Dekompozycja ciągu uczącego dla rozproszonej bazy danych

Teaching sequence decomposition FOR DISTRIBUTED DATABASE

Swietłana Lebiediewa[1]

Abstract - The problem of decomposition of a teaching sequence (TS) for distributed database is formulated. Three types of TS decomposition for a distributed database are formulated. Theorems concerning memory occupancy by the TS after decomposition are proved. The results of a calculation experiment are presented, that illustrate the memory occupancy depending upon the decomposition type, the redundancy of features in the tree and upon the type of the dispersion.

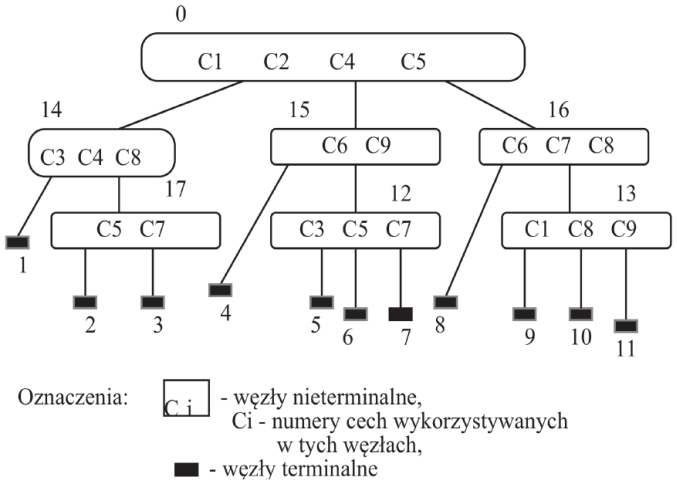
Streszczenie - Sformułowano problem dekompozycji ciągu uczącego (CU) dla rozproszonej bazy danych. Zdefiniowano trzy rodzaje dekompozycji. Przedstawiono twierdzenia dotyczące zajętości pamięci przez zdekomponowany ciąg uczący. Zaprezentowano wyniki badań symulacyjnych ilustrujących zajętość pamięci w zależności od typu rozproszenia i rodzaju dekompozycji.

Słowa kluczowe: rozproszona baza danych, rozpoznawanie wieloetapowe, ciąg uczący, dekompozycja

1. Sformułowanie problemu

Rozpoznawanie wielostopniowe polega na *dekompozycji* problemu decyzyjnego, czyli zastępowaniu jednorazowego rozpoznawania sekwencją tzw. rozpoznawań lokalnych przeprowadzanych w poszczególnych węzłach zgodne z zadaną konstrukcją drzewa decyzyjnego (DD), rys. 1. Żeby zaklasyfikować rozpoznawany obiekt do jednej z klas, należy przejść ścieżką w DD startując od korzenia. W każdym napotykanym węźle wewnętrznym (**decyzyjnym**) należy podjąć decyzję o dalszej drodze w grafie, aż osiągnięty zostanie węzeł terminalny. Klasa z nim związana reprezentuje końcowy wynik rozpoznawania wielostopniowego. Tak więc w trakcie klasyfikacji wielostopniowej rozpoznawany obiekt podlega sekwencji decyzji na ścieżce od korzenia do węzła terminalnego.

Algorytmy rozpoznawania z uczeniem korzystają z ciągu uczącego (CU) [1,2,3,4]. CU jest ciągiem poprawnie zaklasyfikowanych obiektów, elementami CU są pary postaci (x_k, k), gdzie x jest wektorem wartości cech obiektu, k – numerem klasy. Drzewo decyzyjne (DD) przedstawiono na rys.1, a odpowiadający mu CU – na rys. 2.



Rys. 1. Drzewo decyzyjne NR1a
Figure 1. Decision tree No. 1a

W CU przedstawionym na rys. 2 w miejscu wartości niektórych cech występuje symbol *, symbol ten oznacza, że dana cecha jest cechą nieistotną dla rozpoznawanego obiektu i nie jest brana pod uwagę przez algorytm rozpoznawania. Tak więc w procesie rozpoznawania wielostopniowego na różnych etapach rozpoznawania wykorzystuje się tylko niektóre elementy wektora cech obiektów, a pewne cechy wcale nie występują na poszczególnych ścieżkach.

