

# Słupy oświetleniowe

Dawid Jarosz, Tomasz Kolbusz

January 25, 2023

## Contents

<b>1 Słupy oświetleniowe</b>	<b>1</b>
1.1 Cel projektu . . . . .	1
1.2 Dane wejściowe . . . . .	1
1.3 Wykorzystane technologie . . . . .	1
1.4 Podział prac . . . . .	1
1.5 Uruchomienie aplikacji . . . . .	1
1.5.1 Załadowanie danych . . . . .	1
1.5.2 Uruchomienie skryptu . . . . .	2
1.6 Algorytm . . . . .	2

## 1 Słupy oświetleniowe

### 1.1 Cel projektu

Projekt ma za zadanie rozmieścić na mapie miasta słupy oświetleniowe co określona odległość w pasie drogowym. Słupy nie mogą stać w tym samym miejscu co elementy infrastruktury miasta tj. bramy wjazdowe, chodniki.

### 1.2 Dane wejściowe

Jako zbiór danych używamy mapy drogowej oraz infrastruktury miasta Washington DC. Dane pochodzą z serwisu [... tutaj link]. Wykorzystaliśmy dane dostępne w zbiorach Roads i Structure<sub>Lines</sub>. Zbiory pobraliśmy w formacie shp, pozwoli to na łatwe załadowanie ich do bazy danych.

### 1.3 Wykorzystane technologie

Do przechowywania danych i ich przetwarzania wykorzystujemy PostgreSQL wraz z rozszerzeniem PostGIS. Do skryptu wyznaczającego położenie słupów wykorzystujemy Deno.

### 1.4 Podział prac

Tomasz Kolbusz wymyślił wstępny algorytm, który w późniejszym etapie okazał się niewystarczający.

Dawid Jarosz ulepszył algorytm, który został wspólnie zaimplementowany.

### 1.5 Uruchomienie aplikacji

#### 1.5.1 Załadowanie danych

Do załadowania danych z plików shp używamy programu shp2sql. Flaga -D zmienia format wyjściowy z programu na postres'owy dump, przyspiesza to ładowanie danych. Flaga -I dokłada indeksy GiST na kolumny z geometrią.

```
shp2pgsql -D -I -s 4326 Structure_Lines_1999.shp Structure_Lines | psql
```

```
shp2pgsql -D -I -s 4326 Roads.shp Roads | psql
```

### 1.5.2 Uruchomienie skryptu

Skrypt jest uruchamiany w deno, czyli środowisku uruchomieniowym typescript. Jedną z cech deno jest brak konieczności instalacji zależności. Zostaną one automatycznie pobrane przy starcie skryptu. Sam skrypt pozwala nam wywołać kolejne polecenia SQL za pomocą których wyznaczamy położenie słupów oświetleniowych.

```
deno run --allow-net app.ts
```

## 1.6 Algorytm

1. Wczytanie danych do bazy oraz utworzenie potrzebnych tabel.
2. Wczytanie ze zbioru Roads wszystkich obiektów według typów.

