JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 405–411
Received: 23 October 2020
Accepted: 13 November 2020

Enviromental data visualisation using Delaunay triangulation

Wizualizacja danych środowiskowych z wykorzystaniem triangulacji Delauney.

Mateusz Nowosad*

Department of Computer Science, Lublin University of Technology, ul. Nadbystrzycka 38, 20-618 Lublin, Poland

Abstract

Graphical data representation is very helpful when analyzing environmental data. It allows for discovering trends in data and analysis of phenomena occurring in the area. There are many possibilities to represent such values graphically. This article contains visualizations generated using Delauney triangulation to represent data on a map. Strengths and weaknesses, comparative analysis with another solution, performance, and usage suggestions will be presented.

Keywords: enviromental data; delaunay; statistical maps

Streszczenie

Graficzna reprezentacja danych jest bardzo pomocna przy analizie danych środowiskowych. Pozwala na dostrzeżenie trendów w danych i analizę zjawisk zachodzących na określonym terenie. Jest wiele możliwości przedstawienia takich wartości. W tym artykule przedstawiono sposób wykorzystujący triangulacje Delauney do reprezentacji danych na mapie. Przedstawione będą mocne i słabe strony, analiza porównawcza z innym rozwiązaniem, wydajność oraz sugestie dotyczące użycia.

Słowa kluczowe: dane środowiskowe; delaunay; mapy statystyczne

1. Wstęp

Stan powietrza ma bardzo duży wpływ na ludzkie zdrowie i środowisko. Według ostatnich szacunków choroby spowodowane jakością powietrza zabijają ok. milion Europejczyków rocznie [1]. Przykładem istotnego parametru badania powietrza jest PM10, który pojawia się czasami w literaturze pod nazwą “pył gruby”. Udowodniono silny związek przyczynowo skutkowy pomiędzy wzrostem stężenia tej mieszaniny a chorobami dróg oddechowych i układu krążenia [2]. Wraz ze wzrastającą świadomością społeczną wzrasta zapotrzebowanie na rozwiązania pozwalające na analizę i kalkulacje ryzyka związanego z poziomem zanieczyszczeń pochodzenia antropogenicznego. Udostępnienie przystępnych narzędzi wizualizujących dane środowiskowe pozwala na świadome decyzje i kalkulacje ryzyka przez zainteresowanych ludzi, a także może być wykorzystywane do kolejnych prac naukowych związanych z tym tematem.

W niniejszym artykule poddano analizie wykorzystanie triangulacji Delaunay. Bardzo często w literaturze pojawiają się wizualizacje oparte na kolorowych punktach albo mapach ciepła, ale rzadko widuje się graficzne reprezentacje oparte o wielokąty. W tym artykule starano się udowodnić przydatności takiego podejścia. Nacisk położony zostanie na dostępne dane ogólnokrajowe. Wizualizacje oraz badanie wydajności zostaną oparte na przygotowanej aplikacji webowej.

2. Uogólniony opis algorytmu

Triangulacja Delaunay jest popularnym narzędziem z dziedziny geometrii obliczeniowej. Jest ona grafem dualnym diagramu Woronoja. Opracowana została przez matematyka Borysa Delone w Rosji 1934 r. Do generowania wizualizacji w tym artykule użyty został algorytm typu „dziel i rządź” opracowany przez Leonidasa J Guibasa i Jorge’a Stolfisa w pracy o tytule ”Primitives for the manipulation of general subdivisions and the computation of Voronoi” [3]. Jego złożoności obliczeniowa wynosi O(nlogn) dla najgorszego przypadku. Pozwala on na stworzenie siatki trójkątów na podstawie punktów o dowolnym rozmieszczeniu i dowolnej ilości na danej płaszczyźnie.

Opis algorytmu opisany na podstawie wcześniej wymienionej pracy. Wykonanie tego algorytmu rozpoczynane jest podzieleniem punktów na dwie połowy, lewą(L) i prawą(R), oddzielone danym koordynatem x. Następnie rekurencyjnie obliczana jest triangulacja Delauneya dla obu wyżej wymienionych podzbiorów. Końcowym krokiem jest połączenie połówek triangulacji w jedną triangulację całego oryginalnego zbioru. Warto tutaj zaznaczyć, że taka rekurencyjna dekompozycja nie

*Corresponding author
Email address: mateusz.nowosad@pollub.edu.pl (M. Nowosad)

może być użyta, gdy liczba punktów jest mniejsza niż cztery, ze względu na to, że jedna ze stron L albo R skończyłaby tylko z jednym punktem (każdy diagram Delaunay musi mieć co najmniej jedną krawędź więc minimum punktów na grupę to dwa). Przypadki z mniejszą ilością punktów niż cztery rozpatrywane są osobno.

