Fast map generalization heuristic with a uniform grid

Salles V. G. de Magalhães
Universidade Federal de Viçosa
Viçosa, Brazil
salles@ufv.br

W. Randolph Franklin
Rensselaer Polytechnic Institute
Troy, NY, USA
mail@wrfranklin.org

Wenli Li
Rensselaer Polytechnic Institute
Troy, NY, USA
liw9@rpi.edu

Marcus V. A. Andrade
Universidade Federal de Viçosa
Viçosa, Brazil
marcus@ufv.br

ABSTRACT

We present *Grid-Gen*, an efficient heuristic for map simplification. *Grid-Gen* deals with a variation of the generalization problem where the idea is to simplify the polylines of a map without changing the topological relationships between these polylines or between the lines and control points. *Grid-Gen* uses a uniform grid to accelerate the simplification process and can handle a map with more than 3 million polyline points and 10 million control points in 9 seconds in a Lenovo T430s laptop.

Categories and Subject Descriptors

I.3.5 [**COMPUTER GRAPHICS**]: Computational Geometry and Object Modeling

General Terms

Algorithms, Performance

Keywords

Algorithms, Computational geometry, Map generalization

1. INTRODUCTION

One important problem of computational geometry is the curve generalization (or simplification) problem, where the objective is to reduce the amount of information needed to represent a curve while keeping it "similar" to the original geometry. The most well-known algorithms to solve this problem are Douglas-Peucker [3, 7] and Visvalingam-Whyatt [10].

Generalization methods are widely used in GIS to reduce the amount of storage of vector maps, to perform map analysis with different level of details, to improve the display quality of a small scale map [8] and to perform progressive transmission of maps [2].

SIGSPATIAL'14, November 04 - 07 2014, Dallas/Fort Worth, TX, USA

http://dx.doi.org/10.1145/2666310.2666421

While methods such as Douglas-Peucker try to simplify lines while keeping them as "similar" to the original input as possible, the direct application of these algorithms to simplify polylines in a map may create undesirable features. For example, if Douglas-Peucker is applied to a county dataset, the boundaries may be simplified in a way that a point representing a city will be in the wrong county. Also, simplifying a polyline may make it cross another line in the map. See Figure 1: if the red polyline is simplified by removing point a, the resulting map will be more similar to the original than the map obtained by removing point b. However, in the former case the topological relationships in the map will change, since point c will be in the other side of the polyline.

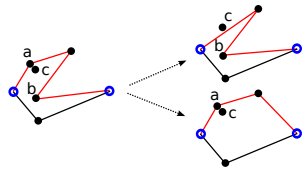

Figure 1: Challenges during map simplification.

In this work we will deal with the following variation of the geometry generalization problem: given a set of polylines and a set of control points, simplify these polylines by removing some of their points (the only points that cannot be removed are the endpoints) such that the topological relations between pairs of polylines and between the polylines and the control point do not change.

We propose *Grid-Gen*, a heuristic that, combined with a uniform grid, can efficiently simplify maps without creating any change in the topology. *Grid-Gen* can efficiently process geometries containing millions of points.

2. THE PROPOSED HEURISTIC

Given a set of control points C and an input map M composed of a set P of polylines, our heuristic simplifies M by iteractively processing each polyline independently. When a polyline is processed, *Grid-Gen* iterates through all its interior points v_i (that is, the points excluding the endpoints) and removes v_i if this deletion would not change the topological relations between the map's elements.

To determine if the deletion of a polyline point v_i would change the map topology, *Grid-Gen* verifies if there is any control point or polyline point inside the triangle whose vertices are v_i and its two adjacent points (i.e., v_{i-1} and v_{i+1}).

Figure 2 presents an example of the possible topological changes that may happen during the deletion of points: Notice that there is a control point x inside the triangle (in red) formed by polyline point a and its two adjacent points. If a polyline is simplified by removing a, then the topological relation between the curve and x will change. Point b also cannot be removed since polyline point y is inside the red triangle containing b as vertex and, thus, the deletion of b would change the topological relation between b's polyline and y's polyline (in fact, the two polylines would cross if b was removed). Therefore, neither a nor b should be removed from the current map. Point c, on the other hand, may be removed without changing the map topology since there is no control or polyline point inside the blue triangle whose vertex is c.

