STUDIA INFORMATICA 2008
Volume 29 Number 1 (75)

Marcin GORAWSKI, Dawid TLATLIK
Politechnika Śląska, Instytut Informatyki

CBA-DRZEWA – KOMPAKTOWA STRUKTURA DLA INDEKSOWANIA PRZESTRZENNYCH OBIEKTÓW NIEPUNKTOWYCH

Streszczenie. Artykuł ten prezentuje nowe struktury danych: CBA-drzewa i QCBA-drzewa zaprojektowane jako alternatywa dla BA-drzew eliminująca niektóre spośród ich wad. Omówiona została ogólna charakterystyka wprowadzonych struktur, a także przeprowadzone dla nich testy porównawcze. Opisany został również problem agregacji przestrzennej, dla którego w głównej mierze adresowane są przedstawione rozwiązania.

Słowa kluczowe: indeksowanie, sumy zdominowane, agregacja przestrzenna

CBA-TREE – COMPACT STRUCTURE FOR SPATIAL OBJECTS INDEXING

Summary. This article presents new data structures for both: CBA and QCBA-trees designed as an options for BA-tree and as an elimination of several flows. As part of this article, general description of introduced structures was not only discussed but also compared by tests. In addition to this, the problem of spatial aggregation, for which the solutions are mainly addressed, was described.

Keywords: indexing, dominance sum, spatial aggregation

1. Wstęp

Problem agregacji przestrzennej obejmuje zagadnienia wyliczania zagregowanych zapytań dla obiektów o niezerowym obszarze w przestrzeni wielowymiarowej. Formalnie możemy go zdefiniować jako: „*dla danego zbioru n obiektów prostokątnych i prostokątnego obszaru zapytania R wyznaczyć sumę wszystkich obiektów przecinających R*”. Dotychczasowa

praca skupiała się przede wszystkim na efektywnej agregacji obiektów punktowych bądź interwałów czasowych, całkowicie pomijając rozważania dla obiektów posiadających określony rozmiar. Dla wielu przestrzennych i czasowo-przestrzennych aplikacji obiekty rozszerzyły się jednak do wielu wymiarów i mogą zostać ulokowane w dowolnym miejscu zdefiniowanej dla nich przestrzeni. Przykładowo rozważmy bazę danych utworzoną dla agencji rolniczej, która przechowuje ślady użycia pestycydów. Każdy rekord reprezentuje działania na określonym obszarze oraz czasie i zdefiniowany jest przez 3-wymiarowy prostopadłościan (tj. obszar 2-wymiarowy reprezentujący spryskany obszar oraz odpowiadający mu przedział czasowy). W artykule tym rozpatrywana jest agregacja *box-sum,* czyli bezpośrednio związana z sumowaniami typu SUM, COUNT i AVG. Przykładem zapytania *box-sum* dla przedstawionej bazy jest *„znajdź łączną objętość pestycydu spryskiwanego w gminie Rudziniec k. Gliwic w maju 2007 roku”.*

Bezpośrednie wyznaczanie zapytań *box-sum* można zrealizować przez indeksowanie danych metodą wielowymiarowego dostępu, przykładowo za pomocą drzewa *R** [1]*,* a następnie redukcję tych zapytań do problemu wyszukiwania. Agregat wyliczany jest jako suma wszystkich obiektów przecinających okno zapytania. Wydajność takiego rozwiązania jest jednak bezpośrednio uzależniona od liczby obiektów przecinających okno i może gwałtownie spadać wraz ze wzrostem jego rozmiaru. Liczbę odwiedzanych wierzchołków można zredukować przez umieszczanie częściowych agregacji w odpowiednich wierzchołkach drzewa (agregacyjne R-drzewa), jednak również w tym przypadku wydajność bezpośrednio zależy od rozmiaru wskazanego okna.

W przedstawionej pracy wykorzystywane zostało odmienne podejście, które redukuje opisywane zagadnienia do problemu sum zdominowanych. Wykorzystywana jest przy tym zoptymalizowana metoda redukcji wyprowadzona w pracy [2]. W pracy tej autorzy proponują nową strukturę danych nazwaną przez nich BA-drzewem (*Box Aggregation Trees),* które do tej pory jest najwydajniejszym rozwiązaniem. Drzewo to posiada bardzo wysoki koszt utworzenia i jak pokazane zostało w przedstawionej pracy, jego praktyczne wykorzystanie ogranicza się do 2 wymiarów. W ramach niniejszego artykułu zaprojektowane zostały w nowe (własne) struktury danych, nazwane odpowiednio CBA-drzewem i QCBA-drzewem, które modyfikują BA-drzewa celem wyeliminowania lub znacznego ograniczenia niektórych jego wad, a także rozszerzenia na większą liczbę wymiarów.

2. Redukcja sumy przestrzennej

2.1. Problem prostej sumy pudełek

Rozróżniamy dwa typy obiektów: obiekty *punktowe* oraz *pudełkowe*. Weźmy dwa *d*-wymiarowe punkty $x = (x_1, \ldots, x_d)$ i $y = (y_1, \ldots, y_d)$, mówimy, że *x dominuje y*, jeśli dla każdego $i \{1, \ldots, d\}$, $x_i \geq y_i$. *D*-wymiarowe pudełko *b* możemy opisać za pomocą dwóch wartości wierzchołkowych: *punktu niskiego*, który jest zdominowany przez wszystkie pozostałe punkty wierzchołkowe *b*, oraz *punktu wysokiego*, który dominuje wszystkie pozostałe punkty wierzchołkowe *b*. Przestrzeń *d*-wymiarowa jest sama w sobie pudełkiem, którego punkty niski i wysoki są odpowiednio reprezentowane przez p_{min} i p_{max}. Wartość każdego z obiektów jest wykorzystywana do agregacji.

Oznaczmy następujące agregaty [15]:

- **(prosta) suma pudełek ((simple) box-sum):** dany jest zbiór S_b obiektów pudełkowych oraz pudełko zapytania *q*, oblicza się SUM{*o.wartość* | $o \in S_b$ i *o.pudełko* przecina *q*},
- **suma zakresów (range sum):** dany jest zbiór S_p obiektów punktowych oraz pudełko zapytania *q*, oblicza się SUM{*o.wartość* | $o \in S_p$ i *o.punkt* zawiera się w *q*},
- **suma zdominowana (dominance-sum):** dany jest zbiór S_p obiektów punktowych oraz punkt zapytania *p*, oblicza się SUM{*o.wartość* | $o \in S_p$ i *o.punkt* jest zdominowany przez *p*}.

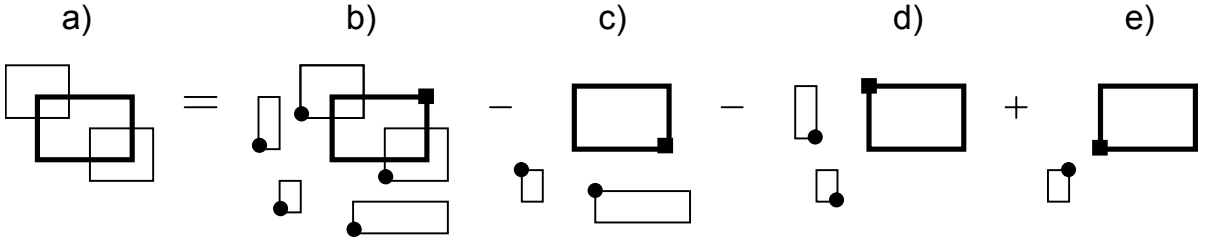
Problem sumy pudełek jest najbardziej ogólny, ponieważ: (1) problem sumy zakresów jest szczególnym przypadkiem sumy pudełek, jeśli pudełko każdego obiektu zredukować do punktu oraz (2) problem sumy zdominowanej jest szczególnym przypadkiem problemu sumy zakresów z pudełkiem zapytania $q = (p_{min}, p)$. Problemy agregacji COUNT (liczebność pudełek, liczebność zakresów i liczebność zdominowana) są szczególnymi przypadkami agregacji SUM, jeśli przyjąć, że wartość każdego obiektu wynosi 1.

2.2. Redukcja sumy pudełek

W pracy [2] autorzy proponują zupełnie nową technikę redukcji. Wprowadzony przez nich algorytm pozwala zredukować *d*-wymiarowe zapytanie *box-sum* do dokładnie 2^d zapytań *dominancie-sum*. Nawet dla małych wartości *d* przedstawiona redukcja dostarcza zdecydowanej poprawy względem stosowanego wcześniej podejścia wymagającego $\Omega(3^d / \sqrt{d})$ zapytań [2]. Przykładowo dla $d = 3$ oryginalna metoda wymagała 26 zapytań, podczas gdy prezentowana poniżej zaledwie 8.