1.Wrocławska Wyższa Szkoła Informatyki Stosowanej „Horyzont” (‘Horyzont’ Wroclaw School of Information Technology) swietlana@lebiediewa.com

C1	C2	C3	C4	C5	C6	C7	C8	C9	NR KLASY
16	3,44	3	6,55	40,0	48	116	*	14,9	7
15	3,45	2	8,45	40,0	*	*	120	*	1
14	2,4	1	8,33	38,1	*	110	*	*	2
18	3,0	*	7,5	35,0	40	*	*	15,0	4
15	2,7	*	6,0	20,0	35	110	112	*	8

Rys.2. Fragment ciągu uczącego dla DD NR1a
Figure 2. A fragment of the teaching sequence for DT No. 1a

W celu oszczędności pamięci proponuje się dekompozycję CU. Z algorytmów rozpoznawania przedstawionych w [2,3] widać, że czas pracy AR w każdym z węzłów składa się z czasu potrzebnego na utworzenie segmentu danych oraz czasu pracy algorytmów rozpoznawania. Odpowiednia postać CU może skrócić czas potrzebny na utworzenie segmentu danych. Celem dekompozycji jest rozbicie CU na takie podciągi, które zarówno minimalizują zajętość pamięci przez CU jak i zmniejszają czas potrzebny rozpoznawania obiektów [5,6]. W pewnych przypadkach wydaje się uzasadnione rozbicie procesu rozpoznawania na kilka lokalizacji, w których znajdują się lokalne bazy danych (LBD), które połączone siecią komunikacyjną stanowią rozproszoną bazę danych (RBD). Model konceptualny scentralizowanej bazy danych (SBD) przedstawiono w [5, 7]. Model konceptualny BD zawiera informacje o rozpoznawanym obiekcie a także informację o CU SEQUENCE i relacje opisujące strukturę DD KLASY, WĘZŁY i CECHY. W relacji SEQUENCE znajduje się informacja o CU. Relacja KLASY opisuje zależność pomiędzy numerem klasy a numerem węzła będącego bezpośrednim poprzednikiem tej klasy. dla każdej klasy wskazany jest jej poprzednik. Relacja WĘZŁY zawiera informacje o numerach każdego węzła, o bezpośrednich następnikach i poprzednikach węzła. W relacji CECHY zawarta jest informacja, w których węzłach wykorzystywane są poszczególne cechy [5, 6]. Model konceptualny RBD różni się od modelu konceptualnego SBD tym tylko, że w opisie relacji WĘZŁY występuje dodatkowy atrybut LOKALIZACJA, wskazujący na lokalizację węzła nieterminalnego [7]. Dekompozycję CU dla SBD przedstawiono w [6]. Przedstawimy dekompozycję CU dla RBD.

2. Dekompozycja ciągu uczącego dla rozproszonej bazy danych

Dla rozproszonej bazy danych (RBD) możemy wyróżnić trzy rodzaje dekompozycji ciągu uczącego: dekompozycję *naturalną*, zmodyfikowaną dekompozycję 1. rodzaju i dekompozycję 2. rodzaju. Ze względu na wysoki koszt transmisji danych wydaje się zasadnym przyjąć założenie, iż CU jest dekomponowany w taki sposób, ażeby algorytm rozpoznawania nie musiał korzystać z informacji zawartej w CU znajdującym się w innej lokalizacji. Można przyjąć, że w komputerze głównym pozostawia się cały CU zdekomponowany lub nie, ze względu na to, że w węźle będącym korzeniem wykorzystywane są cechy występujące we wszystkich elementach CU, można też założyć całkowite rozproszenie CU. W dalszym ciągu zakładamy całkowite rozproszenie CU (jeżeli założyć zachowanie CU w głównej lokalizacji, zajętość pamięci będzie nie mniejsza niż przy całkowitym rozproszeniu). Aby umożliwić ponowne otrzymanie niezdekomponowanego CU, do każdego wiersza CU zdekomponowanego dodajemy identyfikator.

