An efficient map generalization heuristic based on the Visvalingam-Whyatt algorithm

Salles V. G. de Magalhães (salles@ufv.br)
Marcus V. A. Andrade (marcus@ufv.br)
Universidade Federal de Viçosa, Brazil

W. Randolph Franklin (mail@wrfranklin.org)
Wenli Li (liw9@rpi.edu)
Rensselaer Polytechnic Institute, USA

ABSTRACT

We present Grid-Gen2, an efficient heuristic for map simplification that 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-Gen2 is a strategy based on the Visvalingam-Whyatt algorithm to create simplified geometries with shapes similar to the original map. The simplification process is accelerated using a uniform grid. Grid-Gen2 can process a map with more than 3 million polyline points and 10 million control points in 24 seconds in a Lenovo T430s laptop.

1. INTRODUCTION

One important problem in 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 [1, 3] and Visvalingam-Whyatt [5].

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 counties’ boundaries may be simplified in a way that a city will be in the wrong county. Also, simplifying a polyline may make it cross another line in the map.

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 (except the endpoints) such that the topological relations between pairs of polylines and between the polylines and the control points do not change. In a previous paper [4], we proposed Grid-Gen, which uses a uniform grid to efficiently generalize maps. This paper presents Grid-Gen2 that is an extension of our previous method Grid-Gen where we included a new heuristic based on the Visvalingam-Whyatt [5] algorithm to generate maps with better quality than the maps generated by Grid-Gen.

2. THE GRID-GEN 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 iteratively processing each polyline independently. When a polyline is processed, Grid-Gen [4] iterates through all its interior points v_i (that is, the points excluding the endpoints) and removes v_i if the deletion would not change the topological relations between the map’s elements. This process is repeated until the number of points in the simplified map reaches a target value defined by the user or until the map cannot be further simplified without changing its topology.

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 t whose vertices are v_i and its two adjacent points (i.e., v_i-1 and v_i+1).

Figure 1 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 polylines y and z are far from it.

Figure 1: Determining if the deletion of some points would change the map topology.

There are two special cases that Grid-Gen needs to deal in order to avoid creating a simplified map with invalid topology. First, 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 Grid-Gen may remove all interior points from p (creating an invalid polygon). Second, 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 polylines in its interior, then Grid-Gen may remove all interior points of p_1 and p_2, creating two coincident line segments.

To solve these two problems, Grid-Gen preprocesses 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, two dummy control points are added at an infinitesimal distance from one of the line segments that forms p. See an example in Figure 2 (a). This ensures that one of these control points will be always in the interior of the polygon defined by p and, therefore, the heuristic will never remove all interior points of p.

Figure 2: Use of dummy control points (in orange) to avoid invalid simplifications.

If an input polyline p has only two points Grid-Gen also adds 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 2 (b) presents an example where all interior points of a polyline are removed and, then, two dummy points are added to the map.

Since the bottleneck of Grid-Gen is to detect if a polyline or control point lies inside a triangle, a uniform grid [2] is used to accelerate this process. More specifically, the idea is to create a N x 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.

3. THE PROPOSED EXTENSION

As explained in section 2, Grid-Gen iteratively process the input map removing from the polylines the interior points whose deletion would not cause a change in the map topology. This strategy can efficiently create simplified maps with no topological inconsistency, but it does not try to keep the simplified map similar to the original geometry.

We propose Grid-Gen2, an extension of Grid-Gen that ranks the interior polyline points based on their “importance” to the map shape and try to perform the simplification by removing the least important points first. More specifically, the points are ranked using the same strategy as Visvalingam-Whyatt [5] algorithm and, then, a simplification process similar to Grid-Gen is performed in the map, processing the points in an order based on their rank. Thus, while the point ranking strategy tries to generate simplified maps similar to the original input data, the topological inconsistency detection strategy derived from Grid-Gen ensures that no topological error is introduced in the output map.

Given a polyline point \( p_i \), the rank of \( p_i \) is defined based on the area (called effective area) of the triangle defined by \( p_i \) and its two adjacent points from its polyline. As shown by Visvalingam-Whyatt [5], points with higher effective areas are usually “more important” than points with smaller areas and, thus, the latter should have a higher priority when choosing which point to remove during the simplification process.

