STUDIA INFORMATICA 2005
Volume 26 Number 2 (63)

Marcin GORAWSKI, Grzegorz WRÓBEL
Politechnika Śląska, Instytut Informatyki

REALIZACJA ZAPYTAŃ KLASY *K*NN W PRZESTRZENNEJ TELEMETRYCZNEJ HURTOWNI DANYCH

Streszczenie. Artykuł dotyczy realizacji zapytań o *k* najbliższych sąsiadów, których odległość od punktu zapytania nie przekracza wartości granicznej, *k* najbliższych par i o pary, między którymi odległość nie przekracza zadanej wartości granicznej. Zapytania są realizowane w przestrzeni sieciowej i euklidesowej. Proponowane algorytmy zapytań można wykorzystać w systemach przestrzennych hurtowni danych lub baz danych.

Słowa kluczowe: przestrzenna hurtownia danych, zapytanie, gwiazda kaskadowa, aR-drzewo, przestrzeń sieciowa, przestrzeń euklidesowa, euklidesowe ograniczenie.

REALIZATION OF *K*NN QUERY TYPE IN SPATIAL TELEMETRIC DATA WAREHOUSE

Summary. The paper describes the realization on the nearest neighbors, range search, closest pairs and e-distance join queries. The queries are evaluated in a spatial and Euclidean space. Proposed query algorythms can be used in spatial warehouse systems or data base systems.

Keywords: spatial data warehouse, query evaluation, cascaded star, aR-tree, spatial space, Euclidean space, Euclidean restriction.

1. Wprowadzenie

System przestrzennej telemetrycznej hurtowni danych (ang. *Spatial Telemetric Data Warehouse* (STDW)) zbiera dane telemetryczne o pomiarach zużycia mediów (gazu, prądu, wody, ciepła) [1, 2, 3]. Rozwiązanie stanowi dwuwarstwowa infrastruktura telemetryczna, którą tworzy:

a) telemetryczny system zintegrowanego odczytu liczników,
b) system przestrzennych hurtowni danych telemetrycznych.

Telemetryczny system zintegrowanego odczytu liczników bazuje na technologii AMR (ang. *Automatic* (*Integrated*) *Meter Reading*) oraz GSM/GPRS. System ten pozwala przesyłać dane z liczników zużycia mediów zlokalizowanych na dużym obszarze geograficznym do serwera telemetrycznego z wykorzystaniem sieci operatorów telefonii komórkowej GSM w technologii GPRS.

System przestrzennych hurtowni danych telemetrycznych (STDW) jest systemem wspomagania podejmowania decyzji taktycznych o wielkości produkcji mediów na podstawie krótkoterminowych prognoz ich zużycia. Prognozy te sporządzane są w oparciu o analizy danych zgromadzonych w hurtowni danych zasilanej z serwera telemetrycznego, ładowanych w procesie ekstrakcji bazy STDW. Podstawą przestrzennej hurtowni danych telemetrycznych jest struktura gwiazdy kaskadowej indeksowanej *aR*-drzewem [2].

Poniżej przedstawiono zaimplementowaną infrastrukturę telemetryczną z uwzględnieniem telemetrycznego systemu zintegrowanego odczytu liczników o nazwie **AIUT_GSM**.

Aplikacja rozszerzeń funkcjonalności **STDW** ma usprawnić proces serwisowania, czyli kalibrację i naprawę liczników **AIUT_GSM**.

Aplikacja ma umożliwiać zadawanie zapytań o k najbliższych sąsiadów, o jednostki znajdujące się w odległości nie przekraczającej wartości granicznej e, o najbliższe pary sąsiadów, o pary, między którymi odległość nie przekracza wartości granicznej e. Algorytmy mają działać bazując na modelu euklidesowego ograniczenia lub rozwinięcia sieciowego.

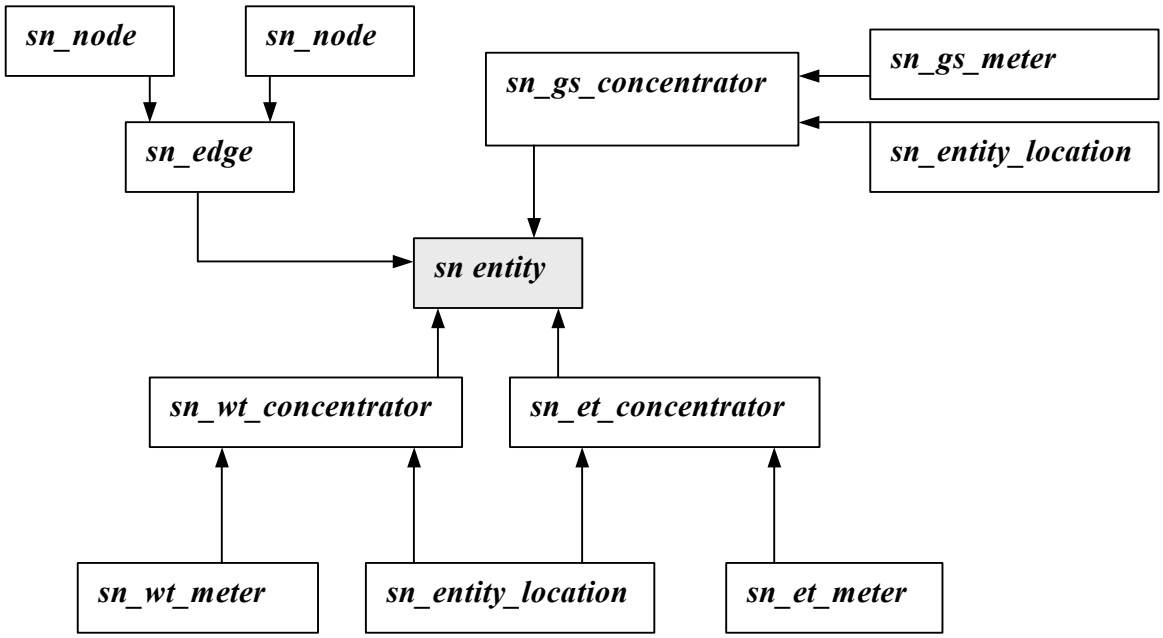
2. Model gwiazdy kaskadowej

Gwiazda kaskadowa [7] jest rozszerzeniem schematu gwiazdy. Model gwiazdy kaskadowej jest rozwinięciem standardowego modelu gwiazdy i składa się z głównej tablicy faktów, wymiarów, z których każdy jest odrębnym, zagnieżdżonym schematem gwiazdy i przechowującym dane wielowymiarowe. Każdy wymiar, oprócz atrybutów, zawiera także informacje opisujące te atrybuty.