Dekompozycja naturalna polega na tym, że z CU usuwa się wiersze, które w kolumnie NR KLASY nie zawierają numerów klas nieosiągalnych z węzłów nieterminalnych lokalnej bazy danych (LBD), następnie usuwa się kolumny zawierające wartości cech, nie wykorzystywanych w danej lokalizacji. Metoda dekompozycji 1. rodzaju dla RBD jest tak zmodyfikowana, że każdy podciąg (lub jego fragment) odpowiadający węzłowi pośredniemu znajdującemu się poza główną lokalizacją, zawiera tylko wartości cech wykorzystywanych w tej lokalizacji. Podciąg ciągu uczącego budujemy dla każdego węzła nieterminalnego, który jest bądź bezpośrednim poprzednikiem węzła nieterminalnego (klasy), bądź jest bezpośrednim poprzednikiem węzła nieterminalnego znajdującego się w innej lokalizacji. W [6] przedstawiono metodę dekompozycji 1. rodzaju dla SBD i metodę dekompozycji 2. rodzaju dla SBD.

Algorytmy dekompozycji korzystają z informacji o CU i strukturze DD zawartej w relacjach SEQUENCE, KLASY, WĘZŁY, CECHY modelu konceptualnego BD.

ALGORYTM 1. (algorytm naturalnej dekompozycji CU dla RBD)

Dane: model konceptualny RBD - relacje SEQUENCE, WĘZŁY, CECHY.

Wyznaczyć: CU zawierający wartości cech, wykorzystywanych w danej lokalizacji do rozpoznawania klas osiągalnych z węzłów nieterminalnych należących do tej lokalizacji.

KROK 0. Do relacji przechowującej CU dodaj kolumnę zawierającą identyfikatory elementów CU.

KROK 1. Dla każdej LBD z relacji WĘZŁY pobierz numery węzłów nieterminalnych znajdujących się w LBD.

KROK 2. Z relacji NASTĘPNIKI wybierz numery klas osiągalnych z węzłów nieterminalnych należących do LBD.

KROK 3. Dla każdej LBD z relacji przechowującej CU wybierz wiersze, dla których

NR KLASY = "klasa osiągalna z węzła nieterminalnego należącego do tej lokalizacji";

KROK 4. Z otrzymanej relacji usuń atrybuty zawierające wartości cech, które nie są wykorzystywane w tej lokalizacji. **STOP**.

Oznaczmy przez *ZPDN* zajętość pamięci przez CU otrzymany w wyniku dekompozycji naturalnej, przez i – numer LBD, przez LC_i – liczbę cech wykorzystywanych w lokalizacji i, przez k_i – liczbę klas osiągalnych z węzłów nieterminalnych należących do lokalizacji i, przez L – liczbę LBD. Zajętość pamięci przez CU dla całej RBD wyraża wzór:

$$ZPDN = \sum_{i=1}^{L} (LC_i + 2) * K_i \qquad (1)$$

ALGORYTM 2. (zmodyfikowany algorytm dekompozycji pierwszego rodzaju CU dla RBD)
Dane: model konceptualny RBD - relacje SEQUENCE, WĘZŁY, CECHY.

Wyznaczyć: dla każdego węzła nieterminalnego będącego bądź poprzednikiem klas, bądź poprzednikiem węzłów nieterminalnych znajdujących się w innej lokalizacji wyznaczyć taki podciąg ciągu uczącego, który zawiera wartości cech wykorzystywanych na ścieżce od "korzenia" poddrzewa do tego węzła.

KROK 0. Do relacji przechowującej CU dodaj kolumnę zawierającą identyfikatory elementów CU.

KROK 1. Z relacji KLASY i WĘZŁY pobierz numery węzłów należących do LBD, które są bądź bezpośrednimi poprzednikami węzłów terminalnych bądź bezpośrednimi poprzednikami węzłów nieterminalnych należących do innej lokalizacji.