Intuicyjnie *d*-wymiarowe pudełko ma 2^d wierzchołków. Dla każdego z obiektów przechowywany jest indeks każdego wierzchołka (np. prawego, górnego). Zapytanie *box-sum* jest

redukowane do 2^d zapytań *dominance-sum*, jednego dla każdego wierzchołka pudełka zapytania. Jako przykład rozważmy pudełko zapytania (oznaczone pogrubioną obwódką) i dwa obiekty przecinające go w przestrzeni 2-wymiarowowej przedstawione na rysunku 1.



Rys. 1. Dwuwymiarowe zapytanie *box-sum* redukowane do czterech zapytań *dominance-sum* [2]
Fig. 1. A 2-dimensional *box-sum* query is reduced to four dominance-sum queries [2]

Zapytanie *box-sum* wylicza wartość całkowitą tych dwóch obiektów. Oznaczając, że pudełko b przecina pudełko zapytania *q*: lewy, dolny wierzchołek *b* na pewno jest zdominowany przez prawy, górny wierzchołek *q*. Rysunek 1b przedstawia kandydujące pudełka. Niektóre z nich są złymi trafieniami, ponieważ znajdują się całkowicie na lewo lub całkowicie poniżej *q*. Te znajdujące się pod *q* odpowiadają tym pudełkom, których lewe, górne wierzchołki są zdominowane przez prawy, dolny wierzchołek *q* (rys. 1c). Natomiast te zupełnie na lewo od *q* odpowiadają tym, których prawe, dolne wierzchołki są zdominowane przez lewy, górny wierzchołek *q* (rys. 1d). Zauważmy, że po odjęciu tych fałszywych trafień od wyniku zapytania, pudełka których prawe, górne wierzchołki są zdominowane przez lewy, dolny wierzchołek *q* (rys. 1e) są odejmowane dwukrotnie. Więc ich całkowita wartość musi zostać ponownie dodana do wyniku końcowego.

Wprowadźmy zgodnie z pracą [2] odpowiednią notację umożliwiającą usystematyzowanie przedstawionych rozważań: niech *q* oznacza pudełko zapytania, *S* zbiór obiektów i $o \in S$ odpowiada obiektowi. Oznaczając ($o.l_1$, ..., $o.l_d$) oraz ($o.h_1$, ..., $o.h_d$) przedstawiamy odpowiednio punkt niski oraz punkt wysoki prostokąta. Analogicznej notacji używamy w stosunku do *q*. Ponadto dla pewnego wymiaru *i* definiujemy $A_i^0(o,q) \equiv o.l_1 < q.h_i$. Czyli $A_i^0(o,q)$ jest warunkiem tego, że punkt niski obiektu *o* jest zdominowany przez punkt wysoki pudełka zapytania *q* w *i*-tym wymiarze. Podobnie definiujemy $A_i^1(o,q) \equiv o.h_1 < q.l_i$ co jest warunkiem na to, że punkt wysoki obiektu *o* jest zdominowany przez punkt niski pudełka zapytania *q* w *i*-tym wymiarze. Obiekt *o* przecina zapytanie *q,* jeśli dla każdego wymiaru *i* projekcje *o* i *q* do wymiaru *i* posiadają część wspólną. Projekcja pudełka do wymiaru jest przedziałem, a dwa przedziały i_1 i i_2 mają część wspólną wtedy i tylko wtedy, gdy zachodzi $i_1.niski < i_2.wysoki$ i nie zachodzi $i_1.wysoki < i_2.niski$. Ostatecznie zapytanie *sum-box* możemy wyrazić zgodnie z pracą [2] jako:

$$boxsum\ (S,q) = \sum_{\substack{\forall (s_1,\ldots,s_d),\\ s_i \in \{0,1\}}} (-1)^{\sum_{i=1}^{d} s_i} \cdot Sum\left\{ o.wartosc \ \middle|\ o \in S \wedge \bigcap_{i=1}^{d} A_i^{s_i}(o,q) \right\} \qquad (1)$$

3. BA-drzewa

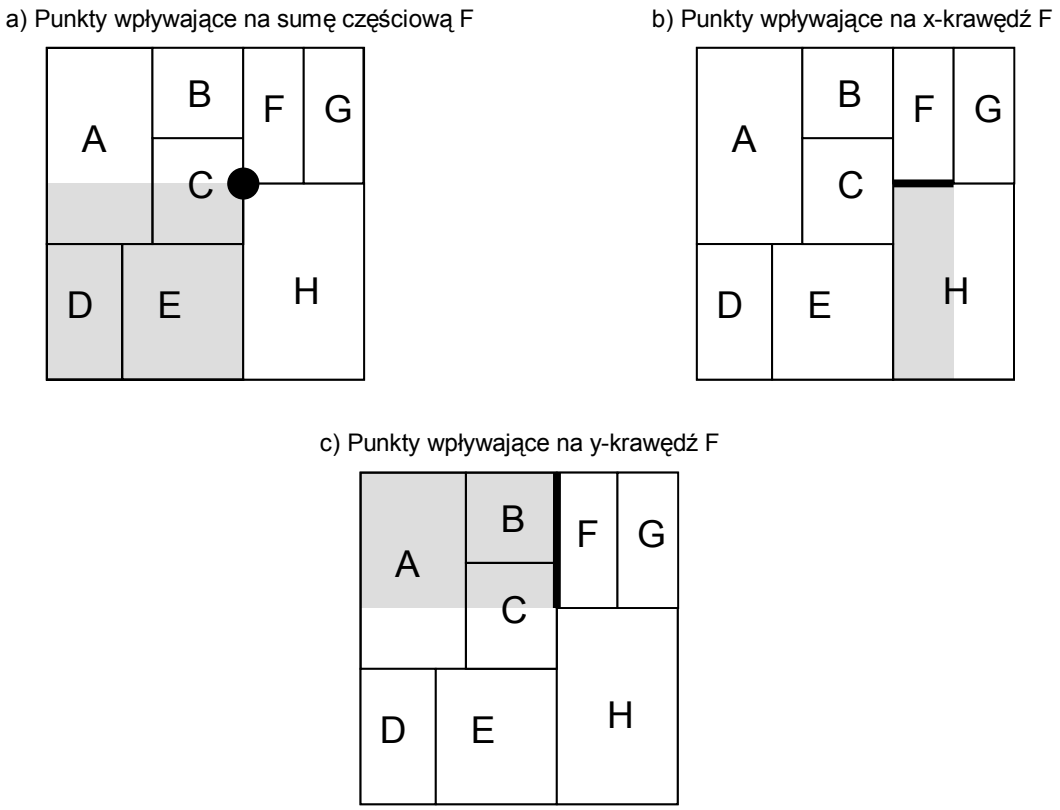
W celu optymalnego wyznaczania sum zdominowanych w pracy [2] autorzy proponują nową strukturę danych, nazwaną przez nich BA-drzewem. BA-drzewo jest strukturą indeksującą, która może zostać zrealizowana za pomocą pamięci dyskowej. Nie przechowuje ona bezpośrednio oryginalnych danych, zamiast tego posługuje się odpowiednimi agregatami pozwalającymi właściwie je scharakteryzować. Przeznaczona jest dla popularnego modelu historycznych hurtowni danych, który pozwala wyłącznie dodawać dane. Do wspieranych operacji należą zatem dodawanie oraz wyszukiwanie danych.

3.1. Opis struktury

BA-drzewo zostało bezpośrednio oparte na K-D-B-drzewie (opisane w pracy [3]), które rozszerzono następnie o pewne informacje krawędziowe. Przykładowy węzeł indeksu BA--drzewa w przestrzeni 2-wymiarowej został przedstawiony na rys. 2. Podobnie jak w k-d-B--drzewie występują dwa typy stron: strony punktów oraz strony regionów. W stronach punktów przechowywane są punkty, natomiast strony regionów reprezentują wewnętrzną strukturę indeksu i przechowują odpowiednie regiony. Regiony poszczególnych rekordów w węźle nie mają części wspólnej, a ich suma tworzy całkowity obszar węzła. Ponieważ *węzeł* jest zaimplementowany jako *strona*, używamy tych pojęć przemiennie. Każdy rekord *r* wskazuje na poddrzewo, zawierające punkty znajdujące się w *r.box*.

Wszystkie rekordy indeksu powiększamy o pewną informację krawędziową. Zabieg ten ma na celu doprowadzenie do takiej sytuacji, w której dowolne zapytanie *dominance-sum* może zostać zrealizowane przez przeszukanie pojedynczego poddrzewa (w gałęzi głównej).

Przypuśćmy, że na rys. 2 punkt zapytania znajduje się w pudełku rekordu *F*. Punktami, które mogą mieć wpływ na wynik zapytania *dominance-sum*, są wszystkie te, które zdominowane zostały przez prawy górny punkt pudełka *F.box*. Takie punkty można podzielić na cztery grupy: (1) punkty zawarte w *F.box*; (2) punkty zdominowane przez punkt niski *F* (zacieniony obszar na rys. 2a); (3) punkty znajdujące się poniżej dolnej krawędzi *F.box* (rys. 2b); oraz (4) punkty znajdujące się na lewo od lewej krawędzi pudełka *F.box* (rys. 2c).



