STUDIA INFORMATICA 2007
Volume 28 Number 3A (72)

Marcin GORAWSKI, Marek ŁUK
Politechnika Śląska, Instytut Informatyki

BEZPIECZNA KLASTERYZACJA BAZUJĄCA NA GĘSTOŚCI POZIOMO ROZPROSZONYCH DANYCH PRZESTRZENNYCH

Streszczenie. Przedstawiony został nowy algorytm klasteryzacji rozproszonej bazujący na gęstości – PPDBDC. Algorytm operuje na danych przestrzennych, które są rozproszone poziomo pomiędzy kilka stron biorących udział we wspólnej eksploracji danych. Został zaprojektowany, obierając się na istniejących algorytmach klasteryzacji rozproszonej DBDC oraz SDBDC, jednak dodatkowo pozwala na zachowanie prywatności przetwarzanych danych.

Słowa kluczowe: odkrywanie wiedzy, eksploracja danych, klasteryzacja, dane przestrzenne, prywatność danych, obliczenia równoległe, poziome rozproszenie danych, DBCSAN, DBDC, SDBDC, PPDBDC.

PRIVACY PRESERVING DENSITY-BASED CLUSTERING OVER HORIZONTALLY PARTITIONED SPATIAL DATA

Summary. The paper proposes a new density-based distributed clustering algorithm - the PPDBDC (Privacy Preserving Density-Based Distributed Clustering) algorithm.. This algorithm can be applied to horizontally distributed spatial data in a data mining process. It is based on existing distributed clustering algorithms: the DBDC algorithm and the SDBDC algorithm. In addition presented solution enables local data privacy preservation.

Keywords: knowledge discovering, data mining, clustering, spatial data, data privacy, parallel computation, horizontally distributed data, DBSCAN, DBDC, SDBDC, PPDBDC.

1. Wprowadzenie

Wraz z rozwojem informatyki i wciąż wzrastającą liczbą komputerów na świecie zwiększa się również nieustannie liczba danych przetwarzanych i gromadzonych przez różnego

rodzaju systemy komputerowe. Dysponujemy coraz większą liczbą baz danych, których objętość również ciągle wzrasta. Bazy te początkowo były tylko zwykłym składowiskiem danych. Kolejnym etapem było stworzenie hurtowni danych, która łączyła heterogeniczne bazy danych, dzięki czemu dane te mogły być dostępne w jednym miejscu i mieć zunifikowany format. Wraz z rozwojem hurtowni danych zaczęły się rozwijać różne techniki eksploracji danych, które mają na celu wydobyć z ogromnych ilości danych wcześniej nieznaną, potencjalnie użyteczną wiedzę. Człowiek nie jest bowiem w stanie przeanalizować samodzielnie tak dużych ilości danych.

Organizacje stosujące hurtownie danych i eksplorację danych są w stanie odkryć wiedzę, której nie ma konkurencja [1, 2]. Wiedza ma wpływ na strategię korporacji, pozwala lepiej dostosować ją do dynamicznie zmieniających się warunków, co czyni ją bardziej wydajną i konkurencyjną w stosunku do innych.

Z czasem zrozumiano, że eksplorując dane wielu organizacji, można odkryć dodatkową wiedzę z korzyścią dla wszystkich stron [3]. Problem jest jednak bardzo złożony, gdyż musimy osiągnąć dwa przeciwstawne cele. Oczywiste jest, że aby móc przeprowadzić eksplorację danych, trzeba w jakiś sposób je udostępnić. Z drugiej strony jednak nie można ujawnić niektórych danych, gdyż są one prawnie chronione, bądź mogą ujawnić wiedzę, która może organizacji zaszkodzić [20, 21]. Aby więc wspólna eksploracja danych miała sens, musimy dysponować mechanizmami gwarantującymi ochronę prywatności danych poszczególnych stron [11, 12].

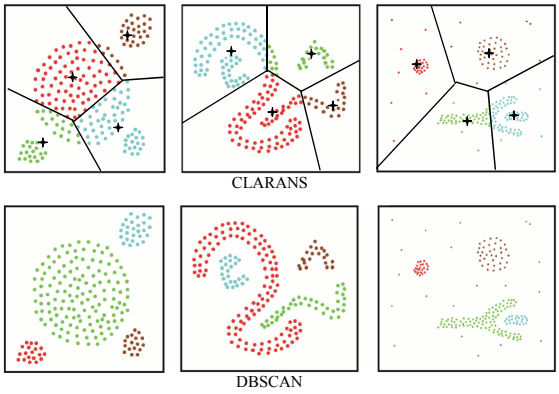
Jedną z metod eksploracji danych jest klasteryzacja [15÷19]. Jest to metoda tzw. uczenia nienadzorowanego. Związana jest z działem statystyki – analizą skupień. Klasteryzacja ma bardzo wiele zastosowań [15 - 19]. Najogólniej rzecz biorąc, może być użyta jako narzędzie samodzielne, dające w wyniku obraz rozkładu danych lub pomocniczo przy przetwarzaniu wstępnym w innych algorytmach. Ten drugi przypadek ma miejsce w procesie projektowania Decyzyjnego Systemu Zintegrowanego Monitorowania i Sterowania Zużyciem Mediów, nad którym pracuje Zespół Algorytmów, Programowania i Systemów Autonomicznych w Zakładzie Teorii Informatyki Instytutu Informatyki Politechniki Śląskiej. Klasteryzacja jest stosowana w tym systemie do optymalnej konstrukcji struktury indeksującej [8, 9]. Inną strukturą jest CHR-Tree, która ma na celu przyspieszenie wykonania przestrzennych zapytań agregujących [6, 7].

W określaniu stopnia podobieństwa obiektów wykorzystuje się funkcję odległości. Wybór funkcji odległości jest zależny od konkretnego zastosowania. W artykule tym zawsze stosowana jest odległość euklidesowa.

Na ogół nie da się jednoznacznie stwierdzić, czy dany algorytm klasteryzacji jest lepszy niż inny, gdyż o wyższości danego algorytmu decyduje wiele czynników.

2. Klasteryzacja bazująca na gęstości

Metody bazujące na gęstości zostały po raz pierwszy opisane w 1996 r. i jednocześnie zaproponowany został algorytm DBSCAN [14]. Posiadają one wiele zalet, z których najważniejsza to możliwość wykrywania klastrów o właściwie dowolnych kształtach. Na rysunku poniżej zostały porównane wyniki klasteryzacji trzech sztucznych zbiorów danych metodami CLARANS i DBSCAN [5].



Rys. 1. Porównanie wyników klasteryzacji metodami CLARANS i DBSCAN
Fig. 1. Results comparison of the CLARANS and DBSCAN methods

Nietrudno zauważyć, że algorytm klasteryzacji podziałowej CLARANS nie radzi sobie dobrze z klastrami o kształtach innych niż kuliste. Niejednokrotnie zdarza się też, że algorytmy klasteryzacji podziałowej, które na ogół są zachłanne, osiągają minimum lokalne i dają złe wyniki, nawet dla klastrów o kształtach kulistych. Taka sytuacja ma właśnie miejsce na rysunku 1 w pierwszym zbiorze danych. Tej wady pozbawione są metody bazujące na gęstości.

Zasada działania tej klasy algorytmów opiera się na obserwacji, że klastry są zbiorami punktów gęsto ułożonych, czyli mających w swoim otoczeniu dużą liczbę innych punktów. Inaczej mówiąc, gęstość punktów w klastrze jest większa niż poza nim. Ponadto, takie podejście daje możliwość odfiltrowania szumu, pod warunkiem że jego gęstość jest mniejsza od gęstości jakiegokolwiek klastra.

2.1. Podstawowe pojęcia

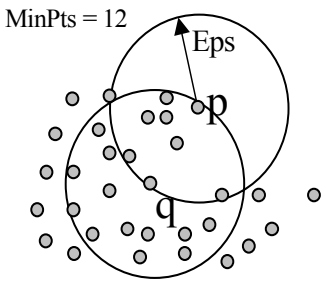
Klasteryzacja bazująca na gęstości polega na znajdowaniu klastrów w zbiorze danych przestrzennych na podstawie rozmieszczenia punktów danych. Gęstość może być określona jedynie w stosunku do jakiejś przestrzeni, tak więc badanie gęstości w zbiorze danych następuje poprzez mierzenie ilości punktów w otoczeniach kolejno analizowanych punktów danych.