Dane, na jakich ma operować system, można zamodelować gwiazdą kaskadową. Elementem najważniejszym i łączącym wszystkie dane jest koncentrator OKO. Tabelą faktów całego schematu jest tabela jednostki koncentratora, nazwana w tym systemie (***sn_entity)***. Tabelami wymiarów, opisującymi koncentrator, są tabele poszczególnych rodzajów koncentratorów (***sn_xx_concentrator***) wraz z atrybutami oraz tabela krawędzi (***sn_edge***), na której znajduje się dana jednostka. Każda krawędź jest opisana przez dwa węzły, między którymi jest umieszczona. Tabelą wymiarów tabeli krawędzi jest tabela węzłów (***sn_node***), zawierająca

atrybuty węzła. Tabele poszczególnych koncentratorów są tabelami faktów dla tabel pojedynczych liczników i tabeli położenia koncentratora. W ten sposób otrzymano gwiazdę kaskadową zagnieżdżoną dwukrotnie.

Relacje między tabelami są przejrzyste. Niewielkim kosztem, łącząc tylko niezbędne tabele, można dotrzeć do każdej danej w bazie hurtowni.

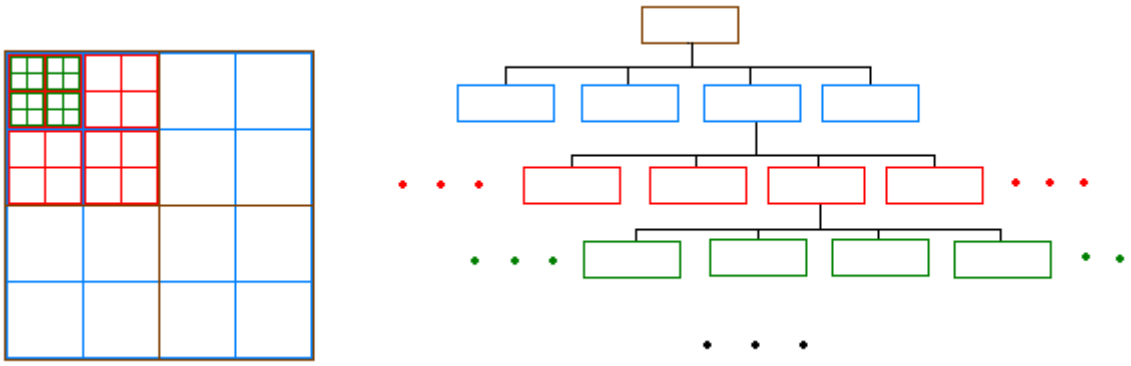


Rys. 1. Schemat gwiazdy kaskadowej
Fig. 1. Cascade star diagram

3. aR – drzewo

W przedstawianym projekcie zaimplementowano *aR*-drzewo [1,2], którego cechą charakterystyczną jest to, że poszczególne minimalne prostokąty ograniczające (ang. *Minimum Bounding Rectangle*_MBR) nie mają części wspólnych. *aR*-drzewa największą rolę odgrywają w zapytaniach zakresowych, gdzie algorytm zaczyna analizę od korzenia, następnie analizowani są ci potomkowie, których MBRy mają cześć wspólną z obszarem wyznaczonym przez punkt zapytania. *aR*-drzewo bardzo skutecznie filtruje dane.

Weźmy dla przykładu *aR*-drzewo, w którym każdy węzeł pośredni zawiera wskaźniki do 4 potomków. MBR każdego z potomków stanowi ¼ MBR rodzica.



Rys. 2. *aR*-drzewo
Fig. 2. *aR*-tree

Najmniejsza jednostka agregacji wyznaczana jest na podstawie danej wysokości drzewa, wielkości przestrzeni i stałej liczby potomków. Aby stworzyć drzewo, dana jednostka porównywana jest z MBR na kolejnych poziomach drzewa. W ten sposób dana jednostka przypisana jest odpowiednim węzłom na wszystkich poziomach drzewa. Mamy do czynienia z odwróconą klasyczną metodą tworzenia drzewa, kiedy to dany poziom tworzy się poprzez zebranie informacji z niższych poziomów.

4. Przestrzeń euklidesowa a sieciowa

Większość literatury koncentruje się na przestrzeni euklidesowej, choć większe znaczenie w życiu codziennym ma przestrzeń sieciowa. W praktyce, obiekty przesuwają się po predefiniowanym zbiorze trajektorii, takich jak szosy, szlaki wodne, powietrzne, linie kolejowe. Istotną sprawą jest dobranie realistycznej struktury, w której są zapamiętane przestrzenne obiekty i cała sieć, zachowując położenie (współrzędne geograficzne) i łączność.

Przestrzeń euklidesowa to przestrzeń, gdzie odległość między punktami jest zdeterminowana przez ich bezwzględne położenie w przestrzeni.

Przestrzeń sieciowa to przestrzeń, gdzie odległość między dwoma punktami zdeterminowana jest przez predefiniowane trajektorie.

Istnieją dwa podejścia do problemu przeszukiwania przestrzeni sieciowej: euklidesowe ograniczenie i rozwinięcie sieci [4].

Model **euklidesowego ograniczenia** bazuje na własności dolnej granicy euklidesowej: $d_N \leq d_{EMax}$, czyli euklidesowa odległość nie może być większa od maksymalnej odległości sieciowej. Przy wykorzystaniu tego warunku odrzucamy część nietrafionych fałszywych rozwiązań.

Model **rozwinięcia sieci** wykorzystuje informacje o łączności. Algorytm zaczyna od analizy punktu zapytania, a następnie analizuje węzły z nim połączone. Algorytm przemieszcza się po krawędziach tak długo, aż znajdzie rozwiązanie. Wykonuje on się bezpośrednio na sieci.

Do najbardziej typowych zapytań zalicza się zapytanie o najbliższych sąsiadów, o sąsiadów w obrębie odległości *e*, o najbliższe pary.

Dosyć istotne jest zdefiniowanie pojęć z zakresu przestrzennych hurtowni:

– odległość sieciowa między punktami *a* i *b* – dystans z punktu *a* do punktu *b* wzdłuż zdefiniowanych trajektorii,

– odległość euklidesowa między punktami *a* i *b* – długość odcinka łączącego punkt *a* i b.

