An Efficient and Topologically Correct Map Generalization Heuristic

Maurício G. Gruppi[1], Salles V. G. de Magalhães[1], Marcus V. A. Andrade[1], W. Randolph Franklin[2] and Wenli Li[2]

[1]*Departamento de Informática, Universidade Federal de Viçosa, Campus da UFV, Viçosa, Brazil*

[2]*Electrical, Computer, and Systems Engineering Department, Rensselaer Polytechnic Institute, 110 8th ST, Troy, U.S.A.*

Keywords: Map Generalization, Computational Geometry, Line Simplification, Geographic Information Systems.

Abstract: We present TopoVW, 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. This process is important for maintaining clarity of cartographic data, avoiding situations such as high density of map features, inappropriate intersections. In practice, high density of features may be represented by cities condensed into a small space on the map, inappropriate intersections may produce intersections between roads, rivers, and buildings. TopoVW is a strategy based on the Visvalingam-Whyatt algorithm to create simplified geometries with shapes similar to the original map, preserving topological consistency between features in the output. It uses a point ranking strategy, in which line points are ranked by their effective area, a metric that determines the impact a point will cause to the geometry if removed from the line. Points with inferior effective area are eliminated from the original line. The method was able to process a map with 4 million line points and 10 million control points in less than 2 minutes on a Intel® Core™ 2 Duo processor.

1 INTRODUCTION

Map generalization, or cartographic generalization, is a process that aims to simplify the way cartographic data is represented by removing information that is not relevant to the viewer, while preserving crucial features on the map. Generalization is an innate trait of every geographical data, maps are generalized representations of reality, and the more generalized is a map, the more remote it becomes from the real world (João, 1998). The output of this process is a map with less complex data and more desirable properties than those from the original map. The biggest concern in this process is to find the balance between simplification and actuality .

Although generalization is a task that has been effortlessly done by humans, it is still hard to automate. One of the main reasons is that information may change during varying of scale, and relationships between features in the map have to be maintained. Another reason is that "a map is a complex mix of metric and topological patterns reflecting individual and interdependent gestaltic properties. Understanding these forms, and conveying salient characteristics requires both cartographic and geographic knowledge" (Mackaness et al., 2011, p.7).

Ideally, situations requiring generalization occur due to failure of the map to meet its goals, that is, during the simplification process, the map is unsuccessful in preserving clarity of the content, at a given scale, for the intended viewer (Shea and McMaster, 1989). Some conditions that occur due to scale reduction may determine the need of generalization on a map, such as high feature density and imperceptibility, when a feature may be invisible, enlarged, or converted to the appearance of a different feature, for instance, when a polygon is condensed so much it acquires the shape of a line or a point.

1.1 Topological Consistency

Preserving consistency of data is crucial on both digital and analogue cartography. As result of generalization processes, both geometries and topological relationships of map features are expected to be changed (Ruas and Lagrange, 1995). This is undesirable since it may result in inappropriate maps, containing inconsistent relationship between features, such as roads (lines) intersecting buildings (polygons, for instance), buildings intersecting other buildings, or overlap be-

G. Gruppi M., V. G. de Magalhães S., V. A. Andrade M., Randolph Franklin W. and Li W..
An Efficient and Topologically Correct Map Generalization Heuristic.
DOI: 10.5220/0005398105160525
In *Proceedings of the 17th International Conference on Enterprise Information Systems* (ICEIS-2015), pages 516-525
ISBN: 978-989-758-096-3

tween rivers and buildings. Therefore, simplifications algorithms must handle such situations, if they occur. Sidedness is another aspect to be preserved. Maintaining sidedness means that, after the process of generalization, features which were simplified will not be on a different side of other features (such as control points).

1.2 Polyline Simplification

One way to perform cartographic generalization is by applying polyline simplification to reduce the complexity of such features in a map. A polyline is a series of line segments defined by a sequence of n points (v_1, v_2, ..., v_n) called vertices, each two neighbouring segments in the series share a common endpoint, namely, a polyline describes a curve starting on a point v_1, passing through every vertex on the sequence until v_n. Figure 1 shows an example input containing three polylines L_1, L_2 and L_3, and also a feature point P. To apply generalization to a polyline implies to simplify its representation by removing points from its sequence, that is, to attempt to represent the same curve with fewer vertices, the process of simplification is illustrated by Figure 2. This, however, can cause inconsistencies of topological relationships between polylines and points. For instance, applying simplification on a county dataset may change its boundaries in such a way that a point representing a city may fall in the wrong county, i.e., during the process of simplification it was not taken into consideration that it was mandatory for the point representing the city to be inside the county polygon by the end of the generalization, that is, sidedness was not kept. Moreover, polyline simplifications can create inappropriate intersections between polylines as shown by Figure 3. Among the most well-known algorithms for performing polyline simplification are Douglas-Peucker, also known as Ramer-Douglas-Peucker (or RDP) (Douglas and Peucker, 1973; Ramer, 1972) and Visvalingam-Whyatt (Visvalingam and Whyatt, 1993).