Aby jakiś obszar został uznany za klaster, gęstość musi przekroczyć pewien próg, czyli liczba punktów należących do otoczenia analizowanego punktu musi przekroczyć określoną wartość, która stanowi parametr algorytmu DBSCAN o nazwie *MinPts*. Drugim parametrem algorytmu jest promień otoczenia *Eps*.

Praca [4] wyjaśnia bardzo dokładnie działanie metod opartych na gęstości, dostarczając dodatkowo niezbędnego aparatu matematycznego. Zatem, niech *D* będzie bazą danych punktów w *k*-wymiarowej przestrzeni euklidesowej *S*.

Kształt otoczenia jest określony przez wybór funkcji odległości, oznaczonej przez dist(*p*,*q*). Przykładowo, gdy zostanie użyta metryka Manhattan w przestrzeni dwuwymiarowej, to sąsiedztwo będzie miało kształt prostokąta. W przypadku odległości euklidesowej otoczenie przyjmie kształt okręgu. Samo działanie algorytmu jest oczywiście niezależne od funkcji odległości.

Należy podkreślić, że nie każdy punkt w klastrze musi mieć w swoim otoczeniu *Eps* co najmniej *MinPts* punktów. Punkty leżące na brzegu gęstych skupisk będą miały dużo mniejszą gęstość otoczenia, niż te znajdujące się w wewnątrz takich skupisk. Konieczne jest zatem odróżnienie punktów *rdzennych* (wewnątrz klastrów) od punktów *brzegowych*. Obrazuje to rysunek 2, na którym punkt *p* jest punktem brzegowym, a punkt *q* rdzennym.



Rys. 2. Dwa rodzaje punktów klastra: punkt rdzenny *q* oraz punkt brzegowy *p*
Fig. 2. Two kinds of cluster's points: core point *q* and boundary point *p*

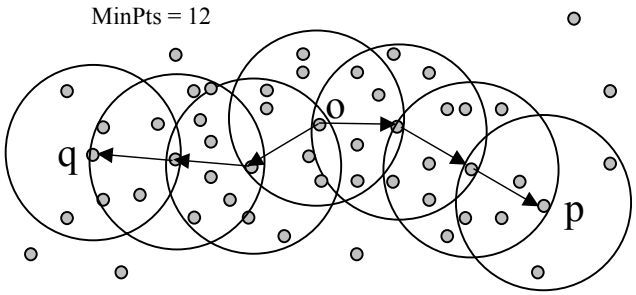
Punkty brzegowe danego klastra *C* będą miały więc otoczenie o gęstości mniejszej niż gęstość wnętrza klastra, ale będą znajdowały się w bezpośrednim sąsiedztwie gęstych obszarów, a więc będą z nich *osiągalne* bezpośrednio. Aby punkt *p* był bezpośrednio osiągalny z punktu *q*, punkt *q* musi mieć w swoim otoczeniu *Eps* co najmniej *MinPts* punktów i zawierać punkt *p*. Punkty brzegowe są osiągalne bezpośrednio jedynie z punktów rdzennych, natomiast punkty rdzenne są wzajemnie osiągalne bezpośrednio.

Dowolny punkt brzegowy danego klastra może być pośrednio osiągalny przez szereg punktów rdzennych. Pojęcie osiągalności jest więc rozszerzeniem pojęcia osiągalności bezpośredniej.

Dwa punkty brzegowe tego samego klastra *C* nie są zazwyczaj osiągalne wzajemnie, ale na pewno każdy z nich musi być osiągalny z dowolnego punktu rdzennego klastra *C*.

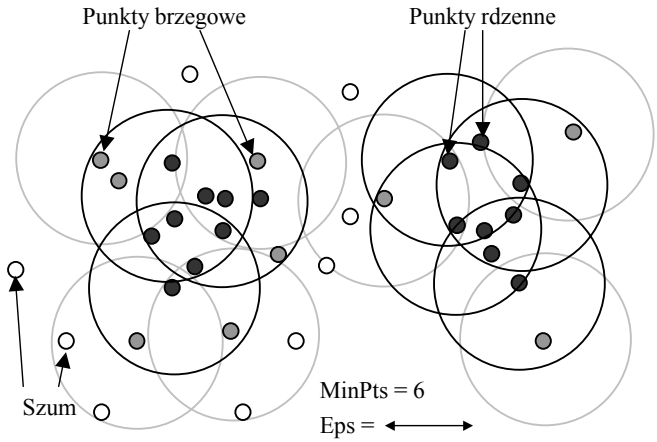
W szczególności każdy z nich musi być osiągalny z tego samego punktu rdzennego. Mówimy wtedy, że punkty te są *połączone gęstościowo*.

Gęstościowe połączenie zobrazowane jest na poniższym rysunku. Punkty *p* i *q* są punktami brzegowymi, nie mają w swoim otoczeniu *MinPts* punktów, nie są więc wzajemnie osiągalne. Punkt *o* jest punktem rdzennym, tak więc punkty brzegowe *p* i *q* są z niego osiągalne, w ten sposób punkty *p* i *q* są z sobą gęstościowo połączone.



Rys. 3. Gęstościowe połączenie: punkty *p* i *q* są połączone gęstościowo poprzez punkt *o*
Fig. 3. Dense connection : points *q* and *p* are densely connected through point *o*

Klaster będzie więc maksymalnym (ze względu na osiągalność) zbiorem punktów połączonych gęstościowo. Szum natomiast można określić jako zbiór wszystkich punktów z bazy danych *D*, które nie należą do żadnego klastra.



Rys. 4. Klasteryzacja bazująca na gęstość – przykład
Fig. 4. Density-based clustering – example

Na powyższym rysunku został przedstawiony wynik klasteryzacji opartej na gęstości. Czarne i szare punkty to odpowiednio rdzenne i brzegowe punkty należące do jednego z dwóch odkrytych klastrów. Otoczenia punktów rdzennych są oznaczone kolorem czarnym, natomiast otoczenia punktów brzegowych kolorem szarym. Punkty znajdujące się poza klastrami są punktami szumu. Nie znajdują się w otoczeniu żadnego z punktów rdzennych, ale mogą znajdować się w otoczeniu punktów brzegowych.

2.2. Algorytm DBSCAN

Na podstawie powyższych definicji można wywnioskować, że klaster jest jednoznacznie określony przez dowolny jego punkt rdzenny. Ten wniosek pozwala zrozumieć działanie podstawowego algorytmu bazującego na gęstości DBSCAN [4].

W celu znalezienia klastra algorytm DBSCAN pobiera dowolny punkt *p* z bazy danych. Jeżeli *p* jest punktem rdzennym, algorytm tworzy nowy klaster i szuka wszystkich punktów osiągalnych z niego dla danych *Eps* i *MinPts*, które są parametrami algorytmu. Gdy punkt *p* nie jest punktem rdzennym, to żaden punkt nie jest z niego osiągalny i sprawdzany jest kolejny punkt w bazie danych. Można więc ten algorytm zapisać następująco:

1. Pobierz dowolny punkt *p* z bazy danych.
2. Jeżeli punkt *p* został już przydzielony, przejdź do kroku 7.
3. Pobierz otoczenie punktu *p* o promieniu *Eps*.
4. Jeżeli otoczenie punktu *p* zawiera mniej niż *MinPts* punktów, to przydziel punkt *p* do zbioru szumu i przejdź do kroku 7.
5. Jeżeli otoczenie punktu *p* zawiera co najmniej *MinPts* punktów, to stwórz nowy klaster.
6. Przydziel do tego klastra punkt *p* oraz wszystkie punkty z niego osiągalne.
7. Jeżeli wszystkie punkty zostały już sprawdzone, to zakończ algorytm. W przeciwnym przypadku pobierz następny punkt *p* z bazy danych i przejdź do kroku 2.

Krok 6 tego algorytmu jest dość złożony i wymaga osobnego rozpisania. Zazwyczaj przy implementacji ma postać osobnej funkcji o następującym działaniu:

1. Dodaj do bieżącego klastra punkt *p.*
2. Dodaj wszystkie punkty znajdujące się w otoczeniu punktu *p* (z wyłączeniem punktu *p*) do zbioru ziaren.
3. Jeżeli zbiór ziaren nie jest pusty, to pobierz z niego dowolny punkt *q* i jednocześnie usuń go z tego zbioru. Jeżeli zbiór ziaren jest pusty, zakończ algorytm.
4. Jeżeli punkt *q* nie jest punktem rdzennym, przejdź do punktu 3.
5. Pobierz otoczenie punktu *q*.
6. Dołącz do zbioru ziaren każdy punkt *r* z tego otoczenia, jeżeli nie został jeszcze przydzielony do żadnego z klastrów ani do szumu.
7. Dołącz każdy punkt *r* z tego otoczenia do bieżącego klastra, jeżeli nie został jeszcze przydzielony do żadnego z klastrów lub jeżeli jest punktem szumu.
8. Przejdź do kroku 3.