For efficiency purposes, Grid-Gen2 initially preprocess the input data computing the effective area of the polyline points. These points are kept in a priority queue with priority based on the point’s effective areas. When a point \( p_i \) is deleted from the map, this may change only the effective area of its two adjacent points \( p_{i-1} \) and \( p_{i+1} \) (in \( p_i \)’s polyline) and, thus, these two areas are recomputed and the new values are used to update the priority of \( p_{i-1} \) and \( p_{i+1} \) in the queue.

4. EXPERIMENTAL EVALUATION

Grid-Gen2 was tested on a laptop with the following configuration: i7-3520M 3.6 GHz processor, 8GB of RAM memory, Samsung 840 EVO SSD (500 GB) and Linux Mint Mate 16 operating system. The tests were performed on the same datasets previously used by Magalhães et al. [4]. However, due to the lack of space in this paper, we will not present the results from datasets 1 and 2 (the smallest datasets).

We compared the processing time of Grid-Gen2 against the processing time of Grid-Gen. Both methods were configured to simplify the maps by removing 50% of their points. The uniform grid size was chosen based on the configuration that led to the fastest performance in Magalhães et al. paper [4]. Table 1 presents the processing time (in milliseconds) for each step of the simplification process (the time for computing the effective area and for initializing the priority queue is included in the the simplification step of Grid-Gen2). Since the time for data I/O and the uniform grid initialization step are the same for both methods (since these steps use the same implementation in the two algorithms), only the simplification time is presented separately.

Observe that, in the worst case, the simplification step of Grid-Gen2 was 8 times slower than the same step in Grid-Gen. However, even though the tests were performed in a machine with a fast SSD drive, in all scenarios both algorithms spent most of their processing time performing I/O. Indeed, if we consider the total running time of the algorithms, Grid-Gen2 was less than 2 times slower than Grid-Gen in the worst test.

Table 1: Processing-time (in milliseconds).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td># input points</td>
<td>8531</td>
<td>3 \times 10^4</td>
<td>3 \times 10^4</td>
<td>3 \times 10^5</td>
<td>4 \times 10^6</td>
</tr>
<tr>
<td>Input reading</td>
<td>10</td>
<td>22</td>
<td>29</td>
<td>257</td>
<td>37092</td>
</tr>
<tr>
<td>Un. grid init.</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>24</td>
<td>1472</td>
</tr>
<tr>
<td>Simp. (Grid-Gen2)</td>
<td>2</td>
<td>15</td>
<td>13</td>
<td>435</td>
<td>23759</td>
</tr>
<tr>
<td>Simp. (Grid-Gen)</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>54</td>
<td>3481</td>
</tr>
<tr>
<td>Output writing</td>
<td>6</td>
<td>21</td>
<td>21</td>
<td>170</td>
<td>1817</td>
</tr>
</tbody>
</table>

Figure 3(a) presents an example of a region from dataset 3. In this Figure, the original dataset and the simplified map obtained by the two methods are overlayed, with the original map (in blue) in the top layer. It is easy to see that Grid-Gen2 maintained the map shape better than Grid-Gen. Indeed, it is difficult to see in this figure regions where the green polylines, that represents the map simplified by Grid-Gen2, are visible. Figure 3(b) displays a zoomed region from the map in Figure 3(a) where it is possible to observe the difference between the three maps. Notice that Grid-Gen2’s output keeps the similarity with the original map better than Grid-Gen.

5. CONCLUSIONS AND FUTURE WORKS

We presented Grid-Gen2, a heuristic that uses techniques based on the Visvalingam-Whyatt [5] algorithm to perform map simplification generating maps that not only are topologically correct but also preserves the shapes of the original map better than Grid-Gen, our previous heuristic. Even though Grid-Gen2 uses a simplification strategy much more sophisticated than Grid-Gen, it is only two times slower (considering the total processing time).

Future work includes evaluating Grid-Gen and Grid-Gen2 by comparing their performance and the quality of their solutions with other methods. Furthermore, another extension is determining an efficient strategy to automatically determine an adequate uniform grid size for each input map.

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

6. REFERENCES


