. 6, December 1992, pp. 33–44.

[Jen1997] JENNINGS N. R., SYCARA K. P., WOOLDRIDGE M., *A Roadmap of Agent Research and Development*, Journal of Autonomous Agents and Multi-Agent Systems, Vol. 1, No. 1, July 1998, pp. 7–36.

[Jen1998] JENNINGS N., WOOLDRIDGE M., *Applications of Intelligent Agents*, [in:] JENNINGS N., WOOLDRIDGE M. (Eds.), *Agent Technology: Foundations, Applications and Markets*, Springer-Verlag, 1998, pp. 3–28.

[Kat1999] KATARZYNIAK R., *Wieloagencki system zarządzania w rozproszonych systemach przetwarzania*, Rozprawa doktorska, Technical Report PRE No. 3, Zakład Systemów Informacyjnych Politechniki Wrocławskiej, Wrocław 1999.

[Kat2002] KATARZYNIAK R., PIECZYŃSKA-KUCHTIAK A., *A consensus based algorithm for grounding belief formulas in internally stored perceptions*, Neural Network World, Vol. 12, No. 5, 2002, pp. 461–472.

[Kat2003] KATARZYNIAK R., *Grounding atom formulas and simple modalities in communicative agents*, Proceedings of the Twenty First IASTED International Conference on Artificial Intelligence, Innsbruck, Austria, 10 – 13 February 2003, pp. 388–392.

[Kay2005a] KAYA M., ALHAJJ R., *Genetic algorithm based framework for mining fuzzy association rules*, Fuzzy Sets and Systems, Vol. 152, 2005, pp. 587–601.

[Kay2005b] KAYA M., ALHAJJ R., *Fuzzy OLAP association rules mining-based modular reinforcement learning approach for multiagent systems*, IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol. 35, No. 2, 2005, pp. 326–338.

[Kay2005c] KAYA M., ALHAJJ R., *A Novel Approach to Multiagent Reinforcement Learning: Utilizing OLAP Mining in the Learning Process*, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Vol. PP, No. 99, 2005, pp. 1–8 (to appear).

[Kaz2001] KAZAKOW D., KUDENKO D., *Machine Learning and Inductive Logic Programming for Multi-Agent Systems*, [in:] LUCK M., MARIK V., STEPANKOVA O. (Eds.), Multi-Agent Systems and Applications, Lecture Notes in Artificial Intelligence (LNAI), Vol. 2086, Springer-Verlag, Berlin Heidelberg 2001, pp. 246–270.

[Kec2002] KECHAGIAS D., SYMEONIDIS A.L., MITKAS P.A., ALBORG M., *Towards Improving Multi-Agent Simulation in Safety Management and Hazard Control Environments*, Proceedings of the Conference on AI, Simulation and Planning in High Autonomous Systems (AIS'2002), Lisbon, Portugal, 7–10 April 2002.

[Kep2002] KEPHART J. O., GREENWALD A. R., *Shopbot Economics*, Autonomous Agents and Multi-Agent Systems, Vol. 5, 2002, pp. 255–287.

[Kim2002] KIM W., KERSCHBERG L., SCIME A., *Learning for automatic personalization in a semantic taxonomy-based meta-search agent*, Electronic Commerce Research and Applications, Vol. 1, 2002, pp. 150–173.

[Kmb2005] KIMBROUGH S.O., LU M., *Simple reinforcement learning agents: Pareto beats Nash in an algorithmic game theory study*, Information Systems and e-Business Management, Vol. 3, 2005, pp. 1–19.

[Kle1994] KLEMETTINEN M., MANNILA H., RONKAINEN P., TOIVONEN H., VERKAMO I., *Finding Interesting Rules from Large Sets of Discovered Association Rules*, [in:] ADAM N.R., BHARGAVA B.K., YESHA Y. (Eds.), Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94), Gaithersburg, Maryland, November - December 1994, p. 401–407.

[Klu2001] KLUSCH M., *Information agent technology for the Internet: A survey*, Data & Knowledge Engineering, Vol. 36, 2001, pp. 337–372.