KROK 2. Dla każdego takiego węzła utwórz relację zawierającą podciąg uczący dla wszystkich klas bezpośrednio osiągalnych z tego węzła i zawierający wartości cech wykorzystywanych w tej lokalizacji (na ścieżce od "korzenia" poddrzewa do tego węzła).

KROK 3. Jeżeli węzeł jest bezpośrednim poprzednikiem węzła nieterminalnego znajdującego się w innej lokalizacji, utwórz podciąg CU zawierający wartości cech wykorzystywanych w tej lokalizacji do rozpoznawania klas będących następnikami tego węzła, otrzymany podciąg dodaj do podciągu otrzymanego w kroku drugim.

STOP.

Niech L, jak wyżej, oznacza liczbę LBD, a i – numer lokalizacji. Oznaczmy przez J_i liczbę węzłów nieterminalnych w lokalizacji i, przez LC_j– liczbę cech wykorzystywanych w danej lokalizacji przez algorytmy rozpoznawania na ścieżce do węzła j, przez K_j – łączną liczbę klas bądź będących bezpośrednimi następnikami węzła j, bądź osiągalnych z węzła – bezpośredniego następnika j należącego w innej lokalizacji. Zajętość pamięci przez CU otrzymane w wyniku dekompozycji pierwszego rodzaju (RZPD1) dla RBD wyraża wzór:

$$RZPD1 = \sum_{i=1}^{L} \sum_{j=1}^{J_I} \left(LC_j + 2\right) * k_j \qquad (2)$$

Algorytm dekompozycji drugiego rodzaju dla RBD nie różni się od algorytmu dekompozycji drugiego rodzaju dla SBD:

ALGORYTM 3. (algorytm dekompozycji 2. rodzaju)
Dane: model konceptualny RBD – relacje SEQUENCE, WĘZŁY, CECHY.
Wyznaczyć: dla każdego węzła nieterminalnego utworzyć taki podciąg ciągu uczącego, który zawiera wartości tylko tych cech, które są wykorzystywane w tym węźle.

KROK 1. Do relacji SEQUENCE przechowującej CU dodaj kolumnę zawierającą identyfi katory elementów CU.

KROK 2. Z relacji WĘZŁY pobierz numery wszystkich węzłów nieterminalnych.

KROK 3. Dla każdego węzła nieterminalnego utwórz podciąg uczący (z CU wybierz wiersze, dla których NR KLASY = "klasa osiągalna z danego węzła", z otrzymanej relacji usuń atrybuty zawierające wartości cech nie wykorzystywanych w tym węźle). **STOP**.

Niech J oznacza liczbę wszystkich węzłów nieterminalnych w bazie danych, j – kolejny numer węzła, LC_j – liczbę cech wykorzystywanych w węźle j, K_j – liczbę klas osiągalnych z węzła j. Wzór na zajętość pamięci przy dekompozycji drugiego rodzaju dla RBD (*RZPD2*) ma postać:

$$RZPD2 = \sum_{j=1}^{J} (LC_j + 1) * K_j \qquad (3)$$

Wniosek Dla zmodyfikowanej dekompozycji 1. rodzaju dla RBD redundancja cech występujących na ścieżce nie ma wpływu na zajętość pamięci, jeżeli występuje w węzłach znajdujących się w jednej lokalizacji, natomiast zwiększa zajętość pamięci, jeżeli występuje w węzłach leżących w różnych lokalizacjach.

Z Algorytmu 2 wynika

Twierdzenie 1. Zajętość pamięci przy dekompozycji drugiego rodzaju nie zależy od sposobu rozproszenia.