Dodatkowo przy wykonywaniu takiego algorytmu warto najpierw posortować rozpatrywane punkty po koordynacie x, przypadki równości rozstrzygać sortując po koordynacie y i odrzucaniu punktów, które się pokrywają. Dzięki temu każda kolejna operacja dzielenia jest o stałym czasie. Rozpatrując krok łączenia dwóch obliczonych triangulacji, może zdarzyć się potrzeba usunięcia kilku krawędzi L - L i R - R lub dodania krawędzi L - R, jednak nigdy nie zostaną dodane nowe krawędzi L - L lub R - R, na co dowód umieszczony jest w artykule, z którego pochodzi ten algorytm.

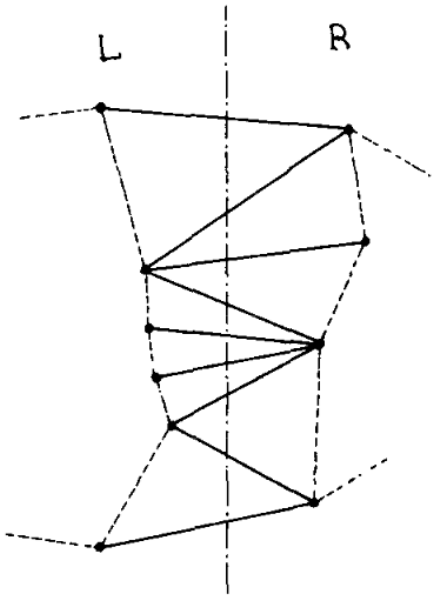
Niezbędnie ważnym dla działania algorytmu jest metoda *inCircle*. Posiada cztery argumenty, będące czterema unikalnymi punktami na badanej płaszczyźnie. Zwraca ona prawdę, gdy tylko punkt D jest we wnętrzu regionu płaszczyzny, który jest ograniczony przez okrąg zawierający ABC i leżąc na lewo od niego. Implikuje to, że D powinno leżeć wewnątrz takiego okręgu, gdy A, B i C tworzą trójkąt zorientowany przeciwnie do ruchu wskazówek zegara, a na zewnątrz, gdy tworzą trójkąt o orientacji zgodnej ze wskazówkami zegara. W przypadku gdzie A, B, C i D leżą na tym samym okręgu funkcja zwraca fałsz.

Dodatkowo potrzebny jest także predykat oznaczony jako $CCW(A, B, C)$ używany często do sprawdzenia, czy punkt X leży po lewej bądź prawej linii krawędzi e. Zwraca on prawdę, wtedy kiedy A, B i C tworzą trójkąt zorientowany przeciwnie do ruchu wskazówek zegara. Jego definicja widnieje poniżej na równaniu 1.

$$\begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} > 0 \tag{1}$$

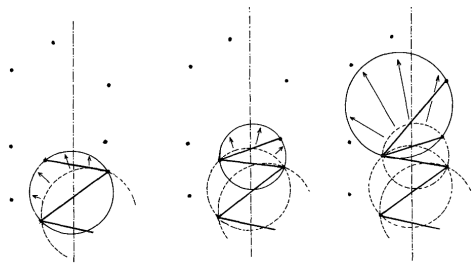
Zgodnie z przyjętą definicją krawędzie L - R muszą przecinać linie równoległą do osi y umieszczoną na określonym x. Trzeba tutaj założyć, że dowolne dwie krawędzie sąsiadujące w kolejności Y mają wspólny wierzchołek. Trzeci bok definiowanego przez nich trójkąta to krawędź L - L lub R - R. W literaturze spotykane jest oznaczenie bazowej krawędzi przecinającej ustalony x mianem *basel*, pamiętając przy tym, że jest ona skierowana od prawej do lewej. Kolejną krawędzią wytworzoną przy działaniu algorytmu będzie krawędź idąca od lewego końca *basel* do któregoś z punktów z grupy R leżącego nad nim lub analogicznie od prawego końca do punktu z grupy L ponad nim, mogą być oznaczone odpowiednio *lcand* i *rcand*.