[Kno1997] KNOBLOCK C.A., AMBITE J.L., *Agents for Information Gathering*, [in:] BRADSHAW J.M., *Software Agents*, AAAI Press / The MIT Press, 1997, pp. 347–374.

[Koh1998] KOHAVI R., PROVOST F., *Glossary of Terms. Special Issue on Applications of Machine Learning and the Knowledge Discovery Process*, Machine Learning, Vol. 30, 1998, pp. 271–274.

[Kri1963] KRIPKE S., *Semantical analysis of modal logic I: normal modal propositional calculi*, Zeitschrift fuer Math. Logic und Grundlagen der Math., Vol. 9, 1963, pp. 67–96.

[Kry2000] KRYSZKIEWICZ M., RYBIŃSKI H., *Legitimate Approach to Association Rules under Incompleteness*, ISMIS 2000, Charlotte, USA, 2000, pp. 505–514.

[Kry2004] KRYSZKIEWICZ M., RYBIŃSKI H., GAJEK M., *Dataless Transitions Between Concise Representations of Frequent Patterns*, Journal of Intelligent Information Systems, Vol. 22, No. 1, 2004, pp. 41–70.

[Kwa1999] KWAŚNICKA H., *Obliczenia ewolucyjne w sztucznej inteligencji*, Praca habilitacyjna, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 1999.

[Kwa2004] KWAŚNICKA H., SPIRYDOWICZ A., *Uczący się komputer. Programowanie gier logicznych*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2004.

[Lai1987] LAIRD J., NEWELL A., ROSENBLOOM P., *Soar: An Architecture for General Intelligence*, Artificial Intelligence, Vol. 33, 1987, pp.1–64.

[Lai1997] LAIRD J.E., PEARSON D.J., HUFFMAN S.B., *Knowledge-directed Adaptation in Multi-level Agents*, Journal of Intelligent Information Systems, Vol. 9, 1997, pp. 261–275.

[Lam1999] LAMMA E., RIGUZZI F., PEREIRA L.M., *Agents Learning in a Three-Valued Logical Setting*, Proceedings of the Workshop on Machine Learning and Intelligent Agents, Advanced Course on Artificial Intelligence 1999 (ACAI'99), Crete, Greece, July 1999.

[Lan1997] LANDER S.E., *Issues in Multiagent Design Systems*, IEEE Expert, March-April 1997, pp. 18–26.

[Lee1994a] LEE J., DURFEE E., *Structured Circuit Semantics for Reactive Plan Execution Systems*, Proceedings of the Twelfth National Conference on Artificial Intelligence, 1994, pp. 1232–1237.

[Lee1994b] LEE J., HUBER M., DURFEE E., KENNY P., *UM-PRS: An Implementation of the Procedural Reasoning System for Multirobot Applications*, Proceedings of the Conference on Intelligent Robotics in Field, Factory, Service and Space (CIRFFSS'94), Houston, Texas, USA, 1994, pp. 842–849.

[LeG2001] LEE G., LEE K.L., CHEN A.L.P., *Efficient Graph-Based Algorithms for Discovering and Maintaining Association Rules in Large Databases*, Knowledge and Information Systems, Vol. 3, 2001, pp. 338–355.

[LeH2005] LEE Y.-C., HONG T.-P., LIN W.-Y., *Mining association rules with multiple minimum supports using maximum constraints*, International Journal of Approximate Reasoning, Vol. 40, 2005, pp. 44–54.

[LeL2004] LEE W.-J., LEE S.-J., *Discovery of Fuzzy Temporal Association Rules*, IEEE Transactions on Systems, Man, And Cybernetics – Part B: Cybernetics, Vol. 34, No. 6, 2004, pp. 2330–2342.

[LeS1998] LEE S.D., CHEUNG D.W., KAO B., *Is Sampling Useful in Data Mining? A Case in the Maintenance of Discovered Association Rules*, Data Mining and Knowledge Discovery, Vol. 2, 1998, pp. 233–262.