In this paper is presented a solution for the following variation of the generalization problem: given a set of polylines and control points, the goal is to simplify these polylines by removing some of their vertices (except endpoints) such that topological relationships between pairs of polylines and between polylines and control points are kept. In practice, polylines may represent boundaries of counties or states, and control points may represent cities within these states. As indicated by Estkowski et al. (Estkowski and Mitchell, 2001), this problem is difficult to solve, even approximately. This paper presents an

Figure 1: Example input of three polylines L_1, L_2 and L_3, and a control point P.

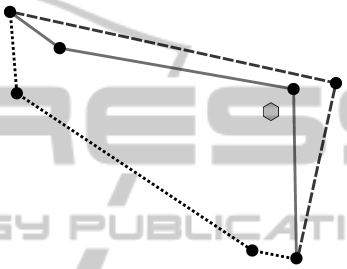

Figure 2: Example of valid simplification output, note that topology consistency is preserved, no intersections were created and sidedness is maintained.

extension of the line simplification method *Grid-Gen* (Magalhães et al., 2014) , in which a new heuristic based on Visvalingam-Whyatt algorithm (Visvalingam and Whyatt, 1993) is included. The method uses a strategy of point ranking to determine the best points to be kept in the output line. Polyline points are ranked by its effective area, namely the area generated by a point and its neighbours, then a cut-off factor is applied for eliminating points from the ranked list, points with lower effective area values will produce less area displacement when eliminated from the line, thus causing less impact to the topology of the output.

2 RELATED WORKS

One alternative for performing map generalization uses a stochastic optimization technique (Ware et al., 2003). It carries out operations such as displacement, size exaggeration, deletion and size reduction in order to resolve conflicts that result from map scale reduction. It uses a trial position approach, for each of n discrete polygonal objects is assigned k candidate trial positions that represent the original, displaced, size exaggerated, deleted and size reduced states of the object, which results in a total of k^n distinct map configurations. It is expected that some of these con-

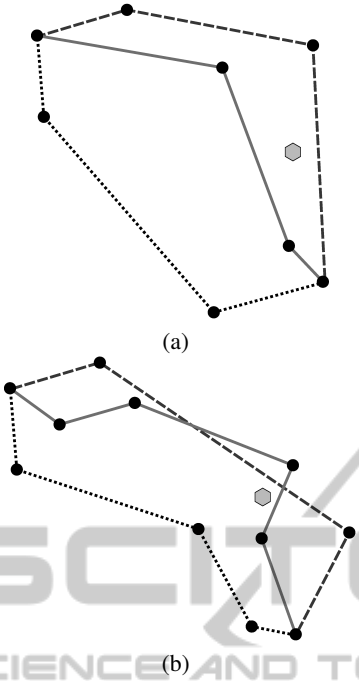

Figure 3: (a) Invalid simplification output, feature point P is placed in the wrong side of line L_2 after the process. ; (b) Intersection created between lines L_1 and L_2. Note that this intersection did not exist prior to the simplification, therefore it is inappropriate.

figurations will retain decreased level of map conflict. The goal is then to find the configuration with the lowest level of graphic conflict, which could be done by an exhaustive search, however, in practical situations this is not viable, i.e., for realistic values of n and k, an exhaustive search cannot provide a solution in efficient time. The method utilizes a simulated annealing strategy (Kirkpatrick et al., 1983) to overcome the exhaustive search issues. The algorithm has been successful in reducing map conflict within reasonable time. Nevertheless, since it makes use of a limited number of k pre-defined trial positions, it may provide inappropriate solutions (Ware et al., 2003).

Ramer-Douglas-Peucker's algorithm has the issue of outputting polylines with inconsistent or incorrect topological relationships. An approach proposed by Saalfeld to polyline simplification using the Douglas-Peucker algorithm attempts to solve this problem by adding tests to the stopping conditions of the original method which guarantee the consistency and correctness of the resulting features (Saalfeld, 1999). Its main idea is to generate polylines with Douglas-Peucker and try to refine it by adding points to the resulting curve (polyline) such that the curve does not present any topological inconsistency, notice, however, that adding points to the curve may erase previous topological inconsistency and create new ones. It is also proved that the search for points to be added to the polyline can be restricted to just a part of the segment (Saalfeld, 1999).

