2009

Redistricting using Heuristic-Based Polygonal Clustering

Deepti Joshi
University of Nebraska-Lincoln, djoshi@cse.unl.edu

Leen-Kiat Soh
University of Nebraska-Lincoln, lsoh2@unl.edu

Ashok K. Samal
University of Nebraska-Lincoln, asamal1@unl.edu

2009 Ninth IEEE International Conference on Data Mining

Redistricting using Heuristic-Based Polygonal Clustering

Deepti Joshi, Leen-Kiat Soh and Ashok Samal
Department of Computer Science and Engineering
University of Nebraska
Lincoln, NE 68588-0115 USA
e-mail: {djoshi, lksoh, samal}@cse.unl.edu

Abstract— **Redistricting is the process of dividing a geographic area into districts or zones. This process has been considered in the past as a problem that is computationally too complex for an automated system to be developed that can produce unbiased plans. In this paper we present a novel method for redistricting a geographic area using a heuristic-based approach for polygonal spatial clustering. While clustering geospatial polygons several complex issues need to be addressed – such as: removing order dependency, clustering all polygons assuming no outliers, and strategically utilizing domain knowledge to guide the clustering process. In order to address these special needs, we have developed the Constrained Polygonal Spatial Clustering (CPSC) algorithm that holistically integrates domain knowledge in the form of cluster-level and instance-level constraints and uses heuristic functions to grow clusters. In order to illustrate the usefulness of our algorithm we have applied it to the problem of formation of unbiased congressional districts. Furthermore, we compare and contrast our algorithm with two other approaches proposed in the literature for redistricting, namely – graph partitioning and simulated annealing.**

Keywords-redistricitng, spatial data mining, polygon, polygonal clustering, district formation.

I. INTRODUCTION

Redistricting is the process of dividing a larger geographic space into smaller regions or districts or zones. In other words, redistricting can be viewed as a set partitioning problem, i.e. the problem is to partition the entire set of units into districts such that a value function is maximized [1]. Each district thus formed is a group of polygons that is spatially contiguous. Moreover, as almost always these districts or zones are formed for some form of jurisdiction, the districts formed should be compact in space in order to facilitate the application of laws and regulations within the district. A compact district can be defined in many ways. It is most commonly measured as an attribute of the shape of the district. A circle is the most compact shape for any district because it covers the most area within the smallest perimeter [3]. We define a compact district as a district that has a shape very close to that of a simple geometric shape and does not meander in space forming a snake-, river-like, or extremely-elongated structure. Examples of simple geometric shapes are a circle, a rectangle, and a square.

Since each district is formed by combining together several small regions or polygons, each can be viewed as a cluster of geospatial polygons. Spatial clustering is the process of grouping together objects with spatial coordinates into groups such that the objects within a cluster are more similar to each other as compared to outside objects. It has recently gained a lot of popularity due to the tremendous amount of data being collected by thousands of satellites, and millions of geographic positioning systems and other sensors.

In spatial clustering, the spatial data is generally organized in the form of a set of points or polygons. Several spatial clustering algorithms have been proposed in the past decade [6]. Most of these algorithms focus on clustering point data. However when the question arises of clustering polygons instead of points, these algorithms fall short of giving accurate results [8]. The main cause of this inadequacy of the current spatial clustering algorithms is that points are relatively simpler geographic objects as compared to polygons. Polygons, especially geospatial polygons, share a spatial and topological relationship that does not exist in the point datasets. For example, two polygons may share boundaries with each other, overlap each other or one may even lie within another polygon.

Furthermore, while clustering is a form of unsupervised learning, making use of the available domain knowledge during the process of clustering can further enhance the quality of the clusters to a large extent. Constraints applied during the process of clustering can be of two types – instance-level constraints and cluster-level constraints. Instance-level constraints are applied to individual objects being clustered. Examples of instance-level constraints are must-link and cannot-link constraints [4]. Cluster-level constraints are applied to the cluster on the whole. Examples of cluster-level constraints are averaging or summation constraints [5].

There are several important applications where the application of constraint-based spatial polygonal clustering can potentially yield significant benefits. In these types of problems, every polygon in the dataset must be accurately assigned to its cluster — thus there will be no noise or outliers in the dataset. Furthermore, each cluster will represent a specific entity conforming to certain properties that would have been predefined in the process of clustering. An example of such applications is the formation of congressional districts where several census tracts are grouped together to form a district.