[Lev1984] LEVESQUE H. J., *A Logic of Implicit and Explicit Belief*, Proceedings of the AAAI-84 Conference, Austin, Texas, USA, (wersja rozszerzona i poprawiona: *Fairchild Technical Report No. 653*), 1984.

[Lgo2004] LE GOC M., GAETA M., *Modelling Structures in Generic Space, a Condition for Adaptiveness of Monitoring Cognitive Agent*, Journal of Intelligent and Robotic Systems, Vol. 41, 2004, pp. 113–140.

[Lia2000] LIAU C. J., *Logical Systems for Reasoning about Multi-agent Belief, Information Acquisition and Trust*, Proceedings of the Fourteenth European Conference on Artificial Intelligence, 2000, pp. 368–372.

[Lie1998] LIEBERMAN H., *Integrating user interface agents with conventional applications*, Knowledge-Based Systems, Vol. 11, 1998, pp. 15–23.

[LiJ2004] LI J., SHEN H., TOPOR R., *Mining Informative Rule Set for Prediction*, Journal of Intelligent Information Systems, Vol. 22, No. 2, 2004, pp. 155–174.

[Lin2002] LIN W., ALVAREZ S.A., RUIZ C., *Efficient Adaptive-Support Association Rule Mining for Recommender Systems*, Data Mining and Knowledge Discovery, Vol. 6, 2002, pp. 83–105.

[Lis2004] LISI F.A., MALERBA D., *Inducing Multi-Level Association Rules from Multiple Relations*, Machine Learning, Vol. 55, 2004, pp. 175–210.

[LiT2000] LIU S., TURBAN E., LEE M.K.O., *Software Agents for Environmental Scanning in Electronic Commerce*, Information Systems Frontiers, Vol. 2, No. 1, 2000, pp. 85–98.

[LiY2003] LIU J., YOU J., *Smart Shopper: An Agent-Based Web-Mining Approach to Internet Shopping*, IEEE Transactions on Fuzzy Systems, Vol. 11, No. 2, April 2003, pp. 226–237.

[Mae1994a] MAES P., *Agents that Reduce Work and Information Overload*, Communications of the ACM, Vol. 37, No. 7, 1994, pp. 31–40.

[Mae1994b] MAES P., *Modeling Adaptive Autonomous Agents* (1994) [in:] Artificial Life: An Overview, MIT Press, Cambridge, MA, USA, 1994, pp.135–162.

[Mal2000] MALOOF M.A., MICHALSKI R.S., *Selecting Examples for Partial Memory Learning*, Machine Learning, Vol. 41, 2000, pp. 27–52.

[Mal2004] MALOOF M.A., MICHALSKI R.S., *Incremental learning with partial instance memory* , Artificial Intelligence, Vol. 154, 2004, pp. 95–126.

[Man1996] MANNILA H., *Data mining: machine learning, statistics, and databases*, Proceedings of the Eighth International Conference on Scientific and Statistical Database Management, Stockholm, 18-20 June 1996, pp.1–8.

[Man1997] MANNILA H., *Methods and problems in data mining. A tutorial.*, [in:] AFRATI F., KOLAITIS P. (Ed.) Proceedings of the International Conference on Database Theory (ICDT'97), Delphi, Greece, January 1997, pp. 41–55.

[Mav2004] MANVI S.S., VENKATARAM P., *Applications of agent technology in communications: a review*, Computer Communications, Vol. 27, 2004, pp. 1493–1508.

[Mar2001] MARSELLA S. et al., *Experiences Acquired in the Design of RoboCup Teams: A Comparison of Two Fielded Teams*, Autonomous Agents and Multi-Agent Systems, Vol. 4, 2001, pp. 115–129.

[Men2000] MENCZER F., BELEW R., *Adaptive Retrieval Agents: Internalizing Local Context and Scaling up to the Web*, Machine Learning, Vol. 39, 2000, pp. 203–242.