Although the previous method handles topological inconsistencies generated by the Douglas-Peucker algorithm it does not consider two topological configurations that may be undesirable: coincidence topology, which consists of the overlapping of two polylines or the overlapping of a feature point and a polyline, and the incidence topology, which concerns the incidence of a polyline point on another polyline p, but not on a vertex of p. Coincidence and incidence topologies are unwanted because they can generate redundancies on a map, such as slivers and gaps (Da Silva and Wu, 2005). A method to handle these configurations work as an expansion of Saalfeld's method, by determining *essential vertices* and adding a proximity criterion to the simplification process, the method eliminates coincidence and incidence from the results.

Another Douglas-Peucker based approach proposed by Li et al. (Li et al., 2013) intends to avoid topological inconsistencies as well as cracks on polygon shapes. Its strategy is based on detection-point identification, which are points lying within a minimum boundary rectangle (MBR) of the bounded face formed by a subpolyline and its corresponding simplifying segment. These detection-points are used for consistency verification of the simplification process. The detection-point identification process starts by getting the MBR $M_{i,j}$, of the bounded face formed by point $P_{i,j}$ and its corresponding simplifying line segment $e_{i,j}$. The next step is to identify features whose MBRs intersect $M_{i,j}$, then, from vertices of the exterior and interior boundaries of such features, identify those lying within $M_{i,j}$ and add such vertices to an output detection-point set R. And improvement to this process is made to avoid unnecessary points arising due to shared edges, that is, edges that belong to more than one line. However, the problem of polyline simplification does not generate cracks, since it works with multiple polygonal chains, and not with polygons which contain shared edges, therefore, the proposed method does not have to handle these particular situations.

A method proposed by Estkowski and Mitchell (Estkowski and Mitchell, 2001) performs map simplification of polygonal subdivisions by using a detour heuristic to prevent intersections of lines, which can result from another simplification method, such as Douglas-Peucker. It first calculates the simplification of an input S using the Douglas-Peucker algorithm, then it calculates all pairs of intersecting lines placing them in a separate list L. Thereafter, given two

intersecting lines s and s' in the list L, each described by a pair of endpoints v_1, v_2 and v'_1, v'_2 respectively, the idea is to search a graph, called detour graph, for a shortest path connecting the endpoints of s (or s'). If this path exists, the vertices are added to the original line in order to eliminate the intersection. Otherwise, it carries out a splitting process which consists of choosing a random vertex u and separating line s into two segments v_1u and uv_2, this is repeated until a solution is found within a certain error bound.

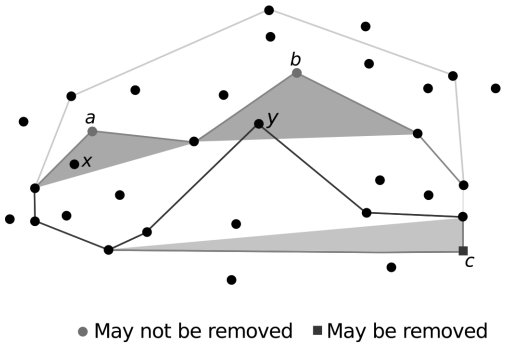

Figure 4: Determining if the deletion of some points would change the map topology.

3 THE PROPOSED METHOD

3.1 The Grid-Gen Heuristic

Given a set of control points C and an input map M composed of a set P of polylines, the heuristic simplifies M by iteractively processing each polyline independently. When a polyline is processed, *Grid-Gen* (Magalhães et al., 2014) 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 relationships 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 4 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 formed by polyline point a and its two adjacent points. If a polyline is simplified by removing a, then the topological relationship between the curve and x will change. Point b also cannot be removed since polyline point y is inside the triangle containing b as vertex and, thus, the deletion of b would change the topological relation between b's polyline and y's polyline (the two polylines would cross if b was removed). Therefore, neither a nor b could be removed from the current map. On the other hand, point c can be removed from the map without changing its topology since there is no control or polyline point inside the triangle generated by vertex c.

Algorithm 1 shows a pseudo-code for the method. Notice that, if no point is removed during one iteration, in the successive iterations no point will be removed either, because the map in the next iteration will not differ from the map in the previous one, therefore, the method can be terminated.

In some situations, Algorithm 1 may create maps with invalid topology. If a polyline p has coincident endpoints and the polygon (or *island*) defined by this polyline does not contain any control points or polylines in its interior, then applying Algorithm 1 may remove all points from p, thus, creating an invalid polygon.