Mają miejsce następujące lematy:

Lemat 1. Przy całkowitym rozproszeniu (każdy z węzłów nieterminalnych znajduje się w innej LBD) ma miejsce następujący wzór:

$$ZPDN = RZPD1 = RZPD2$$

Lemat 2. Jeżeli w LBD żadna para węzłów nieterminalnych nie znajduje się w relacji poprzednik–następnik, ma miejsce równość:

$$RZPD1 = RZPD2,$$

w przeciwnym przypadku

$$RZPD1 < RZPD2$$

Z lematów 1. i 2. wynika

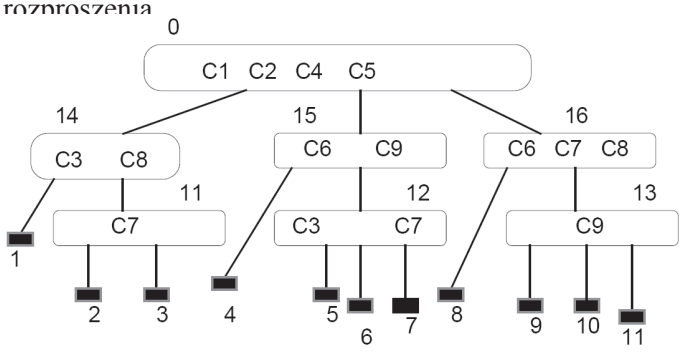
Twierdzenie 2. Dla RBD ma miejsce zależność:

$$RZPD1 \leq RZPD2$$

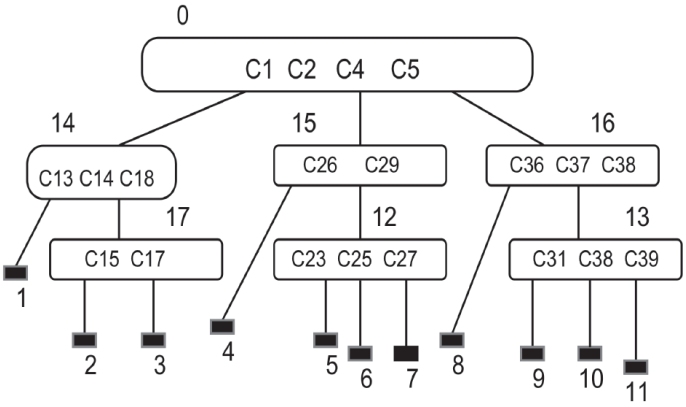
3.Eksperyment obliczeniowy

Przedstawimy wyniki eksperymentu obliczeniowego ilustrującego zajętość pamięci CU otrzymanych w wyniku zmodyfikowanej dekompozycji pierwszego i drugiego oraz

dekompozycji naturalnej dla rozproszonej bazy danych. Rozpatrujemy zajętość pamięci dla drzew 1a (Rys. 1), 1b (Rys 3.), i 1c (Rys 4.) w zależności od rodzaju dekompozycji i typu rozproszenia.



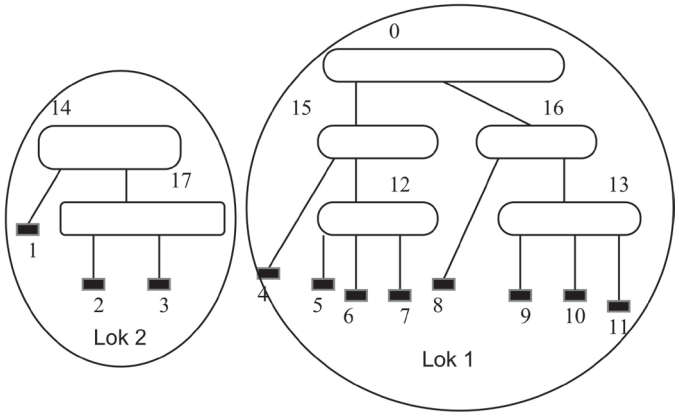
Rys. 3 Drzewo decyzyjne 1b



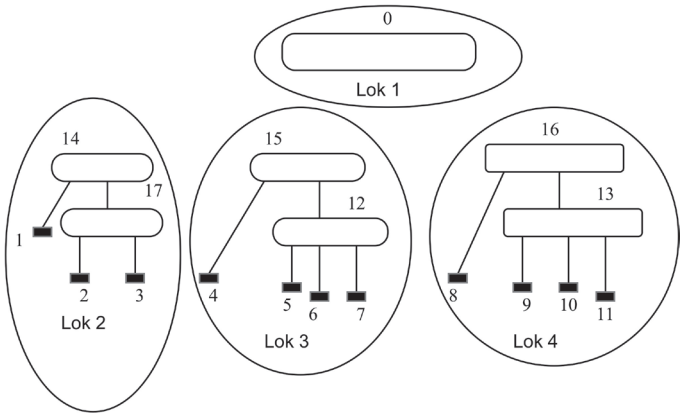
Rys. 4. Drzewo decyzyjne 1c
Figure 4. Decision tree No. 1c