Aby zobrazować działanie tego algorytmu, autorzy opisują, że należy wyobrazić sobie okrąg, który pnie się w górę, przeplatając pomiędzy L i R. W efekcie otrzymujemy krawędzie przecinające ustalony X. Co za tym



Rysunek 1: Poglądowa struktura krawędzi L - R [3].

idzie, gdy mamy koło opisujące trójkąt określony przez basel i poprzednią przecinającą krawędź. Należy kontynuować przekształcanie tego okręgu w inne okręgi mające *basel* jako cięciwę, ale leżące dalej w półpłaszczyźnie powyżej podstawy. Jest to korzystne, bo istnieje tylko jeden stopień swobody, ponieważ środek okręgu jest ograniczony do położenia na symetralnej cięciwy *basel* widoczne na rysunku 2. Okręgi będą się rozszerzały, aż napotkany zostanie kolejny punkt należący do L albo R, z wyjątkiem sytujacji gdzie *basel* jest już najwyższą możliwą krawędzią dla zbiorów L i R, co da początek nowemu trójkątowi z okręgiem opisanym, który też będzie transformowany aż do znalezienia następnego punktu. Nowa krawędź L - R tego trójkąta jest następną poprzeczną krawędzią wyznaczoną przez ciało głównej pętli algorytmu. Kandydat *lcand* jest obliczany tak, aby jako miejsce docelowe miał pierwszy punkt L napotkany w tym procesie a *rcand* pierwszy punkt R. Ostatecznie test wybiera punkt spośród tych dwóch, który był napotkany jako pierwszy. Iterację rozpoczyna się od znalezienia pierwszej możliwej krawędzi przecinającej L i R.



Rysunek 2: Poglądowa wizualizacja działania tworzenia krawędzi L - R [3].

3. Pseudokod algorytmu

Pseudokod algorytmu triangulacji Delauney przystosowany do zastosowania w różnych językach programowania jest przedstawiony na rysunku 3. Warto zwrócić uwagę na fakt, że jego złożoność obliczeniowa, wynoszącą $O(nlogn)$ to kolejna wskazówka, że jest on w grupie algorytmów typu "dziel i rządź" takich jak sortowanie przez kopcowanie, scalanie i sortowanie szybkie. Sprawia to, ze sam trzon jego struktury wydaje się znajomy dla programistów.

```
PROCEDURE Delaunay [S] RETURNS [le, re]:
  IF |S| = 2 THEN
    |Let s1, s2 be the two sites, in sorted order. Create an edge a from s1 to s2:|
    a ← MakeEdge[ ]; a.Org ← s1; a.Dest ← s2; RETURN [a, a.Sym]
  ELSIF |S| = 3 THEN
    |Let s1, s2, s3 be the three sites, in sorted order.|
    |Create edges a connecting s1 to s2 and b connecting s2 to s3:|
    a ← MakeEdge[ ]; b ← MakeEdge; Splice[a.Sym, b];
    a.Org ← s1; a.Dest ← b.Org ← s2; b.Dest ← s3;
    |Now close the triangle:|
    IF CCW[s1, s2, s3] THEN c ← Connect[b, a]; RETURN [a, b.Sym]
    ELSIF CCW[s1, s3, s2] THEN c ← Connect [b, a]; RETURN [c.Sym, c]
    ELSE |The three points are collinear:| RETURN [a, b.Sym] FI
  ELSE {|S| ≥ 4. Let L and R be the left and right halves of S.|
    [ldo, ldi] ← Delaunay[L]; [rdi, rdo] ← Delaunay[R];
    |Compute the lower common tangent of L and R:|
    DO
      IF LeftOf[rdi.Org, ldi] THEN ldi ← ldi.Lnext
      ELSIF RightOf[ldi.Org, rdi] THEN rdi ← rdi.Rprev
      ELSE EXIT FI
    OD;
    |Create a first cross edge basel from rdi.Org to ldi.Org:|
    basel ← Connect[rdi.Sym, ldi];
    IF ldi.Org = ldo.Org THEN ldo ← basel.Sym FI;
    IF rdi.Org = rdo.Org THEN rdo ← basel FI;
    DO |This is the merge loop.|
      |Locate the first L point (lcand.Dest) to be encountered by the rising bubble,|
      |and delete L edges out of basel.Dest that fail the circle test.|
      lcand ← basel.Sym.Onext;
      IF Valid[lcand] THEN
        WHILE InCircle
          [basel.Dest, basel.Org, lcand.Dest, lcand.Onext.Dest]
          DO t ← lcand.Onext; DeleteEdge[lcand]; lcand ← t OD
      FI;
      |Symmetrically, locate the first R point to be hit, and delete R edges:|
      rcand ← basel.Oprev;
      IF Valid[rcand] THEN
        WHILE InCircle
          [basel.Dest, basel.Org, rcand.Dest, rcand.Oprev.Dest]
          DO t ← rcand.Oprev; DeleteEdge[rcand]; rcand ← t OD
      FI;
      |If both lcand and rcand are invalid, then basel is the upper common tangent:|
      IF NOT Valid[lcand] AND NOT Valid[rcand] THEN EXIT FI;
      |The next cross edge is to be connected to either lcand.Dest or rcand.Dest.|
      |If both are valid, then choose the appropriate one using the InCircle test:|
      IF NOT Valid[lcand] OR
        (Valid[rcand] AND
          InCircle[lcand.Dest, lcand.Org, rcand.Org, rcand.Dest])
        THEN |Add cross edge basel from rcand.Dest to basel.Dest:|
          basel ← Connect[rcand, basel.Sym]
        ELSE |Add cross edge basel from basel.Org to lcand.Dest:|
          basel ← Connect[basel.Sym, lcand.Sym]
      FI
    OD;
    RETURN [ldo, rdo]
  FI
END Delaunay.
```