5. Zapytania w przestrzeni sieciowej

5.1. Zapytania o k najbliższych sąsiadów

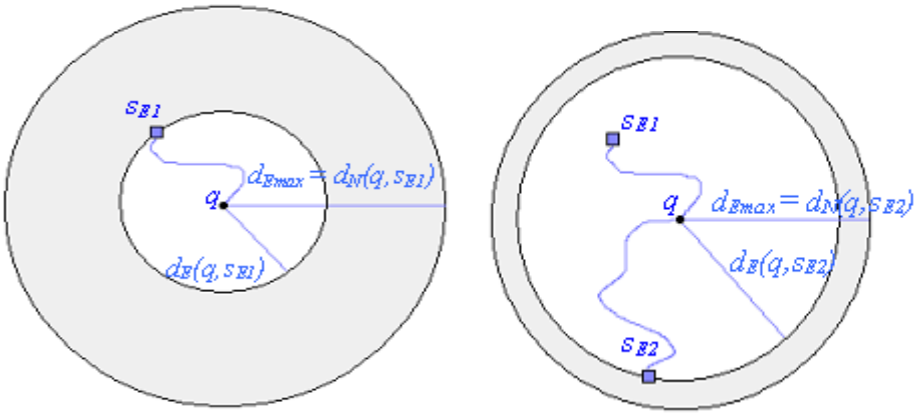
Definicja 5.1

Dany jest punkt startowy q i zbiór jednostek S. Zapytanie o k najbliższych sąsiadów (kNN) zwraca k jednostek zbioru S takich, że odległość sieciowa (euklidesowa) punktu q od nich jest najmniejsza.

5.1.1. Model euklidesowego ograniczenia

Rozważmy przypadek, kiedy chcemy znaleźć jednego, najbliższego sąsiada punktu zapytania *q* [4]. Na początku szuka się najbliższego euklidesowego sąsiada i zostaje obliczona *sieciowa odległość* punktu zapytania od najbliższego sąsiada. Obliczona odległość staje się równocześnie granicą dla dalszych euklidesowych poszukiwań. Następnie wyszukiwany jest kolejny najbliższy euklidesowy sąsiad i obliczana jest jego *odległość sieciowa.* Najmniejszy dystans sieciowy ustawiany jest jako granica euklidesowego obszaru poszukiwań. Operacje są powtarzane aż do momentu, kiedy euklidesowa odległość kolejnego euklidesowego sąsiada jest większa od dolnej granicy $d_{Emax} = d_N(q,s_{E1})$ (rys. 3). Kolejni odnalezieni sąsiedzi powinni znajdować się w szarym obszarze.

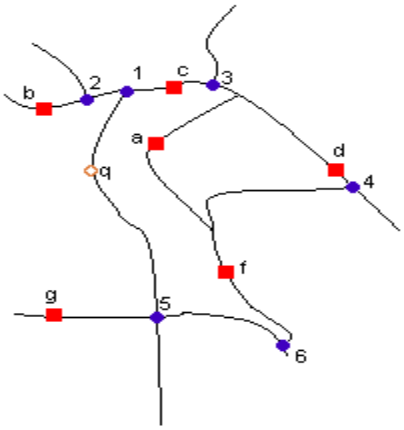
Algorytm wyszukujący *k* najbliższych sąsiadów różni się nieznacznie (rys. 3). Na początku wyszukiwanych jest *k* najbliższych sąsiadów euklidesowych, obliczane są ich *odległości sieciowe* od punktu zapytania, wyniki są sortowane wg rosnących odległości. Maksymalna wartość ustawiana jest jako maksymalna odległość euklidesowa d_{Emax}, w zasięgu której może być usytuowany obiekt. Następnie, znajdywani są przyrostowo kolejni sąsiedzi euklidesowi, obliczana jest odległość sieciowa od punktu zapytania, aktualizowane jest d_{Emax}. Algorytm kończy działanie, gdy odległość kolejnego euklidesowego sąsiada będzie większa od d_{Emax}.



Rys. 3. Pierwszy i drugi euklidesowy sąsiad
Fig. 3. First and second euclidean neighbour