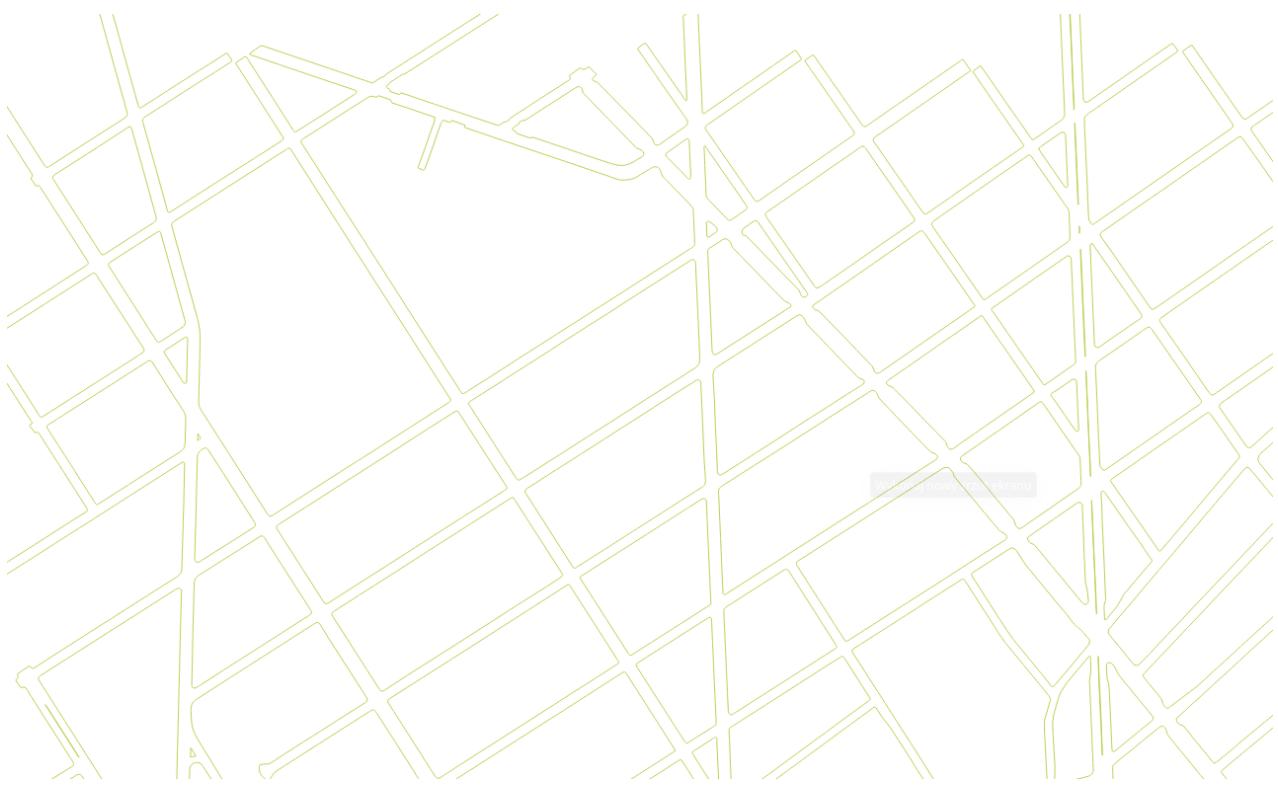
```
insert into objects2 select description, st_transform(geom, 2855) from roads;
```



1. Wyciągnięcie konturów dróg

Wykorzystując `st_boundary` i `st_buffer` możemy wytyczyć obrrys drogi w zadanej odległości.

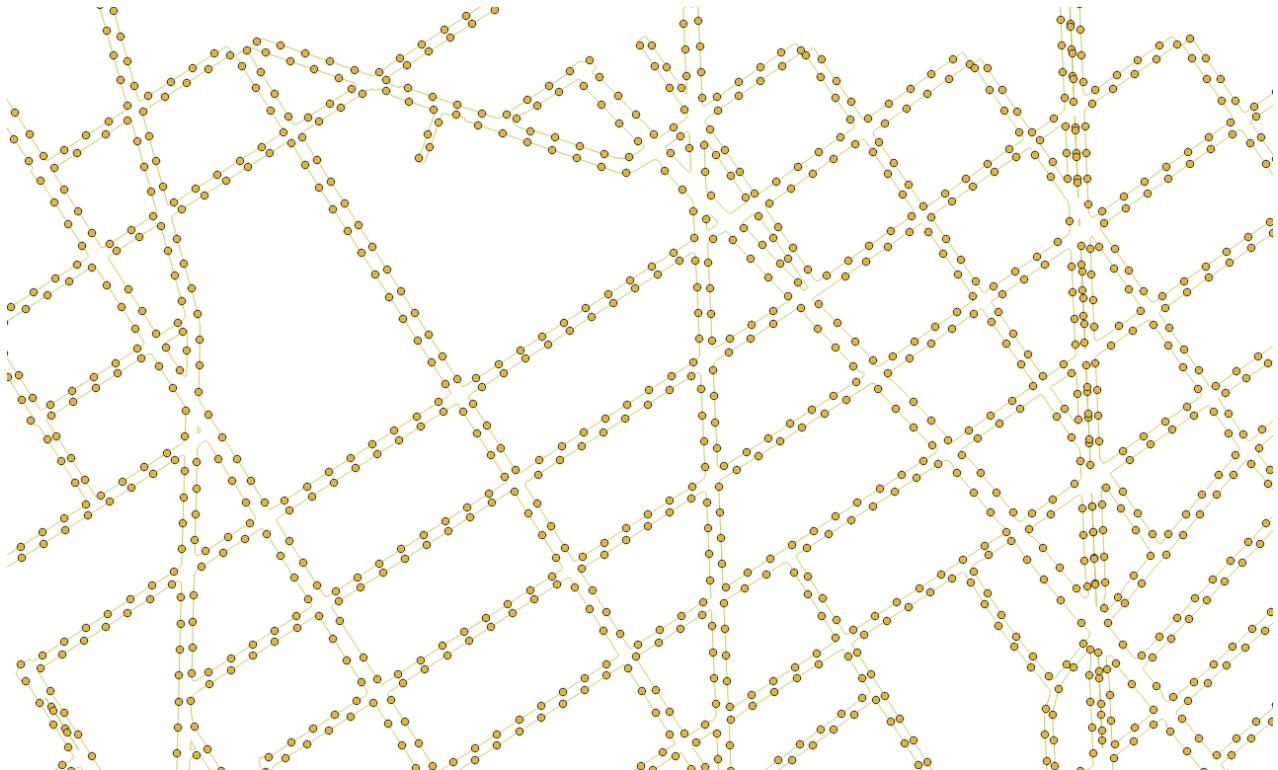
```
insert into roadsContours
select
  (st_dump(st_boundary(st_union(st_buffer(geom, ${distanceFromRoad}))))).geom
from objects
where description = 'Road' or description = 'Intersection';
```



### 1. Ustawienie słupów wzdłuż dorgi co określony dystans

Do wyznaczenia początkowego położenia słupów wykorzystujemy funkcje `st_lineinterpolatepoints`, która pozwala wyznaczyć punkty na linii co zadany dystans.

```
insert into initialPolesPlacement
select st_lineinterpolatepoints(geom, ${poleSpacing}/st_length(geom))
from roadsContours
where st_length(geom) > ${poleSpacing};
```



## 1. Rozwiązywanie problemu skrzyżowań

W przypadku skrzyżowań, nie chcieliśmy żeby nasz algorytm ustawił słupy w bliskiej odległości od skrzyżowania, a jednocześnie żeby w pewnej zadanej odległości ustawić słupy, które mogłyby doświetlać przejścia dla pieszych.

### a. Wyznaczenie dwóch stref wokoło skrzyżowania

W tym punkcie wyznaczamy dwie strefy - większą z której usuniemy postawione słupy, drugą mniejszą, której przeciecie krawędzi z liniami według których wyznaczaliśmy pierwotne położenie słupów. Pozwoli to nam ustawić słupy w zadanej odległości od skrzyżowania (można wykorzystać jako oświetlenie przejść), a uniknąć sytuacji w której postawione słupy będą znajdować się bardzo blisko już postawionych słupów.

```
insert into intersections
select 'clearZone', st_union(st_buffer(geom, ${clearZoneSize}))
from objects
where description = 'Intersection';

insert into intersections
select 'placeZone', st_union(st_buffer(geom, ${placeZoneSize}))
from objects
where description = 'Intersection';
```