1550-4786/09 $26.00 © 2009 IEEE
DOI 10.1109/ICDM.2009.126

In this paper we present an algorithm for clustering polygons in the presence of certain constraints. We call our algorithm the Constraint-Based Polygonal Spatial Clustering (CPSC) algorithm. The uniqueness of this algorithm is that it makes use of the spatial and topological relationships between the polygons as well as the domain knowledge present in the form of constraints. The algorithm is divided into three main steps: (1) select seed polygons, (2) determine the best cluster to grow, and (3) select the best polygon to be added to the best cluster. Several novel strategies of the algorithm include:

- *We make use of heuristic functions to apply the constraints during the clustering process*. A heuristic function has two components: (1) a distance function that measures the distance of the current state of the cluster from the desired goal, and (2) a cost function that measures the reduction in the flexibility of the growth of all the other clusters. The use of these heuristic functions makes the summation type cluster-level constraints easier to achieve.
- *Our process of seed selection is based on the inter-cluster and intra-cluster constraints of the domain.*

In order to validate our algorithm, we have applied CPSC—and compared the results with two other techniques in literature: graph partitioning [2] and simulated annealing [10]—to the *congressional redistricting problem*. Congressional redistricting has been inflicted with issues such as gerrymandering [7] and unequal population distribution for a long time. In our study, we find that our algorithm outperforms the other two algorithms by producing districts that have almost equal population and are spatially more compact.

The paper is organized as follows. In Section II we present an introduction to other redistricting algorithms. Section III presents the CPSC algorithm. Section IV describes the application domains and implementation of the CPSC algorithm in each domain. In Section V we show the results of the application of our CPSC algorithm in the application domain. Finally in Section VI we discuss our conclusions and present the directions for future work.

II. Related Work

Redistricting is essentially an optimization problem where the global optimum solution is difficult to find. This is because the size of the dataset can be enormous. A simple brute force search through all the possible solutions is impractical. As a result, the problem of redistricting has been considered to be impossible to solve precisely and efficiently [1]. Moreover, due to the size of the real datasets, most of the current techniques used for automated redistricting resort to unproven guesswork [1] and random selection, and are therefore highly inefficient.

Several meta-heuristic approaches have been proposed in the past to solve this problem. These meta-heuristic approaches are often based on genetic algorithms, simulated annealing, or graph partitioning techniques. In this section we give an overview of graph partitioning and simulated annealing redistricting methods highlighting their advantages and disadvantages.

A. Graph Partitioning

The problem of partitioning a geographic area into a collection of contiguous, approximately equal population districts can be viewed as a graph theory problem of partitioning a network into a fixed number of sub-sections such that the sum of the weights of the nodes within each sub-section is equal, and each sub-section is contiguous. Each sub-region or polygon in the area becomes a node in the network. Two nodes are connected if they share a boundary. The outline of the graph partitioning algorithm for redistricting proposed by [2] is as follows.

In Step 1 of the algorithm, the input to the algorithm is a directed graph representation of the geographic area and the number of partitions to be formed. Labels are assigned to each node in the graph. Each label has three components – the cluster number to which the node belongs, the weight of the cluster, and the predecessor of the node in the graph. The seeds or the gravity center of the clusters or sub-sections are selected randomly. Each seed is assigned to separate clusters. Subsequently a pass is made through all the nodes in the graph and each node is assigned to a cluster based on the weight of the cluster and the predecessor of the node.

Step 2 of the algorithm takes the clusters produced in Step 1 and improves them by exchanging polygons within clusters where required. When an exchange is made, or a polygon is shifted from one cluster to another, the spatial contiguity of both the clusters is checked, and it is ensured that the contiguity is not broken due to the move.

The advantages of this approach are that it is computationally fast, and presents the user with several potentially useful plans. However there are several disadvantages with this approach. First, it does not terminate at an optimum solution. That is, the procedure terminates at one of the many solutions that may be obtained. Second, there are no guidelines provided in this methodology to select the best plan. Also, this method does not work well when the number of clusters is large.

