ISSN: 0828-3494
ISBN:0-7731-0489-5

ObGen and SynGeoDataGen: Data Generators for Obstacle-Facilitator-Constrained Clustering

Xin Wang & Howard J. Hamilton
Technical Report CS-2004-8
April 2004

Department of Computer Science
University of Regina
Regina, Saskatchewan
CANADA S4S 0A2

ObGen and SynGeoDataGen: Data Generators for Obstacle-Facilitator-Constrained Clustering

Xin Wang and Howard J. Hamilton
Department of Computer Science
University of Regina
Regina, SK, Canada S4S 0A2
{wangx, hamilton}@cs.uregina.ca

1. Introduction

Finding clusters in spatial data is an active research area, with recent research on effectiveness and scalability of algorithms. Dealing with constraints due to obstacles and facilitators is an important topic in constraint-based spatial clustering. An ***obstacle*** is a physical object that obstructs the reachability among the data objects, and a ***facilitator*** is a physical object that connects distant data objects or connects data objects across obstacles. Conceptually, an obstacle increases the distance between objects while a facilitator decreases the distance. Handling constraints due to obstacles and facilitators can lead to effective and fruitful data mining by capturing application semantics [1, 2].

Several algorithms have been proposed to solve constraint-spatial clustering in the presence of obstacles and facilitators and a variety of experiments on synthetic datasets have been conducted [3, 4, 5]. However, for these experiments, the synthetic data were based on manual generation. To this point, no automatic generator of synthetic spatial data with obstacles and facilitators has been available for public use.

In this report, we introduce two data generators that we developed for generating synthetic obstacle datasets, facilitator datasets, and point datasets when obstacles and facilitators are present.

The remainder of this report is organized as follows. In Section 2, the ObGen obstacle generator is introduced. In Section 3, the SynGeoDataGen data generator is described. Conclusions are given in Section 4.

2. Synthetic Obstacle Dataset Generator ObGen

ObGen is a synthetic obstacle generator. Given an obstacle-specification file as input, it generates a set of obstacle files as output. It can generate three types of obstacles in a 5000 × 5000 area.

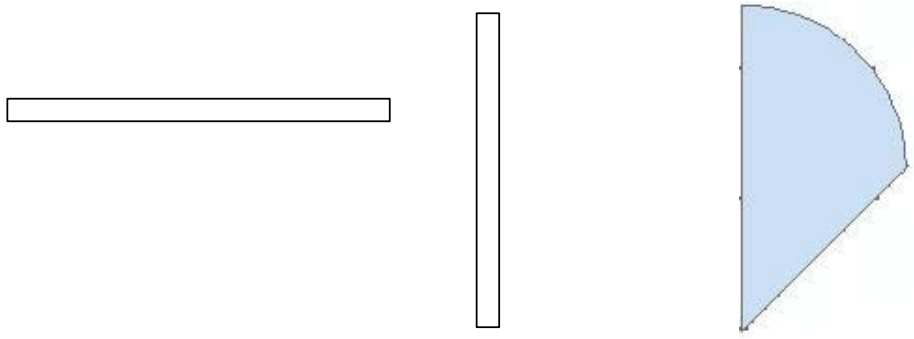

Figure 1. Three Types of Obstacles

```
...
8       408     603
8       306     603
8       204     603
8       102     603
8       0       603
9       670     5007.1
9       666     5007.96
9       662     5008.73
9       658     5009.42
9       654     5010.03
9       650     5010.55
9       646     5011
```

Figure 2. An Example Obstacle File

The three types of obstacles are shown in Figure 1. A ***type1 obstacle*** is a thin rectangle with long horizontal edges. A ***type2 obstacle*** is a rectangle with long vertical edges. A ***type3 obstacle*** is an irregular shape. The apparent curve on a type3 obstacle is actually made of many short edges. Each obstacle is a polygon represented by a set of (x,y) vertices. An ***obstacle file*** is composed of a series of vertices of polygons. Each line in the obstacle file represents one vertex and is composed of three parts, including ObstacleId, vertex_x, vertex_y. The vertices are given in sequential order around the obstacle from an arbitrary starting vertex. Figure 2 shows an example of part of an obstacle file. In the first line, "8" is the obstacle id, and the (408, 603) pair gives the (x, y) coordinates of the vertex.