Zauważmy, że punkty są przyporządkowywane od razu do właściwego klastra. Zmiana przyporządkowania może mieć miejsce jedynie w przypadku punktów szumu, gdyż punkty zakwalifikowane jako szum w kroku 4 algorytmu mogą zostać przydzielone do któregoś z klastrów w kroku 7 procedury poszukiwania punktów osiągalnych. Taka sytuacja ma miejsce w przypadku punktów brzegowych. Nie są one dodawane do zbioru ziaren, gdyż wiemy,

że punkty początkowo zakwalifikowane jako szum nie mogą być punktami rdzennymi, a tylko te punkty dodawane są do zbioru ziaren.

3. Rozproszona klasteryzacja bazująca na gęstości

W literaturze można spotkać się właściwie z dwoma algorytmami rozproszonej klasteryzacji bazującej na gęstości. Są to algorytmy:

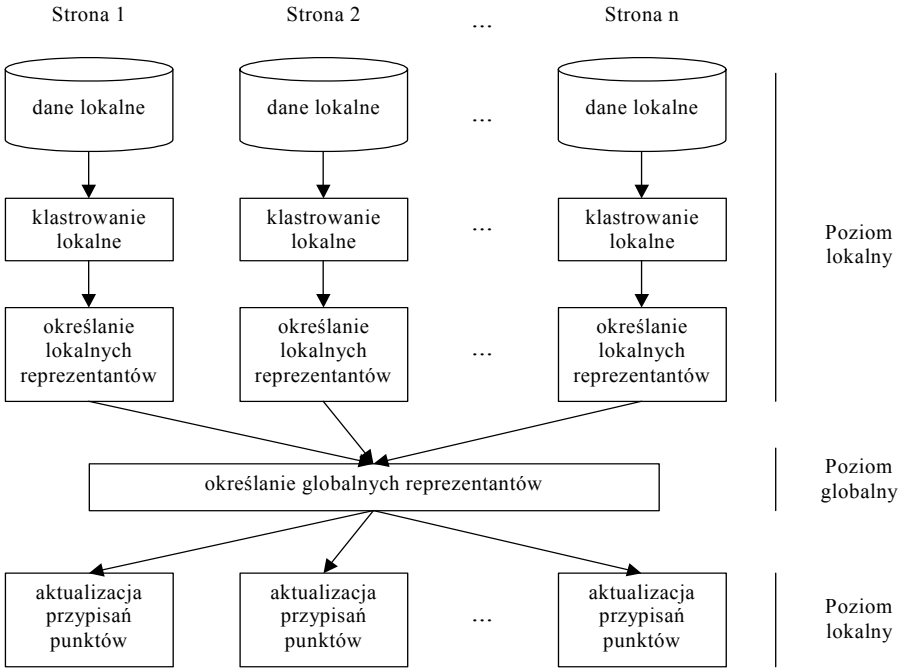
- DBDC (*Density-Based Distributed clustering*) [14],
- SDBDC (*Scalable Density-Based Distributed clustering)* [13].

Każdy z tych algorytmów ma pewne wady, co zostanie szerzej omówione w dalszej części tego artykułu. Poza tym nie były one projektowane pod kątem zachowania prywatności danych stron biorących udział we wspólnej eksploracji danych. Dlatego po przedstawieniu sposobu działania obu algorytmów zostanie zaprezentowane nowe podejście, łączące zalety obu algorytmów i dodatkowo zachowujące prywatność surowych danych.

3.1. Algorytm DBDC

W algorytmie DBDC klasteryzacja przeprowadzana jest dwukrotnie – lokalnie oraz globalnie [14]. Po klasteryzacji lokalnej strony budują tzw. model lokalny. Krok ten polega na wyznaczeniu reprezentantów, czyli punktów, które będą reprezentowały inne punkty. Ma to na celu przyspieszenie obliczeń globalnych, gdyż zamiast klastrować wszystkie punkty danych, klastruje się jedynie reprezentantów, uzyskując tzw. model globalny. Gdy reprezentanci lokalni znajdują się blisko siebie, są łączeni w klaster globalny. Na model globalny składają się połączeni z sobą i niepołączeni reprezentanci lokalni. Model ten przesyłany jest wszystkim stronom biorącym udział w klasteryzacji, które następnie aktualizują przypisania punktów do klastrów na podstawie modelu globalnego. Schemat działania tego algorytmu został przedstawiony na rysunku 5.

Algorytm DBDC jest więc algorytmem przybliżonym. Jego dokładność zależy w dużej mierze od metody wyznaczania reprezentantów. Utrata dokładności w stosunku do klasteryzacji lokalnej wszystkich danych jest nieunikniona, więc najlepiej jest ją wykorzystać jako sposób na ochronę prywatności danych lokalnych.



Rys. 5. Algorytm DBDC
Fig. 5. The DBDC algorithm

Autorzy proponują dwa następujące modele lokalne:

- model składający się ze szczególnych punktów rdzennych,
- model składający się z klastrów wyznaczonych metodą *k*-średnich.

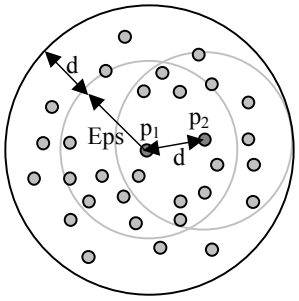
Definicja [14]

Niech $C \subseteq D$ będzie klastrem przy danych *Eps* i *MinPts*. Niech $Cor_C \subseteq C$ będzie zbiorem punktów rdzennych tego klastra. Zbiór $Scor_C \subseteq C$ będzie zbiorem szczególnych punktów rdzennych klastra *C*, jeżeli spełnione będą poniższe warunki:

$$Scor_C \subseteq Cor_C,$$
$$\forall s_i, s_j \in Scor_C : s_i \notin N_{Eps}(s_j),$$
$$\forall c \in Cor_C \exists s \in Scor_C : c \in N_{Eps}(s).$$

Może istnieć bardzo wiele zbiorów $Scor_C \subseteq C$ spełniających powyższe warunki. Drugi warunek gwarantuje, że szczególne punkty rdzenne nie mogą zawierać się wzajemnie w swoich otoczeniach o promieniu *Eps*. Powoduje to znaczną redukcję punktów danych, zwłaszcza w gęstych skupiskach. Trzeci warunek gwarantuje, że wszystkie punkty rdzenne muszą zostać pokryte przez szczególne punkty rdzenne.

Model lokalny składający się ze szczególnych punktów rdzennych musi jeszcze dodatkowo zawierać informację o obszarze pokrycia tych punktów. Szczególny punkt rdzenny musi bowiem reprezentować obszar swojego otoczenia *Eps* oraz otoczenia punktów rdzennych, które pokrywa.



Rys. 6. Obszar reprezentowany przez szczególny punkt rdzenny
Fig. 6. Area presented through particular core point

Drugi wariant modelu lokalnego to model składający się z klastrów wyznaczonych metodą k-średnich. Opiera się on również na szczególnych punktach rdzennych, ale służą one jedynie jako parametry klasteryzacji metodą *k*-średnich. Ilość szczególnych punktów rdzennych będzie więc ilością klastrów, które chcemy odkryć, a współrzędne tych punktów będą punktami startowymi dla klasteryzacji odpowiednio zmodyfikowanym algorytmem *k*-średnich. Tak więc dla każdego klastra *C* znalezionego lokalnie, algorytm k-średnich określa $|Scor_C|$ centroidów wewnątrz klastra *C*. Centroidy te zostaną reprezentantami. Do modelu lokalnego dołączamy jeszcze informacje o obszarze pokrycia przez każdego reprezentanta podobnie jak w poprzednim przypadku. Ten wariant modelu lokalnego pozwala na uzyskanie nieco większej dokładności.