Struktura drzew 1a, 1b i 1c jest identyczna, drzewo 1b różni się od drzewa 1a tym, że na poszczególnych ścieżkach nie ma redundancji cech, a w drzewie 1c w każdym z węzłów występują inne cechy.

Rozpatrzmy dwa rodzaje rozproszenia dla drzew 1a-1c: rozproszenie typu *A* i rozproszenie typu *B*, rys.5. W przypadku rozproszenia typu *A* rozproszona baza danych składa się z dwóch LBD. Do pierwszej LBD należą węzły 0, 16, 13, 15 i 12, do drugiej – węzły 14 i 17. W przypadku rozproszenia typu *B* rozproszona baza danych składa się z czterech LBD, do pierwszej LBD należy węzeł numer 0 (korzeń drzewa), do drugiej – węzły 14 i 17; do trzeciej – węzły 15 i 12; do czwartej – węzły 16 i 13. Rysunki 6 i 7 ilustrują zajętość pamięci przez CU dla drzew 1a-1c odpowiednio dla rozproszenia typu *A* i rozproszenia typu *B*. Jak widać różnice w zajętości pamięci przez CU otrzymywane w wyniku dekompozycji naturalnej, dekompozycji pierwszego rodzaju i dekompozycji drugiego rodzaju zmniejszają się w miarę wzrostu rozproszenia aż do zaniku różnic (por. Lemat 1).

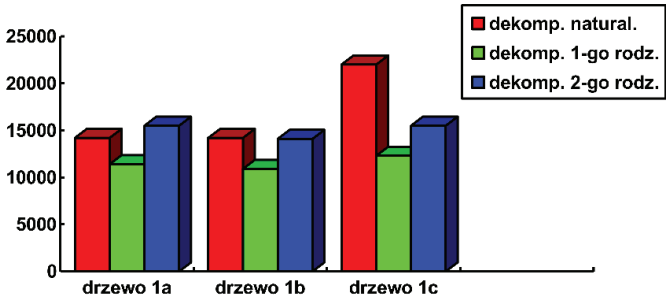


Rozproszenie typu A

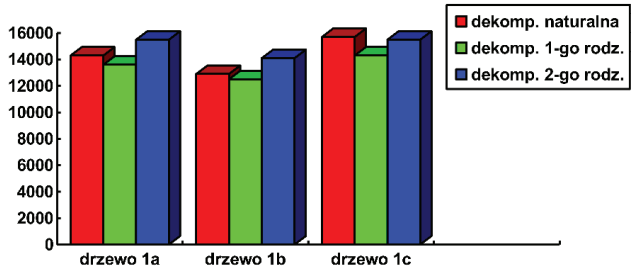


Rozproszenie typu B

Rys. 5. Rozproszenie typu A i B dla drzew 1a – 1c.
Figure 5. The dispersion of type A and B for the trees 1a - 1c.