The ObGen program generates the obstacle file(s) while reading from a text file called the ***obstacle-specification file***. Each line of the obstacle-specification file provides the parameters for generating one obstacle file. Figure 3 shows an example obstacle-specification file that generates 10 obstacle files. In each line, the first parameter is the name of the obstacle file to be generated; the second parameter is the total number of obstacles. In Figure 3, the first five files have 100 obstacles, and the last five files have 200 obstacles. The third parameter is the number of vertices in each obstacle. In the example file, all obstacles have 100 vertices. The last three parameters in the line give the number of type1, type2, and type3 obstacles in the file. For example, in the first line of Figure 3, 30, 40, 30 means we need to generate 30 type1 obstacles, 40 type2 obstacles, and 30 type3 obstacles.

Although an example obstacle file was shown in Figure 2, it may be helpful to illustrate the obstacle dataset in its surrounding area. Figure 4 shows an obstacle dataset with 30 horizontal type1 obstacles, 30 vertical type2 obstacles, and 40 wedged-shaped type3 obstacles. The horizontal and vertical lines are actually very thin rectangles. The irregular wedge shapes are the type3 obstacles.

The size of the whole area is 5000 × 5000. The type1 and type2 obstacles are distributed evenly in that area. The locations of the type3 obstacles are generated randomly. In the figure, some of the obstacles (lines) appear darker, but this difference is due to inconsistent output from the ArcGIS software that was used to produce the images.

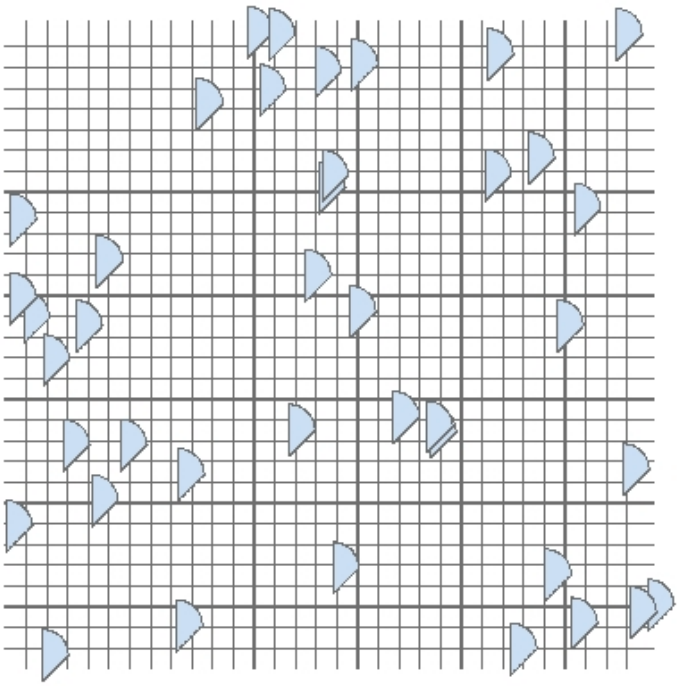

Figure 4. An Example Obstacle Dataset

3. The SynGeoDataGen Synthetic Data Generator

The ***SynGeoDataGen data generato***r generates synthetic datasets that include obstacles and facilitators. The program was orginially written by Howard Hamilton in June 2002 and revised by Camilo Rostoker and Xin Wang in November 2003.

SynGeoDataGen generates the synthetic data while reading a text file. Each line in the file includes 15 parameters, separated by commas (","). SynGeoDataGen generates one dataset for each line. The 15 parameters are as follows.

1) the filename of the facilitator dataset;
2) the number of facilitators in the facilitator dataset;

```
obs1_1  100     100     30, 30, 40
obs1_2  100     100     40, 30, 30
obs1_3  100     100     50, 20, 30
obs1_4  100     100     60, 20, 20
obs1_5  100     100     10, 10, 80
obs2_1  200     100     70, 80, 50
obs2_2  200     100     60, 40, 100
obs2_3  200     100     100, 40, 60
obs2_4  200     100     30, 100, 70
obs2_5  200     100     40, 40, 120
```