B. Simulated Annealing

Simulated annealing is a general purpose optimization procedure based upon the thermodynamic process of annealing of metals by slow cooling. In a redistricting problem where the goal is to draw a plan such that some constraints are met (e.g. each cluster should have equal population) simulated annealing can be easily applied. Simulated Annealing Redistricting Algorithm (SARA) [10] uses this approch. An outline of the algorithm is as follows:

Step 1: Select an over-populated region for the removal of a zone; Step 2: Choose the zone to be removed from the donor region; Step 3: If contiguity would be lost by this transfer, return to step 2; Step 4: Select a recipient region for the chosen zone from amongst the regions to which neighboring zones belong; Step 5: (a) if the transfer would decrease the combined population deviation of the donor and recipient regions then accept it; (b) if the transfer would increase the combined population deviation of the donor and recipient regions then accept it with a probability governed by the size of the deviation and the value of the temperature parameter; Step 6: If the transfer is accepted, calculate the

new regional population deviations and add one to the count of successful transfers; Step 7: If the aggregate regional population deviation is within the target range then stop; if the threshold numbers of successful and unsuccessful swaps have not been exceeded then go to Step 1; if the thresholds have been exceeded then reduce the value of the temperature parameter then go to Step 1.

While simulated annealing based methods perform much better than informal or manual methods, one of the major disadvantages of this approach is that an initial solution needs to be provided. Furthermore, more than one spatial constraint cannot be easily incorporated in algorithms such as SARA. Finally, there is no guarantee that a global optimum will be found.

III. Constraint-Based Polygonal Spatial Clustering (CPSC) Algorithm

Our constraint-based polygonal spatial clustering (CPSC) algorithm is used to cluster polygons that are spatially contiguous. CPSC depends heavily on the constraints that are applied to the cluster growth process. It begins with growing clusters from seed polygons. The seeds are selected systematically from the database based on the knowledge available about the target clusters. This knowledge about the target clusters and the domain is embedded in the clustering process in the form of constraints. Initially each seed represents a cluster and we grow the clusters from them. Towards this, we make use of the notions of *intra-cluster constraints*, and *inter-cluster constraints*. Intra-cluster constraint prescribes why two data points should be grouped into the same cluster while each inter-cluster constraint specifies the conditions for two data points to be grouped into separate clusters. When the initial cluster seeds are selected, they must violate all *intra-cluster constraints*, and abide by *inter-cluster constraints*.

Once the seeds are selected, the initial clusters come into existence. Assume that the initial clusters (consisting of the individual seeds) are the start state, and the target clusters are the goal state. Each cluster is then grown from the start state by adding polygons to the cluster one by one until the target state is achieved. To make our algorithm order independent, at the beginning of each iteration the best cluster is selected to be grown. The best cluster is selected using a heuristic function F, which is a combination of (1) a function that approximates the distance of the current state of the cluster to the goal state (H), and (2) a cost function that measures the reduction in flexibility on the growth of the clusters (G).

Upon the selection of the best cluster, i.e. the cluster with the largest F, the best polygon is selected that can be added to the cluster. To select the best polygon, first a set of candidate polygons is selected that may be added to the best cluster. These are the neighbors of the cluster that abide by all the inter- and intra-cluster constraints specified by the user. Furthermore, since the neighboring polygon could be a polygon that has recently been acquired by the neighboring cluster, we assign a move count with each polygon, and limit the count to two to prevent the polygon from oscillating between two clusters. Once the neighboring polygons are selected, the best cluster assigns a score to each of its neighbors. The score for each neighboring polygon is equal to the number of polygons within the cluster that share boundary with the neighboring polygon multiplied by the total boundary shared by between them divided by the area of the neighboring polygon. This score is based on the need for compact clusters, and determines which polygon is the closest to the polygon and will make the cluster more compact. It does so by assigning a higher score to a polygon that shares its boundary with two polygons within the cluster, as opposed to a polygon that only shares its boundary with one polygon within the cluster. If more than one polygon is assigned the same score, then a selection between them is made on the basis of the heuristic function F. Furthermore, none of these polygons must violate any of the intra-cluster constraints, and none of them must abide by the inter-cluster constraints. Once a polygon is selected to be added to the best cluster, it is necessary to check that by its addition to the best cluster the spatial contiguity of the cluster it is being removed from, and the best cluster is still maintained. If all clusters are spatially contiguous, the selected polygon is added to the best cluster. This process goes on until all the polygons have been assigned to a cluster, and the target state clusters are produced.