[Men2002] MENCZER F., STREET W.N., MONGE A.E., *Adaptive Assistants for Customized E-Shopping*, IEEE Intelligent Systems, November/December 2002, pp. 12–19.

[Meo1998] MEO R., PSAILA G., CERI S., *An Extension to SQL for Mining Association Rules*, Data Mining and Knowledge Discovery, Vol. 2, 1998, pp. 195–224.

[Mey2002] MEYSTEL A..M., ALBUS J.S., *Intelligent Systems. Architecture, Desing, and Control*, John Wiley & Sons, New York 2002.

[Mil2000] MILCH B., KOLLER D., *Probabilistic Models for Agents' Beliefs and Decisions*, Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-00), Stanford, California, June 2000, pp. 389–396.

[Mit2002] MITKAS P. et al., *An Agent Framework for Dynamic Agent Retraining: Agent Academy*, Proceedings of the Twelth Annual Conference and Exhibition on eBusiness and eWork (e2002), Prague, Czech Republic, 16–18 October 2002.

[Mos1986] MOSER L.E., TURNBULL A.A., *Proverbs for Programming in Pascal*, John Wiley & Sons, New York 1986.

[Muk2003] MUKHERJEE R., SAJJA N., SEN S., *A Movie Recommendation System – An Application of Voting Theory in User Modeling*, User Modeling and User-Adapted Interaction, Vol. 13, 2003, pp. 5–33.

[Mue1997] MÜLLER H. J., *Towards Agent Systems Engineering*, Data & Knowledge Engineering, 1997, Vol. 23, pp. 217–245.

[Mul1996] MULAWKA J.J., *Systemy ekspertowe*, Wydawnictwa Naukowo-Techniczne, Warszawa 1996.

[Mye1997] MYERS K., WILKINS D., *The Act Formalism, Version 2.2*, SRI International Artificial Intelligence Center, Technical Report, Menlo Park, CA, 1997.

[Nan2004] NANOPOULOS A., MANOLOPOULOS Y., *Memory-adaptive association rules mining*, Information Systems, Vol. 29, 2004, 365–384.

[Nas2004] NASON S., LAIRD J.E., *Soar-RL: Integrating Reinforcement Learning with Soar*, Proceedings of the Sixth International Conference on Cognitive Modeling (ICCM2004), Pittsburgh, Pennsylvania, USA, 30 July – 1 August 2004.

[Nas2005] NASON S., LAIRD J.E., *Soar-RL: integrating reinforcement learning with Soar*, Cognitive Systems Research, Vol. 6, 2005, pp. 51–59.

[Ndu1998] NDUMU D.T., COLLIS J.C., NWANA H.S., *Towards desktop personal travel agents*, BT Technology Journal, Vol. 16, No. 3, July 1998, pp. 69–78.

[New1990] NEWELL A., Unified Theories of Cognition, Harvard University Press, Cambridge 1990.

[NeS1997] NEWELL S.C., *User Models and Filtering Agents for Improved Internet Information Retrieval*, User Modeling and User-Adapted Interaction, Vol. 7, 1997, pp. 223–237.

[NgK2004] NG K.S., *Alkemy: A Learning System Based on an Expressive Knowledge Representation Formalism*, Computer Sciences Laboratory, The Australian National University, Canberra, Australia, August 2004.

[Ngu2002] NGUYEN N.T., *Metody wyboru consensusu i ich zastosowanie w rozwiązywaniu konfliktów w systemach rozproszonych*, Praca habilitacyjna, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2002.

[Nwa1996] NWANA H., *Software Agents: An Overview*, Knowledge Engineering Review, Vol. 11, No. 3, 1996, pp. 205–244.

[Nwa1999] NWANA H., NDUMU D.T., *A Perspective on Software Agents Research*, The Knowledge Engineering Review, Vol. 14, No. 2, 1999, pp. 1–18.

[Oli1999] OLIVEIRA E., FISCHER K., STEPANKOWA O., *Multi-agent systems: which research for which applications*, Robotics and Autonomous Systems, Vol. 27, 1999, pp. 91–106.

