Huerta, Schade, Granell (Eds): Connecting a Digital Europe through Location and Place. Proceedings of the AGILE'2014 International Conference on Geographic Information Science, Castellón, June, 3-6, 2014. ISBN: 978-90-816960-4-3

An algorithm for segmenting a feature set into equitable regions

Md. Imran Hossain
University of the Bundeswehr Munich
Institute for Applied Computer Science
Werner-Heisenberg-Weg 39
85577 Neubiberg, Germany
Imran.Hossain@unibw.de

Wolfgang Reinhardt
University of the Bundeswehr Munich
Institute for Applied Computer Science
Werner-Heisenberg-Weg 39
85577 Neubiberg, Germany
Wolfgang.Reinhardt@unibw.de

Abstract

A set of geographic features of the same class representing a geographic area is often required to be divided in to several subsets/regions so that the sum of a numeric attribute of the features in each subset/region remains almost equal and the bounding polygon of regions do not overlap with each other. This kind of non-overlapping regions formation with similar collective feature value is of great importance especially in the field of optimization and spatial decision support. The paper presents a novel algorithm to solve the above mentioned spatial analysis work. The algorithm is further implemented, tested and the results are discussed.

Keywords: Spatial algorithm, Spatial Analysis, Vector Segmentation, Equitable Region

1 Introduction

Segmentation of a feature set of the same class into several equitable and non-overlapping[1] regions depending on a feature property is often required especially in the domain of optimization and spatial decision support. For example, an evacuation assistance providing authority having 4 emergency evacuation units (vehicles) might be interested to divide the whole emergency area into 4 parts in a way that each part consists of approximately the same number of evacuees and the bounding polygon of the parts do not overlap so that the evacuation process ensure optimization. Another practical need of such segmentation could be apprehended with the scenario that a service provider wants to cover an area (neighbourhood) with service centres of same capability. Let us assume that with the limited resources the service provider can provide only 4 service centres to cover the neighbourhood of 205 peoples. In such a case the service provider will be interested to segment the whole area into 4 regions so that each region consists of approximately the same number of people (in this case ~ 51) so that the service centres operates in an optimized way. Beyond the mentioned examples, a number of other different application areas could be found where the need of such a segmentation of a feature set is inevitable. A more precise definition of the problem that we address in this paper is given below.

A geographic area G defined by a feature set consisting of n number of features with a numeric attribute A has to be completely divided into N ($N \in \mathbb{N} \mid 2 \leq N \leq n$) number of subsets/regions based on following 3 criteria.

Criteria 1: Each region should consist of a certain number of complete features of the feature set. A splitting of a feature is not allowed.

Criteria 2: The sum of the value of A of all features in any region R_i must be equal to $T \pm d$ [where T is calculated by summing up the values of A of all n features of the geographic area G and then divided by N and d is a deviance]. Maximum value of d is equal to the maximum value of A of any given feature within G. The deviance d has to be considered as a splitting of the features is not allowed. Besides, as it may not possible in all cases of given data sets that the value of the sum of A of all regions is within $T \pm d$, the number of regions not following the criteria has to be minimized.

Criteria 3: The bounding polygon of any region should not overlap with any other region means it can only touch other or/and remain as disjoint.

The main goal of this paper is to present a novel algorithm (section 3 for more detail) to solve the problem stated herein. The authors developed the algorithm and successfully implemented it with c# and ArcObjects library. Implementation of the algorithm and the results of its application are discussed in section 4.

2 Related works

Automated zone design (AZD) or regionalisation is a technique for which Shortt [2] has given the overview of its concept, terminology and methods. AZD is an umbrella term for quite a number of approaches to create zones from a set of basic blocks following given criteria. Among the automated zone design algorithms automated zone design procedure (AZP) is the most popular and widely used one. It was introduced by Openshaw [6, 7]. The AZP has been enhanced by Openshaw and Rao [8], Alvanides [4] and Alvanides et al [3]. Cockings et. al. [5] used automated zone design techniques to dynamically maintain existing zoning systems. There are also a lot of other application of AZP algorithm such as climate zoning, location optimization and many more. The AZP algorithm iteratively combinines and recombines sets of blocks in order to create output zones which are optimised based on a set of pre-specified design criteria [8].