Rys. 6. Wykres zajętości pamięci przez CU otrzymane w rezultacie dekompozycji pierwszego i drugiego rodzaju oraz dekompozycji naturalnej dla drzew 1a-1c, rozproszenie typu A
Figure 6. Chart memory use by the CU obtained as a result of decomposition of the first and second kind, and for the decomposition of natural trees 1a-1c, the dispersion of type A



Rys. 7. Wykres zajętości pamięci przez CU otrzymane w rezultacie dekompozycji pierwszego i drugiego rodzaju oraz dekompozycji naturalnej dla drzew 1a-1c, rozproszenie typu B
Figure 7. Chart memory use by the CU obtained as a result of decomposition of the first and second kind, and for the decomposition of natural trees 1a-1c, the dispersion of type B

Najlepsze rezultaty daje dekompozycja 1-go rodzaju dla rozproszenia typu A. Dla CU otrzymanych w wyniku dekompozycji 1-go rodzaju wzrost zajętości pamięci następuje wraz ze wzrostem rozproszenia i wzrostem redundancji cech. Zajętość pamięci przez CU otrzymane w wyniku dekompozycji 2-go rodzaju wzrasta wraz ze wzrostem redundancji. Rodzaj rozproszenia nie ma wpływu na zajętość pamięci przez CU otrzymane w wyniku dekompozycji 2-go rodzaju. Dekompozycja naturalna daje najlepsze rezultaty w przypadku braku redundancji i rozproszenia typu C. Najgorsze rezultaty daje dekompozycja naturalna w przypadku rozproszenia typu B (większa zajętość pamięci niż w przypadku CU nie zdekomponowanego).
Dla każdej dekompozycji korzystny wpływ na zajętość CU ma rozpoznanie części klas na wcześniejszych etapach.

Wnioski:

Dla RBD zmodyfikowana dekompozycja 1. rodzaju daje wyraźnie lepsze wyniki, gdy liczba etapów jest duża i w poszczególnych lokalizacjach znajdują się węzły należące do długich ścieżek. Im większe rozproszenie, tym mniejsza różnica między rodzajami rozproszenia. Przy całkowitym rozproszeniu każda z dekompozycji daje ten sam rezultat. Dekompozycja 2. rodzaju nie zawsze daje oszczędność pamięci. Dekompozycja 2. rodzaju na ogół potrzebuje więcej pamięci niż dekompozycja 1. rodzaju, a w niektórych przypadkach – także więcej niż dekompozycja naturalna. Wraz z wzrostem rozproszenia (wzrostem liczby lokalizacji) maleje różnica w zajętości pamięci przez CU otrzymywane w wyniku dekompozycji naturalnej, dekompozycji pierwszego i drugiego rodzaju. Przy dużym rozproszeniu najbardziej korzystna wydaje się być dekompozycja 2-go rodzaju, ponieważ CU otrzymane w wyniku dekompozycji 2. rodzaju są identyczne z fragmentami CU wykorzystywanych przez algorytmy rozpoznawania w węźle, dlatego w przypadku stosowania dekompozycji 2. rodzaju nie ma potrzeby tworzenia specjalnej relacji *Fragment CU w węzle* przy tworzeniu modelu zewnętrznego, a różnica w zajętości pamięci przez CU otrzymane w wyniku dekompozycji 1-go i 2-go rodzajów jest nieznaczna lub żadna w przypadku całkowitego rozproszenia.

References

[1] Bubnicki, Z. ‘Knowledge-Based Approach as a Generalization of Pattern Recognition Problems and Methods’. Systems Science Vol. 19, No 2 (1993), pp. 5–21
[2] Flasiński, M. Wstęp do sztucznej inteligencji. Warsaw: PWN, 2011
[3] Kurzyński, M. Algorytmy rozpoznawania wieloetapowego oraz ich zastosowania medyczne i techniczne. Wrocław: Wyd. PWr., 1987
[4] Józefczyk, J. ‘Rozpoznawanie i zastosowania biomedyczne’ [in:] Problemy automatyki i informatyki, Wrocław: Wyd. Ossolineum, 1998, pp. 45–58
[5] Lebiediewa, S, ‘System informatyczny dla wieloetapowego rozpoznawania obiektów’. Biuletyn Naukowy WWIS. Informatyka, 2014
[6] Lebiediewa, S, ‘Training sequence decomposition’. Biuletyn Naukowy WWIS. Informatyka, 2015
[7] Lebiediewa, S. Metodologia projektowania problemowo zorientowanych baz danych do systemów wielostopniowego podejmowania decyzji. Wrocław: Wyd. Pwr., 1998