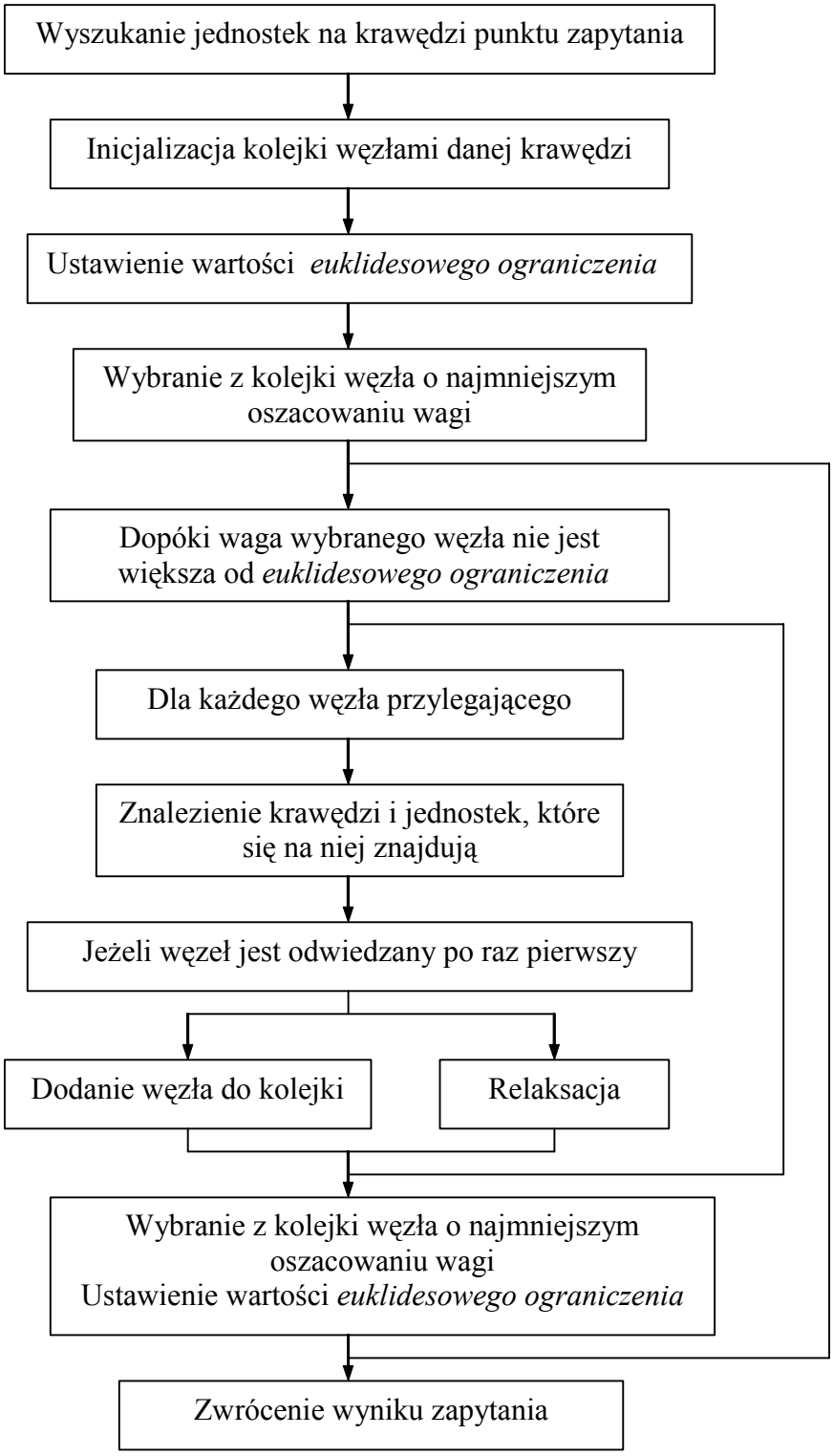
5.1.2. Model rozwinięcia sieci

Algorytm euklidesowego ograniczenia najlepszą wydajność wykazuje dla danych, dla których odległości sieciowe i euklidesowe nieznacznie się różnią. W przeciwnym przypadku algorytm sprawdza sporo jednostek, które są jedynie rozwiązaniem dla euklidesowego modelu a nie dla modelu sieciowego.



Rys. 4. Lokalizacja k najbliższych sąsiadów
Fig. 4. Location of k nearest neighbours

Dla przypadku z rys. 4 najbliższym sąsiadem euklidesowym dla punktu *q* jest *a*, natomiast najbliższym sąsiadem sieciowym jest *c*. Algorytm bazujący na własności euklidesowej dodatkowo sprawdzi jednostki *a* i *b*, zanim znajdzie rzeczywistego najbliższego sąsiada, którym jest *c*.



Rys. 5. Schemat blokowy algorytmu odpowiedzi na zapytanie o *k* najbliższych sąsiadów (model rozwinięcia sieci)
Fig. 5. Block chart of algorithm *k* nearest neighbour query (framework network expension)

Algorytm bazujący na rozwinięciu sieciowym również wykona dodatkowe obliczenia. Na początku obliczy dystans (q, 1), następnie zbędnie (q, 5) i (1, 2), zanim znajdzie najbliższego

sąsiada *c*. Nadmiarowość ta jest jednak znacznie mniejsza. Algorytm na początku znajduje krawędź, na której znajduje się punkt zapytania, sprawdza, czy nie ma na niej jednostek – potencjalnych najbliższych sąsiadów, jeżeli są, dodaje je do zbioru najbliższych sąsiadów. Wartość maksymalnej odległości sieciowej ustawiana jest na *k*-tą odległość potencjalnych sąsiadów d_{max}. Osobno tworzona jest kolejka zawierająca węzły i odległości z punktu zapytania *q* do węzła – posortowana rosnąco według odległości. Następnie, sekwencyjnie, tak długo jak odległość do najbliższego węzła jest mniejsza od maksymalnej dopuszczalnej odległości sieciowej d_{max} , sieć jest rozwijana przez analizowanie kolejnych sąsiadujących krawędzi.

Schemat blokowy algorytmu odpowiedzi na zapytanie o *k* najbliższych sąsiadów (model rozwinięcia sieci) przedstawiono na rys. 5.

5.2. Zapytanie o wszystkich sąsiadów będących w odległości sieciowej e od punktu zapytań

Definicja 5.2

Dany jest punkt startowy *q*, wartość *e* oraz zbiór jednostek *T*. Zapytanie o wszystkich sąsiadów, będących w odległości sieciowej *e* od punktu zapytań (w skrócie: zapytanie z zakresem *e*), zwraca wszystkie jednostki zbioru *T*, których odległość sieciowa od punktu zapytania *q* nie jest większa od *e*.

5.2.1. Model euklidesowego ograniczenia

Algorytm realizacji zapytań z zakresem *e* na początku zwraca zbiór kandydatów, czyli wszystkich jednostek leżących w zasięgu euklidesowej odległości *e* od punktu zapytania ($d_N(q,t) \le e \Rightarrow d_E(q,t) \le e$). W ten sposób eliminuje się sporą część fałszywych trafień – jednostek poza zasięgiem euklidesowym. Następnie, metodą rozwinięcia sieci sprawdzane są wszystkie odcinki, których odległość sieciowa do punktu zapytania nie przekracza wartości *e*. Wszystkie jednostki znajdujące się na tych odcinkach dopisywane są do wyniku i równocześnie usuwane ze zbioru kandydatów. Wyszukiwanie kończy się, gdy zbiór kandydatów jest pusty lub przebadane zostały wszystkie odcinki w zasięgu *e*.

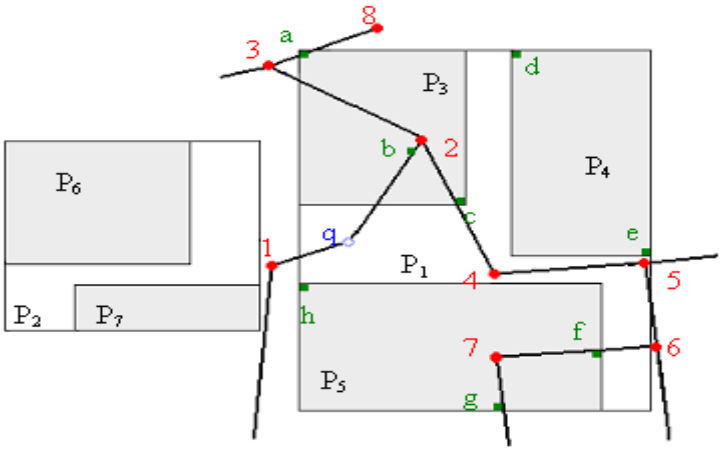
W pierwszym etapie zwracani są euklidesowi kandydaci, w drugim etapie uściślającym odrzucane są jednostki nie spełniające zapytania. Osobnego rozpatrzenia wymagają obiekty leżące na odcinkach, które tylko częściowo spełniają warunek z zakresu *e*.

5.2.2. Model rozwinięcia sieci

Algorytm działa rekurencyjnie w oparciu o *aR*-drzewo. Zaczynając od korzenia, szuka odcinków w zasięgu odległości e od punktu zapytania – oznaczmy ten zbiór odcinków QO. Następnie, podobną procedurę przeprowadza dla potomków, dla których zbiór kwalifikujących się odcinków nie jest pusty. O ile za pierwszym razem należało zastosować algorytm

rozwinięcia sieci, o tyle dla potomków wystarczy przeprowadzić operacje sprawdzenia, czy MBR danego odcinka ma część wspólną z obszarem danego potomka. Funkcja jest wywoływana rekurencyjnie aż do momentu dotarcia do liści drzewa. Na poziomie liści znajdowane są obiekty spełniające warunek zapytania z zakresem e.