AZP is not applicable to our task described in the introduction as firstly, AZP is applicable only to continuous and connected feature sets whereas in our case continuous and

[1] Non-overlapping regions means the boundaries of the regions are disjoint or/and in touch with each other.

discrete feature sets must be treated. Secondly in AZP a zone can exist in a disconnected multi-polygon form which means a zone's bounding polygon may intersect with other zone's bounding polygon which is prohibited in our case. Also it is required in our approach that the bounding polygon of each region must not overlap with any other region.

The territory design tool of ESRI [1] offers functionality to create, automatically balance, and maintain territories. The tool establishes potential franchise areas and assigns sales territories consisting of multiple variables and levels. Again, the territory design tool works on continuous and connected feature sets. Manual intervention is often required to make all territory balanced. In contrast to the territory design tool, our goal is to balance the regions (territory in territory design tool) automatically and to cover discrete feature sets as mentioned earlier.

3 The algorithm

Firstly, the input, output and criteria of the algorithm are defined hereafter.

Input:
- Geographic area G = {f_n | f_n ∈ F (set of features), f_n has a numeric attribute A}
- N ($N \in \mathbb{N} \mid 2 \leq N \leq n$) = number of required subsets of G, N has to be defined by the user.

Output:
- N number of subsets R_n (subsets/regions)

Criteria:
- Region cannot be formed with splitted feature means a feature of a region is not allowed to be in a form like $f_i/m \mid m \in \mathbb{N}$.
- The sum of $|A|$ ($|A|$ is the value of attribute A) of any region defined herein with $O(R_i) = T \pm d$
- The bounding polygon of any subset $BNDline(R_i)$ do not overlap with the bounding polygon of any other.

The value of T is calculated by equation 1and the value of d is an element of set D. The value of d can ranges from 0 to the maximum value of $|A|$ of a given feature set G (equation 2).

$$T = \frac{\sum_{f=1}^{fn} |A|(G)}{N} \ldots \qquad (1)$$

$$d \in \mathrm{D} = \{\mathrm{q} \in \mathbb{Q} \mid 0 \leq q < MAX\ (|A|(G))\ \}\ldots \qquad (2)$$

In general, the algorithm prioritizes forming regions along the bounding line $BNDline(G)$ of the input feature set G. This approach prevents features being unclassified and also prevent big differences among the regions. A region R_i is formed by grouping features around the bounding line until $O(R_i) = T \pm d$. Once no region formation is possible along the $BNDline(G)$, another bounding line is created for the features which are not classified into regions and regions are again formed along the new bounding line. This process continues until N-1 regions are formed. The N^{th} region is formed with remaining unclassified features after formation of N-1^{th} region and consequently it´s possible that the sum of $|A|$ may not be within $T \pm d$ in this case. The algorithm is described in more detail through the following steps.

Step1. Objective function: Objective function returns the value of T on which each region is formed. The algorithm of the objective function is given by figure 1.