[Pan1992] PANKOWSKI T., *Podstawy baz danych*, Wydawnictwo Naukowe PWN, Warszawa, 1992.

[Par1998] PARUNAK H.V.D., *Practical and Industrial Applications of Agent-Based Systems*, Industrial Technology Institute, 1998.

[Pas2005] PASQUIER N., TAOUIL R., BASTIDE Y., STUMME G., LAKHAL L., *Generating a Condensed Representation for Association Rules*, Journal of Intelligent Information Systems, Vol. 24, No. 1, 2005, pp. 29–60.

[Paz1997] PAZZANI M.J., BILLSUS D., *Learning and Revising User Profiles: The Identification of Interesting Web Sites*, Machine Learning, Vol. 27, 1997, pp. 313–331.

[Paz2002] PAZZANI M.J., BILLSUS D., *Adaptive Web Site Agents*, Autonomous Agents and Multi-Agent Systems, Vol. 5, 2002, pp. 205–218.

[Per1997] PERKOWITZ M., DOORENBOS R.B., ETZIONI O., WELD D.S., *Learning to Understand Information on the Internet: An Example-Based Approach*, Journal of Intelligent Information Systems, Vol. 8., 1997, pp. 133–153.

[Pie2003] PIECZYŃSKA-KUCHTIAK A., *Funkcja decyzyjna w algorytmie wyboru komunikatu*, [in:] BUBNICKI Z., GRZECH A. (Eds.), *Inżynieria Wiedzy i Systemy Ekspertowe*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2003, Tom 2, pp. 271–280.

[Piv2002] PIVK A., GAMS M., *Domain-dependent information gathering agent*, Expert Systems with Applications, Vol. 23, 2002, pp. 207–218.

[Pro2003] PROTAZIUK G., RYBIŃSKI H., *Association Rules in Incomplete Transactional Databases*, Proceedings of the First Symposium on Databases, Data Warehousing and Knowledge Discovery, Baden Baden, Germany, July 2003, pp. 7–20.

[Ran2003] RANKINS R., JENSEN P., BERTUCCI P., *Microsoft SQL Server 2000. Księga eksperta.*, HELION, Gliwice 2003.

[Rao1991] RAO A., GEORGEFF M., *Modeling Rational Agents within a BDI-Architecture*, Proceedings of the Second International Conference on Artificial Intelligence, Sydney, Australia, Morgan Kaufman, 1991, pp. 473–484.

[Rao1995] RAO A., GEORGEFF M., *BDI Agents: From Theory to Practice*, Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA, June 1995, pp. 312 –319.

[Ras2002] RASTOGI R., SHIM K., *Mining Optimized Association Rules with Categorical and Numeric Attributes*, IEEE Transactions on Knowledge and Data Engineering, Vol. 14, No. 1, 2002, pp. 29–50.

[Rau2005] RAUCH J., *Logic of Association Rules*, Applied Intelligence, Vol. 22, 2005, pp. 9–28.

[Rib2002] RIBEIRO C., *Reinforcement Learning Agents*, Artificial Intelligence Review, Vol. 17, 2002, pp. 223–250.

[Rin1997] RING M.B., *CHILD: A First Step Towards Continual Learning*, Machine Learning, Vol. 28, 1997, pp. 77–104.

[Rip2000] RIPPER P.S., FONTOURA M.F., NETO A.M., LUCENA de C.J.P., *V-Market: A framework for agent e-commerce systems*, World Wide Web, Vol. 3, 2000, pp. 43–52.

[Rus1995] RUSSELL S.J., NORVIG P., *Artificial Intelligence, a modern approach*., Prentice Hall, New Jersey, USA, 1995.

[Sch2004] SCHMIDT R.A., TISHKOVSKY D., HUSTADT U., *Interactions between Knowledge, Action and Commitment within Agent Dynamic Logic*, Studia Logica, Vol. 78, 2004, pp. 381–415.