Największym problemem przy klasteryzacji metodą *k*-średnich jest dobór punktów startowych. W tym wypadku dzięki zastosowaniu szczególnych punktów rdzennych mamy gwarancję, że będą one rozłożone dość równomiernie i praktycznie zupełnie niezależnie od gęstości. Nie zdarzy się więc taka sytuacja, że dwa punkty startowe znajdą się blisko siebie, gdyż szczególne punkty rdzenne nie mogą się nawzajem zawierać w swoich otoczeniach o promieniu *Eps*.

Liczba reprezentantów jest więc taka sama jak w poprzednim podejściu. Zmienia się jedynie ich położenie, które w tym wypadku nie musi odpowiadać położeniu jakiegoś konkretnego punktu lokalnego. Ten model idealnie nadaje się więc do ochrony prywatności danych, gdyż nie są ujawniane rzeczywiste współrzędne żadnego z punktów lokalnych. Jednocześnie więc zyskujemy prywatność danych oraz większą dokładność w porównaniu do poprzedniego rozwiązania.

Aby wykonać klasteryzację globalną, niezbędne jest określenie nowych parametrów *Eps* i *MinPts*. Twórcy algorytmu proponują, aby ustawić je następująco:

$$Eps_{global} = 2 \cdot Eps_{local},$$

$$MinPts_{global} = 2.$$

Taka wartość *MinPts* pozwala na połączenie dwóch reprezentantów, jeżeli tylko znajdują się blisko siebie. Problem doboru wartości *Eps* jest dużo bardziej złożony. Wartość domyślna

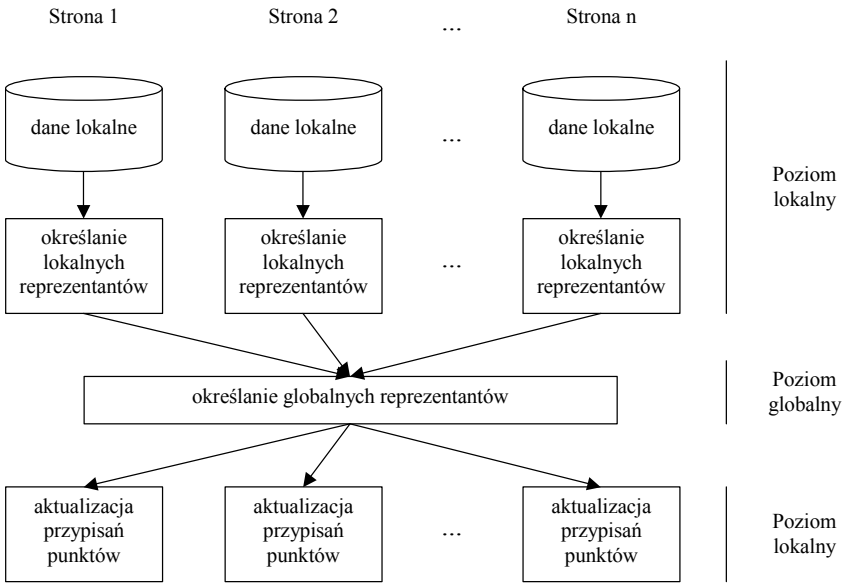
nie zawsze jest odpowiednia, co zresztą twórcy algorytmu sami pokreślają. Tak więc właściwie mamy trzy parametry wejściowe: Eps_{global}, Eps_{local} oraz $MinPts_{local}$. Jest to poważna wada tego algorytmu, gdyż, jak zostało to wcześniej omówione, nawet dobranie dwóch parametrów algorytmu DBSCAN sprawia wiele trudności.

Drugą wadą algorytmu DBDC jest to, że nie możemy w żaden sposób regulować dokładności poprzez wybór ilości reprezentantów. Ich liczba jest bowiem określona jedynie przez liczbę szczególnych punktów rdzennych.

Kolejna wada tego algorytmu wiąże się z tym, że nie bierze on w ogóle pod uwagę szumu lokalnego. Może się bowiem zdarzyć taka sytuacja, że kilka stron będzie posiadało na tym samym obszarze punkty szumu. Lokalnie więc gęstość tego obszaru będzie niższa od minimalnej gęstości klastra określonej przez *Eps* i *MinPts*. Jednak gdyby strony połączyły wszystkie swoje dane, to na tym obszarze powstałby klaster. Algorytm DBDC niestety nie wykryje takiego klastra, gdyż reprezentanci lokalni muszą znajdować się wewnątrz klastrów lokalnych.

3.2. Algorytm SDBDC

Twórcy algorytmu DBDC postanowili ze względu na wspomniane wady ulepszyć ten algorytm, tworząc w ten sposób algorytm SDBDC [16]. Podejście jest podobne do poprzedniego. Główna różnica polega na sposobie wyznaczania reprezentantów oraz na tym, że nie jest już przeprowadzana klasteryzacja na poziomie lokalnym. Oto schemat tego podejścia:



Rys. 7. Algorytm SDBDC
Fig. 7. The SDBDC algorithm

Aby uzyskać lepszą efektywność algorytmu, autorzy proponują zredukować liczbę reprezentantów poprzez zwiększenie ich jakości. Jest to możliwe dzięki dwóm miarom: StatRep i DynRep [13]. Miary te określają jakość potencjalnego reprezentanta. Pierwsza miara określa, który obiekt będzie najlepszym reprezentantem. Miara DynRep podobnie, z tym że nie są brane pod uwagę obiekty pokryte przez reprezentantów. Miara StatRep wyznaczana jest następująco [13]:

$$\mathrm{StatRep}(o, Eps) = \sum_{o_i \in N_{Eps}(o)} Eps - d(o_i, o)$$

Miara DynRep obliczana jest identycznie, z tym że nie są brane po uwagę punkty pokryte przez innych reprezentantów [16]:

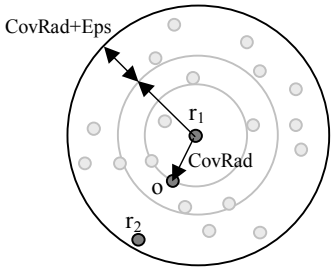
$$\mathrm{DynRep}(o, Eps, Rep) = \sum_{\substack{o_i \in N_{Eps}(o) \\ \forall r \in Rep_i : o_i \notin N_{Eps}(r)}} Eps - d(o_i, o)$$

Algorytm wyznaczania reprezentantów przy użyciu tych dwóch miar wygląda następująco [13]:

1. Wykonaj zapytania o otoczenie wokół każdego obiektu lokalnego danej strony.
2. Sortuj obiekty malejąco ze względu na ich miarę StatRep.
3. Usuń pierwszy element z listy i dodaj go do zbioru reprezentantów.
4. Oblicz miary DynRep dla każdego lokalnego obiektu danej strony, niebędącego reprezentantem, a następnie sortuj te obiekty malejąco ze względu na miarę DynRep.
5. Jeżeli liczba reprezentantów nie jest jeszcze wystarczająca, to kontynuuj algorytm przechodząc do kroku 3, w przeciwnym wypadku zakończ działanie.

Miary StatRep i DynRep pozwalają dość dobrze wybrać reprezentantów, jednak nie odzwierciedlają pokrytych ilości punktów. Do tego celu służy parametr CovCnt, określający właśnie ilość pokrytych elementów. Ważna jest jednak zasada, że dany obiekt może być pokryty tylko przez jednego reprezentanta. Obliczanie parametru CovCnt musi się więc odbywać w procesie wyznaczania reprezentantów. Oprócz tego twórcy algorytmu proponują zastosować jeszcze promień pokrycia CovCnt, który jest odległością danego reprezentanta od najdalszego pokrytego przez niego punktu. Atrybuty te dołączane są do modelu lokalnego dla każdego punktu.

Klasteryzacja globalna została nieznacznie zmodyfikowana. Zamiast ilości reprezentantów w danym otoczeniu sumowane są wagi reprezentantów, co powoduje, że otrzymujemy rzeczywistą liczbę punktów lokalnych na danym obszarze. Promień pokrycia danego reprezentanta jest dodawany do parametru *Eps*, dzięki czemu brani są zawsze pod uwagę reprezentanci osiągalni bezpośrednio z punktów reprezentowanych. Dobrze obrazuje to rys. 8.



Rys. 8. Zapytanie o sąsiedztwo poszerzone o promień pokrycia
Fig. 8. The neighboring query extended by covering ray

Dzięki poszerzeniu *Eps* o promień pokrycia CovRad możliwe stało się dotarcie do punktu r_2 z punktu r_1, gdyż punkt r_2 znajduje się w otoczeniu punktu *o* reprezentowanego przez r_1.