Figure 1 outlines the algorithm. The CPSC algorithm presented in Figure 1 can be applied to any domain given the dataset of polygons, the number of clusters, a set of constraints, and the heuristic function F based on the constraints. The algorithm does however inherently produce spatially contiguous and compact clusters.

The CPSC Algorithm

Input: Dataset D of n polygons, Set C of constraints, Heuristic function $F=G+H$, Number of seeds k

1. **Select *k* seeds** such that:
   a. The seed should be a polygon with a larger F than the other non-seed polygons.
   b. Each seed must violate all intra-cluster cluster constraints.
   c. Each seed must abide by the inter-cluster constraints.
2. **Initialize clusters**: each seed is assigned to a cluster
3. **Grow clusters** from seeds until all polygons have been assigned to a cluster and all clusters achieve their target state.
   a. ***Compute F*** for each cluster
   b. ***Select the best cluster***, i.e., the cluster with largest F
   c. ***Find candidate polygons*** that may be added to the best cluster as follows:
      i. Set of neighboring polygons such that:
         - None of the intra-cluster constraints are violated.
         - All the inter-cluster constraints are violated.
         - The polygon shares their boundary with at least one polygon in the cluster.
         - The polygon has been moved less than two times (this is to prevent oscillations)
      ii. If none such polygon exists, then all the neighboring polygons that share their boundary with

the cluster are added to the list of possible polygons, and their move count is set to 1.

d. ***Select the best polygon*** to be added to the best cluster as follows:
   i. The best cluster assigns a score for each possible polygon. The score for each polygon = (no. of polygons within the cluster that share boundary with the polygon × total boundary shared by the polygon and the cluster) ÷ area of the polygon.
      1. Find the maximum score assigned to any possible polygon.
      2. If there is only one polygon that has been assigned the maximum score, then it is the best polygon to be added to the cluster.
      3. Else, if more than one polygon have been assigned the maximum score, then find the best polygon as follows:
         - Compute the resulting *F*'s for the best cluster with the addition of all the polygons with the maximum score one at a time.
         - The best polygon will be the polygon whose addition to the best cluster will lead to the smallest *F* unless specified otherwise, among all the possible polygons.

e. ***For the best polygon*** selected:
   i. If it belongs to another cluster, check to make sure contiguity of that cluster is not broken upon its removal.
   ii. If the contiguity is not broken, add the best polygon to the best cluster.
   iii. Else, reject this polygon, and repeat the process of selecting the best polygon with the new list formed by the removal of this polygon.

Figure 1: The CPSC Algorithm

Note: In some application the best cluster may be the one with the smallest F value.

IV. Application to A Real-World Problem

In order to show the usefulness of our algorithm, we have applied CPSC algorithm to the problem of congressional redistricting. This problem can be interpreted as a problem of cluster formation where each cluster represents a district. Each district or cluster is formed by grouping together polygons which follow certain constraints. Given below are the details of the problems along with the constraints that must be used the clustering process.

The Congressional Redistricting Problem

The problem of congressional redistricting has been vexing our society for a long time. Once a state learns that it has been assigned *k* seats, it must divide its territory into *k* districts. The constraints that define a "good district" are as follows [3]: (1) All the districts within a state should be equal in population, (2) Each district should be a single continuous territory, (3) Districts should be compact; Tentacles wriggling through the landscape are considered a bad design, (4) Districts should recognize the exiting communities of interest, (5) Districts should conform to existing natural and political boundaries when possible, and (6) Finally, under the Voting Rights Act a district must not be drawn with the intent of excluding the minority candidates from election.

In case of any conflict among the above constraints, the highest priority is given to numerical equality and spatial contiguity. In our implementation we take into consideration only the first three constraints as they define the overall structure of the algorithm. Constraints 4, 5, and 6 are not incorporated due to lack of data. However, they can be applied as must-link and cannot-link constraints while selecting the possible set of polygons in step 3(c) of the algorithm. Therefore, the problem statement is to divide the geographic area of a state into *k* districts such that the total population within each district is within 1% margin of error, i.e. the difference in the actual population of the cluster and the target population. Each of these *k* districts must be spatially contiguous. Finally all of the *k* districts must be as compact as possible.