Figure 3. An Example Obstacle-Specification File

3) the number of vertices used to define each facilitator;
4) the obstacle file, i.e., the filename of the obstacle dataset;
   If you have no obstacle file, put an "X" for this parameter. As mentioned in Section 2, each line of the obstacle file represents a vertex of an obstacle.
5) the filename of the synthetic point dataset that is to be generated;
6) the number of non-spatial attributes: i.e., the number of property values;
7) the number of spatial attributes (for a 2D dataset, set it to 2);
8) the range of values for spatial attributes; for example, 500 means the spatial attributes will range from 0 to 499.
9) a zero-one value describing the subset of the spatial attributes that contributes to defining a cluster; it could be fixed (1) or variable (0). It is usually set to 1;
10) the chance (out of 100) that an attribute should have a nonrandom value, where 100 indicates that all attributes should be given values near to the center of the cluster;
11) the number of clusters. To generate pure noise, set this parameter to 0.
12) the maximum cluster radius; the maximum distance (in any one of the spatial attribute values) between the center and a point belonging to the largest cluster, .
13) the minimum cluster radius; the maximum distance (in any one of the spatial attribute values) between the center and a point belonging to the smallest cluster.
14) the number of points in each cluster. For example, if you set this parameter to 1000 and the 11th parameter to 25, the dataset will include 25,000 points. Since the number of points for each cluster is fixed, we can use parameters 12 and 13 to adjust the density of the clusters.
15) the percentage of purity, which will usually be set to 100.

```
fac25k.txt, 10, 1000, X, pf25k_data, 1, 2, 5000, 1, 100, 36, 20, 25, 625, 100
none.txt, 0, 0, fac25k.txt, pf25k_noise, 1, 2, 5000, 1, 100, 1, 2500, 2500, 2500, 100
fac50k.txt, 10, 1000, obstacle1.txt, pf50k, 1, 2, 5000, 1, 100, 20, 53, 58, 2500, 100
```

Figure 5. An Example Input File for SynGeoDataGen

Figure 5 shows an example input file. The first line specifies that a point dataset and a facilitator dataset should be generated without any obstacles, since the fourth parameter is an "X." According to this line, 10 facilitators should be generated and saved as fac25k.txt. Each facilitator should have 1000 vertices. The point dataset should have 1 non-spatial attribute and 2 spatial attributes. Since the value of the eighth parameter is 5000, the values of the spatial attributes should be integers in the range from 0 to 4999. The subset of spatial attributes should be fixed and the chance that an attribute has a nonrandom value should be 100%. There should be 36 clusters in the dataset and each cluster should have 625 points (22500 points in total). The minimum and maximum cluster radius should be 20 and 25, respectively. The percentage of purity of clusters should be 100%. The second and third line specify that point and facilitator datasets should be generated with obstacles defined as specified in "fac25k.txt" and "obstacle1.txt" , respectively.

Given a specification file as input, SynGeoDataGen can generate both a facilitator dataset and a point dataset or only a point dataset. To generate only a point dataset, the second and third parameters should be set to 0, because the number of facilitators and the number of vertices for each facilitator are 0. The clusters of data points generated by SynGeoDataGen are square-shaped. The centers of the clusters are distributed randomly in a 5000 × 5000 area, i.e., for each center, the value for each attribute is picked randomly from a uniform distribution of possible values for the attribute. When generating a cluster, SynGeoDataGen first picks a center for the

cluster randomly, then distributes points within a specified distance (the radius) of that center. The number of points in the cluster and the radius of the cluster are specified in the input text file. If the center occurs near the edge of the 5000x5000 area, all points are still included, but the cluster will be shaped like a rectangle rather than a square and its density will be higher.

SynGeoDataGen can also generate facilitator datasets. The facilitator datasets are generated during the same run as the point datasets. First, the centers of the cluster are picked randomly. Then, for each facilitator, two cluster centers are selected randomly to be its two vertices. Based on the two vertices and the edge between them, a rectangle-shaped facilitator is generated fore ach such pair of vertices. Finally the required number of points are distributed within the specified radius of each center. The generated points do not overlap with facilitators.

Finally, SynGeoDataGen also can generate clusters in the presence of obstacles. When an obstacle overlaps with a planned cluster, SynGeoDataGen avoids distributing any points inside the obstacle polygon.

Figure 6 shows a snapshot of the datasets that would be generated from the last line in Figure 5. There are 20 clusters (shown as small black square) and 5 facilitators (shown as black lines connecting the clusters). Each cluster includes 2500 points and each facilitator has 1000 vertices. The obstacles shown in Figure 4, which were stored in the obstacle file called "obstacle1.txt" were used as the obstacle dataset.