Dzięki temu, że klasteryzacja jest przeprowadzana tylko raz, wszystkie dane traktowane są równorzędnie. Algorytm DBDC „faworyzował" bowiem punkty rdzenne klastrów, gdyż tylko one mogły zostać reprezentantami.

W przeciwieństwie do algorytmu DBDC, algorytm SDBDC nie wymaga podania dodatkowego parametru Eps_{global}.

Algorytm wyznaczania reprezentantów jest iteracyjny i w każdej kolejnej iteracji wyznaczany jest w sposób zachłanny nowy reprezentant. Istnieje możliwość przerwania tego procesu w dowolnym momencie, przez co można płynnie regulować dokładność i szybkość. Im bowiem mniej reprezentantów, tym klasteryzacja globalna trwa krócej, ale też wyniki są mniej dokładne.

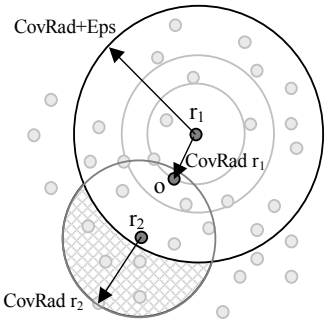
3.3. Nowe podejście – algorytm PPDBDC

Po próbnej implementacji algorytmu SDBDC okazało się, że ma on jednak dość istotne wady. Zaproponowany sposób wyznaczania reprezentantów jest dość nieefektywny. Nie ma sensu stosowanie dwóch miar, gdyż początkowo miara StatRep odpowiada mierze DynRep dla pustego zbioru reprezentantów. Po pewnych optymalizacjach i tak okazało się, że obliczanie wartości miar DynRep zajmuje bardzo dużo czasu. Twórcy algorytmu SDBDC uzyskali więc przyspieszenie globalnej klasteryzacji poprzez redukcję liczby reprezentantów, wynikającą z bardziej efektywnego ich wyznaczania. Jednak odbywa się to kosztem bardzo dużego narzutu obliczeniowego związanego z obliczaniem wartości miar w zachłannym algorytmie wyznaczania reprezentantów.

Testy wykazały, że losowy wybór reprezentantów może być również bardzo skuteczny, a za to nie wnosi tak dużego narzutu obliczeniowego.

W wyniku zastosowania poszerzonych zapytań podczas klasteryzacji niestety nie mogą być zastosowane takie same wartości *Eps* i *MinPts*, co w przypadku zwykłego algorytmu

DBSCAN. Problem stanowi poszerzanie zapytania o wartość CovRad dla danego reprezentanta. Zostało to przedstawione na następującym rysunku.



Rys. 9. Poszerzone zapytanie. Zakreskowany obszar powoduje zawyżenie liczby punktów objętych przez zapytanie
Fig. 9. Extended query. Checked space causes inflation of the number of points covered by the query

W przedstawionej sytuacji w wyniku wykonania poszerzonego zapytania brany pod uwagę jest również punkt r_2, pomimo że znajduje się poza otoczeniem *Eps* punktu zapytania r_1. W wyniku tego brana jest, pod uwagę waga punktu r_2, a więc wynik zostanie poszerzony także o punkty znajdujące się w zakreskowanym obszarze (rys. 9). Aby więc uzyskać wynik klasteryzacji odpowiadający wynikowi wykonania DBSCAN, niezbędne jest powiększenie parametru *MinPts*. Testy wykazały, że różnica może być bardzo duża. W rozważanym zbiorze testowym dla wartości *Eps* = 1,7 optymalna wartość parametru *MinPts* = 360. Aby uzyskać podobny wynik klasteryzacji przy użyciu algorytmu SDBDC, należało powiększyć ten parametr do wartości ok. 500. Taka sytuacja sprawia, że parametry wejściowe algorytmu stają się jeszcze trudniejsze do określenia. Nie można więc stosować opracowanych do tej pory metod wyznaczania parametrów wejściowych, a metod tych i tak jest bardzo niewiele.

Algorytm SDBDC nie umożliwia ponadto zachowania prywatności danych, gdyż ujawnia współrzędne reprezentantów. W wyniku tych obserwacji i przeprowadzonych badań zaprojektowaliśmy algorytm o nazwie **Privacy Preserving Density-Based Distributed Clustering** (w skrócie - **algorytm PPDBDC),** który pozwala w zadowalającym stopniu spełnić wymagania postawione w tej pracy.

Schemat działania tego algorytmu jest taki sam jak w przypadku algorytmu SDBDC. Do klasteryzacji używany jest zmodyfikowany algorytm DBSCAN, operujący podobnie jak w przypadku SDBDC na punktach ważonych (reprezentantach). Różnica jednak polega na tym, że nie jest poszerzany promień otoczenia *Eps*. Całkowicie zmieniony został natomiast proces wyznaczania reprezentantów.

Promień otoczenia używanego przy wyznaczaniu reprezentantów wcale nie musi odpowiadać wartości parametru *Eps* stosowanego w klasteryzacji. Dlatego też dodatkowo oprócz parametrów klasteryzacji *Eps* i *MinPts* potrzebny będzie parametr Eps_{rep} używany jedynie podczas wyznaczania reprezentantów. Wartość tego parametru powinna być mniejsza od

połowy *Eps*, aby wykonywanie poszerzonych zapytań nie było już konieczne. Dzięki temu parametry *Eps* i *MinPts* będą miały takie samo działanie jak w przypadku algorytmu DBSCAN.

Dobór reprezentantów będzie więc przeprowadzany następująco:

1. Stwórz kopię zbioru danych. Będzie to zbiór kandydatów na reprezentantów.
2. Pobierz punkt *r* ze zbioru kandydatów i dołącz go do zbioru reprezentantów.
3. Określ wagę reprezentanta jako liczbę punktów znajdujących się w jego otoczeniu Eps_{Rep}.
4. Określ współrzędne reprezentanta, np. poprzez uśrednienie współrzędnych punktów kandydatów w jego otoczeniu Eps_{Rep}, celem zachowania prywatności.
5. Usuń wszystkie punkty znajdujące się w otoczeniu Eps_{Rep} reprezentanta ze zbioru kandydatów.
6. Gdy liczba reprezentantów jest wystarczająca lub zbiór kandydatów na reprezentantów jest pusty, zakończ działanie. W przeciwnym wypadku przejdź do kroku 2.

Zauważmy, że dzięki takiej metodzie wyznaczania reprezentantów można zrezygnować z dołączania promienia pokrycia do modelu lokalnego, tak więc model lokalny składał się będzie jedynie z reprezentantów i ich wag.

Szczególną uwagę należy zwrócić na krok 4. Zapewnia on prywatność danych lokalnych poszczególnych stron, a jednocześnie pozwala na zwiększenie dokładności (podobnie jak w przypadku metody wyznaczania reprezentantów za pomocą algorytmu *k*-średnich w algorytmie DBDC). Gdy waga reprezentanta jest mała, np. 1 lub 2, to należy dodatkowo zaszumić współrzędne reprezentanta, tak aby nie było możliwe dokładne określenie współrzędnych punktów źródłowych. Zaszumienie może się odbyć np. poprzez dodanie do współrzędnych reprezentanta wartości losowej z przedziału $\left(0\ ;\ 0{,}5 \cdot Eps_{Rep}\right)$.

Krok 5 algorytmu gwarantuje, że dany punkt zostanie wliczony do wagi tylko jednego reprezentanta.

Sposób działania tego algorytmu daje więc możliwość wyważenia pomiędzy dokładnością, szybkością a prywatnością. Odbywa się ono poprzez zwiększanie lub zmniejszanie Eps_{Rep}. Im większa będzie ta wartość, tym mniejsza będzie dokładność klasteryzacji, ale za to algorytm będzie działał szybciej i skuteczniej zachowa prywatność danych stron biorących udział we wspólnej klasteryzacji. Im mniejsza będzie wartość Eps_{Rep}, tym więcej zostanie wyznaczonych reprezentantów, co wpłynie korzystnie na wynik klasteryzacji, ale prywatność danych będzie zachowana w mniejszym stopniu.

Prywatność danych nie jest więc zachowana na poziomie modelu idealnego, zapewniana jest ochrona surowych danych. W wyniku działania algorytmu strony poznają rozkład gęstości danych pozostałych stron. Nie są jednak ujawniane współrzędne żadnego obiektu.