Heuristics Used. The heuristic function *F*, used by CPSC to determine 1) the best cluster to grow and 2) the best polygon to add to the best cluster, is defined based on the input dataset, and the constraints defined before the clustering process. For the congressional redistricting problem, the inputs to the algorithm are:

Dataset: US Census Tracts as the set of polygons

Number of seeds: *k*

Target: *k* spatially contiguous and compact clusters $\{C_1, C_2, \dots, C_k\}$ each containing the same population (*x*), with a margin of error of 1%.

Set of constraints:

Intra-Cluster Constraints: (CS1) Each cluster must be spatially contiguous, (CS2) Each cluster must be compact, (CS3) Each cluster must contain equal population, (CS4) Set of spatial constraints as a set of must-link constraints between the census tracts. *Inter-Cluster Constraints:* (CS5) Set of spatial constraints as a set of cannot-link constraints between the census tracts.

Based on the above inputs, we define the heuristic function *F* as follows:

$$F = G + H,$$

where $H = x - \textit{current population of the cluster}$, and *G* is the cost of the reduction in the flexibility of the growth of the cluster, where *x* is the total population of the state divided by *k* or the number of clusters.

While selecting the *best cluster* to grow, the cost of reduction in the flexibility of the growth of the cluster ($G_{cluster}$) is defined as:

$$G_{cluster} = \max_{i=0 \text{ to } k} \max_{j=0 \text{ to } n} \left[\frac{O(C_i) - O'(C_{i,j})}{O(C_i)}\right], \quad (1)$$

where *k* is the number of clusters, *n* is the number of polygons surrounding a cluster—i.e., neighbors—that have not yet been assigned to any cluster, $O(C_i)$ is the (outer) boundary of a cluster *i* (assuming all polygons within the cluster are contiguous) that is shared with polygons that are still not assigned to any cluster, and, $O'(C_{i,j})$ is the resulting new boundary of the cluster *i* after adding a new polygon *j*. With

this cost function, our objective is to select a cluster to be grown that will preserve the maximum degree of flexibility for the other clusters to grow. Thus, together with *H*, the best cluster is one with the highest value of *F*, meaning one that is (1) furthest away from the target population and/or the least compact, and (2) the costliest to grow (akin to the min-conflict algorithm in conventional constraint satisfaction problems). As alluded to earlier, we use the same rationale in designing the cost function for selecting the *best polygon* to add to the best cluster C_i:

$$G_{neighbor} = \sum_{i=0}^{k} \left[\frac{O(C_i) - O\prime(C_{i,j})}{O(C_i)}\right] \qquad (2)$$

To select the best polygon to add to a cluster, we select the neighbor that reduces the open boundary of the cluster the most, and takes the cluster closest to its target. Thus together with *H*, the best polygon will (1) increase the population of the cluster, (2) make it more compact, and (3) reduce the open boundary of the cluster. Note that while we use a maximum function in Eq. (1), we use a summation function in Eq. (2). This is because when a free polygon (i.e. a polygon not yet assigned to any cluster) is added to a cluster, this action may considerably hinder the growth of another cluster. Therefore, we include the cumulative effect of the addition of a polygon to a cluster.

Taken together, Eq. (1) allows us to pick the least costly cluster to grow, and Eq. (2) allows us to pick the least costly neighbor to add to that cluster. In our application here, selecting the seeds from the dataset reduces the problem to sorting them by *F = H = x – current population of the cluster* in descending order and selecting the top *k* polygons. Furthermore, since the seeds cannot be spatially contiguous we enforce a physical distance between them as follows: The physical distance between two seeds must be a function of *r and k* unless specified otherwise. That is, for seeds s_i and s_j the distance between them $dist(s_i, s_j) = f(r, k)$. where $r = \sqrt{((Area\ of\ MBR)/(k \times \pi))}$, $Area\ of\ MBR$ is the area of the enclosing minimum bounding rectangle of the dataset, and k is the number of seeds.