Rysunek 3: Pseudokod algorytmu obliczającego triangulacje Delaunay [3].

Samo działanie opisane jest w poprzednim rozdziale, a także jako przypisy w tym kodzie. Poniżej umieszczono także definicje wybranych pomocniczych funkcji $RightOf$, $LeftOf$ i $Valid$.

Algorytm 1: Definicje funkcji RightOf i LeftOf

input : X - współrzędne badanego punktu, e - zorientowana krawędź e w formie obiektu zawierającego punkty Org i $Dest$ odpowiadające odpowiednio puntkowi poczatkowemu i końcowemu
output: prawda lub fałsz

RightOf(X, e) **return** CCW$(X, e.Dest, e.Org)$
LeftOf(X, e): **return** CCW$(X, e.Org, e.Dest)$

Reszta definicji używanych funkcji zostanie pominięta ze względu na ich obszerność. Dostępnę są one w pracy "Primitives for the manipulation of general subdivisions and the computation of Voronoi" wraz z dowodami ich poprawności [3].

Algorytm 2: Definicja funkcji Valid

input : e, $basel$ - badana krawędź bazowa w takim samym formacie co e
output: prawda lub fałsz

Valid$(e, basel)$: **return** RightOf$(e.Dest, basel)$

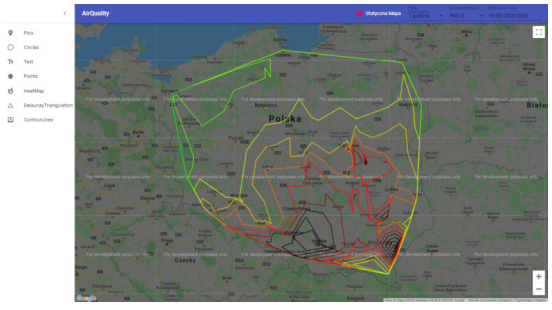
4. Program testowy

Na potrzeby przeprowadzenia testów przygotowana została aplikacja webowa typu "mashup" [4]. Wykorzystany stos technologiczny, często określany skrótem MERN [5] opiera się na technologiach:

- MongoDB – rozwiązanie bazodanowe,
- Express.js – serwerowy szkielet programistyczny,
- React – biblioteka służąca do tworzenia interfejsów użytkownika,
- Nodejs - środowisko uruchomieniowe oparte na silniku V8 firmy Google.

Uzasadnieniem takiego wyboru może być popularność tych technologii. Warunki eksperymentów starano się doprowadzić do jak najbliższych rzeczywistości. Zgodnie z ankietą przeprowadzoną na stronie StackOverflow [7], JavaScript ósmy rok z rzędu okazał się najpopularniej używanym językiem wśród respondentów, otrzymując 69.7%. React.js uplasował się na drugim miejscu z 36.8% spośród wszystkich szkieletów webowych. W kategorii „różnych" najczęściej używanym narzędziem okazał się Node.js z udziałem 51.9%. Zapewniło to środowisko bliskie docelowej implementacji.

Ważnym elementem aplikacji jest też oczywiście baza danych NoSQL MongoDB. Nierelacyjne systemy bazy danych coraz częściej znajdują wykorzystanie w SPA (ang. Single Page Application). Według powyższej ankiety używa go 26.7% badanych, więc skłania to do sprawdzenia wydajności takiego rozwiązania oraz poznania wyzwań, jakie towarzyszą implementacjom przy użyciu takiej technologii.