[Scu2005] SCHUSTER A., WOLFF R., TROCK D., *A high-performance distributed algorithm for mining association rules*, Knowledge and Information Systems, Vol. 7, 2005, pp. 458–475.

[Sen1999] SEN S., WEISS G., *Learning in Multiagent Systems*, [in:] WEISS G. (Ed.), Multiagent systems, The MIT Press, 1999, Chapter 6, pp. 259–298.

[Snk2002] ŞENKUL S., POLAT F., *Learning Intelligent Behavior in a Non-stationary and Partially Observable Environment*, Artificial Intelligence Review, Vol. 18, 2002, pp. 97– 115.

[She2000] SHEN W., MATURANA F., NORRIE D.H., *Enhancing the Performance of an Agent-Based Manufacturing System Through Learning and Forecasting*, Journal of Intelligent Manufacturing, Vol. 11, 2000, pp. 365–380.

[ShS1999] SHEN L., SHEN H., CHENG L., *New algorithms for efficient mining of association rules*, Information Sciences, Vol. 118, 1999, pp. 251–268.

[Sil1998] SILVERSTEIN C., BRIN S., MOTWANI R., *Beyond Market Baskets: Generalizing Association Rules to Dependence Rules*, Data Mining and Knowledge Discovery, Vol. 2, 1998, pp. 39 – 68.

[Sin1991] SINGH M. P., *Intentions, Commitment and Rationality*, Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society, Chicago, Illinois, August 1991.

[Sin1992] SINGH M. P., *A Critical Examination of the Cohen-Levesque Theory of Intention*, Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI), Vienna, Austria, August 1992.

[Sin1995a] SINGH M. P., *Semantical Considerations on Some Primitives for Agent Specification*, *IJCAI Workshop on Agent Theories, Architectures, and Languages (ATAL)*, Montreal, Canada, August 1995.

[Sin1995b] SINGH M. P., *Formalizing Actions in Branching Time: Model-Theoretic Considerations*, Proceedings of the Second International Workshop on Temporal Representation and Reasoning (TIME), Melbourne Beach, Florida, April 1995.

[Sin1998] SINGH M. P., *Semantical Considerations on Intention Dynamics for BDI Agents*, Journal of Experimental and Theoretical Artificial Intelligence, 1998.

[Sun2003] SUNG S.Y., LI Z., TAN C.L., NG P.A., *Forecasting Association Rules Using Existing Data Sets*, IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No. 6, 2003, pp. 1448–1459.

[Sym2002] SYMEONIDIS A.L., MITKAS P.A., KECHAGIAS D.D., *Mining Patterns And Rules For Improving Agent Intelligence Through An Integrated Multi-Agent Platform*, Proceedings of the Sixth IASTED International Conference on Artificial Intelligence and Soft Computing (ASC 2002), Banff, Alberta, Canada, 17–19 July 2002.

[Tak1998] TAKADAMA K., NAKASUKA S., TERANO T., *Printed Circuit Board Design via Organizational-Learning Agents*, Applied Intelligence, Vol. 9, 1998, pp. 25–37.

[Tes2002] TESAURO G., KEPHART J.O., *Pricing in Agent Economies Using Multi-Agent Q-Learning*, Autonomous Agents and Multi-Agent Systems, Vol. 5, 2002, pp. 289–304.

[Tha2002] THAWONMAS R., HIRAYAMA J., TAKEDA F., *Learning from Human Decision-Making Behaviors - An Application to RoboCup Software Agents*, [in:] HENDTLASS T., ALI M. (Eds.): IEA/AIE 2002, Lecture Notes in Artificial Intelligence, Vol. 2358, Springer-Verlag, Berlin Heidelberg 2002, pp. 136–146.

[Tho1998] THOMAS S., SARAWAGI S., *Mining Generalized Association Rules and Sequential Patterns Using SQL Queries*, Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), New York, USA, 1998.

[Thr1995] THRUN S., MITCHELL T.M., *Lifelong Robot Learning*, Robotics and Autonomous Systems, Vol. 15, 1995, pp. 25–46.