Dla przykładu, z rys. 6 algorytm znajduje zbiór odcinków spełniających zapytanie {(*q*, 1), (*q*, 2), (2, 3), (2, 4), (4, 5), (3, 8)}, następnie oblicza zbiór odcinków dla odpowiednich potomków. Jedynie zbiór dla P_3 nie jest pusty, algorytm dociera do liści drzewa, znajduje jednostki (*a*,*b*,*c*) i kończy swoje działanie.



Rys. 6. Zapytanie z zakresem (model rozwinięcia sieci)
Fig. 6. Range query (framework network expansion)

5.3. Zapytanie o k najbliższe pary

Definicja 5.3

Dane są dwa zbiory jednostek *T*, *S* i wartość *k*. Zapytanie o k najbliższe pary zwraca *k* par (*s*,*t*), $s \in S$, $t \in T$, które są najbliżej położone względem siebie.

5.3.1. Model rozwinięcia sieci

Różnica pomiędzy zapytaniem o najbliższe pary a poprzednimi typami zapytań (zapytaniem z zakresem *e* i *k* najbliższych sąsiadów) jest taka, że teraz nie mamy punktu zapytania, który może być użyty jako źródło do rozwinięcia sieciowego. Zatem, jedyną opcją jest użycie jako źródła wszystkich punktów zbioru jednostek (tego z mniejszą licznością zbioru). Algorytm najbliższych par wywołuje algorytm o najbliższych sąsiadów w modelu rozwinięcia sieci, żeby wyszukać *k* najbliższych sąsiadów $t_1,...,t_k$ ($\in T$) pierwszego obiektu s_1 zbioru *S*. Odległość $d_N(s_1,t_k)$ stanowi granicę d_{Nmax} dla następnych rozwinięć. Gdy bliższe pary są odkrywane, granica ta jest stopniowo zmniejszana. Ten sam proces jest powtarzany dla kolejnych jednostek zbioru *S*. Algorytm kończy pracę, gdy zostaną przebadane wszystkie jednostki zbioru *S*.

5.3.2. Model euklidesowego ograniczenia

Algorytm działa podobnie do algorytmu wyszukiwania najbliższych par. Na początku obliczanych jest k najbliższych sąsiadów dla pierwszego elementu zbioru S, następnie obliczane są odległości sieciowe do znalezionych jednostek. Zbiór jednostek zostaje posortowany. k-ta odległość staje się równocześnie granicą euklidesową d_{Emax}. Następnie wyszukiwani są kolejni euklidesowi sąsiedzi, obliczana jest wartość dystansu sieciowego, jaki dzieli jednostki od punktu zapytania, jeżeli odległość jest mniejsza od najmniejszej dotychczasowej odległości, jednostka jest wstawiana do zbioru rozwiązania, uaktualniana jest granica euklidesowa. Operacje są powtarzane aż do momentu, kiedy kolejny euklidesowy sąsiad leży poza zasięgiem d_{Emax}. Pętla powtarzana jest dla kolejnych elementów zbioru T.

5.4. Zapytania o złączone pary sąsiadów w odległości nie większej niż e od siebie

Definicja 5.4

Dane są dwa zbiory jednostek S, T i wartość e. Zapytania o złączone pary sąsiadów w odległości nie większej niż e od siebie zwraca wszystkie pary (s,t) $s \in S$, $t \in T$ takie, że $d_N(s,t) \le e$, czyli wszystkie pary, między którymi odległość sieciowa nie przekracza wartości e.

6. Algorytm najkrótszych ścieżek

Zakładamy, że istnieje ważony graf skierowany $G = (V,E)$ z funkcją wagową w: $E \rightarrow R$ przyporządkowującą krawędziom wagi o wartościach rzeczywistych. Wagą ścieżki $p = \langle v_0, v_1, ..., v_k \rangle$ jest suma wag tworzących ją krawędzi.

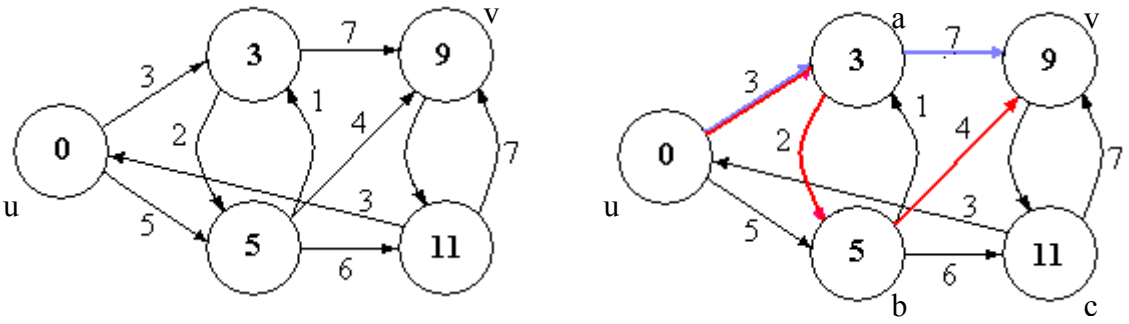
$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) \tag{1}$$

Wagę najkrótszej ścieżki z jednego punktu do drugiego można zdefiniować jako:

$$\delta(u,v) = \begin{cases} \min\{w(p) : u \xrightarrow{p} v\} \\ \infty \end{cases} \tag{2}$$

Załóżmy, że mapa samochodowa jest zamodelowana za pomocą grafu, w którym wierzchołki reprezentują skrzyżowania dróg, a krawędzie odcinki między skrzyżowaniami. Waga krawędzi powinna jak najlepiej odzwierciedlać koszt przejazdu przez nią. W najprostszym przypadku może to być długość odcinka. Rzeczywistość jest jednak znacznie bardziej skomplikowana. Wartością najlepiej odzwierciedlającą koszt jest czas, jaki jest potrzebny do przebycia danej drogi. O wadze może decydować odległość między węzłami, standard drogi oraz stopień zatłoczenia.