4. Testy

Stanowisko testowe zrealizowano na komputerze o parametrach: procesor AMD Athlon XP 2000+ (taktowanie 1,66 GHz), Windows XP Professional SP 2, RAM 1 GB, Java 1.5.0.6.

4.1. Struktura indeksująca

W algorytmach klasteryzacji bazujących na gęstości, operacją zabierającą najwięcej czasu jest zapytanie o sąsiedztwo. W dodatku tych operacji przeprowadza się zazwyczaj bardzo dużo. Ze względu na zapytania zakresowe podstawowy algorytm klasteryzacji oparty na gęstości DBSCAN ma czasową złożoność obliczeniową rzędu $O(n)$. Bez zastosowania struktury indeksującej zapytanie zakresowe działa na zasadzie wyszukiwania liniowego, którego czasowa złożoność obliczeniowa jest rzędu $O(n)$ ze względu na liczbę punktów danych przeznaczonych do przeszukania.

Przy odpowiednio dużych zbiorach danych obszar zapytania jest dużo mniejszy w stosunku do obszaru całego zbioru danych. Można więc z góry odrzucić dużą przestrzeń, co do których wiemy, że nie ma części wspólnej z obszarem zapytania.

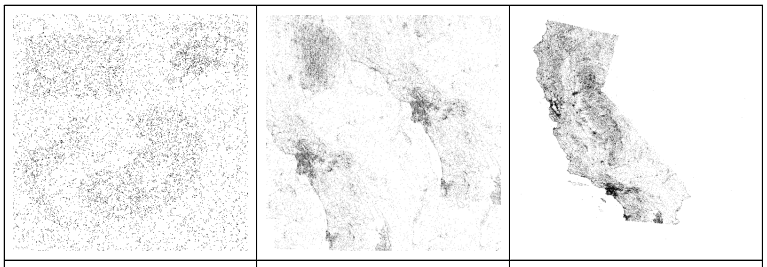
Do tego celu idealnie nadają się drzewa czwórkowe (w dwuwymiarowej przestrzeni). Drzewo czwórkowe dzieli przestrzeń rekurencyjnie za każdym razem na cztery części (każdy wymiar dzielony jest w połowie). Gdy dana ćwiartka na rozpatrywanym poziomie drzewa nie ma części wspólnej z obszarem zapytania, to nie ma potrzeby dalszego analizowania jej ani ćwiartek podrzędnych.

Takie podejście nie ogranicza się tylko do dwóch wymiarów. Właściwie liczba wymiarów może być dowolna, z tym że wraz ze wzrostem liczby wymiarów rośnie również stopień drzewa. W trójwymiarowej przestrzeni stosuje się więc drzewa ósemkowe. Dla *n*-wymiarowej przestrzeni stopień drzewa będzie równy 2*n* (po dwóch potomków dla każdego wymiaru na dowolnym poziomie drzewa). W programie drzewa zostały tak właśnie zaimplementowane, aby mogły być stosowane do danych o dowolnej wymiarowości.

Aby przetestować wydajność algorytmów, w każdym przypadku stosowano takie samo drzewo czwórkowe.

4.2. Zbiory danych testowych

Algorytmy klasteryzacji zostały przetestowane na trzech zbiorach danych testowych, które zostały opisane na poniższym rysunku:



Zbiór danych 1	Zbiór danych 2	Zbiór danych 3
sztuczny	sztuczny	rzeczywisty
13147 punktów	38754 punkty	56417 punktów

Rys. 10. Zbiory danych testowych
Fig. 10. Test data sets

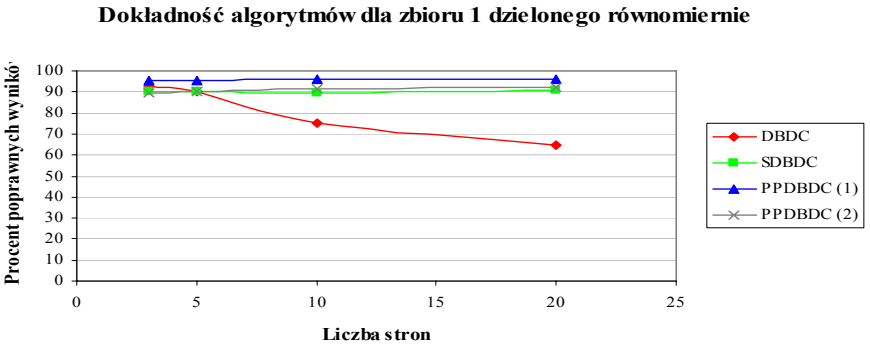
Zbiór danych rzeczywistych to dane pochodzące z testu SEQUOIA 2000, stosowanego również w pracy [4]. Użyty został zbiór „Point".

Wszystkie zbiory zostały również podzielone losowo (rozkład równomierny) na części w celu przetestowania wpływu rozproszenia danych na dokładność i szybkość. Zbiór 1 został dodatkowo podzielony arbitralnie, aby przetestować odporność algorytmów na taki rodzaj podziału.

4.3. Dokładność algorytmów

Kolejne testy związane są z badaniem dokładności poszczególnych algorytmów klasteryzacji rozproszonej w stosunku do wzorcowego algorytmu DBSCAN.

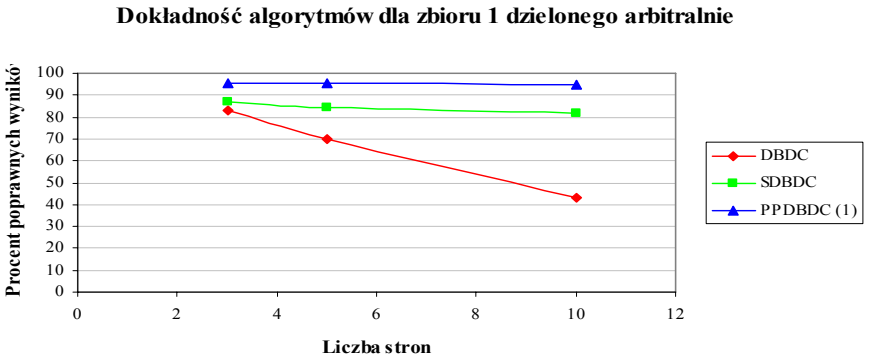
Rysunek 11 obrazuje skalowalność poszczególnych algorytmów klasteryzacji w zależności od liczby stron. Algorytm DBDC nie jest mocno skalowalny, jego dokładność mocno spada wraz ze wzrostem liczby stron. Pozostałe algorytmy cechują się dużą dokładnością niezależnie od liczby stron. Należy zaznaczyć, że dokładność poniżej 85% jest w rzeczywistości bardzo niska. W zbiorze danych nr 1 klastry znajdują się w stosunkowo dużej odległości od siebie, więc błędne wyniki sprowadzają się zazwyczaj do złego oznaczania punktów szumu, których gęstość jest niższa niż klastrów. Błędne oznaczenie dużych powierzchni szumu nie wpływa zatem mocno na zmierzoną dokładność. Oznaczenie (1) przy algorytmie PPDBDC oznacza klasteryzację z parametrem $Eps_{rep} = 0{,}5\ Eps$, a (2) z parametrem $Eps_{rep} = 0{,}9\ Eps$.



Rys. 11. Dokładność algorytmów w zależności od liczby stron przetwarzających dane (zbiór 1 regularly decomposed dzielony równomiernie)

Fig. 11. Accuracy of algorithms in dependence of the number of pages processing data (set 1)

W przypadku algorytmu DBDC konieczna była regulacja parametru *MinPts* w każdym z testów. Algorytm SDBDC wymagał do prawidłowej klasteryzacji dużo większej wartości parametru *MinPts* niż w przypadku wzorcowego algorytmu DBSCAN, jednak wartość ta była niezależna od liczby stron. Ze względu na równomierne rozproszenie wzorów w algorytmie DBDC wartość parametru *MinPts* była zawsze dużo niższa od wartości właściwej dla DBSCAN.