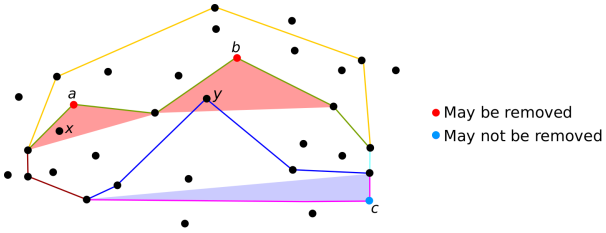

Figure 2: Determining if the deletion of some points would change the map topology.

Algorithm 1 presents a pseudo-code of the method. Notice that, if during one iteration of the while loop no point is removed from the map, then in the next iterations no point will be removed (since the map does not change when no point is removed) and, thus, the heuristic can be terminated.

There are two situations where Algorithm 1 may create simplified maps with an invalid topology. If one polyline p has coincident endpoints and the polygon (or *island*) defined by this polyline does not have any control point or other polylines in its interior, then Algorithm 1 may remove all interior points from p (creating an invalid polygon). Figure 3 presents an example where an island is simplified creating a "polyline" composed only by the two coincident endpoints.

Also, if two polylines p_1 and p_2 have the same endpoints and the polygon formed by them does not contain any control point or polyline in its interior, then Algorithm 1 may remove all interior points of p_1 and p_2, creating two coincident line segments. See the example in Figure 4.

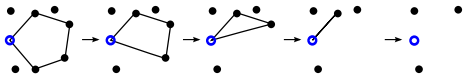

Figure 3: Simplification of a polyline with coincident endpoints (in blue) and with no control point or other polylines in the interior of the polygon that it defines.

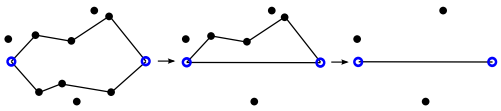

Figure 4: Simplification of two polylines with coincident endpoints (in blue) and with no control point or other polylines in the interior of the polygon that it defines.

To solve these two problems, we preprocess the input adding *dummy* control points that ensure that the heuristic would never simplify the polylines to an invalid state. If a polyline p has coincident endpoints, we add two dummy control points at an infinitesimal distance around one of the line segments that forms p. See an example in Figure 5. This ensures that one of these control points will be always in the interior of the polygon defined by p. Thus, Algorithm 1 would never create an invalid simplification such as the one showed in Figure 3, since this would change the topological relations between the polyline and the *dummy* points.

Figure 5: Use of *dummy* control points (in orange) to avoid invalid simplifications: notice that, in this dataset, the simplification algorithm stops after removing the second interior point of the polyline since one of the *dummy* control points is in the interior of a triangle defined by the polyline vertices.

If an input polyline p has only two points (that is, if it does not have any internal points) we also add two *dummy* control points in an infinitesimal distance around p. Furthermore, if during the simplification all the internal points of a polyline are removed, the dummy points are also added around the resulting polyline. This ensures that no simplification would create a polyline coincident to p. Figure 6 presents an example where all interior points of a polyline p are removed and, then, two *dummy* points are added to the map.

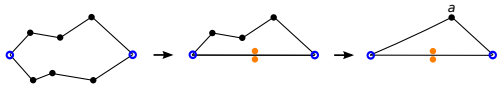

Figure 6: Use of *dummy* control points to avoid the creation of coincident polylines. Notice that *Grid-Gen* does not remove point a, since the polygon contains one of the dummy points.