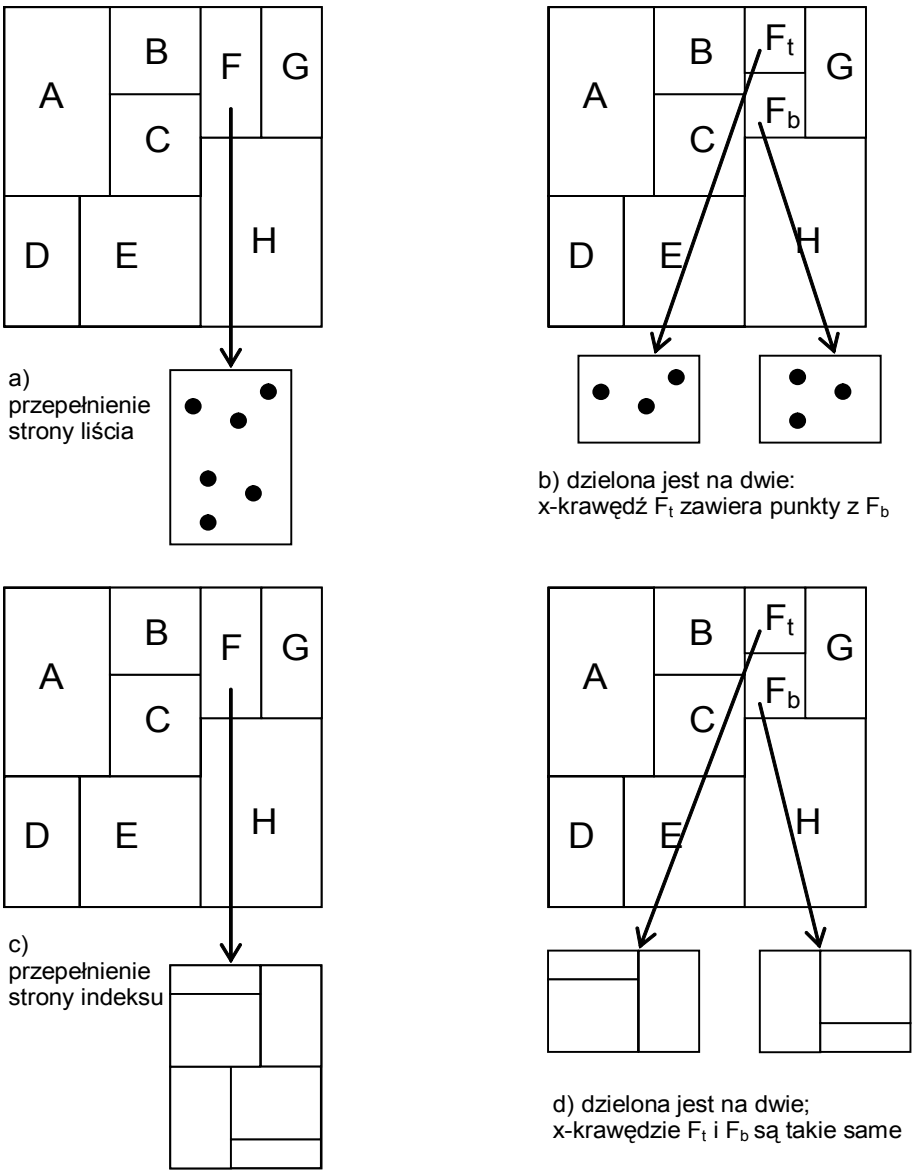
Rys. 2. BA-drzewo jako K-D-B-drzewo powiększone o informacje krawędziowe [2]
Fig. 2. The BA-tree is a K-D-B-tree with augmented border information [2]

W celu wyznaczenia sumy zdominowanej dla punktów z pierwszej grupy wykonywane jest rekurencyjne przeglądanie poddrzewa (F). Dla punktów z drugiej grupy rezerwujemy w rekordzie F pojedynczą wartość zwaną *sumą częściową*, która jest łączną sumą wszystkich tych punktów. Celem obliczenia sumy zdominowanej w trzeciej grupie tworzymy x-*krawędź* dla F zawierającą pozycje x i wartości wszystkich punktów należących do tego zbioru. Szukana jest suma zdominowana, następnie redukowana do 1-wymiarowego zapytania dla tej krawędzi. Do przechowywania wskazanych pozycji x możemy wykorzystać 1-wymiarowe BA-drzewo. Analogicznie dla punktów z czwartej grupy tworzymy y-kr*awędź*, będącą 1-wymiarowym BA-drzewem obejmującym pozycje y punktów wszystkich elementów tego zbioru.

3.2. Przepełnienia

Rozróżniamy dwa przypadki, w zależności od tego, czy przepełnienie występuje na stronie liścia, czy indeksu. Rysunek 3 przedstawia stronę liścia wskazywaną przez rekord F, w której występuje przepełnienie. Strona ta zostaje podzielona na dwie strony: F_t i F_b (rys. 3b). Z uwagi na to, że jest to przepełnienie y, krawędź y rekordu F podzielona zostaje na dwie krawędzie, jedną przechowywaną w F_t i drugą w F_b. Krawędź x dolnego rekordu F_b

pozostaje taka sama jak dla rekordu F. Jednakże krawędź x górnego rekordu F_t składa się z krawędzi x rekordu F oraz z punktów na stronie F_b. Rysunek 3c rozważa przypadek, w którym przepełniona została strona regionów. Wynik podziału pokazany jest na rys. 4d.



Rys. 3. Przepełnienia w BA-drzewie [2]
Fig. 3. Split in the BA-tree [2]

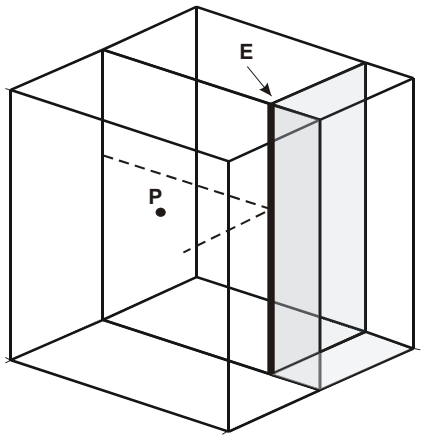
Tak jak poprzednio krawędź y rekordu F podzielona została na dwie oddzielne, należące do nowych rekordów. Jednak inaczej niż w przypadku podziału liścia krawędzie x obu rekordów F_t i F_b pozostają takie same jak dla rekordu F.

W celu zweryfikowania poprawności takiego podziału, rozważmy zapytanie *dominance-sum*, w którym punkt zapytania p zawarty jest w pudełku $F_t.box$. Oczywiście punkty w poddrzewie (F_b) powinny mieć wpływ na wynik zapytania. Jednak krawędź x rekordu F_t została skopiowana z F, a co za tym idzie nie zawiera żadnego punktu z obszaru F_b przed podziałem.

Nie stanowi to problemu, gdyż zapytanie rekurencyjnie zbada stronę indeksu wskazywaną przez F_t, gdzie informacja zawarta w krawędzi zawiera punkty z obszaru F_b.

3.3. Rozszerzenie wymiarowości

W pracy [2] autorzy twierdzą, że strukturę BA-drzewa można bezpośrednio rozszerzyć również na większą liczbę wymiarów. W swoich założeniach przyjęli oni, że *d*-wymiarowe BA-drzewo jest K-D-B-drzewem, w którym każdemu rekordowi przypisana została pojedyncza wartość sumy częściowej oraz *d* różnych krawędzi, z których każda zrealizowana została opierając się na $(d-1)$-wymiarowym BA-drzewie. Takie podejście obarczone jest niestety poważnym błędem. Problem ten zilustrowany został na rys. 3, na którym umieszczono przykładowy punkt P, znajdujący się w 3-wymiarowym BA-drzewie. Zgodnie z oryginalnym algorytmem wstawiania danych dla BA-drzew punkt ten nie zostanie włączony do żadnej krawędzi regionu, a także sumy częściowej, niemniej jego wartość jest niezbędna do prawidłowego wyznaczenia sumy zdominowanej dowolnego punktu, znajdującego się w wyróżnionym regionie. Eliminacja tego błędu opiera się na wprowadzeniu dodatkowej 1-wymiarowej krawędzi *E*. Krawędź ta obejmuje wszystkie punkty, które nie znalazły się w pozostałych krawędziach regionu, jednak zdominowane zostały przez punkty bezpośrednio należące do E. Reprezentujemy ją za pomocą 1-wymiarowego BA-drzewa. Wprowadzona modyfikacja powoduje konieczność zmiany algorytmu wyznaczania sumy zdominowanej, która od teraz do wyniku końcowego włączać musi również wartość zapytania *dominancie-sum* wykonanego dla dodatkowych krawędzi rogowych.