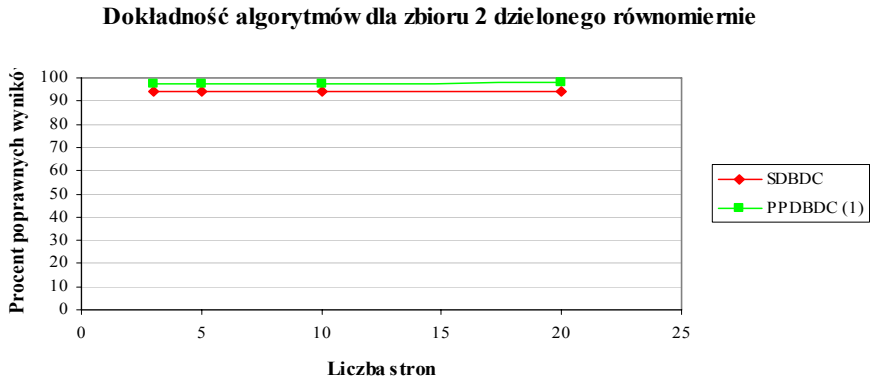


Rys. 12. Dokładność algorytmów w zależności od liczby stron przetwarzających dane (zbiór 1 dzielony arbitralnie)

Fig. 12. Accuracy of algorithms in dependence of the number of pages processing data (set 1)

W przypadku ręcznego podzielenia zbioru danych powstały większe skupiska punktów należących do poszczególnych stron. W przypadku algorytmu DBDC nie trzeba było już mocno pomniejszać wartości *MinPts*, ale jego dokładność i tak jeszcze bardziej spadła. Pozostałe algorytmy okazały się mniej czułe na ten rodzaj rozproszenia danych, chociaż wraz ze wzrostem liczby stron ich dokładność nieznacznie spada. W przypadku równomiernego rozproszenia dokładność algorytmów zwiększa się wraz ze wzrostem liczby stron, gdyż zbiór reprezentantów staje się coraz większy.

Rysunek 13 przedstawia dokładność algorytmów SDBDC oraz PPDBDC w zależności od liczby stron dla zbioru danych nr 2. Wyniki uzyskane za pomocą algorytmu DBDC były bardzo złej jakości i w ogóle nie zostały wzięte pod uwagę. Dokładność przedstawionych algorytmów jest bardzo wysoka i praktycznie niezależna od ilości stron. Dokładniejszy okazał się znów algorytm PPDBDC, który w przypadku 20 stron przetwarzających dane miał dokładność przekraczającą 98%. W tym wypadku jednak prywatność danych nie jest już skutecznie zachowana, gdyż liczba punktów przypadających na jednego reprezentanta jest coraz mniejsza. Da się to jednak wyregulować za pomocą parametru Eps_{rep}.

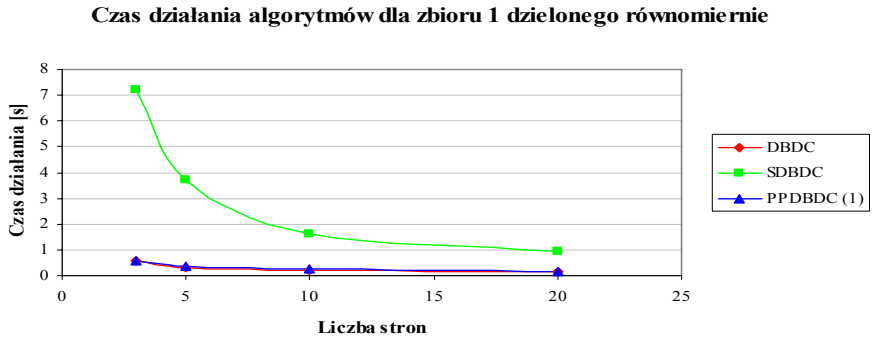


Rys. 13. Dokładność algorytmów w zależności od liczby stron przetwarzających dane (zbiór 2 dzielony arbitralnie)
Fig. 13. Accuracy of algorithms in dependence of the number of pages processing data (set 2)

4.4. Szybkość algorytmów

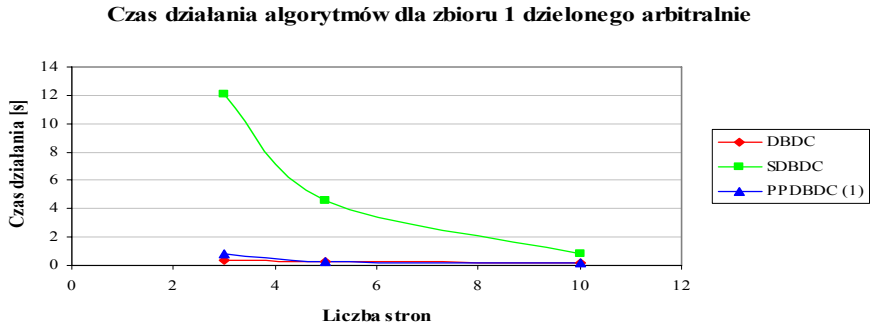
Na rysunku 14 przedstawiony został czas klasteryzacji przy użyciu algorytmu DBSCAN z indeksem oraz bez indeksu w zależności od wielkości zbioru danych. Bez indeksu algorytm działa znacznie wolniej (złożoność kwadratowa).

Rys. 14. Czas klasteryzacji przy użyciu algorytmu DBSCAN
Fig. 14. The DBSCAN algorithm clustering time

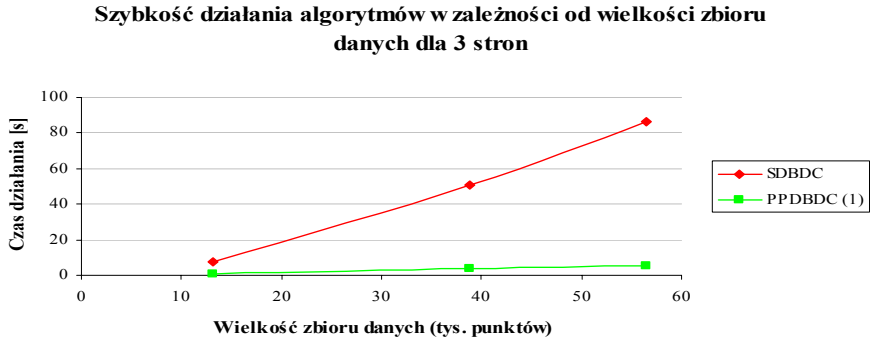


Rys. 15. Porównanie czasu działania algorytmów w zależności od liczby stron
Fig. 15. Comparison of algorithms worktime in dependence of the pages quantity

Z rys. 15 wynika, że algorytm SDBDC jest dużo wolniejszy od pozostałych, co jest zgodne z oczekiwaniami. Tak duży narzut obliczeniowy jest powodowany koniecznością wyznaczania miar DynRep dla każdego punktu w procesie wyznaczania reprezentantów. Wzrost liczby stron powoduje zmniejszenie czasu wykonania algorytmu, jednak ta zależność nie jest proporcjonalna ze względu na obliczenia, które są wykonywane szeregowo. Podobnie jest w przypadku ręcznego podziału danych pomiędzy strony. Czas wykonania wydłużył się ze względu na nierówny podział zbioru.



Rys. 16. Czas potrzebny na klasteryzację w zależności od wielkości zbioru danych
Fig. 16. Time needed for clustering in dependence of data set's size



Rys. 17. Porównanie czasu działania algorytmów w zależności od liczby stron
Fig. 17. Comparison of algorithms worktime in dependence of the pages quantity

Na wykresie 17 widać, że przy coraz większych zbiorach danych różnica w czasie działania algorytmów SDBDC oraz PPDBDC jest coraz większa.

4.5. Wnioski

Wyniki przeprowadzonych testów są zgodne z oczekiwaniami. Potwierdzają w pełni wszystkie wcześniejsze przypuszczenia. Program symulujący klasteryzację działał za każdym razem prawidłowo.

Przeprowadzone testy pozwalają stwierdzić, że prezentowany przez autorów nowy algorytm PPDBDC działa prawidłowo. Wyniki klasteryzacji rozproszonej przy użyciu tego algorytmu cechowały się wysoką jakością. Dokładność mierzona była w stosunku do lokalnej klasteryzacji wszystkich danych przy użyciu algorytmu DBSCAN. Algorytm PPDBDC praktycznie w każdym z testów osiągał dokładność powyżej 90%. We wszystkich testach wynik jego działania jest lepszy niż w przypadku pozostałych algorytmów.

Drugą bardzo ważną zaletą jest jego szybkość. Testy wykazały, że jest on znacznie szybszy od algorytmu SDBDC, dając jednocześnie lepsze wyniki. Podobną szybkością cechuje się algorytm DBDC, jednak jakość klasteryzacji przy użyciu tego algorytmu jest dużo niższa, zwłaszcza dla dużej liczby stron.