[Tsa1999] TSAI P.S., LEE C.-C., CHEN A.L., *An Efficient Approach for Incremental Association Rule Mining*, [in:] ZHONG N., ZHOU L. (Eds.), PAKDD'99, Lecture Notes in Artificial Intelligence (LNAI), Vol. 1574, Springer-Verlag, Berlin Heidelberg 1999, pp. 74–83.

[Tsa2004] TSAI P.S., CHEN C.-M., *Mining interesting association rules from customer databases and transaction databases*, Information Systems, Vol. 29, 2004, pp. 685–696.

[TsC2005] TSAY Y.-J., CHIANG J.-Y., *CBAR: an efficient method for mining association rules*, Knowledge-Based Systems, Vol. 18, 2005, pp. 99–105.

[URLSoar] The Soar Home Page, http://sitemaker.umich.edu/soar.

[Van1991a] VAN DER HOEK W., *On the Semantics of Graded Modalities*, Technical report No. IR-246, vrije Universiteit, Amsterdam, May 1991.

[Van1991b] VAN DER HOEK W., MEYER J.-J., *Graded Modalities in Epistemic Logic*, Logique & Analyse, Vol. 133–134, 1991, pp. 251–270.

[Van1997] VAN DER HOEK W., VAN LINDER B., MEYER J.-J., *An Integrated Modal Approach to Rational Agents*, Proceedings of the Second AISB Workshop on Practical Reasoning and Rationality, Manchester, UK, April 1997, pp. 123–159.

[Van1999] VAN DER HOEK W., VAN LINDER B., MEYER J.-J., *A logical approach to the dynamics of commitments*, Artificial Intelligence, Vol. 113, 1999, pp. 1–40.

[Vid1997] VIDAL J.M., DURFEE E.H., *Agents Learning About Agents: A Framework and Analysis*, Proceedings of the AAAI-97 Workshop on Multiagent Learning, Providence, Rhode Island, USA, 28 July 1997.

[Vid2003] VIDAL J.M., DURFEE E.H., *Predicting the Expected Behavior of Agents that Learn About Agents: The CLRI Framework*, Autonomous Agents and Multi-Agent Systems, Vol. 6, 2003, pp. 77–107.

[Wag2003] WAGNER T., *Applying Agents for Engineering of Industrial Automation Systems*, [in:] SCHILLO M. et al. (Eds.), MATES 2003, *Lecture Notes in Artificial Intelligence*, Vol. 2831, Springer-Verlag, Berlin Heidelberg 2003, pp. 62–73.

[Wal1996] WALLEY P., *Measures of uncertainty in expert systems*, Artificial Intelligence, Vol. 83, 1996, pp. 1–58.

[Wei1996] WEISS G., *Adaptation and learning in multi-agent systems: Some remarks and a bibliography*, [in:] WEISS G., SEN S. (Eds.), Adaptation and learning in multi-agent systems, Lecture Notes in Artificial Intelligence, Vol. 1042, Springer-Verlag, 1996, pp. 1–21.

[Wei1999] WEISS G., DILLENBOURG P., *What is 'multi' in multi-agent learning?*, [in:] DILLENBOURG P. (Ed.), Collaborative learning. Cognitive and computational approaches., Pergamon Press, 1999, Chapter 4, pp. 64–80.

[WgP2005] WANG H., PERNG C.-S., MA S., YU P.S., *Demand-driven frequent itemset mining using pattern structures*, Knowledge and Information Systems, Vol. 8, 2005, pp. 82–102.

[WgS2004] WANG F.-H., SHAO H.-M., *Effective personalized recommendation based on time-framed navigation clustering and association mining*, Expert Systems with Applications, Vol. 27, 2004, pp. 365–377.

[WgU2005] WANG Y.-C., USHER J.-M., *Application of reinforcement learning for agent-based production scheduling*, Engineering Applications of Artificial Intelligence, Vol. 18, pp. 73–82.

[Wil2004] WILLIAMS A.B., *Learning to Share Meaning in a Multi-Agent System*, Autonomous Agents and Multi-Agent Systems, Vol. 8, 2004, pp. 165–193.