Rysunek 4: Główny widok programu testowego

Główny widok programu jest zaprezentowany na rysunku 4. Dane po pobraniu z bazy przekazywane są do algorytmu w formię współrzędnych i wartości wybra-

nego zanieczyszczenia. Decyzja o okresie agregacji nie wpływa na ilość danych przekazanych do algorytmu, co za tym idzie, nie zwiększa czasu jego wykonania. Wynika to z tego, że agregacja danych i wybór danego związku chemicznego wykonywane są już na poziomie bazy danych, dzięki temu każdy punkt na mapie ma przypisaną jedną uśrednioną wartość danego zanieczyszczenia a ilość stacji pomiarowych to ilość punktów używanych do stworzenia siatki na mapie. Do algorytmu trafiają współrzędne każdej stacji. Zgodnie z opisem algorytmu każda ze stacji staje się wtedy wierzchołkiem trójkąta. Kolory wypełnienia widoczne na mapie obliczane są na podstawie średniej arytmetycznej z zanieczyszczenia w każdym wierzchołku na zasadzie Zielony-Czerwony-Czarny, gdzie od zielonego do czerwonego mamy zakres wartości do limitu ustawionego przez WHO albo inny podmiot, a przyciemnianie czerwonego aż do czarnego zarezerwowane jest do wartości przekraczających limit.

5. Implementacja wizualizacji

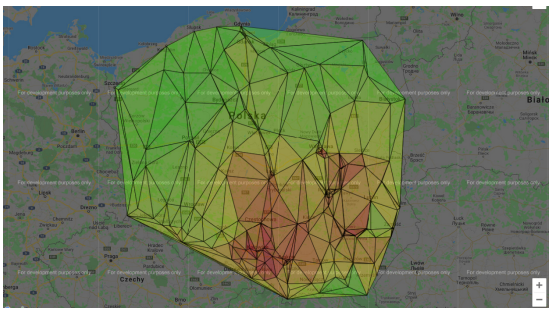
Dla wizualizacji z triangulacją Delauney wykorzystana została dodatkowa biblioteka „faster-delaunay". Została ona oparta na algorytmie opisanym w poprzedniej części tej pracy. Jest to także najszybsza dostępna biblioteka, jaka została sprawdzona. Ze względu na format argumentu, jaki przyjmuje funkcja dostarczana przez tę bibliotekę, dane wymagały przetworzenia.

Najważniejszym krokiem przy implementacji wizualizacji na mapie było prze-mapowanie danych pomiarowych na tablice zawierające współrzędne punktów pomiarowych. Wartości współrzędnych pomnożone zostały przez milion, aby uniknąć, problemów związanych z operacjami na liczbach zmiennoprzecinkowych dla tej biblioteki. Przed wykorzystaniem gotowe współrzędne wierzchołków nowo powstałych trójkątów zostaną podzielone ponownie przez milion, aby można było je nałożyć na mapę zgodnie z ich prawdziwym położeniem. Następnie ponownie przyporządkowano wartości do punktów pomiarowych o konkretnych współrzędnych. Nastąpiło także wyliczenie średniej z danych z wierzchołków trójkąta, aby można było określić wypełniający kolor. Dzięki tak przygotowanym danym samo wyświetlenie ich na mapie nie stanowi większego problemu. Wysyłane są przy użyciu biblioteki do obsługi map i zwracana zostaje finalna mapa ze zdefiniowanymi figurami.

6. Wizualizacje

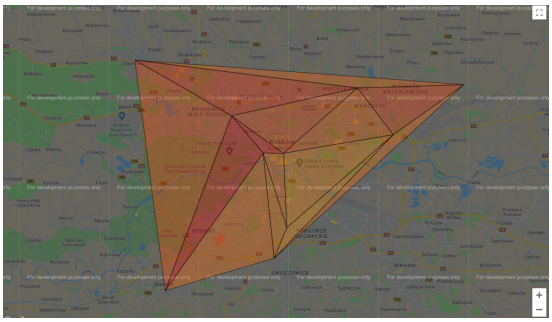
Oceniając tą wizualizacje warto pamiętać o celu wykorzystywania map statystycznych. W pracy pod tytułem "Graficzna prezentacja danych statystycznych Wykresy, mapy, GIS" umieszczona została bardzo trafna definicja: "Mapy statystyczne, w odróżnieniu od tabel statystycznych, czy wykresów statystycznych, pozwalają na jednoczesne zaprezentowanie przestrzennego rozmieszczenia opisywanego zjawiska (jego występowania) i jego wartości. Ich zadaniem nie jest pokazywanie precyzyjnej wartości zjawiska dla danego obiektu (choć jest to często możliwe do odczytania z mapy), tylko jego przybliżonej wartości wraz z relacjami przestrzennymi pomiędzy poszczególnymi obiektami" [6].