Rozważmy prosty przykład z rys. 7, gdzie poszukuje się najkrótszej ścieżki z *u* do *v*. Rozwiązaniem - drzewem najkrótszych ścieżek o korzeniu w *s* będzie podgraf. Przykład pokazuje, w jakiej kolejności będą rozpatrywane ścieżki. Z wielu ścieżek prowadzących do *v* najkrótsza prowadzi przez <*u*, *a*, *b, v*>. Ścieżka <*u*, *a, v*>, mimo iż składa się z mniejszej ilości skrzyżowań, jest dłuższa od ścieżki <*u*, *a*, *b, v*>. W tym przypadku najpierw zostanie znaleziona ścieżka dłuższa <0, 3, 9>, a dopiero później krótsza <0, 3, 5, 9>.



Rys. 7. Algorytm najkrótszych ścieżek
Fig. 7. The shortest path algorithm

6.1. Relaksacja

Relaksacja [9] to wielokrotne zmniejszanie oszacowania rzeczywistej wagi najkrótszej ścieżki, aż do momentu kiedy ograniczenie stanie się faktyczną wagą najkrótszej ścieżki.

Górne ograniczenie wagi najkrótszej ścieżki nazywamy oszacowaniem wagi. Relaksacja polega na sprawdzeniu, czy przechodząc przez wierzchołek *u*, można znaleźć krótszą od dotychczas najkrótszej ścieżki do *v*, jeśli taka możliwość istnieje, oszacowanie jest aktualizowane. Podczas relaksacji może zostać zmniejszona wartość oszacowania wagi najkrótszej ścieżki i może zmienić się poprzedni węzeł.

Relaksacja powoduje monotoniczną zbieżność oszacowań wag najkrótszych ścieżek do ich rzeczywistych wag.

6.2. Algorytm Dijkstry

Algorytm Dijkstry [8] jest jednym z najbardziej rozpowszechnionych algorytmów rozwiązywania problemu najkrótszych ścieżek z jednym źródłem w grafie ważonym $G=(V,E)$.

W algorytmie pamiętany jest zbiór węzłów *S*, dla których zostały już obliczone wagi. *Algorytm Dijkstry* polega na wielokrotnym powtarzaniu operacji wybrania wierzchołka ze zbioru *V*-*S* o najmniejszym oszacowaniu najkrótszej ścieżki, dodania wierzchołka *u* do *S*, i wykonania relaksacji krawędzi wychodzących z wierzchołka *u* . Wierzchołki ze zbioru *V*-*S* znajdują się w kolejce priorytetowej *Q*. Kluczem, według którego posortowane są elementy ko-

lejki, jest oszacowanie najkrótszej krawędzi. Graf może być reprezentowany albo przez listę sąsiedztwa, albo przez macierz połączeń. Dla grafów rzadkich lepiej implementować macierz połączeń, dla gęstszych listę sąsiedztwa.

DIJKSTRA(*G*, *w*, *s*)
1. INICJACJA-POJEDYNCZEGO-ŹRÓDŁA(*G*,*s*)
2. *S*←∅
3. *Q*←[*G*]
4. while Q≠∅ do
5. *u* = znajdź_min(*Q*)
6. *S*←*S*∪{*u*}
7. for każdy wierzchołek *v*∈listy sąsiedztwa[*u*]
8. do RELAX (u,v,w)

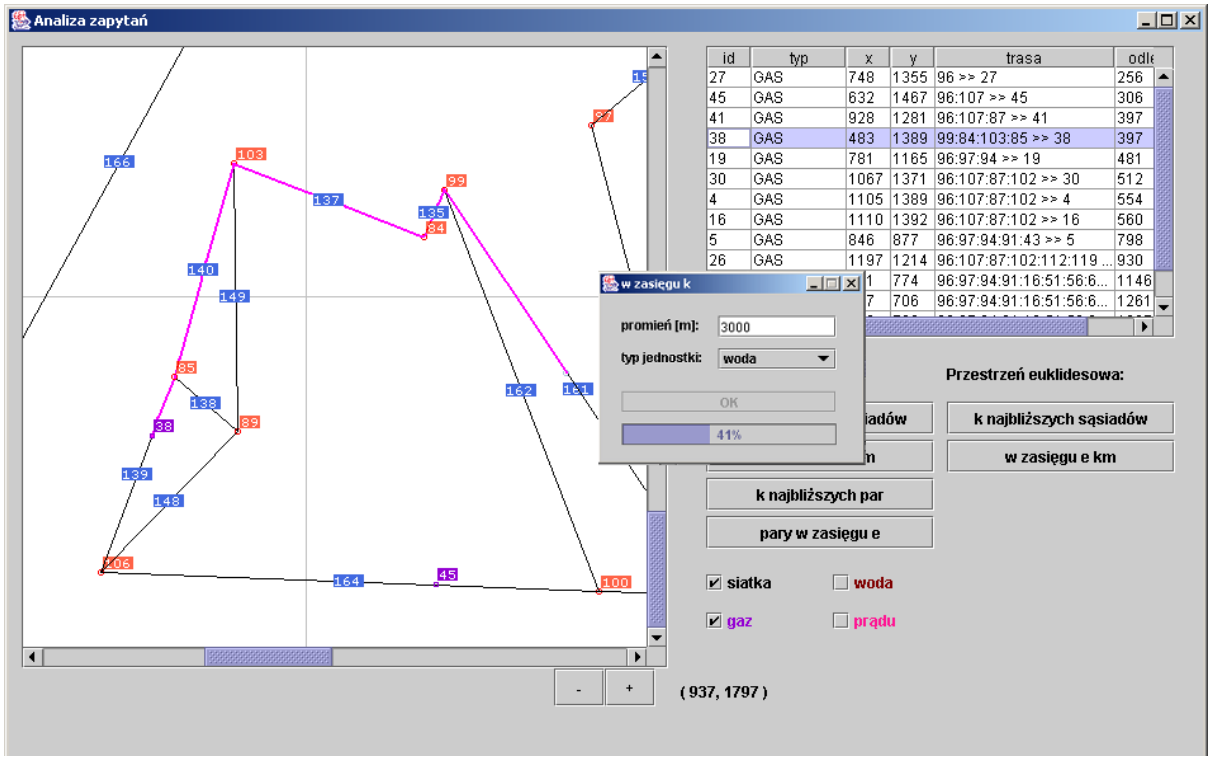
Na początku algorytm nadaje parametrom wartości początkowe. Kolejka priorytetowa zawiera wszystkie węzły grafu. W każdej iteracji usuwany jest węzeł z najmniejszym kluczem i wstawiany jest do zbioru *S*. Następnie, na każdej krawędzi wychodzącej z *u* wykonywana jest relaksacja. Jeżeli droga przez ten wierzchołek jest krótsza od wcześniej wyznaczonej, aktualizowany jest poprzednik oraz waga osiągnięcia tego wierzchołka. Można zauważyć, że każdy węzeł jest tylko raz usuwany z kolejki, w związku z tym iteracji będzie dokładnie tyle, ile węzłów grafu. Algorytm ma charakter zachłanny.