Rys. 4. Przykładowy węzeł dla 3-wymiarowego BA-drzewa
Fig. 4. Node for 3-dimensioanl BA-tree

Uogólniając rozważania na większa liczbę wymiarów, przyjmujemy, że *d*-wymiarowe BA-drzewo jest K-D-B-drzewem, w którym każdemu rekordowi przypisana została pojedyncza wartość sumy częściowej, *d* różnych krawędzi, z których każda zrealizowana została

opierając się na ($d-1$)-wymiarowym BA-drzewie, a także pojedynczej ($d-2$)-wymiarowej krawędzi rogowej zrealizowanej na ($d-2$)-wymiarowym BA-drzewie.

4. CBA-drzewa

Opisane w poprzednim rozdziale BA-drzewa posiadają kilka wad. Najpoważniejszą z nich jest ogromny koszt obliczeniowy związany z utworzeniem pełnej struktury drzewa. Koszt ten bezpośrednio wynika z konieczności podziału krawędzi, który wymaga zastosowania stosunkowo złożonej operacji podziału BA-drzew, komplikującej się dodatkowo dla większej liczby wymiarów. Kolejną wadą BA-drzew jest brak zrównoważenia, w wyniku którego złożoność odpowiedzi dla najgorszego przypadku może być nawet liniowa. Brak zrównoważenia powoduje również ograniczenie możliwości stosowania BA-drzew ze względu na trudność oszacowania czasu odpowiedzi dla pojedynczego zapytania, co uniemożliwia ich efektywnie wykorzystanie w przypadku optymalizatora zapytań. BA-drzewo zostało bezpośrednio oparte na K-D-B-drzewie, więc również wprowadza pewną redundancję konieczną do reprezentacji węzłów wewnętrznych. Co więcej, realizacja BA-drzew jest dość skomplikowana, szczególnie dla większej liczby wymiarów, co wiąże się ze stosunkowo dużą podatnością na wszelkiego rodzaju błędy.

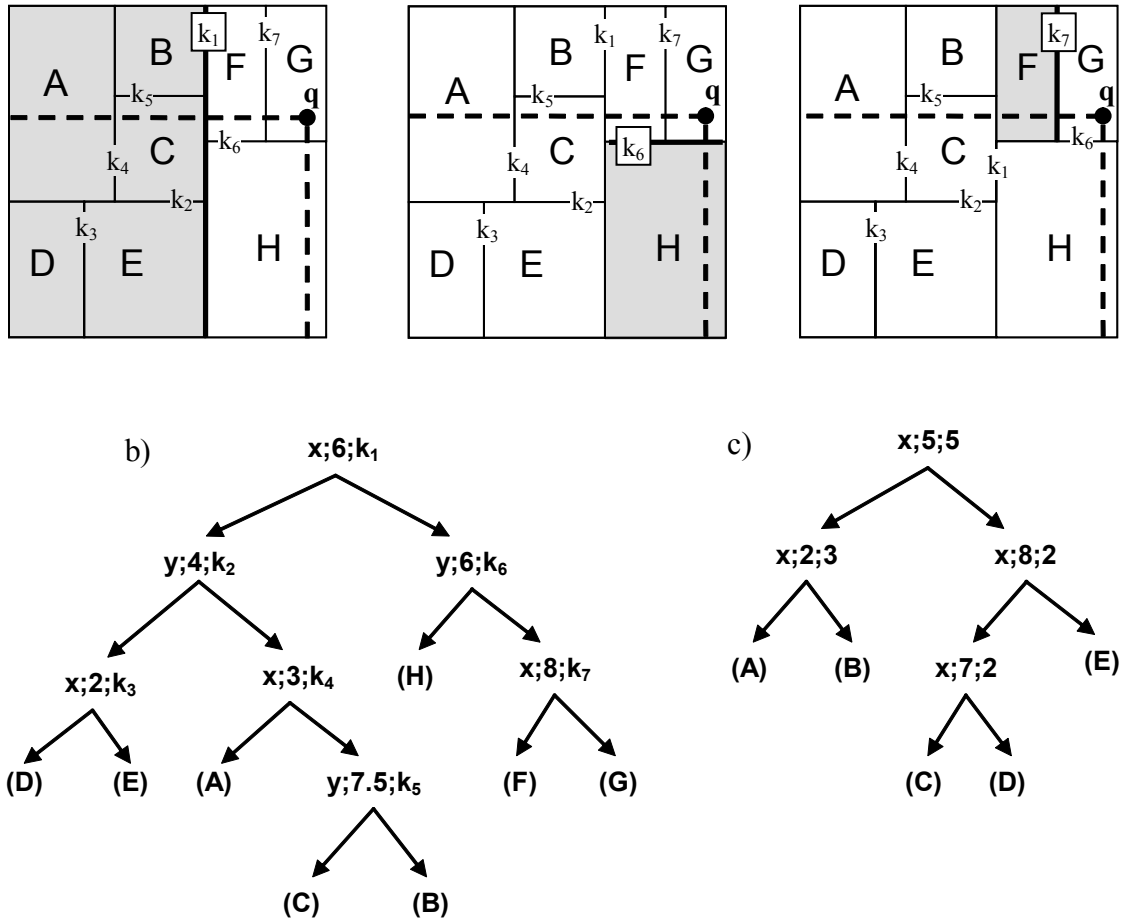
Wszystkie przedstawione wady BA-drzew były motywacją opracowania nowej struktury danych, nazwanej przez autorów CBA-drzewem (ang. *Compact Box Aggregation Tree*, *CBA--tree*), która stara się wyeliminować lub znacznie ograniczyć wymienione wcześniej wady. Przeznaczenie CBA-drzew jest identyczne jak BA-drzew, co pozwala wykorzystywać obie struktury do tej samej klasy zadań.

4.1. Opis struktury

W przeciwieństwie do BA-drzew opartych na niezrównoważonym KDB-drzewie podstawą opracowanego CBA-drzewa jest drzewo KDB_{KD} [4]. Takie rozwiązanie ma wiele zalet. Przede wszystkim KDB_{KD}-drzewo jest drzewem zrównoważonym, wykorzystuje efektywną metodę FD podziału węzłów wewnętrznych (ang. *First Division,* w przypadku przepełnienia węzłów wewnętrznych podział węzła wzdłuż pierwszej płaszczyzny podziału) oraz usuwa redundancję w opisie węzłów wewnętrznych. Stosowana dla nich metoda organizacji krawędzi eliminuje występowanie krawędzi rogowych opisanych w poprzednim rozdziale. Największą zaletą CBA-drzew jest jednak całkowite wyeliminowanie konieczności podziału krawędzi, co zdecydowanie poprawia efektywność operacji tworzenia drzewa, a także umożliwia wykorzystanie tej struktury dla dowolnej liczby wymiarów. Tak jak pokazane zostało

w poprzednim rozdziale, z praktycznego punktu widzenia BA-drzewa nadają się wyłącznie dla danych 2-wymiarowych, ponieważ ich koszt utworzenia dla większej liczby wymiarów jest zbyt duży.

Podstawową cechą KDB_{KD}-drzew jest wykorzystywanie KD-drzew [5] do opisu węzłów wewnętrznych. W opisywanej strukturze każdy węzeł KD-drzewa rozszerzony został o pewną dodatkową informację zależną od liczby wymiarów. Dla jednego wymiaru informacja ta reprezentuje sumę wszystkich punktów, znajdujących się na lewo od węzła, natomiast dla ich większej liczby stanowi wskaźnik do krawędzi przechowującej wszystkie punkty, leżące po lewej stronie płaszczyzny podziału przypisanej do węzła (tzn. wszystkie punkty do poprzedzającej równoległej płaszczyzny podziału). Przykładowy węzeł indeksu CBA-drzewa w przestrzeni 2-wymiarowej zaprezentowany został na rys. 5.



Rys. 5. Struktura CBA-drzewa: a) krawędzie mające wpływ na wynik zapytania dla dowolnego punktu regionu G, b) reprezentacja węzła za pomocą KD-drzewa, c) przykładowy węzeł dla drzewa 1-wymiarowego

Fig. 5. The structure of CBA-tree: a) borders having impact on the result of the query for region G, b) the representation of node with KD-tree, c) node for 1-dimensional tree

Podobnie jak dla KDB_{KD}-drzew występują tutaj dwa typy stron: strony punktów oraz strony regionów. W stronach punktów przechowywane są punkty, natomiast strony regionów

reprezentują wewnętrzną strukturę indeksu i przechowują odpowiednie regiony. Poszczególne pudełka wyznaczone są przez odpowiednie płaszczyzny dzielące KD-drzewa. Regiony poszczególnych rekordów w węźle nie mają części wspólnej, a ich suma tworzy całkowity obszar węzła. Ponieważ *węzeł* jest zaimplementowany jako *strona*, używamy tych pojęć przemiennie. Każdy liść KD-drzewa wskazuje na poddrzewo obejmujące obszar przypisany węzłowi.