Mając na uwadze cel, który powinna taka reprezentacja osiągnąć po wykonaniu kilku eksperymentów nakładka wydaje się ciekawą możliwością do wykorzystania przy obrazowaniu trendów na większym terenie np. w skali krajowej. Nie wyklucza to możliwości wykorzystania jej na obszarze np. miastowym, co przedstawione jest poniżej (rysunek 6). Na rysunku 5 umieszczona została mapa, wygenerowana na podstawie algorytmu, oznaczona kolorami zgodnie z opisem w rozdziale 5. Na pierwszy rzut oka mapa przypomina kartogram, i rzeczywiście techniki wykonania są podobne, przedstawia także uśrednione wartości dla zanieczyszczenia z każdego wierzchołka danego trójkąta co dalej utwierdza to podobieństwo. Dokładność mapy ściśle zależy od ilości dostępnych punktów pomiarowych. Im mniejszy rozmiar każdego trójkąta, tym więcej szczegółów i trendów na mniejszym terenie możemy zobaczyć.



Rysunek 5: Wizualizacja oparta o Triangulacje Delaunay.

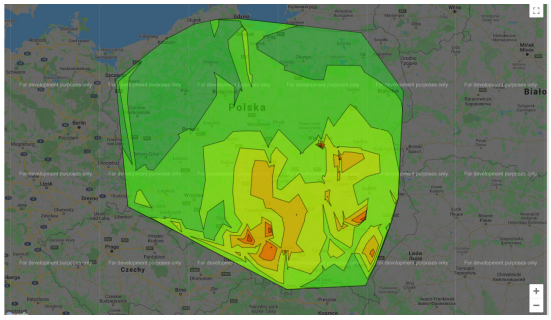
Na teren testowy wybrane zostało miasto Kraków i okolice. Zachowano kolorowanie na podstawie limitów, aby nie wprowadzać nieścisłości przy interpretacji.



Rysunek 6: Triangulacja Delaunay na małym obszarze.

Widać jednak, że są lepsze alternatywy dla przekazywania takich informacji. Ilość punktów pomiarowych, na obszarze miasta, nie była wystarczająca do uzyskania tak dobrego efektu co na skalę krajową. Mapa pozostała czytelna, lecz nie widać na niej trendów tak jasno, jak na poprzedniej. Warto tutaj zaznaczyć, że nie ma wizualizacji idealnej, która będzie perfekcyjna dla każdego przypadku. Trzeba iść na kompromisy i dobierać je dla konkretnych sytuacji.

Nie można jednak rozpatrywać takiej wizualizacji nie poruszając innych. W ramach porównania opisana zostanie także wizualizacja oparta o izolinie. Jest to znana i sprawdzona forma reprezentacji danych na mapie.

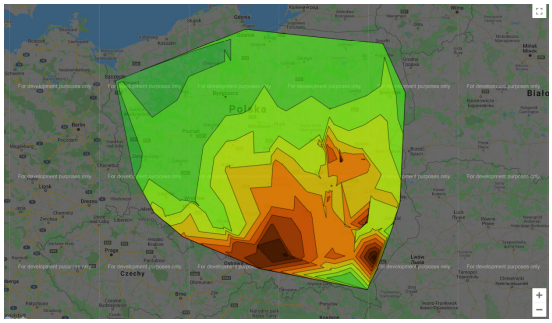


Rysunek 7: Przykładowy widok mapy z izoliniami.

Powyżej umieszczony został przykład wygenerowanej mapy (rysunek 7). Aby oddać wartości, skorzystano z kolorów wypełniających pole w wielokątach utworzonych przez izolinie. Tak samo, jak w poprzednich nakładkach kolory odpowiadają limitom zanieczyszczeń, a nie skali od najmniejszej wartości pomiaru na widocznym terenie do największej. Nie skorzystano też z żadnego wygładzania krawędzi, aby zachować jak najdokładniejsze oddanie rzeczywistości.