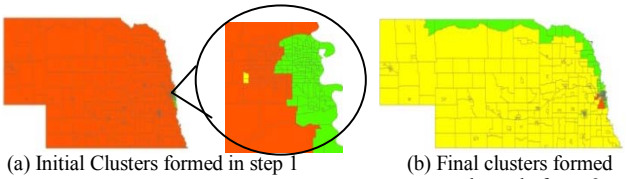
V. Experimental Evaluation

In this section we evaluate our algorithm by comparing the results obtained by applying the graph partitioning, the simulated annealing algorithm (SARA), and the CPSC algorithm on the datasets for the problems described in Section IV. Both the algorithms have to use clearly defined constraints, and the resulting clusters need to be spatially contiguous and compact.

Congressional Redistricting

For this experiment, we used the census tract dataset for the state of Nebraska. The total number of polygons (census tracts) in Nebraska is 505. The state of Nebraska has been assigned 3 seats in the congress. Therefore the number of clusters (*k*) is equal to 3. Based on the census data, the approximate population of each cluster or district must be equal to 570421.

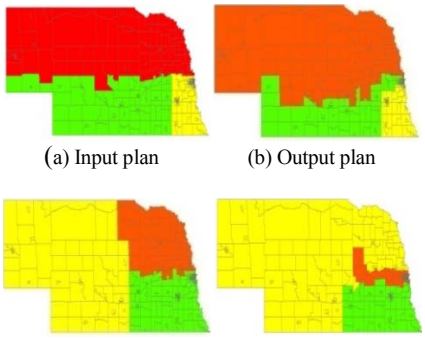
We first applied the graph partitioning algorithm presented in Section 2.1. Figure 2(a) shows the initial clusters formed based on a random run. Figure 2(b) shows the final output of the graph partitioning algorithm.



(a) Initial Clusters formed in step 1 (b) Final clusters formed at the end of step 2

Figure 2: Results of Graph Partitioning Algorithm

Next, we applied SARA (presented in Section 2.2) on the same dataset. A random input plan is shown in Figure 3(a). Figure 3(b) shows the final output obtained at the end of the execution of SARA. A visual inspection of the input and the output plans show that the output plan is fairly dependent on the input plan. Furthermore, the algorithm does not promote the formation of compact districts. Figure 3(c) shows another input plan, and the corresponding input plan is presented in Figure 3(d). The same phenomenon is observed once again.



(a) Input plan (b) Output plan

(c) Input plan (d) Output plan

Figure 3: Results of Simulated Annealing

We then applied the CPSC algorithm on the Nebraska census tract dataset and obtained the three clusters shown in Figure 4. A visual inspection of the districts shows that the districts formed are spatially contiguous and compact. Finally, we also compare the current congressional districts as in Figure 5 with the districts formed using the above three algorithms.

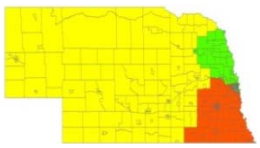


Figure 4: Results of CPSC

Figure 5: Current Congressional Redistricting Plan

Table I lists the total population of each cluster formed using all the methods listed before. For each of the district formed using all the methods listed above the margin of error (that is, the difference in the actual population of the cluster and the required population) is computed and listed in Table II. We can see that CPSC produces clusters/districts that fit the input criteria the best. In the experiment described above CPSC produces clusters that are spatially contiguous, compact, and conform to the other constraints presented to the

algorithm as inputs. The main reason behind CPSC's excellent performance is the use of heuristic function in seed selection, and in deciding which cluster to grow and which polygon to add to the selected cluster. This feature of parallel growth of all the clusters and unbiased selection of polygons is the novelty of CPSC and makes it better than other redistricting algorithms. Other than this the holistic integration of constraints and the use of inter-cluster and intra-cluster constraints make the resulting clusters a lot closer to the desired target. Finally, another unique feature of CPSC is the use of the cost function as a part of the heuristic that measures the reduction in flexibility of clustering with every assignment of a polygon to a cluster.