Wszystkie węzły KD-drzewa powiększamy o pewną informację krawędziową. Krawędzie te realizowane są za pomocą (d−1)-wymiarowych CBA-drzew. Dzięki temu zabiegowi dowolne zapytanie *dominance-sum* może zostać zrealizowane na podstawie przeszukania pojedynczego poddrzewa (w gałęzi głównej).

Rozważmy dla przykładu, że na rys. 5 punkt zapytania znajduje się w regionie G. Punktami, które mogą mieć wpływ na wynik zapytania *dominance-sum*, są wszystkie te, które zdominowane zostały przez prawy górny punkt pudełka G oraz punkty znajdujące się w samym pudełku. W celu wyznaczenia sumy wszystkich punktów znajdujących się w pierwszej grupie sumujemy odpowiedzi *dominance-sum* odpowiednich krawędzi leżących na ścieżce dostępu do pudełka G. Dla przedstawionego przykładu będą to krawędzie k_1, k_6, k_7. Odpytywane są tylko te krawędzie, dla których w kolejnym kroku przechodzimy do prawego potomka. Dla punktu zapytania znajdującego się w regionie C wybrane zostałyby krawędzie k_2 i k_6. Punkt zapytania dla wszystkich krawędzi wyznaczany jest przez usunięcie odpowiedniego wymiaru z punktu zapytania q – wymiaru definiującego ortogonalną płaszczyznę podziału przypisaną do węzła. Przykładowo, dla krawędzi k_1 punkt zapytania powstaje przez usunięcie wymiaru x z punktu q, natomiast dla krawędzi k_6 przez usunięcie wymiaru y. W przypadku jednego wymiaru algorytm wygląda podobnie, z tą różnicą, że zamiast zapytań *dominancie-sum* dodajemy sumy częściowe przypisane do węzłów. Przykładowo, dla pustego liścia D na rys. 5c wynik końcowy wyniósłby $5 + 2 = 7$.

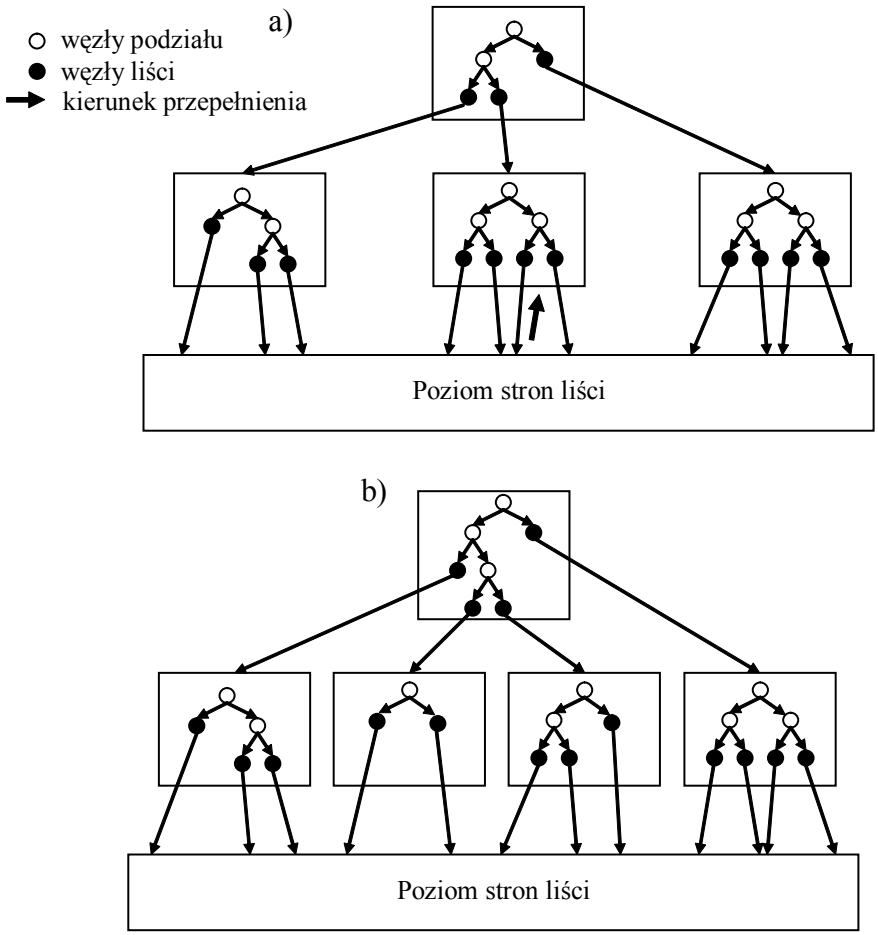
Sposób wyznaczenia punktów zdominowanych dla samego pudełka G zależy od jego typu. Dla strony regionów wykonywane jest rekurencyjne przeglądanie poddrzewa G, natomiast dla strony punktów wartość tę wyznaczamy bezpośrednio.

4.2. Przepełnienia

Największą zaletą CBA-drzew w stosunku do BA-drzew jest całkowite wyeliminowanie operacji podziału krawędzi. W momencie wystąpienia przepełnienia dla strony liści F przedstawionej na rys. 6 zostaje podzielona ona na dwie nowe strony F_t i F_b, do których równomiernie przesunięte zostają wszystkie punkty. W kolejnym kroku utworzona zostaje nowa krawędź k, która zawiera wszystkie punkty, znajdujące się po lewej stronie płaszczyzny podziału, a więc w tym przypadku punkty strony F_b. W przypadku pokrywania się z płaszczy-

zną podziału któregoś z punktów strony F, dołączony zostaje on wyłącznie do krawędzi *k*. Tak utworzona krawędź wraz z płaszczyzną podziału zostaje następnie dodana do węzła rodzica jako kolejny węzeł reprezentującego go KD-drzewa, zastępując odpowiedni węzeł liścia.

W momencie wystąpienia przepełnienia dla węzłów wewnętrznych następuje ich podział wzdłuż pierwszej płaszczyzny podziału, co powoduje utworzenie nowych KD-drzew reprezentujących odpowiednie regiony. Węzeł korzenia zawierający płaszczyznę pierwszego podziału zostaje przesunięty do strony rodzica, natomiast jego potomkowie wskazują na nowo utworzone strony. Algorytm ten przedstawiony został na rys. 6.



Rys. 6. Przepełnienie węzłów wewnętrznych w CBA-drzewie: a) przed podziałem, b) po podziale węzłów
Fig. 6. Splitting an interior page: a) before split, b) after split

4.3. Złożoność obliczeniowa

Dokładna analiza złożoności CBA-drzew pokazuje podstawową ich słabość. Wprawdzie dane rozłożone są w sposób zrównoważony, jednak czas odpowiedzi na pojedyncze zapyta-

nie *dominance-sum* może się znacznie różnić w zależności od położenia punktu zapytania. Przyjmijmy liczbę punktów wynoszącą *N* oraz pojemność stron punktów ustaloną na wartość *B*. Przy takich parametrach liczba pojawiających się płaszczyzn podziału wynosi $K = N/B - 1$, a więc liczba odpytywanych krawędzi dla pojedynczego zapytania zawiera się w przedziale [0, *K*]. Średnia liczba odpytywanych krawędzi w tym przypadku wynosi więc $Avr_Q = K/2$ dla równomiernie rozłożonych danych. Analogiczną analizę można przeprowadzić dla operacji wstawiania, która dla najgorszego przypadku wymaga aktualizacji *K* krawędzi, przy średniej ich liczbie wynoszącej $Avr_U = K/2$.

5. QCBA-drzewo

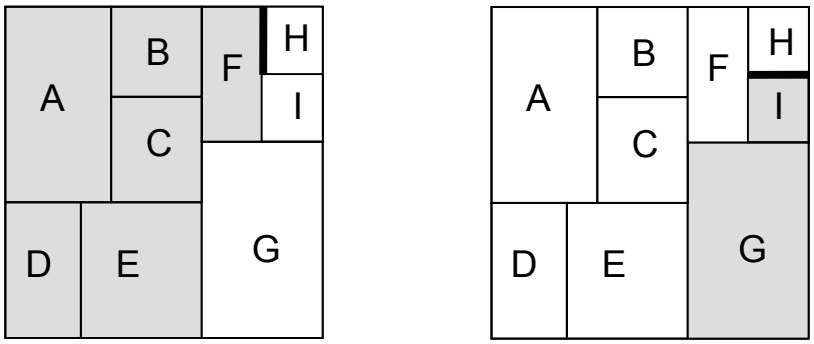
Opisane w poprzednim rozdziale CBA-drzewa pod wieloma względami przewyższają BA-drzewa. Jedynym obszarem, w którym posiadają gorsze parametry od swojego pierwowzoru, jest wydajność dla pojedynczych zapytań. Związane to jest z charakterystycznym sposobem organizowania danych, dla którego w najgorszym przypadku konieczne jest odpytanie wszystkich krawędzi podziału. Niedogodność tę próbuje wyeliminować zaprezentowana poniżej nowa struktura danych, nazwana QCBA-drzewem.