Also, it is necessary to consider the special case where some line segments are colinear to the endpoints of a polyline. For example, in Figure 7 there is no polyline or control point in

Algorithm 1 Pseudo-code of the map simplification algorithm

```
np2rem ← Number of points to remove
npar ← 0 //Number of points already removed from the map
while npar < np2rem do
    for each polyline p in the input map do
        for each interior point v_i in p do
            if v_{i-1}, v_i, v_{i+1} are colinear OR there is no polyline point or control point in triangle v_{i-1}, v_i, v_{i+1} then
                npar ← npar + 1
                Remove v_i from p
                if npar ≥ np2rem then
                    Stop the algorithm
                end if
            end if
        end for
    end for
    if no point was removed from the input map then
        Stop the algorithm
    end if
end while
```

the triangle v_{i-1}, v_i, v_{i+1} and, therefore, Algorithm 1 could remove vertex v_i, creating coincident segments. This special case is treated by considering that points in the border of the triangle v_{i-1}, v_i, v_{i+1} are inside the triangle and, therefore, Algorithm 1 does not remove v_i.

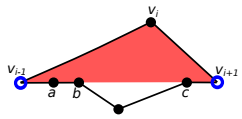

Figure 7: Example of two polylines with the same endpoints (in blue): if v_i is removed, the resulting polyline will have segments coincident to the segments of the other polyline. This problem is solved by considering that points in the border of the triangle v_{i-1}, v_i, v_{i+1} (points a, b and c) are effectively inside the triangle.

3. ACCELERATING THE POINT IN TRIANGLE TESTING

The bottleneck of Algorithm 1 is the test to detect if a polyline or control point lies inside a triangle. Several spatial indexing techniques may be used to accelerate this process like, for example, R-trees [5] or uniform grids [4]. This work uses a uniform grid to perform this kind of optimization.

More specifically, the idea is to create a $N \times M$ grid (where N and M are parameters defined by the user), superimposed over the map being simplified. Each cell c of the grid contains a list of all points (polyline and control points) inside it. Given a triangle t, only the points in the cells that intersect t need to be checked in order to verify if there is any point in t. If a polyline is simplified, the point removed from the polyline is also removed from the uniform grid.

Figure 8 presents an example of a 3×5 uniform grid superimposed on the map. Only the 6 points laying in the detached grid cells need to be checked to determine if there is a point in the detached triangle.

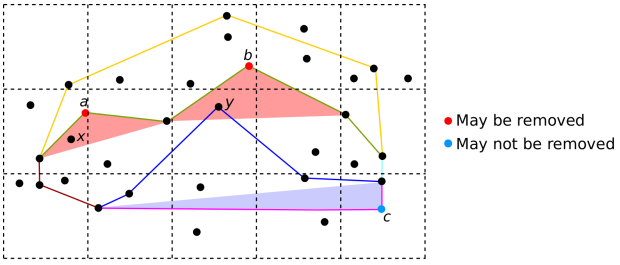

Figure 8: Example of a 3×5 uniform grid superimposed over a map.

4. EXPERIMENTAL EVALUATION

*Grid-Gen*was tested on a laptop with the following configuration: i7-3520M 3.6 GHz processor, 8GB of RAM memory, Samsung 840 EVO SSD (500 GiB) and Linux Mint Mate 16 operating system.

Tests were performed on 7 datasets, the first 5 datasets were used by the GISCUP 2014 [1] organizers to evaluate the solutions submitted to the GISCUP contest; the polygons in dataset 6 were obtained from *IBGE*'s (the Brazilian geography agency) website [6] and represents the Brazilian county divisions; dataset 7 represents the United States continental county division and was obtained from the United States Census website [9]. The control points in datasets 6 and 7 were selected in random positions in the maps. Table 1 presents the number of points in each map and, also, the maximum number of points that *Grid-Gen* removed during the simplification.

Initially, we evaluated the processing time (excluding I/O time) of *Grid-Gen* considering the 7 datasets and different dimensions for the uniform grid. Table 2 presents the processing time (in milliseconds). We also evaluated the heuristic using an 8000×8000 uniform grid, but these results were not included because *Grid-Gen* was slower in all datasets using this grid size.

Notice that for the smaller datasets, the overhead caused by the management of a larger uniform grid does not balance

Table 1: Number of control and polyline points in each dataset and number of points removed after the map simplification process.

Dataset	1	2	3	4	5	6	7
Number of control points	26	127	151	256	1607	10000	10000000
Number of polyline points	992	1564	8531	28014	28323	342738	3645559
Number of points removed	928	1435	7545	25212	23411	308992	3613026

the possible reduction in the number of cells evaluated during the point in triangle test. On larger datasets, on the other hand, increasing the grid size significantly improves the heuritic's performance (for example, there is an improvement of 18 times in the performance when the grid size is increased from 125^2 cells to 4000^2 cells in dataset 7).