7. Funkcjonalność systemu STDW

Generator umożliwia wygenerowanie grafu oraz losowych danych hurtowni z możliwością ustawienia rozmiarów mapy, gęstości węzłów oraz poszczególnych koncentratorów.

7.1. System analizy zapytań

System analizy zapytań umożliwia wykonywanie zapytań o najbliższych sąsiadów, o sąsiadów, których odległość od punktu zapytania nie przekracza *e*, o najbliższe pary i pary, między którymi odległość nie przekracza *e* (rys. 8). Wiele funkcji związanych jest z wyświetlaniem danych bazy.

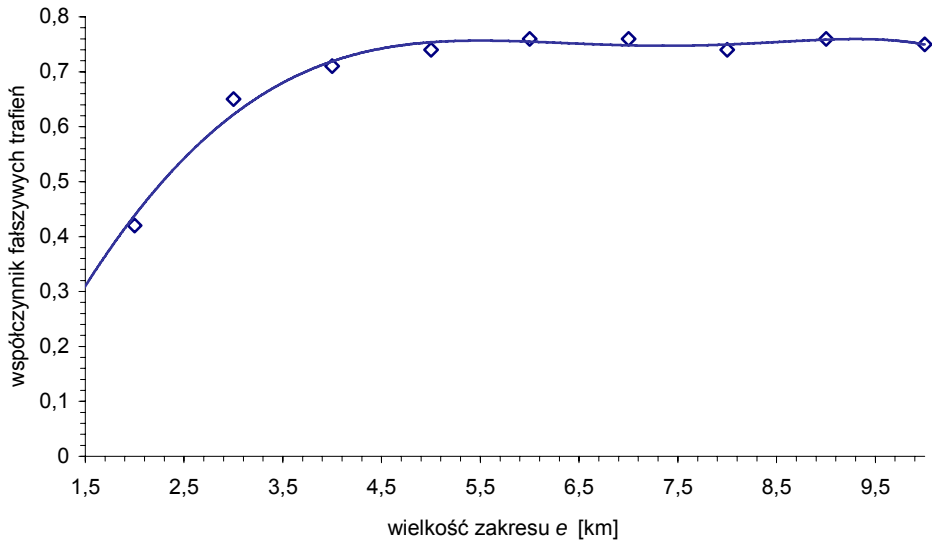


Rys. 8. System analizy zapytań
Fig. 8. System analysis of queries

8. Testy

8.1. Zapytanie z zakresem *e*

Testy zostały przeprowadzone na grafie wygenerowanym za pomocą generatora dołączonego do projektu. Dla wielkości mapy 25km*25km, przy gęstości występowania danego koncentratora 2/1 km^2, gęstość skrzyżowań 1/1 km^2, gęstość pozostałych jednostek – 0.5/1 km^2. Badania zostały przeprowadzane na zapytaniach z zakresem *e*, gdzie zakres $e \in <2$ km, 10 km$>$.

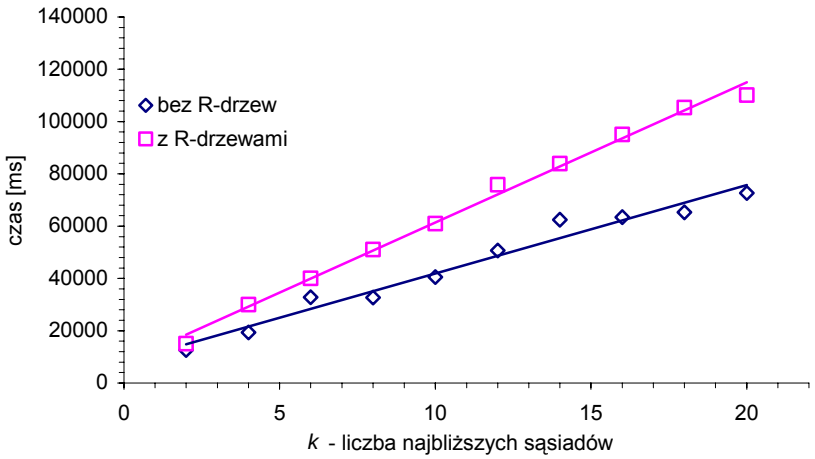


Rys. 9. Wykres zależności współczynnika fałszywych trafień od wielkości zakresu *e*
Fig. 9. False hit ratio vs. range *e*

Dla bardzo małych wartości *e* (zakresu) współczynnik fałszywych trafień jest niski, ale szybko rośnie, by ustabilizować się na wartości 0.8 (rys. 9). Współczynnik fałszywych trafień zdefiniowany jest przez stosunek *liczby fałszywych trafień* do *liczby wszystkich analizowanych jednostek*. Dla dużych wartości zakresu *e* korzystanie z algorytmu opartego na euklidesowym ograniczeniu wydaje się nieoptymalne. Średnio na cztery badane jednostki przypada jednostka spełniająca warunki rozwiązania.

8.2. Zapytanie kNN

Testy zostały przeprowadzone na danych z bazy stworzonych za pomocą generatora. Przyjęte zostały następujące założenia: graf ma wymiary 15 km * 15 km, średnia gęstość skrzyżowań wynosi 1/1km^2, gęstość jednostek, na których oparte są badania 2/1 km^2, gęstość pozostałych jednostek 0.5/1 km^2, wysokość *aR*-drzewa, na którym bazowały algorytmy, jest równa 5. Badano wydajność czasową dla wartości $k \in <2, 20>$, gdzie k – liczba najbliższych sąsiadów.



Rys. 10. Wykres wydajności algorytmu *k*NN w zależności od *k*
Fig. 10. Cost vs. *k* nearest neighbours

Wbrew oczekiwaniom, czasy wykonania algorytmu z *aR*-drzewami są większe od czasu wykonania algorytmu z bezpośrednim dostępem do bazy (rys.10). Wiąże się to z faktem, że algorytm bez *aR*-drzew został udoskonalony poprzez wprowadzenie identyfikatora położenia, który umożliwia odczytanie większej liczby danych, wykonując zapytanie jednokrotnie. Kolejnym czynnikiem spowalniającym pracę z *aR*-drzewami jest algorytm wymiany stron.

9. Wnioski końcowe

Praca dotyczy rozwiązań intensywnie rozwijanych w Pracowni Hurtowni Danych i Systemów Inteligentnych w Zakładzie Teorii Informatyki, w Instytucie Informatyki Politechniki Śląskiej. Jednym z nowatorskich pomysłów jest zastosowanie gwiazdy kaskadowej jako schematu hurtowni. Przyjęty w pracy schemat danych się sprawdził. Nie ma nadmiarowości łączeń tabel w tworzonych zapytaniach.