5.1. Opis struktury

Drzewo QCBA łączy ze sobą cechy CBA-drzew oraz BA-drzew. Do zarządzania węzłami wewnętrznymi wykorzystuje mechanizmy występujące w BA-drzewach i analogicznie do CBA-drzewa wykorzystuje KD-drzewa do opisu węzłów wewnętrznych, modyfikując jednak znaczenie poszczególnych krawędzi. W CBA-drzewie pojedyncza krawędź przechowuje informację o wszystkich punktach znajdujących się po lewej stronie płaszczyzny podziału, natomiast w QCBA-drzewie krawędź taka przechowuje punkty obejmujące tzw. kaskadę. Pojęcie kaskady stosowane jest do definiowania obszaru, który powstaje w wyniku zaproponowanego algorytmu tworzenia nowych krawędzi. Przykładowe kaskady utworzone dla krawędzi pudełka *H* przedstawione zostały na rys. 7. Obszar definiowany przez pojedynczą kaskadę zależy ściśle od hierarchii kolejnych płaszczyzn podziału. Dla d-wymiarowego drzewa dowolny region może posiadać maksymalnie *d* zdefiniowanych kaskad. Wszystkie kaskady utworzone dla pojedynczego regionu łącznie obejmują obszar dominancji wysokiego punktu tego regionu, tak więc pozwalają wyznaczyć sumę zdominowaną dla wszystkich punktów znajdujących się na zewnątrz pudełka.

Kaskada powstaje w wyniku charakterystycznej metody tworzenia krawędzi – w momencie przepełnienia strony punktów tworzona jest nowa krawędź, która w pierwszej kolejności

kopiuje krawędź przylegającą, a następnie uzupełnia ją o punkty lewej strony podziału. Krawędź przylegająca regionu dla danego wymiaru jest to taka krawędź, która pokrywa się z odpowiednim bokiem regionu. Krawędzią otaczającą w danym wymiarze nazywamy pierwszą krawędź określonego wymiaru, którą napotykamy przy przechodzeniu drzewa podziału począwszy od jego liścia aż do korzenia. Takie rozróżnienie wynika bezpośrednio z opisu węzła za pomocą KD-drzewa i okazuje się konieczne, ponieważ bardzo często krawędź przylegająca oraz krawędź otaczająca nie są sobie odpowiadające.



Rys. 7. Przykładowe kaskady zdefiniowane dla krawędzi pojedynczego regionu
Fig. 7. Cascades defined for the single region

Rozważmy dla przykładu, że na rys. 7 punkt zapytania znajduje się w pudełku H. Punktami, które mogą mieć wpływ na wynik zapytania *dominance-sum*, są wszystkie te, które zdominowane zostały przez prawy górny punkt pudełka *H (*z wyłączeniem jego zawartości) oraz punkty znajdujące się w samym pudełku. W celu wyznaczenia sumy wszystkich punktów znajdujących się w pierwszej grupie sumujemy odpowiedzi *dominance-sum* dla wszystkich krawędzi przylegających regionu *H*. Sposób wyznaczenia punktów zdominowanych dla samego pudełka *H* zależy od jego typu. Dla strony regionów wykonywane jest rekurencyjne przeglądanie poddrzewa *H*, natomiast dla strony punktów wartość ta może zostać wyznaczona bezpośrednio.

Przedstawiony sposób organizacji danych pozwala w znacznym stopniu ograniczyć liczbę odpytywanych krawędzi, ponieważ w trakcie realizacji zapytania dla pojedynczej strony indeksu odpytanych zostanie maksymalnie *d* krawędzi, gdzie *d* oznacza wymiarowość utworzonego drzewa. Punkt zapytania dla krawędzi wyznaczany jest analogicznie do CBA-drzew i polega na usunięciu odpowiedniego wymiaru z punktu zapytania *q*.

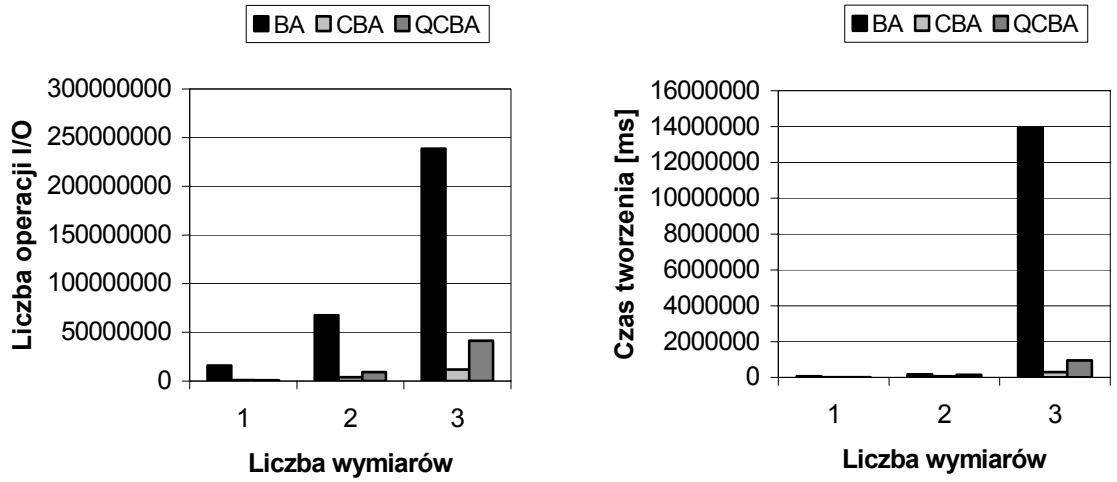
6. Testy

W ramach przeprowadzonych testów szczegółowej analizie poddane zostały opisane struktury danych w celu porównania ich z tradycyjnymi BA-drzewami. Ze względu na gwał-

towny wzrost rozmiaru przy większej liczbie wymiarów testy przeprowadzone zostały dla dwóch różnych zbiorów testowych: pierwszy z nich składający się z 700 tys. punktów o rozkładzie równomiernym służący do testowania maksymalnie dwóch wymiarów oraz drugi zbiór liczący 200 tys. punktów, wykorzystywany także dla 3 wymiarów. Liczebność przedstawionych zbiorów dobrana została w sposób gwarantujący możliwość całkowitego umieszczenia tworzonych struktur w pamięci fizycznej komputera, celem uwiarygodnienia wyników uzyskanych dla pomiarów czasu. Przyjęty rozmiar pojedynczej strony wyniósł 4 [kB]. Dla tak utworzonych struktur wykonane zostały następnie testy wydajności. Polegały one na wykonaniu 10000 zapytań *dominance-sum* dla losowo wybranych punktów o rozkładzie równomiernym i uśrednieniu otrzymanych wyników.

6.1. Tworzenie drzewa

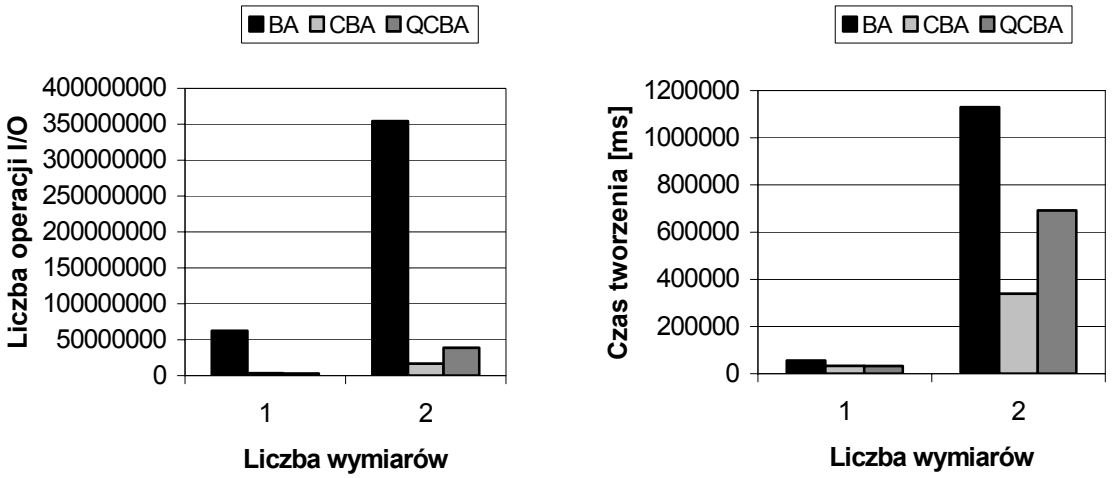
Wyniki testów porównujących cykl tworzenia prezentowanych struktur umieszczone zostały na rys. 8 i rys. 9. Uwydatniają one podstawową zaletę CBA-drzew, którą jest zdecydowanie najmniejszy koszt utworzenia spośród wszystkich prezentowanych rozwiązań. Otrzymane różnice są ogromne. Dla pierwszego zbioru 700 tys. punktów 2-wymiarowe CBA--drzewa potrzebują ponad 20-krotnie mniej operacji I/O. Jeszcze większe dysproporcje pojawiają się przy większej liczbie wymiarów, ponieważ taka sama relacja dla 3 wymiarów zachodzi już dla 200 tys. punktów i wraz ze wzrostem ich liczby jeszcze bardziej się pogłębia. Druga z opracowanych struktur, QCBA-drzewo, wypada już nieco gorzej od CBA-drzew, wymagając 3-krotnie więcej operacji I/O na etapie tworzenia drzewa, wynik ten nadal jest jednak zdecydowanie lepszy od rezultatów osiąganych przez BA-drzewa.