Using our software, there are two ways to generate a dataset with a specified percentage of noise. One way is to set the percentage of purity of the clusters to the desired value. This can be done using the 15th parameter of SynGeoDataGen. The other way is to generate the noise points and non-noise points in two separate datasets, and then combine the two datasets. For example, we can use the first two lines of the specification given in Figure 5 to generate a dataset with 25000 points including 10% noise points. First, we generate a facilitator dataset called "fac25k.txt" and a 22,500 point dataset called "pf25k_data" using SynGeoDataGen. Then, we use the "fac25k.txt" facilitator dataset as an obstacle dataset (parameter #4) to produce 2500 noise points, which can be thought of as a single big cluster with a radius of 2500 (which covers the whole 5000 x 5000 area). The reason we use the facilatator dataset as the obstacle dataset is to avoid generating noise points inside the facilitator polygons. Finally, we combine the two point datasets as one 25000 point dataset. Before combining two datasets, the values of the non-spatial attribute in the noisy dataset should be set to different values from the non-spatial attribute in the non-noise dataset. For example, all values for the non-spatial attribute of the noisy dataset can be changed from "0" to the character "n" which is not used for the non-spatial attribute in the non-noise dataset. This can be easily accomplished using standard spreadsheet software, such as Microsoft Excel. This transformation is required because the noise points generating by the second method are evenly distributed throughout the whole clustering area, while the noise points generated by first method will overlap with every single cluster.

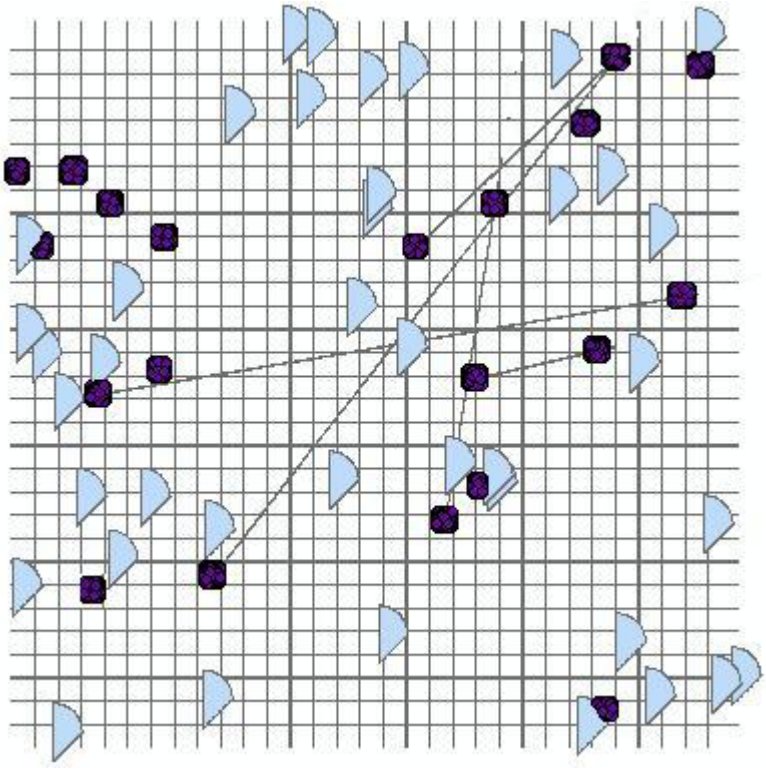

Figure 6. An Example A Point Dataset in the Presence of Obstacles and Facilitators

4. Download available

The ObGen and SynGeoDataGen generators were both written in C++. These files are available for downloading as .zip files at www.cs.uregina.ca\~wangx\synGeoDataGen.html. Some sample datasets are also available.

5. Conclusion

Constraint-based clustering is an important topic in spatial clustering. To facilitate generating synthetic datasets, we created the ObGen software to generate obstacle files and the SynGeoDataGen software to generate point files and facilitator files. When the three types of output files are combined together, we can represent clustering problems featuring data points, facilitators, and obstacles.

References:

[1] Han, J., Lakshmanan, L.V.S., and Ng, R.T.: Constraint-Based Multidimensional Data Mining. *Computer* 32(8) (1999) 46-50.

[2] Tung, A.K.H., Han, J., Lakshmanan, L.V.S., and Ng, R.T.: Constraint-Based Clustering in Large Databases. In *Proc. 2001 International Conference on Database Theory*, London, U.K., (2001) 405-419.

[3] Tung, A.K.H., Hou, J., and Han, J.: Spatial Clustering in the Presence of Obstacles. In *Proc. 2001 International Conference On Data Engineering*, (2001) 359-367.

[4] Estivill-Castro, V. and Lee, I. J.: AUTOCLUST+: Automatic Clustering of Point-Data Sets in the Presence of Obstacles. In *Proc. of International Workshop on Temporal, Spatial and Spatio-Temporal Data Mining*, Springer, (2000) 133-146.

[5] Zaïane, O.R., and Lee, C.H.: Clustering Spatial Data When Facing Physical Constraints. In *Proc. of the IEEE International Conference on Data Mining*, Maebashi, Japan, December, (2002) 737-740.

Appendix (Source Code)