Algorytm PPDBDC przyjmuje takie same parametry jak algorytm DBSCAN. Jest to kolejna jego ważna zaleta. Algorytmy DBDC oraz SDBDC są pod tym względem dość nieprzewidywalne. DBDC z uwagi na zastosowanie klasteryzacji lokalnej nie bierze pod uwagę lokalnego szumu, przez co zachowuje się różnie dla danych rozproszonych równomiernie i arbitralnie. W zbiorach rozproszonych równomiernie należy bowiem zmniejszać wartość parametru *MinPts*, aby uzyskać efekt analogiczny do klasteryzacji algorytmem DBSCAN. W przypadku algorytmu SDBDC w wyniku powiększenia testowanego otoczenia przez promień pokrycia należy znacznie zwiększyć wartość *MinPts*, aby uzyskać efekt analogiczny do użycia algorytmu DBSCAN.

Podczas rozproszonej klasteryzacji przy użyciu algorytmu PPDBDC nie są ujawniane współrzędne żadnego punktu danych innym stronom. Przy użyciu parametru Eps_{rep} można dokonać wyważenia pomiędzy dokładnością klasteryzacji a prywatnością danych.

5. Podsumowanie

Opracowany został nowy algorytm bezpiecznej klasteryzacji rozproszonej bazujący na gęstości dla poziomo rozproszonych danych – algorytm o nazwie Privacy Preserving Density-Based Distributed Clustering (w skrócie – algorytm PPDBDC). Umożliwia on przeprowadzenie klasteryzacji przez kilka niezależnych stron, zachowując prywatność ich danych. W celu przetestowania algorytmu PPDBDC napisany został specjalny generator danych testowych oraz program symulujący rozproszoną klasteryzację.

Przeprowadzone testy na różnych zbiorach danych potwierdziły zalety nowego algorytmu. Pozwala on na uzyskanie większej szybkości przetwarzania danych oraz jednoczesne uzyskanie dobrej jakości wyników. Ze względu na to, że jest to algorytm klasteryzacji bazującej na gęstości, potrafi wykrywać klastry o różnych kształtach.

Praca ta może posłużyć jako podstawa do dalszych badań w tej dziedzinie. Ze względu na to, że nowy algorytm nie pozwala na zachowanie prywatności danych na poziomie modelu idealnego, jednym z kierunków dalszego rozwoju powinno być ulepszanie tego algorytmu pod kątem bezpieczeństwa.

Istnieje również potrzeba opracowania algorytmu wyznaczania parametrów wejściowych dla klasteryzacji bazującej na gęstości, który działa na danych poziomo rozproszonych i pozwala zachować prywatność danych stron biorących udział w obliczeniach.

LITERATURA

1. Clinton C., Kantarcioglu M., Vaidya J.: Defining Privacy for Data Mining. Proceedings of the National Science Foundation Workshop on Next Generation Data Mining, November 1-3, 2002, Baltimore, MD.
2. Clifton C., Kantarcioglu M., Vaidya J., Lin X., Zhu M. Y.: Tools for privacy preserving distributed data mining. ACM SIGKDD Explorations Newsletter, v.4 n.2, s. 28÷34, December 2002.
3. Clinton C.: Privacy Preserving Distributed Data Mining. November, 2001 www.cs-.purdue.edu/homes/clifton/DistDM/CliftonDDM.pdf.
4. Ester M., Kriegel H., Sander J., Xu X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. Proc. 2nd int. Conf. on Knowledge Discovery and Data Mining (KDD '96), Portland, Oregon, 1996, AAAI Press, 1996.
5. Ester M., Kriegel H., Sander J., Xu X.: Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications. Data Mining and Knowledge Discovery, An International Journal 2(2): 169÷194, June 1998. MA.
6. Gorawski M., Malczok R.: Eps Algorithm: A Heuristic Approach to Calculation Density-Based Clustering Eps Parameter. 4th International Conference, Advances in Information Systems, ADVIS06, October 18÷20, 2006, Izmir, Turkey, Lecture Notes in Computer Science, s. 90÷99.
7. Gorawski M., Malczok R: Calculation of Density-Based Clustering Parameters Supported with Distributed Processing. 8th International Conference Data Warehousing and Knowledge Discovery, DaWaK 2006, Krakow, Poland, September 4 – 8, 2006, Lecture Notes in Computer Science, s. 417÷426.

8. Gorawski M., Słabiński Ł.: Data Privacy Preserving Distributed k-means Clustering (Ochrona prywatności danych podczas rozproszonego klastrowania metodą k-średnich). Computer Methods and Systems, Kraków, Poland. Oprogramowanie Naukowo-Techniczne, ISBN 83-916420-3-8, Vol. 2, 2005, s. 351÷356.
9. Gorawski M., Stachurski K.: On Association Rules Mining Algorithms with Data Privacy Preserving. Advances in Web Intelligence Third International Atlantic Web IntelligenceConference, AWIC 2005, Lodz, Poland, June 6÷9, 2005, Lecture Notes in in Artificial Intelligence 3528 Springer 2005, ISBN 3-540-26219-9, s. 170÷175.
10. Gorawski M.: Association Rule Mining and Data Privacy Preserving in Spatial Data Warehouses – Performance Analysis (Odkrywanie reguł asocjacji i ochrona prywatności danych w przestrzennych hurtowniach danych – analiza wydajnościowa) 4 th Conference on Information Technology May 21÷24, 2006, Gdańsk, Faculty of ETI Annals in Information Technologies, ZN.ETI Nr. 4, T1, 2006, s. 673÷680.
11. Jagannathan G., Pillaipakkamnatt K., Wright R.: A New Privacy-Preserving Distributed k-Clustering Algorithm. Proceedings of the 2006 SIAM International Conference on Data Mining (SDM), 2006.
12. Jagannathan G., Wright R.: Privacy-Preserving Distributed k-Means Clustering over Arbitrarily Partitioned Data.Proceeding of the eleventh ACM SIGKDD international conference on Knowledge Discovery in Data Mining, 2005.
13. Januzaj E., Kriegel H., Pfeifle M.: Scalable Density-Based Distributed Clustering. Proc. 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), Pisa, Italy, 2004.
14. Januzaj E., Kriegel H., Pfeifle M.: Towards Effective and Efficient Distributed Clustering. Proceedings of Int. Workshop on Clustering Large Data Sets, 3rd Int. Conf. on Data Mining (ICDM 2003), Melbourne, FL, 2003, s. 49÷58.
15. Jha S, Kruger L., McDaniel P.: Privacy Preserving Clustering. 10th European Symposium on Research in Computer Security (ESORICS), Milan, Italy, September 2005.
16. Klusch M., Lodi S., Moro G.: Distributed clustering based on sampling local density estimates. The Eighteenth biennial International Joint Conference on Artificial Intelligence (IJCAI'2003), Morgan Kaufmann, Mexico, August 2003.
17. Merugu S., Ghosh J.: A Privacy-sensitive Approach to Distributed Clustering. Pattern Recognition Letters, vol. 26, 2005, s. 399÷410.
18. Merugu S., Ghosh J.: Privacy-preserving Distributed Clustering using Generative Models. Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM), November 2003.

19. Oliveira S., Zaïane O.: Privacy Preserving Clustering By Data Transformation. Proceedings of the 18th Brazilian Symposium on Databases (SBBD 2003), Manaus, Brazil, October 2003, s 304÷318.
20. Vaidya J., Clinton C.: Privacy Preserving Association Rule Mining in Vertically Partitioned Data. The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002).
21. Zhang N., Zhao W.: Distributed Privacy Preserving Information Sharing. Proceedings of the International Conference on Very Large Data Bases (VLDB), Trondheim, Norway, September 2005.

Recenzent: Prof. dr hab. inż. Henryk Rybiński

Wpłynęło do Redakcji 19 marca 2007 r.

Abstract

The paper proposes a new density-based distributed clustering algorithm - the PPDBDC (Privacy Preserving Density-Based Distributed Clustering) algorithm.. This algorithm can be applied to horizontally distributed spatial data in a data mining process. It is based on existing distributed clustering algorithms: the DBDC algorithm and the SDBDC algorithm. In addition presented solution enables local data privacy preservation. However the privacy is not ideally preserved, we can make a tradeoff between data privacy and clustering accuracy. Performed tests' results confirmed high performance and scalability of the PPDBDC algorithm in dependence of dataset's size and a number of pages. Furthermore the algorithm is relatively easy to implement.

Adresy

Marcin GORAWSKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, M.Gorawski@polsl.pl.
Marek Łuk: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, M.Luk@polsl.pl.