[Wls2003] WILSON B., *The Machine Learning Dictionary for COMP9414* (http://www.cse.unsw.edu.au/~billw/mldict.html), 2003.

[Woo1995a] WOOLDRIDGE M., JENNINGS N.R., *Intelligent Agents: Theory and Practice*, Knowledge Engineering Review, Vol. 10, No. 2, 1995.

[Woo1995b] WOOLDRIDGE M., *An Abstract General Model and Logic of Resource-Bounded Believers*, [in:] COX M., FREED M. (Eds.), *Representing Mental States and Mechanisms*, Proceedings of the 1995 AAAI Spring Symposium, AAAI Press, March 1995.

[Woo1997a] WOOLDRIDGE M., *Agent-based Software Engineering*, [in:] IEE Proceedings on Software Engineering, Vol. 144, No. 1, February 1997, pp. 26–37.

[Woo1997b] WOOLDRIDGE M., *A Knowledge-Theoretic Semantics for Concurrent MetateM*, [in:] MUELLER J., WOOLDRIDGE M., JENNINGS N. R. (Eds.), *Intelligent Agents III*, Springer-Verlag, Berlin Heidelberg 1997.

[Woo1999a] WOOLDRIDGE M., JENNINGS N.R., *Software Engineering with Agents: Pitfalls and Pratfalls*, IEEE Internet Computing, May/June 1999, pp. 20–27.

[Woo1999b] WOOLDRIDGE M., JENNINGS N.R., KINNY D., *A Methodology for Agent-Oriented Analysis and Design*, [in:] ETZIONI O., MULLER J.P., BRADSHAW J. (Eds.), Proceedings of the Third International Conference on Autonomous Agents (Agents '99), Seattle, WA, USA, May 1999.

[Woo1999c] WOOLDRIDGE M., *Intelligent Agents*, [in:] WEISS G. (Ed.), *Multiagent Systems*, The MIT Press, 1999.

[Yan2004] YANG Q., LI T., WANG K., *Building Association-Rule Based Sequential Classifiers for Web-Document Prediction*, Data Mining and Knowledge Discovery, Vol. 8, 2004, pp. 253–273.

[Yao2002] YAO Y.Y., HAMILTON H.J., WANG X., *PagePrompter: An Intelligent Web Agent Created Using Data Mining Techniques*, [in:] ALPIGINI J. J. et al. (Eds.), RSCTC 2002, Lecture Notes in Artificial Intelligence, Vol. 2475, Springer-Verlag, Berlin Heidelberg 2002, pp. 506–513.

[Yen1996] YEN S.J., CHEN A.L.P., *An efficient approach to discovering knowledge from large databases*, Proceedings of the IEEE/ACM international conference on parallel and distributed information systems, 1996, pp. 8–18.

[You2005] YOUNG R.M., *The data learning problem in cognitive architectures*, Cognitive Systems Research, Vol. 6, 2005, pp. 89–97.

[Zak1999] ZAKI M.J., *Parallel and Distributed Association Mining: A Survey*, IEEE Concurrency, Vol. 7, No. 4, 1999, pp. 14–25.

[Zak2000] ZAKI M.J., *Scalable Algorithms for Association Mining*, IEEE Transactions on Knowledge and Data Engineering, Vol. 12, No. 3, 2000, pp. 372–390.

[ZhE2001] ZHOU Z., EZEIFE C.I., *A Low-Scan Incremental Association Rule Maintenance Method Based on the Apriori Property*, [in:] STROULIA E., MATWIN S. (Eds.), AI 2001, Lecture Notes in Artificial Intelligence, Vol. 2056, Springer-Verlag, Berlin Heidelberg 2001, pp. 26–35.

[Zho2002] ZHONG F., WU D.J., KIMBROUGH S.O., *Cooperative Agent Systems: Artificial Agents Play the Ultimatum Game*, Group Decision and Negotiation, Vol. 11, 2002, pp. 433–447.