Rys. 8. Porównanie operacji tworzenia rodziny BA-drzew dla 200 tys. punktów
Fig. 8. Building trees for 200K points

Taki ogromny skok wydajności udało się osiągnąć dzięki całkowitemu wyeliminowaniu operacji dzielenia krawędzi charakterystycznych dla BA-drzew. Uzyskane wyniki pokazują,

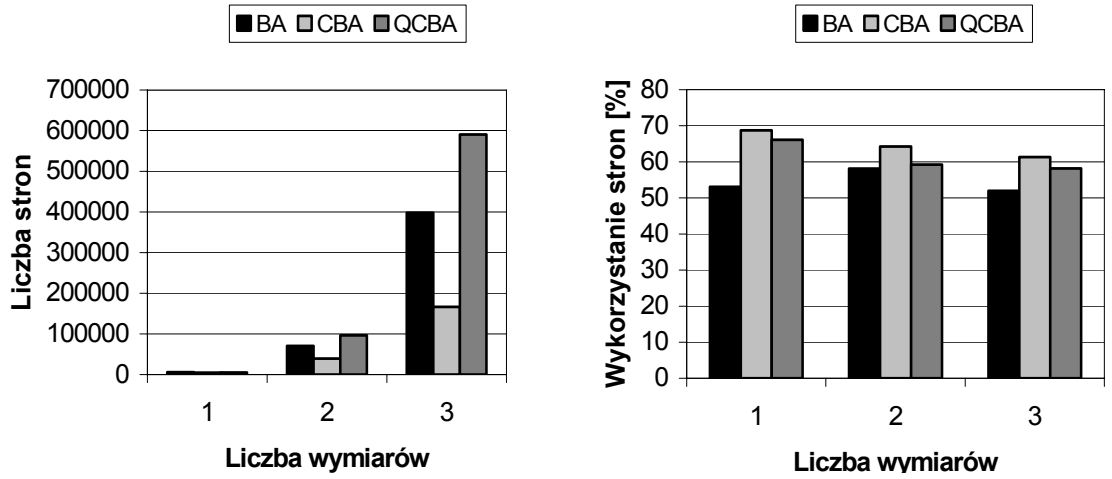
jak wysoki jest koszt tej operacji, szczególnie dla większej liczby wymiarów. W przypadku QCBA-drzew operacja podziału zastąpiona została kopiowaniem odpowiednich krawędzi, która okazuje się wydajniejsza, jednak jej koszt nadal pozostaje stosunkowo wysoki.



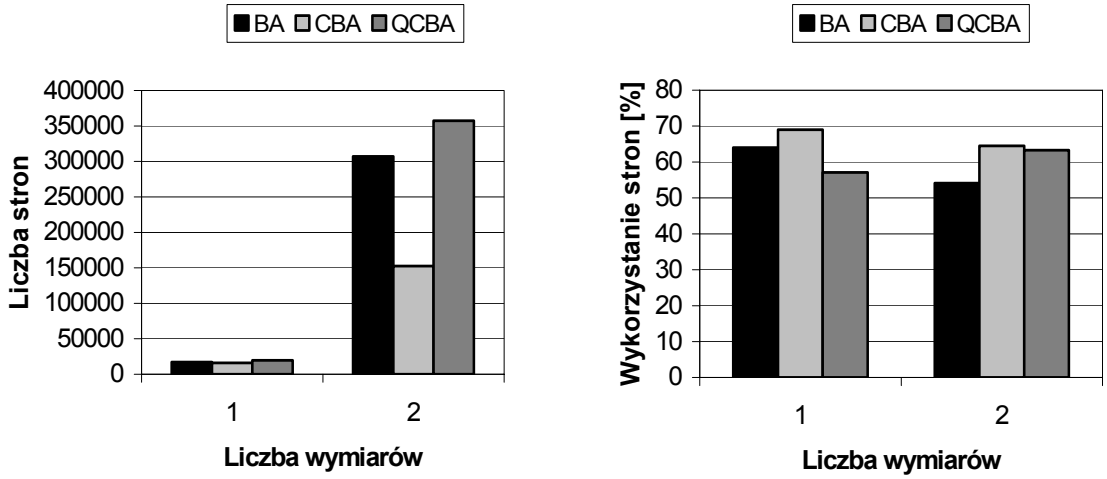
Rys. 9. Porównanie operacji tworzenia rodziny BA-drzew dla 700 tys. punktów
Fig. 9. Building trees for 700K points

6.2. Rozmiar struktury

Porównanie rozmiaru utworzonych struktur zaprezentowane zostało na rys. 10 oraz rys. 11. Zgodnie z otrzymanymi wynikami CBA-drzewa posiadają ponaddwukrotnie mniejszy rozmiar w stosunku do tradycyjnych BA-drzew, głównie dzięki wyeliminowaniu redundancji występującej w opisie krawędzi. Zastosowane w QCBA-drzewa algorytmy budowania krawędzi powodują znaczne zwiększenie rozmiaru całej struktury. Drzewo QCBA wymaga prawie dwa razy więcej pamięci dyskowej od BA-drzew, co przekłada się na 3-krotny wzrost rozmiaru względem CBA-drzew.



Rys. 10. Porównanie rozmiaru struktur rodziny BA-drzew dla 200 tys. punktów
Fig. 10. Size of trees for 200K points

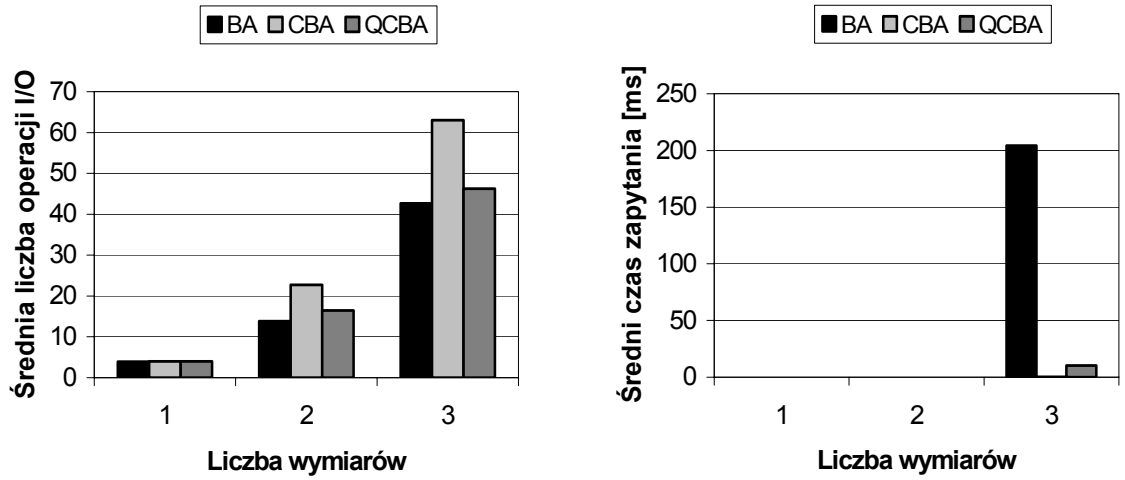


Rys. 11. Porównanie rozmiaru struktur rodziny BA-drzew dla 700 tys. punktów
Fig. 11. Size of trees for 700K points