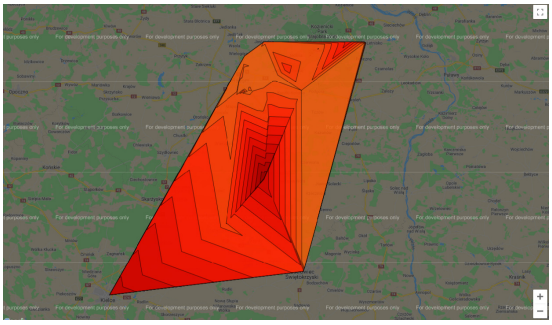
Ulepszenie wyglądu takiej mapy wymagałoby więcej punktów pomiarowych. Jest to cecha wspólna z poprzednią wizualizacją. Pewien poziom wygładzania mógłby ulepszyć wizualne aspekty tej mapy, jednak tą, decyzje najlepiej podjąć już na etapie konkretnej implementacji.

Przedstawione na tej mapie dane są spójne z poprzednimi. Można jednak przedstawić jeszcze dodatkowy obraz wygenerowanej mapy na podstawie poziomów innego zanieczyszczenia. Ta nakładka wygląda zdecydowanie lepiej wizualnie niż poprzednia. Jest to swoistego rodzaju najlepszy przypadek z dostępnych danych. Pokazuje to potencjał tego podejścia (rysunek 8).



Rysunek 8: Przykładowy widok mapy z izoliniami.

W następnej części analizowana jest przydatność tej mapy przy obszarach miejskich. Mniejsza ilość czujników i fakt, że wartości zanieczyszczeń są na podobnych poziomach, utrudnia wygenerowanie przydanej mapy. Tak jak w poprzednim eksperymencie zadecydowano zachować te same kolory ze względu na spójność danych.



Rysunek 9: Przykładowy widok mapy z izoliniami.

Jak widać poniżej ciężko odczytać jakieś wartościowe informacje z tej mapy. Z prezentowanych jak do tej pory wizualizacji ta prezentuje się najgorzej (rysunek 9). Nie nadaje się w tym stanie do przestawiania danych miejskich, chyba że mamy wystarczającą liczbę punktów pomiarowych oraz kolory zostaną wyskalowane zgodnie z najmniejsza i największą badaną wartością. Przy takiej reprezentacji trzeba umieścić legendę dotyczącą skali, aby nie wprowadzać użytkowników w błąd. Pozwoli to na znalezienie punktów problematycznych na danym obszarze.

7. Testy wydajnościowe

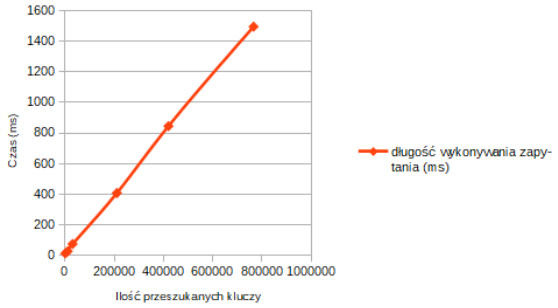
Wykorzystanie rozwiązań zaprezentowanych w tej pracy będzie wiązało się ściśle z szybkością działania. Wymagania współczesnego użytkownika co do responsywności aplikacji internetowych. Platforma testowa wyposażona była w procesor Intel i7 4720HQ o taktowaniu 3,60 GHz oraz 16 GB pamięci ram DDR3L CL9 o prędkości 1600MHz. Pierwszy aspekt aplikacji poddany analizie to, jak sprawdza się model bazodanowy, stworzony na potrzeby tego badania. Aktualny stan kolekcji zawiera 1373039 obiektów pomiarów i zajmuje, po wyeksportowaniu, do pliku BSON 195 MB. Czasy prezentują się następująco:

Tabela 1: Czasy agregacji dla różnych okresów uśredniania

Okres uśredniania (h)	Średnia czasu wykonania zapytania z pięciu prób (ms)	Liczba dokumentów zebranych do uśredniania (ms)	Ilość przeszukanych kluczy (n)
1	14.22	250	1238
8	28.39	2239	11160
24	75.82	6239	31064
168	408.33	42274	210501
336	845.62	84365	420144
672	1495.47	157155	765429

Wybrano takie okresy, ponieważ wydawały się zbliżone do tego, jak mogłyby być wykorzystane te dane i wizualizacje. Czasy te są akceptowalne i pozwalają na wygodne korzystanie z aplikacji bez większych okresów

oczekiwania. Warto jeszcze zauważyć, że użyto unikalnych indeksów zawierających pole daty, aby maksymalnie przyspieszyć te operacje.



Rysunek 10: Triangulacja Delaunay na małym obszarze.