Figure1: Objective function algorithm

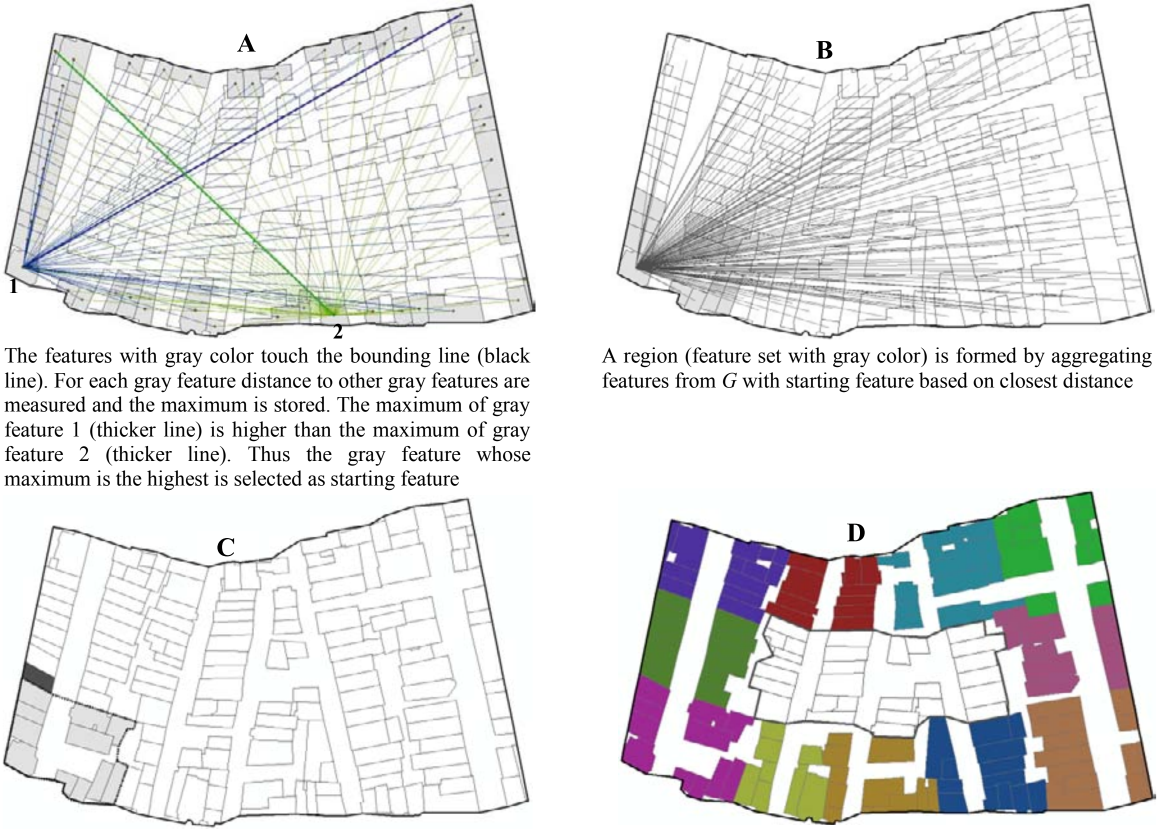
```
Data: Input FeatureSet G, Attribute A and No. of expected region N
Result: A double value representing T
FeatureSetTotal ← 0
for each Feature ∈ FeatureSet do
    FeatureSetTotal = FeatureSetTotal + |A|
return FeatureSetTotal/N
```

Step2. Selection of the starting feature for the first region: The starting feature for the first region is selected by two criteria. Firstly, it has to be along the $BNDline(G)$ and secondly it has to be located in an appropriate corner of G. Therefore the starting feature is selected by firstly making an array of features that touches the $BNDline(G)$. Secondly a feature is picked up from that array and distances are calculated from that feature to all other features of that array. The maximum distance is then stored with each picked up feature. This process is carried out for all features in the array. Finally, the feature that has the maximum distance value compared to other features in the array is selected as a starting feature for the first region formation (fig. 3A). The algorithm of this step is given in the following figure.

Figure 2: Starting feature selection algorithm

```
Data: Input FeatureSet G
Result: Feature f
    /* the function for getting the bounding line of a feature set
Function BoundLine (FeatureSet)
    return BoundLine
/* applying the function to the input feature set
BNDline ← BoundLine (G)
/*declaring an empty feature array
AlongBNDlineFeatureArray ← ∅
/* adding feature to the array that intersects the bounding line
for each Feature ∈ G do
    if Feature ∩ BNDline do
        Add Feature to the AlongBNDlineFeatureArray
/*find the suitable corner feature
for each Feature1 ∈ AlongBNDlineFeatureArray do
    /* an array for holding the distance between a feature and all
    /*other features of the AlongBNDlineFeatureArray
    FeatureDiastanceArray ← ∅
    /*calculating distances between the a feature and all other
    /* features of the AlongBNDlineFeatureArray
    for each Feature2 ∈ AlongBNDlineFeatureArray do
        Calculate distance between Feature1 and Feature2
        Add the distance in the FeatureDiastanceArray
    /*finding the maximum distance for each feature by sorting
    /*the array descending and get the first element
    sort descending FeatureDiastanceArray (distance)
    MAXVvalue ← first element of FeatureDiastanceArray
    Tag the MAXValue to Feature1
/*find the feature by sorting AlongBNDlineFeatureArray
/*discending with regards to the tagged value and get the first
/* feature of the array
sort descending AlongBNDlineFeatureArray (Tagged value)
return first element of AlongBNDlineFeatureArray
```