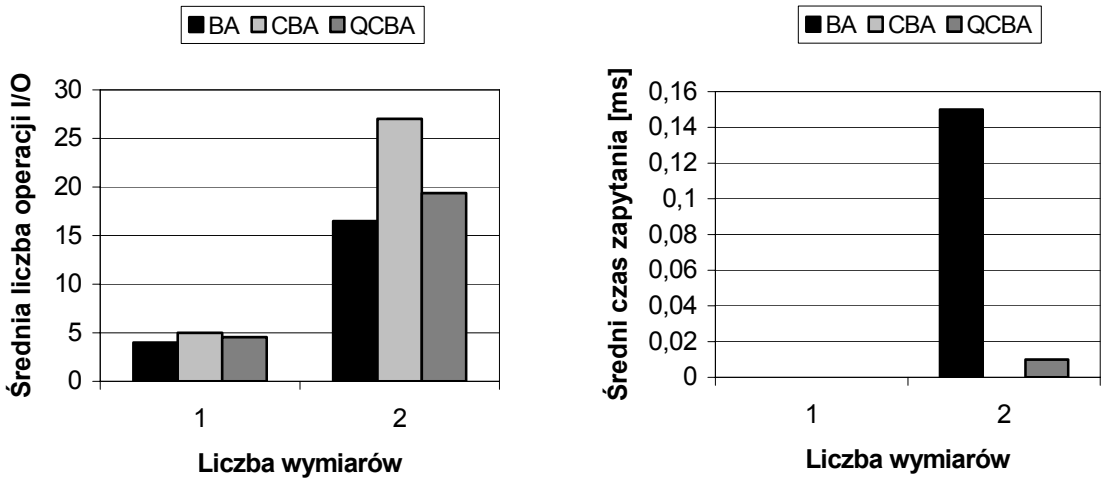
6.3. Zapytania dominance-sum

Zgodnie z prognozami wyeliminowanie redundancji dla CBA-drzew skutkuje koniecznością odpytywania większej liczby krawędzi, co z kolei przekłada się na wzrost średniej liczby operacji I/O dla pojedynczego zapytania. Wzrost ten jest dość znaczny, ponieważ wynosi blisko 50%, jednak nie zawsze musi się on bezpośrednio przedkładać na taki sam spadek wydajności. Jak pokazują wyniki pomiaru czasu z rys. 12 oraz rys. 13, przetwarzanie zapytań dla CBA-drzew jest znacznie wydajniejsze, ponieważ wykorzystywane do opisu węzłów wewnętrznych KD-drzewa pozwalają zredukować obliczenia wykonywane dla pojedynczej strony. Kolejnym aspektem, który powinien mieć pozytywny wpływ na wydajność CBA-drzew, jest korzystanie z buforowania. Przypuszczalnie kilka głównych krawędzi drzewa powinno być stale przechowywanych w buforze (przy zastosowaniu algorytmu LRU), eliminując tym samym koszt dostępu do części stron.

Wyniki uzyskane dla QCBA-drzew sugerują, że łączenie ze sobą krawędzi nie jest efektywną metodą zwiększania wydajności. Wprawdzie QCBA-drzewa mają zbliżoną do BA-drzew średnią liczbę operacji dla pojedynczych zapytań, jednak kosztem 50% wzrostu rozmiaru całej struktury.



Rys. 12. Porównanie zapytań *dominance-sum* dla rodziny BA-drzew przy 200 tys. punktów
Fig. 12. Query performance for 200K points



Rys. 13. Porównanie zapytań *dominance-sum* dla rodziny BA-drzew przy 700 tys. punktów
Fig. 13. Query performance for 700K points

7. Wnioski

W świetle uzyskanych wyników zaprezentowane nowe struktury danych stanowią interesującą alternatywę dla BA-drzew. Algorytmy zaprojektowane dla CBA-drzew pozwoliły na 2-krotne zmniejszenie rozmiaru całej struktury w stosunku do tradycyjnych BA-drzew, a także ponad 20-krotne zwiększenie wydajności procesu budowania drzewa. Odbyło się to jednak kosztem wzrostu o 50% średniej liczby operacji I/O dla pojedynczego zapytania *dominance-sum*. Trudno nie przeceniać tak otrzymanych wyników. Zgodnie z opisaną w pracy redukcją w trakcie indeksowania obiektów przestrzennych osobno indeksujemy każdy

z wierzchołków, co dla 3 wymiarów w konsekwencji przekłada się na ponad 160-krotny wzrost kosztu utworzenia oraz 16-krotny wzrost rozmiaru dla struktury opartej na BA--drzewach. Takie rozważania pozostają jednak czysto teoretyczne, ponieważ znaczny wzrost komplikacji algorytmów dla większej liczby wymiarów i związany z nimi drastyczny spadek wydajności sprawiają, że BA-drzewa mogą być wykorzystywane maksymalnie dla dwóch wymiarów. Jednak również w tej dziedzinie CBA-drzewa mają nad nimi znaczną przewagę – oparty na nich indeks przestrzenny nadal posiada ponad 80-krotnie mniejszy koszt utworzenia oraz 8-krotnie mniejszy rozmiar w stosunku do takiego samego rozwiązania opartego na BA-drzewach, przy czym te relacje jeszcze bardziej się pogłębiają przy wzroście liczby obiektów. Widzimy więc, że opracowane CBA-drzewa doskonale sprawdzają się przy indeksowaniu wielowymiarowych obiektów przestrzennych. Analizę wpływu na rzeczywistą wydajność przetwarzanych zapytań należałoby uzupełnić o testy wykonane dla kontenerów implementujących mechanizmy buforowania, ponieważ korzystanie wyłącznie z kontenera pamięciowego nie pozwala w pełni uwzględnić wszystkich aspektów charakterystycznych dla aplikacji działających również poza środowiskiem laboratoryjnym.

Druga z opracowanych struktur, QCBA-drzewo, była próbą zwiększenia wydajności CBA-drzew. W celu zredukowania liczby operacji I/O w charakterystyczny sposób łączy ona ze sobą odpowiednie krawędzie. Jak wykazały przeprowadzone badania, kosztem trzykrotnego zwiększenia rozmiaru względem CBA-drzew udało się uzyskać wyniki zbliżone do poziomu BA-drzew, przy zachowaniu 6-krotnie mniejszego kosztu ich utworzenia.

Zaprezentowane w artykule algorytmy nie wyczerpują wszystkich możliwości poprawy parametrów analizowanych struktur. W ramach kolejnych badań będzie podjęta próba zwiększenia procentowego wykorzystania stron, co pozwoliłoby na dalsze zredukowanie rozmiaru całej struktury. Dodatkowo należałoby rozpatrzyć możliwość wykorzystania przedstawionych struktur również dla innych klas zapytań. Wybranymi obszarami zainteresowań, w których przedstawione struktury potencjalnie mogłyby znaleźć zastosowanie, jest wyznaczanie aproksymowanych odpowiedzi, wyznaczanie agregacji temporalnych dla interwałów czasu, zapytania o najbliższego sąsiada, a także użycie w środowisku rozproszonym.

LITERATURA

1. Beckmann N., Kriegel H., Schneider R., Seeger B.: The R* tree: An Efficient and Robust Access Method for Points and Rectangles. Proc. Of SIGMOD, 1990.
2. Zhang D., Tsotras V. J., Gunopulos D.: Efficient Aggregation over Objects with Extent. Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, Madison, Wisconsin, June 03-05, 2002.

3. Robinson J. T.: The K-D-B-Tree: A Search Structure for Large Multidimensional Dynamic Indexes. Proceedings of the 1981 ACM SIGMOD international conference on Management of data, Ann Arbor, Michigan, April 29-May 01, 1981.
4. Yu B., Orlandic R., Bailey T., Somavaram J.: KDBKD-Tree: A Compact KDB-Tree Structure for Indexing Multidimensional Data. ITCC, Proceedings of the International Conference on Information Technology: Computers and Communications, 2003, s. 676.
5. Bentley J. L.: Multidimensional Binary Search Trees Used for Associative Searching, Communications of the ACM, vol. 18, no. 9, Sept. 1975, s. 509÷517.

Recenzent: Dr hab. inż. Zbyszko Królikowski, prof. Pol. Poznańskiej

Wpłynęło do Redakcji 5 grudnia 2007 r.

Abstract

Both, high cost of building the BA-trees and problems which occur when considering larger number of dimensions, limits the possibilities of their broader use. The CBA-tree, proposed as part of this article, eliminates many from among these flaws. Mainly, they help to reduce cost of building the tree, which can be created 20 times more efficient. Thanks to the removal of redundancy for interior nodes, they also allow the 2 times decrease of whole structure size. However, it involves the increase of number I/O operations for single query. The great advantage of CBA-tree in comparison to BA-tree, is also it's possibility to use larger number of dimensions directly.

The second introduced structure – the QCBA-tree, through the specific joining of borders increase the performance of single query. The results are closed to level of BA-tree, requiring however considerable increase of whole structure size.

Adresy

Marcin GORAWSKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, m.gorawski@polsl.pl .
Dawid TLATLIK: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, dawid.tlatlik@polsl.pl.