Kolejny aspektem poddanym analizie wydajnościowej będzie przygotowanie danych do generowania samych wizualizacji. Przygotowanie będzie oznaczać czas potrzebny do wykonania potrzebnych algorytmów albo obróbki danych, jeżeli nie można ich wykorzystać bezpośrednio z bazy. Z tego powodu dane będą dotyczyły dwóch wizualizacji mapy z izoliniami i opartej o triangulacje Delauney. Sam czas odpowiedzi Google Maps API będzie wyłączony z tego badania ze względu na zbyt wiele zmiennych wpływających na długość oczekiwania takich jak lokalizacja, prędkość sieci, obciążenie serwerów w danym momencie itp. na które badający nie ma wpływu.

Tabela 2: Czasy agregacji dla różnych okresów uśredniania

Wizualizacja	Czas wykonywania algorymów (ms)	Całkowity czas przygotowania danych (ms)
Triangulacja Delauney	9	24
Izolinie	16	17

8. Mocne i słabe strony

Rozpatrując mocne i słabe strony przedstawionego rozwiązania możemy zidentyfikować:

Mocne strony:

- Ciekawa alternatywa dla innych map statystycznych przy odpowiednio dużej ilości czujników.
- Nowatorskość rozwiązani może zachęcać do zapoznania się z przekazywanymi informacjami.
- Blisko stuprocentowe pokrycie mapy Polski uśrednionymi danymi, mało pustych obszarów bez żadnych danych.

Słabe strony:

- Silna zależność dokładności od ilości punktów pomiarowych.
- Duże uśrednienie powoduje niedokładność danych przy obszarach z małym zagęszczeniem stacji pomiarowych.

W ramach porównania dla wizualizacji opartych o izolinie można wymienić:

Mocne strony:

- Znajomy format wizualizacji, duże prawdopodobieństwo, że odbiorca będzie mógł korzystać z dotychczasowej wiedzy przy analizie.
- Dobrze widoczne uogólnione trendy na obszarach z dużą ilością punktów pomiarowych.
- Tak jak poprzednia wizualizacja, prawie stuprocentowe pokrycie Polski uśrednionymi danymi, mało pustych obszarów bez żadnych danych.

Słabe strony:

- Najgorsza z przedstawionych w reprezentacji obszarów o niewielkiej ilości stacji pomiarowych.
- Nakładające się kolory zasłaniają kluczowe elementy mapy. Ten efekt można zniwelować kolorując same izolinie.

9. Wnioski

Temat zanieczyszczeń powietrza będzie towarzyszył społeczeństwu akademickiemu przez dłuższy czas. Jego obszerność i bezprecedensowe znaczenie daje bardzo duże możliwości dopasowania go do różnych dyscyplin naukowych. Przedstawione w tym artykule problemy i ich analiza dostarczają narzędzi do lepszego zrozumienia problemu stanu środowiska, jak i zwiększaniu świadomości społecznej. Wykorzystywanie wizualizacji przy użyciu niespotykanych technik bądź algorytmów może przynieść nowe ciekawe metody przekazywania informacji. Wizualizacja będąca tematem tej pracy zdecydowanie spełnia oczekiwania, jeżeli chodzi o prezentowanie trendów w danych i przydatność. Ciekawym rozszerzeniem tych badań mogłoby być zbadanie innych algorytmów opartych na diagramach Vornoia do wygenerowania map przedstawiających zanieczyszczenia powietrza.

Literatura

[1] J. Lelieveld,Cardiovascular disease burden from ambient air pollution in Europe reassessed using novel hazard ratio functions, Eur Heart J, t. 40, nr 20, s. 1590–1596, maj 2019, doi: 10.1093/eurheartj/ehz135.

[2] D.W. Dockery, C.A. Pope, Acute respiratory effects of particulate air pollution, Annual review of public health, t. 15, nr 1, s. 107–132, 1994.

[3] L. Guibas, J. Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi, ACM Trans. Graph., t. 4, nr 2, s. 74–123, kwi. 1985, doi: 10.1145/282918.282923.

[4] D. Fichter, What is a mashup, Library Mashups. Exploring new ways to deliver library data. Medford, NJ: Information Today, Inc, 2009.

[5] V. Subramanian, Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node. Apress, 2017.

[6] M. Pieniążek, B. Szejgiec, M. Zych, A. Ajdyn, G. Nowakowska, Graficzna prezentacja danych statystycznych Wykresy, mapy, GIS. Warszawa: Główny

Urząd Statystyczny Departament Badań Regionalnych i Środowiska, 2014.

[7] „Stack Overflow Developer Survey 2020", Stack Overflow. https://insights.stackoverflow.com/survey/2020/?utm˙source=social-share&utm˙medium=social&utm˙campaign=dev-survey-2020, (udostępniono cze. 25, 2020).