Table 2: Processing-time (in milliseconds) to simplify each of the 7 maps. The smallest time for each dataset is in boldface.

Map	Grid Size (# cells)					
	125^2	250^2	500^2	1000^2	2000^2	4000^2
1	**0**	4	7	23	99	405
2	**0**	2	5	17	65	269
3	**2**	3	6	15	51	171
4	11	**9**	12	19	50	168
5	10	**8**	10	17	45	149
6	333	170	134	**132**	203	376
7	163875	48940	22529	14307	10708	**9172**

In the second set of tests, presented in Table 3, we evaluated the amount of time spent by each step of *Grid-Gen*. For each dataset, *Grid-Gen* was executed using the uniform grid size that presented the best result considering the tests from Table 2. Notice that, even though the tests were performed in a machine with a fast SSD drive, in all situations most of the processing time was spent performing I/O. This suggests that further improvements in the simplification heuristic (for example, using parallelization) would not improve significantly the total running time. Observe also that the amount of time used to write the output is much smaller than the time spent to read the input map. This could be explained because, as shown in Table 1, *Grid-Gen* was able to significantly simplify all datasets.

Table 3: Processing-time (in milliseconds) spent in each step of *Grid-Gen*. Row *Initialize* presents the total time for the initialization of the uniform grid.

Dataset	1	2	3	4	5	6	7
Grid size	125	125	125	250	250	1000	4000
Read input	1	2	9	23	27	275	37688
Initialize	0	0	0	1	1	25	1567
Simplify	0	0	2	8	6	107	7605
Write output	0	0	2	4	8	40	38

5. CONCLUSIONS AND FUTURE WORKS

We presented *Grid-Gen*, a heuristic that, as showed in our experiments, can perform map simplification very efficiently. Even though the tests were performed using a fast SSD drive, most of the processing time was spent in I/O operations. Also, *Grid-Gen* never changes the topological relationships in the map.

Future work includes comparing *Grid-Gen* with other methods and determining an efficient strategy to automatically determine an adequate uniform grid size for each input map. Another extension is adapting it not only to perform the simplification while satisfying the topological constraints, but also trying to keep the output data as similar as the input as possible. This could be achieved by incorporating ideas from other generalization (like Visvalingam-Whyatt method) algorithms to *Grid-Gen*.

6. ACKNOWLEDGMENTS

This research was partially supported by NSF grant IIS-1117277 and by CAPES (Ciencia sem Fronteiras).

7. REFERENCES

[1] ACM SIGSPATIAL Cup 2014. GIS Cup sample data. mypages.iit.edu/∼xzhang22/GISCUP2014/ (accessed on 09/01/2014).

[2] M. Bertolotto and M. J. Egenhofer. Progressive transmission of vector map data over the world wide web. *GeoInformatica*, 5(4):345–373, 2001.

[3] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica*, 10(2):112–122, 1973.

[4] W. R. Franklin, D. Sun, M.-C. Zhou, and P. Y. Wu. Uniform grids: A technique for intersection detection on serial and parallel machines. In *Proceedings of Auto Carto 9*, pages 100–109, Maryland, April 1989.

[5] A. Guttman. *R-trees: a dynamic index structure for spatial searching*, volume 14. ACM, 1984.

[6] IBGE: Instituto Brasileiro de Geografia e Estatistica. Malhas digitais dos municipios Brasileiros. geoftp.ibge.gov.br/malhas_digitais/municipio_2007/ (accessed on 09/01/2014).

[7] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972.

[8] W. Shi and C. Cheung. Performance evaluation of line simplification algorithms for vector generalization. *The Cartographic Journal*, 43(1):27–44, 2006.

[9] United States Census. US counties Shapefiles. www.census.gov/cgi-bin/geo/shapefiles2013/main (accessed on 09/01 /2014).

[10] M. Visvalingam and J. Whyatt. Line generalisation by repeated elimination of points. *The Cartographic Journal*, 30(1):46–51, 1993.