TABLE I. POPULATION OF CLUSTERS

	Population			
	Graph Partitioning	***Simulated Annealing***	***CPSC***	***Current Districts***
1	382,209	529,243	569,662	569,018
2	75,227	589,206	570,100	574,845
3	1,253,827	592,814	571,501	567,451
Stdev	611,426	35,706	960	3,896

TABLE II. MARGIN OF ERROR OF CLUSTERS

	Margin of Error			
	Graph Partitioning	***Simulated Annealing***	***CPSC***	***Current Districts***
1	-32.99%	-7.22%	-0.13%	-0.25%
2	-86.81%	3.29%	0.05%	0.78%
3	119.81%	3.93%	0.19%	-0.52%
Stdev	107.189	0.063	0.002	0.007

VI. CONCLUSION AND FUTURE WORK

In this paper we have proposed a new spatial clustering approach to cluster polygon datasets. This approach can make use of the available domain knowledge in the form of constraints that guide the clustering process. Our algorithm, called constrained polygonal spatial clustering (CPSC), views the clustering process as a search process, with seeds as the start states, and the desired clusters satisfying or optimizing the constraints as the goal states, and embedding the constraints into the heuristics that guide the "search" process. Specifically, CPSC strategically uses inter-cluster and intra-cluster constraints to select the initial seeds for the clusters, to compute the distance and cost functions to select the best cluster to grow next, and to select the best neighbor to add to the best cluster. CPSC is thus able to remove order dependency from the clustering results, systematically moving towards meeting the soft constraints optimally, and increasing the likelihood of meeting the hard constraints.

We successfully applied CPSC to the difficult problem of congressional redistricting. There are several other such applications that can greatly benefit by using CPSC redistricting algorithm; for example, electricity dispersion zones, traffic analysis zones, and police precincts.

In terms of future work, our immediate next step is to look into the congressional redistricting problem more comprehensively by considering additional constraints such as the must-link constraint for minority-population areas. In addition, we plan to consider other spatial characteristics such as topological relationships and how polygons of irregular or natural shapes might behave under our CPSC algorithm. In particular, we intend to apply our framework to water resource management and drought mitigation. Further, there are also soft and hard constraints that are *temporal* (or seasonal) that we will need to consider.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grants No. 0219970 and 0535255.

REFERENCES

[1] Altman, M. "Is Automation the Answer? The Computational Complexity of Automated Redistricting", *Rutgers Computer & Technology Law J.* 23, 2001, pp: 81-142.

[2] Bodin, L. D. "A Districting Experiment with a Clustering Algorithm", Democratic Representation and Apportionment: Quantitative Methods, Measures and Criteria, Annals of New York Academy of Sciences , New York, 1973.

[3] Clayton, D. M., *African Americans and the Politics of Congressional Redistricting*, New York Garland Publishing Co., 2000. pp: 138 – 140.

[4] Davidson, I., and Ravi, S. S. "Clustering with constraints: Feasibility issues and the k-means algorithm". *Proc. SIAM Int. Conf. Data Mining*, 2005.

[5] Davidson, I., and Ravi, S. "Towards efficient and improved hierarchical clustering with instance and cluster level constraints", *Technical Report, Department of Computer Science, University of Albany*.

[6] Han, J., Kamber, M., and Tung, A. "Spatial clustering methods in data mining: A Survey," in Miller, H., and Han, J., eds., *Geographic Data Mining and Knowledge Discovery*. Taylor and Francis. 2001.

[7] Hayes, B. "Machine Politics", *American Scientist*, 1996, 84, pp:522-526.

[8] Joshi, D., Samal, A. K., and Soh, L. K. "Density-Based Clustering of Polygons", *IEEE Symposium Series on Computational Intelligence and Data Mining*, 2009, pp: 171-178.

[9] Law, M. H.C., Topchy, A., and Jain, A. K. "Clustering with Soft and Group Constraints". Joint IAPR International Workshop on Syntactical and Structural Pattern Recognition and Statistical Pattern Recognition, 2004.

[10] Macmillan, W. "Redistricting in a GIS environment: An optimization algorithm using switching points", *Journal of Geographical Systems*, Springer-Verlag, 2001, 3, pp: 167-180.

[11] Mann, H., and Fowle, W. B. *The Common School Journal,* Boston: Marsh, Capen, Lyon, and Webb, pp: 1838-1851.

[12] Ruiz,C., Spiliopoulou, M., and Ruiz, E. M. C-DBSCAN: Density-Based Clustering with Constraints. In 11th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing, 2007, pp 216 – 223.

[13] Wagstaff, K., Cardie, C., Rogers, S., and Schroedl S., "Constrained K-Means Clustering with Background Knowledge", *Proc. 18th ICML, 2001,* pp. 577-584.