Figure 3: Visual illustration of different steps of the algorithm



The features with gray color touch the bounding line (black line). For each gray feature distance to other gray features are measured and the maximum is stored. The maximum of gray feature 1 (thicker line) is higher than the maximum of gray feature 2 (thicker line). Thus the gray feature whose maximum is the highest is selected as starting feature

A region (feature set with gray color) is formed by aggregating features from *G* with starting feature based on closest distance

A feature (dark gray) from *G* is selected as a start feature for subsequent region building if it is closest to the bounding line of (thick dotted line) previously formed region and touches the bounding line of the input feature set *G*

All the features along the bounding line (outer black line) of *G* are classified into regions. A second bounding line (inner black line) is formed for the unclassified features.

Step3. Formation of the first region: At the beginning the first region R_1 is formed only with the starting feature. Then the region is grown by grouping features from *G* on the basis of minimum distance, which means a feature from *G* is allowed to be grouped with the starting feature if the distance between them is a minimum compared to the distance of other features in *G* (fig. 3B). This grouping or region building is continued until $O(R_1) = T \pm d$ criteria is fulfilled. Since a feature in *G* is not allowed to divide according to the underlying data model, it is only possible to completely include or exclude a feature to a region. Which means the feature cannot be sliced. So, $O(R_1)$ cannot always be exactly equal to *T*. The maximum possible deviation of $O(R_1)$ with *T* for R_1 to R_{N-1} will be thus the maximum value of $|A|$ of any given *G*.

Once a region R_i is formed, a static variable *StatN* is updated with the number of region formed and the feature set on which the process will be continued is obtained by $G - R_i$. The process terminates and goes out of scope when $N\text{-}1 = StatN$. For example, if 3 regions are expected and 2 regions have already been completed then remaining features of *G* automatically form a region and the process goes out of scope. The figure 4 presents the algorithm of Step3.

Figure 4: Algorithm for first region formation

```
Data: Starting feature
Result: FeatureSet (Region), updated $G$
/* static variable for keeping track of the regions
RegionFormed = 0
/*an array of features representing a region
RegionFeatureArray ← ∅
Add starting feature to RegionFeatureArray
/*declaring an empty feature array
FeatureArrayWithDistance ← ∅
/* adding feature to the array with distance to start feature
for each Feature ∈ G do
    Calculate Distance between Feature and Starting Feature
    Tag Distance to Feature
    Add Feature with distance to FeatureArrayWithDistance
/*sorting the array of feature with regard to the distance
sort ascending FeatureArrayWithDistance (distance)
/*adding closest features to the region array until region's
/* $O(Ri) = T \pm d$
for each Feature ∈ FeatureArrayWithDistance do
    if $\sum|A|$ of RegionFeatureArray < $T$ do
        Add Feature to RegionFeatureArray
    else do
        finalize RegionFeatureArray
        RegionFormed = RegionFormed +1
/*returning the region and updated $G$
return RegionFeatureArray
return $G \leftarrow G$ - RegionFeatureArray
```

Step4. Start feature selection for subsequent regions: As stated earlier, the algorithm prioritizes forming regions along the bounding line *BNDline(G)* of the input feature set *G*. Therefore, a start feature for any subsequent region R_{i+1} should be located next to the former region R_i and also should touch the *BNDline(G)* (fig. 3C). These are two simple criteria for selecting a start feature for any subsequent region building. The algorithm of this step is given in next page with figure 5.

Figure 5: Algorithm for start feature for subsequent regions

```
Data: Feature Set R_i (last region), updated G
Result: Feature f
/* the function for getting the bounding line of a feature set
Function BoundLine (FeatureSet)
    return BoundLine
/* applying the function to the input region feature set g_i
BNDline_Ri ← BoundLine (Ri)
/* applying the function to the feature set G
BNDline_G ← BoundLine (G)
/*declaring an empty feature array
FeatureSet_near_ BNDline_Ri ← ∅
/* adding feature from G to the array that are with a certain
/* distance (5m) from the BNDline_Ri
for each Feature ∈ G do
    if Feature is within 5m of BNDline_Ri do
        Calculate distance from Feature to BNDline_Ri
        Tag distance to Feature
        Add Feature to the FeatureSet_near_ BNDline_Ri
/*sort FeatureSet_near_ BNDline_Ri in ascending way so that
/* the feature with shortest distance comes first in a loop
sort ascending FeatureSet_near_ BNDline_R_i(distance)
/*find the start feature
for each Feature ∈ FeatureSet_near_ BNDline_R_i do
    if Feature touches BNDline_G do
        return Feature
```

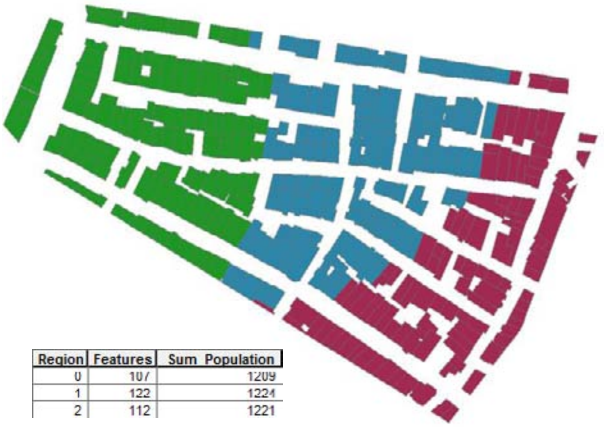
Step5. Repetition: Step 3 to 4 are repeated until no start feature is returned by step 4 and the required number of regions is still not achieved. A null feature return by step 4 means all the features along the bounding line of *G* are classified into regions. If this is the case, a new bounding line is created for the set of non-classified features (fig. 3D). The *BNDline(G)* which is created in step2 is replaced by the new bounding line and the process starts continuing from step 2.

4 Implementation, application and results

The algorithm we presented in section 3 has been implemented using c# programming language and ArcObjects library of ESRI. Figure 6 and 7 shows the result of 2 examples of an application of the implemented algorithm. Each feature (polygon) in both figures represents residential buildings and has an attribute called population (no. of residents). The maximum value of *d* of the input feature set was 21.

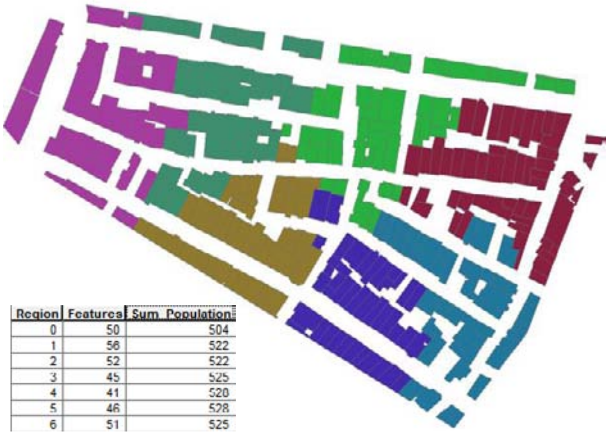
In figure 6, the expected number of equitable regions was 3 based on the population attribute which means the feature set has to be divided into 3 non-overlapping regions so that the total population for each region remains approximately equal. In figure 7 the expected region number was 7. Both figures show a distinct division of the feature set into regions. None of the region in both figure overlap with others. However, the shape of the regions gets more irregular with the increase number of regions (fig.7). The important point to be noted here is that region no.0 in both figures differs significantly from other regions in terms of total population and the difference goes beyond the *MAX(d)* in figure 7. The differences among other regions are minimal and within *MAX(d)*.

Figure 6: Input feature set *G* divided into 3 equitable regions



Region	Features	Sum Population
0	107	1209
1	122	1224
2	112	1221

Figure 7: Input feature set *G* divided into 7 equitable regions



Region	Features	Sum Population
0	50	504
1	56	522
2	52	522
3	45	525
4	41	520
5	46	528
6	51	525

Region 0 is in fact the last region formed with the remaining feature set once *N-1* regions are formed. If the other regions formed with a positive value of *d* (section 3, step 3) then the effect goes on to the last N^{th} region (region 0) which is forced to be formed with a total value deduced by the cumulative positive *d* of the former regions. Thus only the N^{th} region's $O(R_n)$ may not be equal to $T \pm d$. The maximum difference between the last region's $O(R_n)$ with other regions $O(R_i)$ is thus expected to be higher with the increased no. of regions. However, this problem can be solved with a constraint that two consecutive regions should form with $+d$ and $–d$ simultaneously which restricts region formation with always $+d$ or $–d$.

Another important point to be noted here that theoretically, a feature set may contain several features which are suitable as starting feature for the formation of first region (step 2, section 3). Selection of each of those suitable features as a start feature would results a different form of the output regions in terms of region's $O(R_i)$ and shape. The algorithm restricts different output possibilities by automatically selecting the best starting feature. But the algorithm could be adopted for allowing the selection of alternative suitable features for obtaining variations in output regions.

5 Conclusion and future works

The paper presented an algorithm for segmenting a feature set into multiple equitable non-overlapping regions and its implementation and the result of its application are discussed. As a first attempt the algorithm has been developed and implemented to deal with polygon and point features set. Due to the limited space examples of point feature sets are not discussed in this paper. Several tests have proven its applicability. The applicability could be extended to polyline features set in future with limited effort. At present the algorithm is dealing with a single attribute and regions are formed based on a value which is approximately equal for each region. As a future work the algorithm could be extended to deal with multiple attributes and other statistical parameters e.g. regions could be formed based on equal standard deviation of one or multiple attributes. The algorithm could be enriched by introducing constraints e.g. region formation can be restricted to cross certain types of roads and other geographical features. Moreover, spatial indexing could be applied to improve computing time for large input datasets.

References

[1] ESRI. White paper on territory design. Redlands, USA, 2010.

[2] N. Shortt. Regionalization/zoning systems. In R. Kitchin and N. Thrift, editors, *International Encyclopedia of Human Geography*, pages 298--301. Elsevier, Oxford, 2009.

[3] S. Alvanides, S. Openshaw and P. Rees. Designing your own geographies. In P. Rees, D. Martin and P. Williamson, editors, *The Census Data System*, pages 47—65. JohnWiley, Chichester, Sussex, 2002.

[4] S. Alvanides. Zone Design Methods for Application in Human Geography. *PhD thesis*, School of Geography, University of Leeds, 2000.

[5] S. Cockings, A. Harfoot, D. Martin and D. Hornby. Maintaining existing zoning systems using automated zone-design techniques: methods for creating the 2011 Census output geographies for England and Wales. *Journal of Environment and Planning*, A 2011, volume 43: 2399 - 2418, 2011.

[6] S. Openshaw. *A geographical solution to scale and aggregation problems in region-building, partitioning and spatial modeling*. Transactions of the Institute of British Geographers, New Series 2, 1977.

[7] S. Openshaw. Algorithm 3: a procedure to generate pseudo-random aggregations of N zones into M zones, where M is less than N. *Journal of Environment and Planning*, A 9: 1423-1428, 1977.

[8] S. Openshaw and L. Rao. Algorithms for re-engineering 1991 Census geography. *Journal of Environment and Planning*, A 27: 425 - 446, 1995.