Segmentation of point-based geographic space

Mateusz Ośko
IQSolution Sp. Z.o.o., ul. Rubież 46, 61-612 Poznań, Poland
Labolaratory of Geoinformatics, Adam Mickiewicz University in 61-680 Poznań, Poland
mosko@iqsolution.pl

Jarosław Jasiewicz[1]
Labolaratory of Geoinformatics, Adam Mickiewicz University in Poznań, Poland, Krygowskiego 10, 61-680 Poznań, Poland
jarekj@amu.edu.pl

Przemysław Szarwark
IQSolution Sp. Z.o.o., ul. Rubież 46, 61-612 Poznań, Poland
`http://iqsolution.pl`
pszarwark@iqsolution.pl

Abstract

In this paper, we present the algorithm aimed to segment the type of geographical space where points are a substantial component. The research problem falls within the mainstream of Automated Unit Design (AUD). The objective function of the solution is a balance between the size of segmented units expressed as an attribute of points datasets and its agreement with constraints provided by the geographic space. An algorithm has three free parameters; two of them allow one to control the objective function: the size of segmented units and allowable deviation from the size. The paper contains a case study where we show how our approach segment the geographic space of the City of Poznań.

Funding *Jarosław Jasiewicz*: This work was founded by NCBiR Grant No: POIR.01.01.01-00-1271/17-00
Przemysław Szarwark: This work was founded by NCBiR Grant No: POIR.01.01.01-00-1271/17-00

1 Introduction

Points datasets are one of the increasingly growing sources of data in recent years. Large data point sets appear on different scales: from micro-scale, like colonies of microorganism, through the medium-scale like clustering addresses of diseases [2], or agricultural application [3]. Point addresses or massive observations like LIDAR or mobile logs are sources of point data explicitly placed in geographical space. In GIScience, algorithms allowing the segmentation of geographic space are vital. If points are located in 3D or 2D Cartesian space without additional geographic constraints, this problem reduces the selection of the optimal clustering algorithm where coordinates are the only input. In more complex situations, where clusters are influenced by scale of units and additional geographic constrains, the problem becomes intricate and cannot be solved by such an ordinary clustering. Geographic space has natural obstacles and varied friction, which may relate to the points on a different level. This is well

[1] corresponding author

known Automated Unit Design (AUD) [8, 1, 7, 5], till now explored mostly based on basic spatial units also refereed as 'building blocks'. Our work extends earlier concepts by setting spatial relationship between points.The segmentation process is based on the procedure proposed recently by [10] for images and later [4] for raster pattern data.

The situation is presented in Figure 1. City space is a typical illustration of the problem. In plate A, space (city limits) is defined by its borders and is dived by hard obstacles (here river - blue line). The city has a street network (black lines) that changes the movement friction and influences the relation (distance) between several point objects (addresses). Also, points do not represent single entities. One address point, for example, can include several flats. Although Cartesian coordinates naturally cluster the points, clustering does not include other constraints provided by the geographic space. We present the desired situation in plate B after segmenting into smaller units, which follow the natural limitations of space and the relationships within the point data sets.

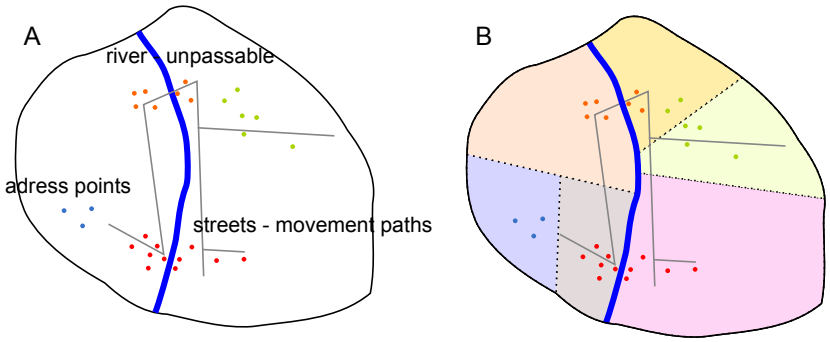

Figure 1 The schematic situation of the input data (A) and expected results (B). See explanation in text. Plate A show results of clustering by coordinates, Plate B show expected results. The colours are used for visual enhancements, without additional meaning

In this paper, we present preliminary results of a new approach to segmenting any geographic space where the pattern of points data set is an essential part of the space. The objective function is the balance between the size and consistency of the segments in terms of the existing geographic constraints. The proposed algorithm requires a space defined by its border, and points within the space must have relations described by the dense distance matrix. The distances do not require Euclidean, and it could be any metric that satisfies three metric axioms (identity, symmetry, triangle inequality). We also present a case study where the proposed algorithm was successfully applied to divide the Poznań City space into smaller spatial units containing a balanced number of flats. Each unit demonstrates spatial cohesion, respects natural infrastructure barriers, and contains a similar amount of residents. Apart from the purely theoretical aspect, our method has significant practical potential in many areas using geographic information systems like marketing, delivery or urban palling [1]. The entire algorithm is implemented in Python as a single class and is intended for medium to large data sets. The source code is implemented in Python and MIT-licensed parts of the source code are available at `https://github.com/IQ-Solution-Poznan/polygon-segmentation-algorithm`.

2 An algorithm

The goal of the algorithm is to divide the space into smaller units denoted as $\hat{U}$. Three free parameters control the division:

- expected size of the units, denoted as ω; not limited to the area of units, also expressed in any of the unit parameters: for example number of flats included in one address;
- allowable deviation from ω, denoted as ϵ, responsible for the balance;
- size of the initial neighborhood, denoted as δ, necessary for designate seed candidates (C') for each forthcoming $\hat{U}$, responsible for initial conditions of the algorithm.

Data preparation is the first step before the main algorithm starts. The hard borders must cut the space into subspaces, and each point must be assigned to one and only one subspace, by the spatial relationships *within*. For each subspace, a between-points dense distance matrix M with a selected metric must be calculated. The ability to store the M in the memory is an upper limit of the number of points.

The data pre-processing is not a part of the proper algorithm and will not be discussed here. The algorithm (Figure 2) starts by dividing each of the subspaces into elementary units. Such an element is the natural neighbourhood of the individual point or Voronoi polygon (*vp*). All *vp*s perfectly fill the extend of the subspace creating irregular gird. In the next step, topological relations are built between the *vp*s, so each *vp* knows all its neighbours. In regular raster grids, the topology can be efficiently established based on cell indices. Unfortunately, such a property does not exist in irregular vector grids, but topological relationships are necessary to merge *vp* into larger units in reasonable computational time.

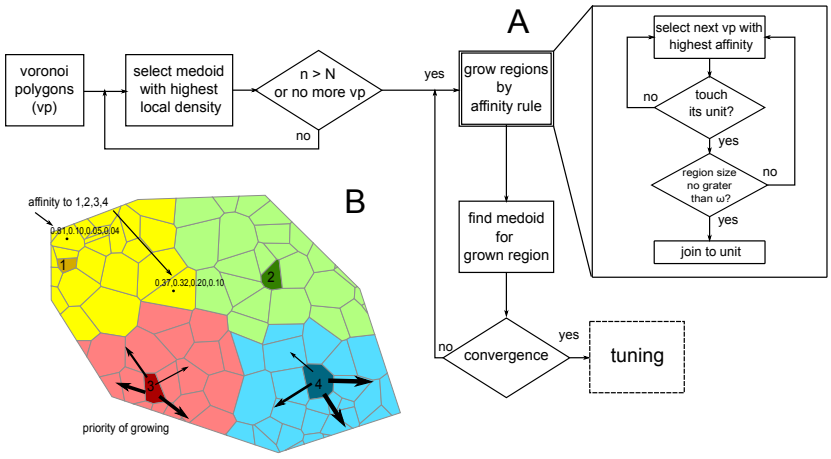

Figure 2 The schematic graph of the algorithm (A) and the scheme of region growing controlled by affinity between *vp* and C' (B)

.

The simple feature spatial model does not support topology, so the information about the neighbourhood is extracted internally from the topological data model provided by to *topoJSON* data structure. The algorithm uses a simplified topology, where each *vp* store a set of IDs of its *vp* neighbours. The expected ω determines the number of target units N. The ω can be described by any numerical parameter that attributes the *vp*. It could be its area or any other parameter, for example, the number of flats assigned to the given city address or the amount of biomass assigned to the given plant.

When N s is known, the segmentation process starts with determining the C' of the units. On the contrary to most clustering algorithms in our approach, this is a deterministic process, and the location of each seed C' is resolved upon the density analysis. Each C' is an existing *vp*. Density analysis requires the presence of M. First the mean distance $\bar{d}$ to other *vp*s is calculated for each *vp*s δ. The δ is not determined by the cardinality but is expressed in relation to ω. This relation is a free parameter, but we recommend setting it at 1.5. It means that each δ is possibly larger than the expected ω. We use such a strategy to limit the influence of discontinuities in the distribution of points. Finally *vp* with lowest $\bar{d}$ is marked as the initial C' and all *vp*s belonging to its δ are removed from the list of possible C's. Next, the process continues with consecutive *vp* till the number of C' reaches N, or there are no more available candidates.

All *vp* not marked as C' are scored in terms of the affinity against each C', so that the affinities for each *vp* must be a norm. Such an approach allows assessing which C's are best to adjoin given *vp*. In classic clustering, the marginal *vp*, distant to all C' would be difficult to join anywhere without breaking other, especially ω, constrains. All *vp*s, starting from most affine, are joined sequentially to its most affinal C'. The normality of the affinity guarantees that marginal $\hat{\text{U}}$ grow primarily towards the regions with the least amount of C', in practice towards the edges. After each joining, the difference in ω of all growing $\hat{\text{U}}$ is tested against the ϵ. If differences between $\hat{\text{U}}$ exceeds assumed ϵ, the growing of the largest $\hat{\text{U}}$ is holding until the balance is restored. The process continues till all *vp* are joined to the $\hat{\text{U}}$.

In such algorithms [6, 9], the initial C' is not a necessary best choice. After growing is finished in all $\hat{\text{U}}$ a new C' is selected based on medoid rule – the minimal mean distance to all *vp* in the unit ($\bar{d}'$). After assigning a new C' the process described in the previous paragraph repeats iteratively till the convergence, namely that in a given or next iteration, the assignment of C' will not change.

There is an optional tuning part of the algorithm, valuable when the point pattern shows the presence of natural micro-clusters, significantly smaller than the expected size of the units. In this step, all micro-clusters are checked if they are not divided between two different $\hat{\text{U}}$. If so, all *vp*s belonging to such micro-cluster are moved in the least expensive way for assumed ω.

3 Case study

To illustrate how the presented algorithm works, we created a spatial database for the city of Poznań containing information on roads, railway lines and main rivers. Data were supplemented with a collection of address points downloaded from `http://www.gugik.gov.pl/pzgik`. Each point was attributed with the number of flats assigned to a given address. The entire space inside the City limits was divided into subspaces by cutting with impassable barriers and main communication obstacles. The road network downloaded from Open Street Map, and Open Source Routing Machine was used to compute dense distance matrices for each sub-space, where the average round trip time $((A \to B) + (B \to A))/2$ was used as a metric. A spatially and geographically consistent units at the same time with the balanced size of units were adopted as an objective function.

We noticed during the experiments that a perfect balance between spatial cohesion and the size of units represented by the number of flats is complicated to achieve. In fact, the trade-off between the spatial consistency and size is controlled by the ϵ parameter. The higher ϵ the balance is shifted towards the cohesion. The lower values of ϵ shift the balance toward the size at the expense of spatial functionality. In the example shown in Figure 3 we

set the $\omega = 3000$ and $\epsilon = 3$, allowing to create units of various size but compact in shape. It means that in this example, the size of the expected units varies with the number of flats between approximately 2000 and 6000.

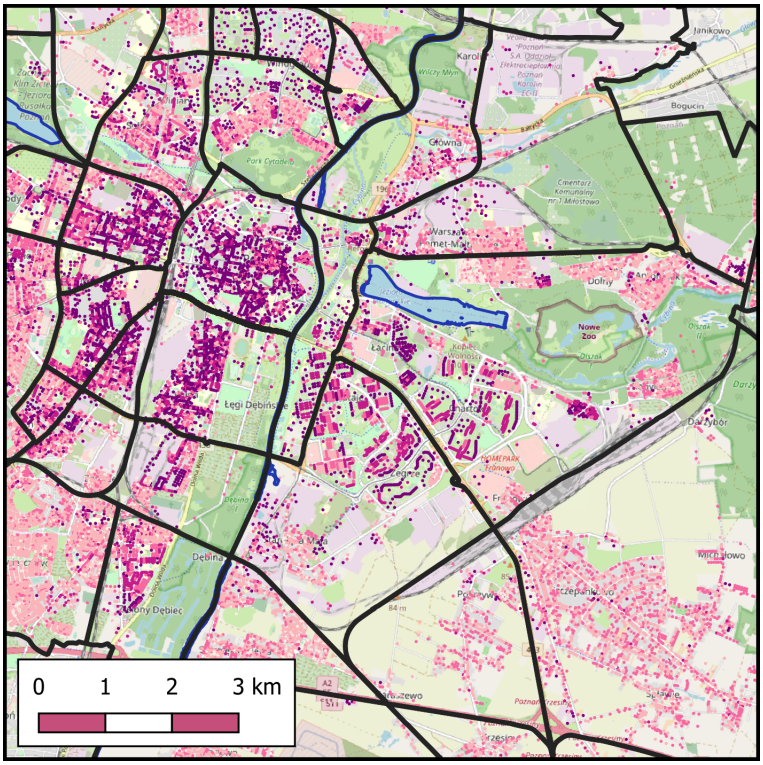

Figure 3 The case study example. Results of he segmentation over the Open Street Map and points distribution. See legend in Figure 4

.

4 Discussion and conclusions

In this short paper, we present the algorithm designed for the segmentation of large point data sets embedded in the geographical space based on priciples of AUD. Although the algorithm is at the primary stage of development, it allows segmenting the geographical space taking into account the geographic constraints. Apart from the theoretical assumptions, the further development of the algorithm is also of great practical importance, wherever the effective segmentation of geographical space translates into economic activity. Examples include marketing, logistics, emergency management or public health.

References

1 Samantha Cockings. *Automated zone design for the spatial representation of population.* PhD thesis, University of Southampton, March 2013. URL: `https://eprints.soton.ac.uk/368175/`.

2 John D Dockerty, Katrina J Sharples, and Barry Borman. An assessment of spatial clustering of leukaemias and lymphomas among young people in new zealand. *Journal of Epidemiology & Community Health*, 53(3):154–158, 1999.

3 Morteza Ghahremani, Kevin Williams, Fiona M. K. Corke, Bernard Tiddeman, Yonghuai Liu, and John H. Doonan. Deep segmentation of point clouds of wheat. *Frontiers in Plant Science*, 12:429, 2021. doi:10.3389/fpls.2021.608732.

4 Jarosław Jasiewicz, Tomasz Stepinski, and Jacek Niesterowicz. Multi-scale segmentation algorithm for pattern-based partitioning of large categorical rasters. *Computers & Geosciences*, 118(September):122–130, sep 2018. doi:10.1016/j.cageo.2018.06.003.

5 M Joseph and R Moeckel. Automated design of gradual zone sy stems. *Open Geospatial Data, Software and Standards*, 2017.

6 Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

7 David Martin. Extending the automated zoning procedure to reconcile incompatible zoning systems. *International Journal of Geographical Information Science*, 17(2):181–196, 2003.

8 Stan Openshaw. A geographical solution to scale and aggregation problems in region-building, partitioning and spatial modelling. *Transactions of the institute of british geographers*, pages 459–472, 1977.

9 Erich Schubert and Peter J Rousseeuw. Faster k-medoids clustering: improving the pam, clara, and clarans algorithms. In *International conference on similarity search and applications*, pages 171–187. Springer, 2019.

10 Frank Y. Shih and Shouxian Cheng. Automatic seeded region growing for color image segmentation. *Image and Vision Computing*, 23(10):877–886, 2005. doi:10.1016/j.imavis.2005.05.015.

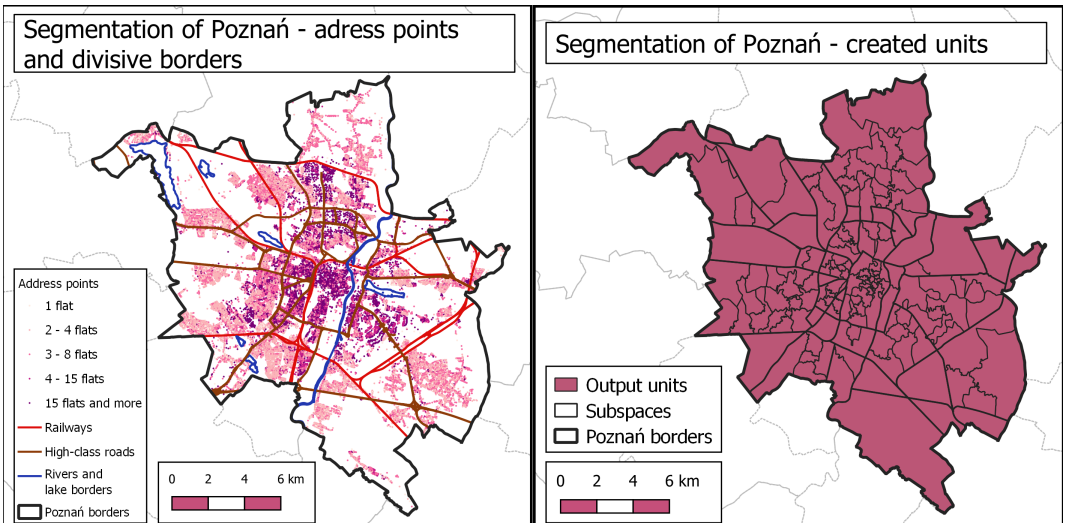

Figure 4 The case study example. The left plate show the data used in the segmentation process. Units were grown with $\omega = 3000$ flats. The Right plate show results. Note that areas of units varies

.