Struktury indeksujące znacznie przyśpieszają pracę systemu. Zastosowanie dodatkowego indeksu położenia, który umożliwiał zbiorowy odczyt danych, przyczyniło się do znacznej poprawy wydajności programu. Jednorazowe odczytanie całej informacji jest znacznie lepszym rozwiązaniem niż odczytanie tej informacji przez kilkakrotne odwołanie się do pamięci.

Wprowadzenie identyfikatora położenia przyczyniło się do zwiększenia efektywności algorytmu. Dzięki temu zabiegowi algorytm wyszukiwania najbliższych ścieżek wykonuje się szybciej niż algorytm wykorzystujący strukturę *aR*-drzewa. *aR*-drzewa warto stosować w systemach wielomaszynowych, gdzie moc obliczeniowa jest duża. Dla pamięci operacyjnej o zbyt dużej pojemności program musi korzystać z systemowych mechanizmów pamięci wirtualnej, powodując znaczące opóźnienia.

Zgodnie z modelem rozwinięcia sieci wykonywane są szybciej zapytania o *k* najbliższych sąsiadów i jednostki znajdujące się w odległości nie przekraczającej *e*. W modelu euklidesowego ograniczenia analizowanych jest wiele fałszywych trafień – jednostek nie spełniających zapytania. Model ograniczenia euklidesowego wydaje się być bardziej wydajny dla zapytań o najbliższe pary i jednostki, między którymi odległość nie przekracza *e*. W modelu rozwinięcia sieci analizowane są wszystkie jednostki ze zbioru z mniejszą licznością, podczas gdy w modelu euklidesowym analizowane są pary, między którymi odległość euklidesowa spełnia warunek dolnej granicy.

Własność dolnej granicy może być wykorzystana jedynie wtedy, gdy wagą krawędzi grafu jest odległość między punktami. Dla przypadków, kiedy krawędź jest funkcją kilku zmiennych, wyznaczenie dolnej granicy euklidesowej może być bardzo trudne.

Testy pokazały, że algorytm korzystający z własności euklidesowej jest bardzo mało wydajny, wykonuje dużo nadmiarowych obliczeń (średnio na cztery badane jednostki jedna jest prawdziwym wynikiem). Analiza euklidesowych sąsiadów jest efektywna jedynie dla gęstych sieci, dla których odległości euklidesowe i sieciowe nieznacznie się różnią.

Przestrzenne hurtownie danych są niewątpliwie jedną z prężniej rozwijających się dziedzin informatycznych. Rozpatrywanie przypadku, kiedy punkt zapytania jest dynamiczny [5], a przestrzeń uwzględnia istnienie przeszkód [6], może okazać się bardzo interesującym kierunkiem dalszych prac.

LITERATURA

1. Gorawski M., Malczok R.: Aggregation and analysis of spatial data by means of materialized aggregation tree. Third Biennial International Conference on Advances in Information Systems, LNCS 3261, Izmir, Turkey 2004.
2. Gorawski M., Malczok R.: Distributed Spatial Data Warehouse Indexed with Virtual Memory Aggregation Tree. 5th Workshop on Spatial-Temporal DataBase Management (STDBM_VLDB'04), Toronto, Canada 2004.
3. Gorawski M., Gabryś M.: Telemetryczny system zintegrowanego odczytu liczników. Praca zbiorowa „Współczesne problemy sieci komputerowych". WNT, Warszawa 2004, s. 203÷211.
4. Papadias D., Zhang J., Mamoulis N., Tao Y.: Query Processing in Spatial Network Databases.Very Large Data Bases Conference (VLDB), pp. 802-813, Berlin 2003.
5. Zhang J., Zhu M., Papdias D., Tao Y., Lee D. L.: Location-based Spatial Queries. ACM Conference on the Management of Data (SIGMOD), pp. 467-478, CA, 2003.

6. Zhang J., Papadias D., Mouratidis K., Zhu M.: Spatial Queries in Presence Obstacle. 9th International Conference on Extending Database Technology (EDBT), s. 366÷384, Greece, 2004.
7. Adam N. R., Atluri V., Yu S., Yesha Y.: Efficient Storage and Management of Environmental Information, IEEE Symposium on Mass Storage Systems, 2002.
8. Schulz F., Wagner D., Weihe K.: Dijkstra's Algorithm On-Line: An Empirical Case Study from Public Railroad Transport. http://i11www.ira.uka.de/algo/people/fschulz/publications/pdf/sww-daole-00.pdf
9. Cormen T., Leiserson E. C., Rivest R. L.: Wprowadzenie do algorytmów. WNT, Warszawa 2001.

Recenzent: Prof. dr hab. inż. Tadeusz Morzy

Wpłynęło do Redakcji 7 grudnia 2004 r.

Abstract

The paper describes the implementation of a spatial data warehouse. The problem is illustrated on a real system of integrated meter readings. The system is based on GSM/GPRS technology. The system consists of two layers: a spatial telemetric data warehouse and a telemetric system of integrated meter readings. The spaial data warehouse is based on a cascaded star schema (fig. 1) indexed with *aR*-tree (fig. 2).

The paper focuses on the nearest neighbors (fig. 5), range search, closest pairs and e-distance join queries. The queries are evaluated in a spatial and Euclidean space. Two frameworks are presented: *Euclidean restricton* and *network expansion* for processing the most common spatial queries. The network expansion uses the modified *Dijsktra* algorithm and the framework Euclidean restriction is based on the *Euclidean lower-bound property* (fig. 3): $d_E(n_i,n_j) \leq d_N(n_i,n_j)$, i.e., the Euclidean distance between two points is equal or smaller than their network distance. A relaxation (fig. 7) method is described in detail. We presented a flow chart for the nearest neighbors.The paper addresses also the application expansion (fig. 8) and functionality.

Test results (fig. 9,10) are presented at the end of article. We compared algorithms using aR-tree structures with those reading collective data. We also studied a missing hits (misses) factor.

The summary provides information which algorithm is more efficient. Namely the algorithm in Euclidean restriction framework is more useful in closest pairs and e-distance join queries calculation. However, the network expansion is more productive in the case of nearest neighbors and range search queries evaluation.

Missing hits factor (fig. 10) initially increases linearly, then keeps big value. Spatial data warehouse is the technology of fast development, hence we consider dynamic query point as a direction of our future studies/research.

Adresy

Marcin GORAWSKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, M.Gorawski@polsl.pl.
Grzegorz WRÓBEL: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska.