# Combined algorithm for Voronoi diagram construction in application to dynamic ride sharing

A. Butenko, J. M. Gómez

Anton Butenko, University of Oldenburg, Germany
email: anton.butenko@uol.de

## resume

Anton Butenko

<u>Work</u>

- University of Oldenburg, Department of Computing science, Germany, since 2021
- University of Bremen, Institute of Geography, Climate Lab, Germany, 2016-2019
- Basque centre for applied mathematics (BCAM), Bilbao, Spain, 2016
- Moscow State institute of electronics and mathematics of National research university «Higher school of economics» (MIEM NRU HSE), Russia, 2012 – 2017
- Space research institute of Russian academy of sciences (IKI RAN), Russia, 2012 -2016

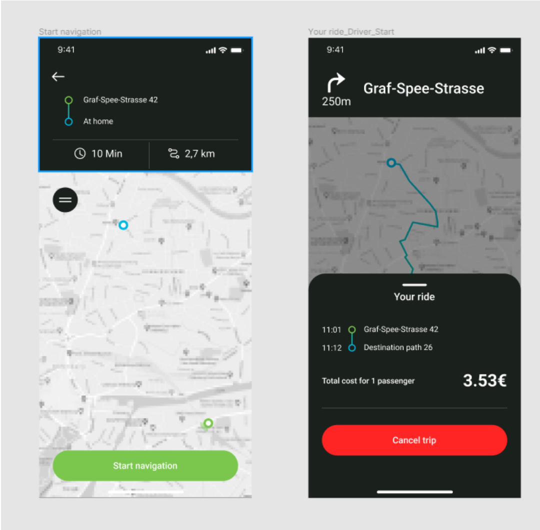
<u>Education</u>

Oryol state university, Russia

M.Sc., Applied mathematics and informatics, 2011 instaride

**Instaride** is a mobile application for spontaneous shared rides. Connection of potential driver and passenger occurs when driver is already on the way.

While driver's compromise is to make route detour, passenger's compromise is to walk to/from pick-up point and drop-off point (meeting points).


user interface

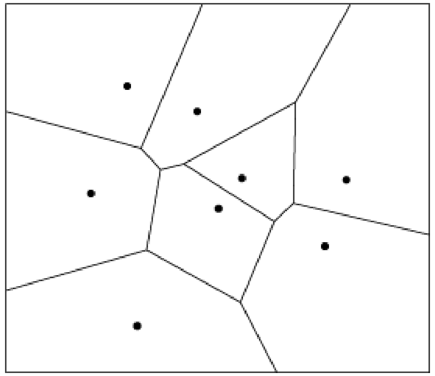
Criteria for meeting points pre-selection:

- Free parking opportunity
- Easily recognizable landmarks
- Pedestrinan zones
- Illumination
- Safety

(bus stops, petrol stations, supermarket parkings, roads intersection etc)

the need to determine the nearest meeting point leads to the nearest neighbor search problem (NNS)

## Standard Voronoi diagram and its generalizations



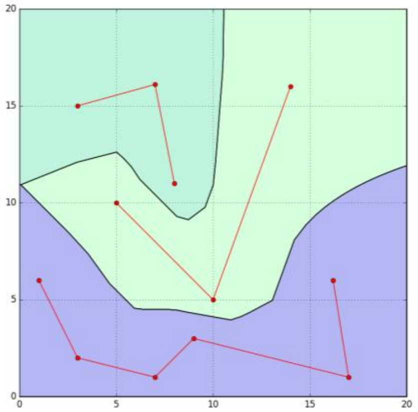
$$\rho(A,B) = \left[\sum_i (x_i - y_i)^2\right]^{1/2}$$

$A, B \in \mathbb{R}^2$

As one of the most effective approach for NNS, Voronoi diagram decomposes plane into areas of proximity based on Euclidean distance.
However, in urban environments its accuracy is insufficient, especially under presence of obtacles: water reservoirs, railway tracks, highways, industrial zones, hilly terrain.
Accuracy can be improved by using generalized diagrams.

generalization of:

a) seed objects




b) distance -> metric function

$\rho(x,y)\colon\ \Omega \to \mathbb{R}$, such as

- $\rho(x,y) = \rho(y,x)$
- $\rho(x,y) = 0 \Leftrightarrow x = y$
- $\rho(x,y) \le \rho(x,z) + \rho(z,y)$

## Combined Voronoi diagram construction

notation:

$\rho : L \times L \to \mathbb{R}^{+} \cup \{0\}$ - metric function

$B(x,y) = \{z : \rho(x,z) = \rho(y,z)\}$ - bisector

$D(x,y) = \{z : \rho(x,z) < \rho(y,z)\}$ - half-plane

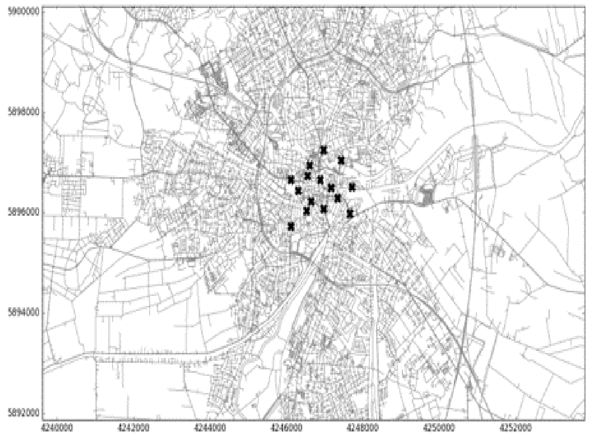
$S = \{s_1, \ldots, s_k\} \in L_\rho$ - seeds

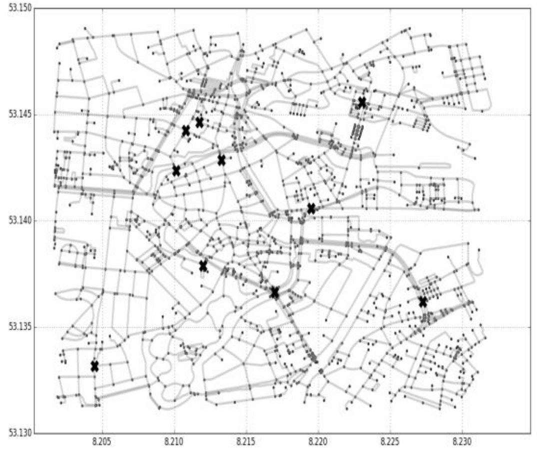
=>

$VR(s_i, S) = \bigcap_{i \neq j} D(s_i, s_j)$ - Voronoi cell

$V(S) = \bigcup_{i \neq j} \overline{VR}(s_i, S) \cap \overline{VR}(s_j, S)$ - Voronoi diagram

City as transportation network:





Considering city as a weighted graph $G(V,E)$, where

$E = \{e_i\}$ – edges = roads and streets;
$V = \{v_i\}$ - vertices = intersections and deadlocks;
$w(e_i)$ – edges' weights = proximity (time, distance..),

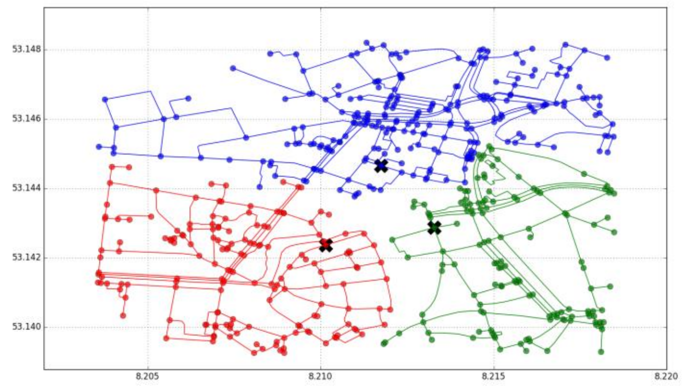
we set metric function as
$\rho_G(v_i, v_j)$ =the shortest path length between 2 vertices.

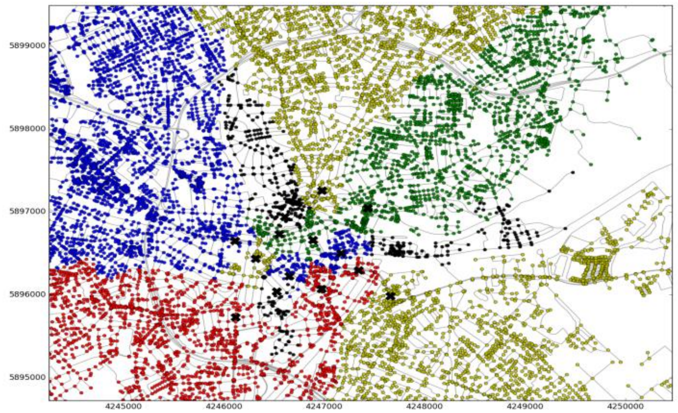
**Combined Voronoi diagram construction. Discrete diagram on the graph.**

For given graph $G(V, E)$

$S = \{s_1, .., s_k\} \in V$ – meeting points = seeds

Voronoi diagram brakes up set $V = V_1 \oplus ... \oplus V_k$ and => $E = E_1 \oplus ... \oplus E_k$ into sets of Voronoi cells



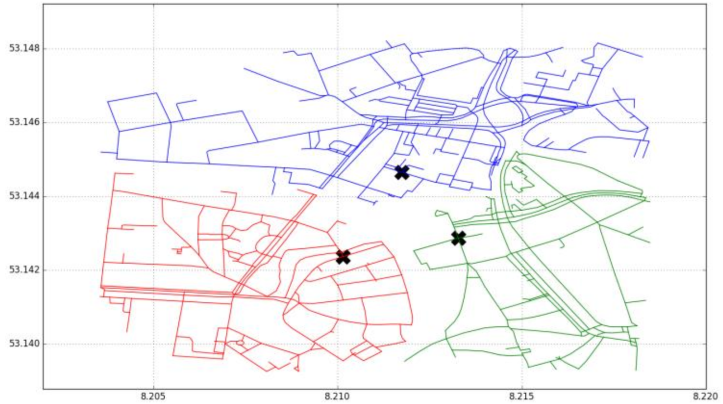

Algorithm steps:

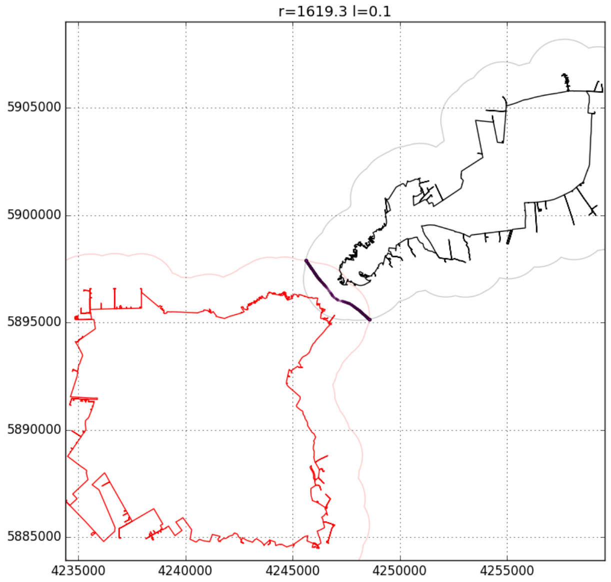
1. City representation as graph is obtained from OpenStreetMap project geodata;
2. Meeting points $S = \{s_1, .., s_k\}$ are added to $V$;
3. Using Dijkstra algorithm, find $\rho_G(v, s_i)\ \forall i$. If for $s_j$ $\rho_G(v, s_i) = min[\rho_G(v, s_i)]$ then add $v$ to $VR(s_j)$;
4. From constructed cells $V_m$ get $E_m$ - subset of edges that belong to the cell

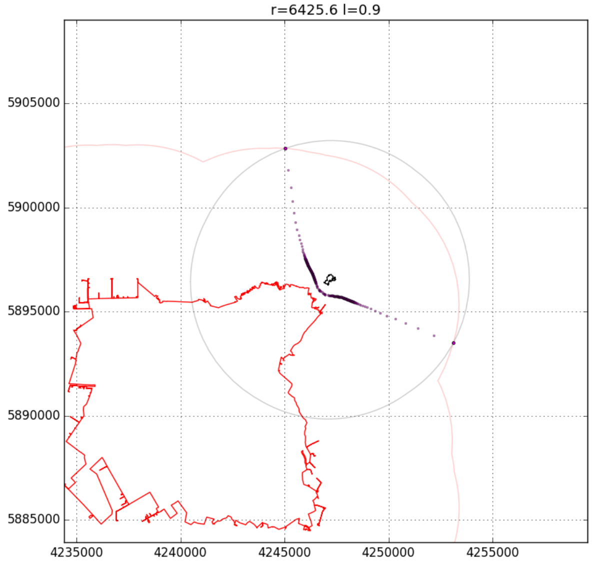
## Combined Voronoi diagram construction. Continuous diagram on the plane.

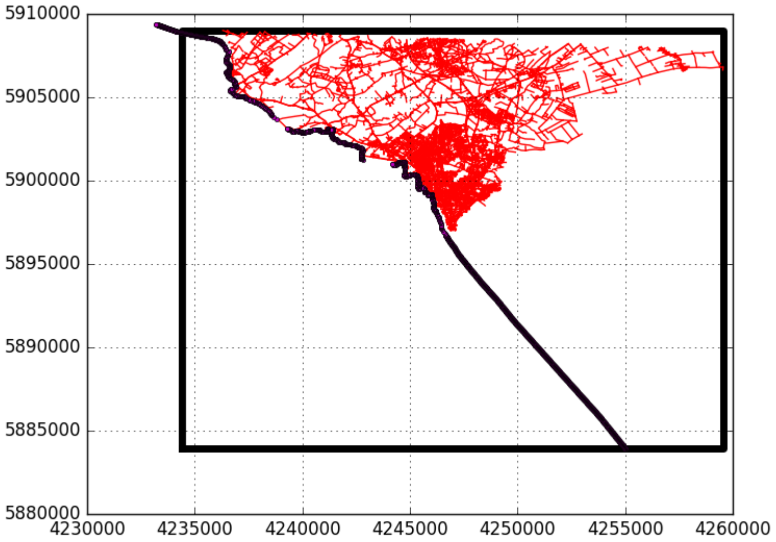
Considering set of edges $E$ as geometrical objects – lines $E'$ – we can construct continuous Voronoi diagram on the plane with Euclidean distance as metric function:

1. To find bisector $B(E'_m, E'_n) = \bigcup[S_r(E'_m) \cap S(E'_n)],\ r \in (0, +\infty)$ interpolate points obtained for $r_{k+1} = r_k + \Delta r_k$ ;
2. Determine corresponding half-plane $D(E'_m, E'_n) \ni E'_m$;
3. Intersection of all half-planes for $m \neq n$ forms Voronoi cell for $E'_m$: $VR(E'_m) = \bigcap B(E'_m, E'_n) \cap \Omega$ ;



## Evaluation. Further work.

Comparison with standard diagram for the central part of Oldenburg:

$$\Delta S = \frac{1}{S(\Omega)} \sum_i \frac{S(C_1^i \triangle C_2^i)}{S(C_1^i \cup C_2^i)} \cdot (S(C_1^i) + S(C_2^i)) \cdot 0.5$$

$\Delta S \approx 0.31$, where $S_{1,2}$- two types of Voronoi cells: standard and combined

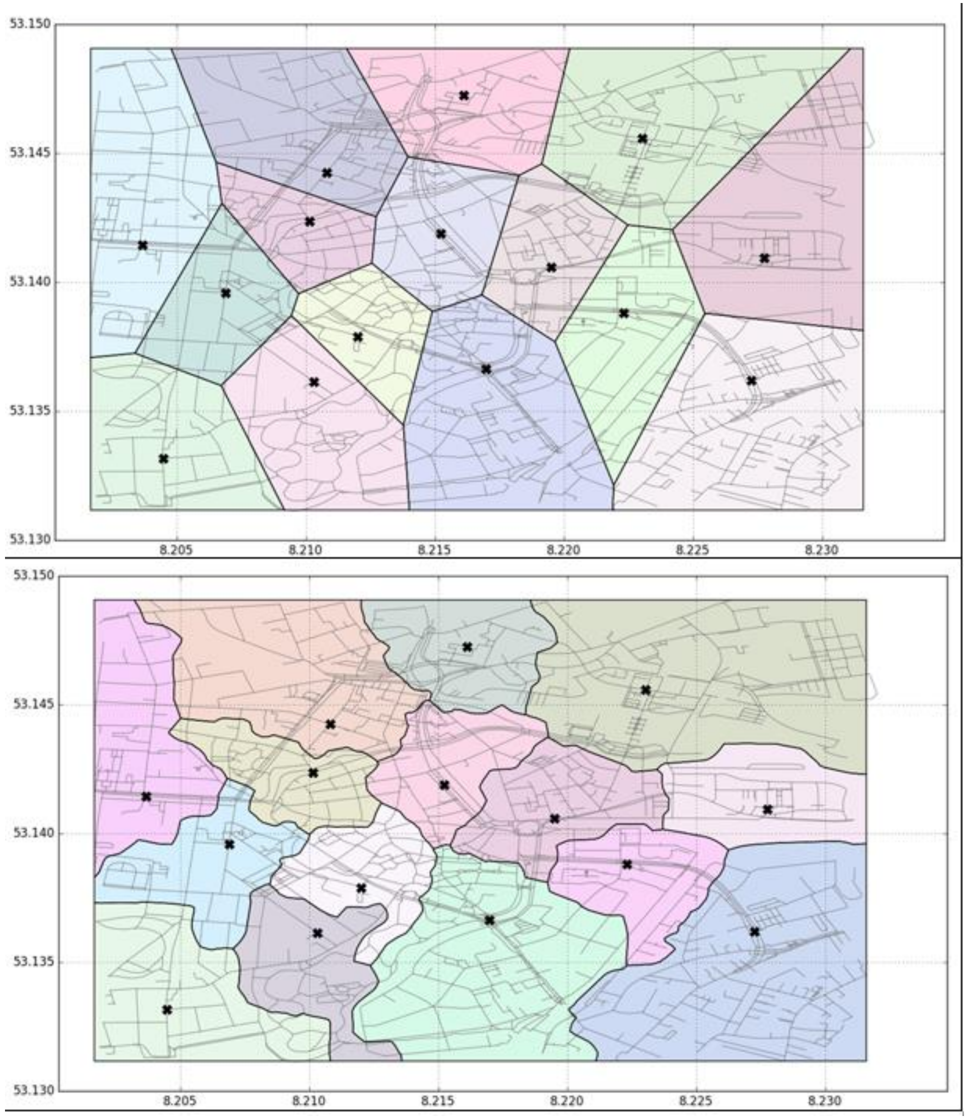
Also for 1000 random locations we obtain nearest meeting point:

a) from standard Voronoi diagram(SVD);
b) combined Voronoi diagram (CVD);
c) by direct route computation using Openrouteservice (ORS)

- nearest meeting points obtained from DE and DC are equal - 792 locations;
- otherwise - 208 locations.

for latter 208:

- meeting points obtained from ORS and CDV are equal - 165 locations;
- meeting points obtained from ORS and SVD are equal - 41 locations;
- Otherwise – 2 locations.




Further work:

- ➢ Using additional bandwidth data
- ➢ Algorithm‘s computatinal complexity reduction
- ➢ Complete algorithm evaluation
- ➢ Optimization complex topography features processing

# Tools

python™

NumPy

matplotlib

OSMNX

OpenStreetMap

NetworkX
Network Analysis in Python

openroute
service

Joblib

# Thank you for your attention