### b. Usunięcie słupów z większej strefy

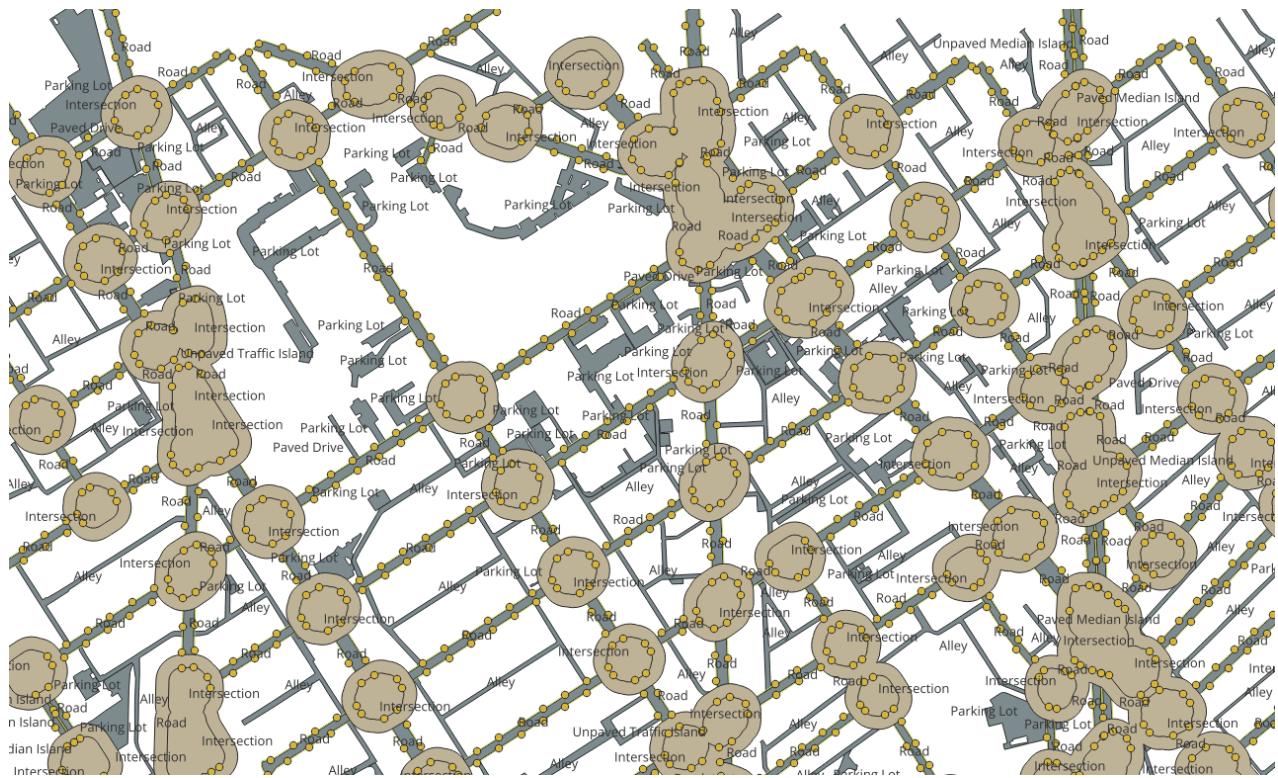
```
insert into polesAfterIntersections
select (st_dump(st_difference(
    (select st_union(geom) from initialpolesplacement),
    (select st_union(geom) from intersections where type = 'clearZone'))))
).geom;
```

### c. Dodanie słupów na przecięciach

```

insert into polesAfterIntersections
select st_intersection(
    select st_boundary(st_union(geom)) from intersections where type = 'placeZone'),
    (select st_union(geom) from roadsContours)
);

```



## 1. Znalezienie przeszkód które uniemożliwiają postawienie słupów.

Tutaj jako przeszkody zakwalifikowaliśmy pozostałe obiekty ze zbioru Roads, takie jak alejki, parkingi, podjazdy oraz zbiór StructureLines.

```

insert into obstacles
select st_buffer(st_union(geom), 3)
from objects
where description != 'Road' and description != 'Intersection';

```



### 1. Usunięcie słupów które kolidują z przeszkodami.

```
insert into clearedFromObstacles
select (st_dump(st_difference(
    (select st_union(geom) from polesAfterIntersections),
    (select st_union(geom) from obstacles))))
).geom;
```

### 1. Rozwiążanie problemu z brakującymi słupami po usunięciu kolidujących.

Po usunięciu kolidujących słupów odległości między niektórymi słupami mogą być znacznie większe niż założone, dlatego postanowiliśmy usunięte punkty przesunąć na pozycję na której nie będą kolidowały z innymi obiektami.

#### a. Wybranie uniętych punktów.

```
insert into toMove
select (st_dump(st_intersection(
    (select st_union(geom) from polesAfterIntersections),
    (select st_union(geom) from obstacles))))
).geom;
```

#### b. Znalezienie linii według których były ustawiane słupy, ale wycięcie części które kolidują z obiekta

```
insert into contoursWithoutObstacles
select st_difference(
    (select st_union(geom) from roadscontours),
    (select st_union(geom) from obstacles)
);
```

#### c. Znalezienie najbliższych punktów, które nie kolidują z przeszkodami.

W tym miejscu dla każdego usuniętego punktu szukamy najbliższego możliwego położenia dla punktu. Efektywnie to przesuwa punkt na linię która przechodzi przez tą przeszkodę, stawiając go po najbliższej stronie przeszkody.

```

insert into movedPoints
select st_closestPoint((select geom from contoursWithoutObstacles), geom)
from tomove;

```



Szare punkty zostały przesunięte do pozycji niebieskich punktów.

1. Znalezienie punktów które postawione są za blisko siebie i usunięcie ich.

```

insert into finalPoles select ST_RemoveRepeatedPoints(st_collect(geom), ${minLength}) from cleared

```