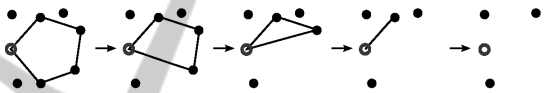

Figure 5: Simplification of a polyline with coincident endpoints and no control points or other polyline in the interior of the polygon that it defines.

In addition, if two polylines p_1 and p_2 have coincident endpoints, and the polygon generated by them does not contain control points or polylines in its interior, Algorithm 1 may remove all interior points from p_1 and p_2, creating two coincident line segments, as shown by Figure 6.

Figure 6: Simplification of two polylines with coincident endpoints and no control points or polylines in the interior of the polygon defined by them.

To solve these two problems, *Grid-Gen* 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, two dummy control points are added at an infinitesimal distance around one of the line segments that forms p. See an example in Figure 7. 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

Algorithm 1: Pseudo-code for the Grid-Gen algorithm.

```
1: np2rem ← Number of points to remove
2: npar ← 0 //Number of points already removed from the map
3: while npar < np2rem do
4:     for each polyline L in the input map do
5:         for each interior point v_i of L do
6:             if colinear(v_{i-1}, v_i, v_{i+1}) OR ∄ polyline point/control point in a triangle v_{i-1}, v_i, v_{i+1} then
7:                 npar ← npar + 1
8:                 Remove v_i from L
9:                 if npar > np2rem then Stop the algorithm
10:                end if
11:            end if
12:        end for
13:    end for
14: end while
```

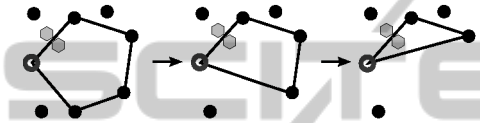

Figure 7: Use of dummy points (hexagon in orange) to avoid invalid simplifications: note that, in this dataset, the process terminates after removing the second interior point of the polyline since a dummy control point is inside the triangle defined by the resulting polyline.

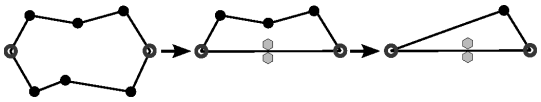

Figure 8: Use of dummy control points (hexagon in orange) to avoid invalid simplifications.

of p. By using this strategy, Algorithm 1 will never generate invalid simplifications as it would in changes in the topological relationships between the polylines and the *dummy* points.

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 8 presents an example where all interior points of a polyline p are removed and, then, two *dummy* points are added to the map.

Another concern is about the special case in which some line segments are colinear to the endpoints of a polyline. For instance, in Figure 9 there are no control points or polylines within the triangle v_i, v_i, v_{i+1} and, therefore, Algorithm 1 could remove vertex v_i, creating coinciding segments. This special case is treated by considering every point along the edges of the triangle v_{i-1}, v_i, v_{i+1} are inside triangle, thus not removing v_i.

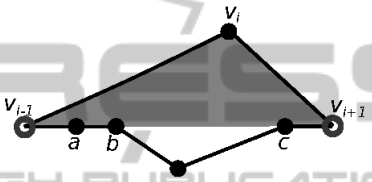

Figure 9: Example of polylines with the same endpoints: if v_i is removed, the resulting polyline will have segments coinciding with segments of the other polyline. To solve that, it is considered that points on the edges of the triangle v_{i-1}, v_i, v_{i+1} (points a, b and c) are effectively inside the triangle.

The bottleneck of *Grid-Gen* is to detect if a polyline or control point lies inside a triangle. To speedup this step, a uniform grid (Franklin et al., 1989) is used. 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 10 presents an example of an uniform grid superimposed on a map.

3.2 Extending Grid-Gen

As explained in section 3.1, *Grid-Gen* iteratively processes the input map removing interior points from the polylines that would not cause a topology inconsistency. Although this strategy can efficiently simplifies a map avoiding topological inconsistency, it does not try to keep the simplified map geometry similar to the original one.

The *TopoVW* is an extension of *Grid-Gen* that ranks the interior polyline points based on their "rele-

Algorithm 2: Pseudo-code for the Visvalingam-Whyatt generalization method.

```
Data: Input line L as a list of points, separate list R of ranked points
Compute the effective area of each point on the line
Delete all points with zero area and store them in a separate list
loop
    Find the point with least effective area and call it current point.
    Delete the current point from the original list L and add it to the ranked list R with its effective area
    Recalculate the effective area of the two adjacent points
    if Size of L = 2 then Terminate
    end if
end loop
```

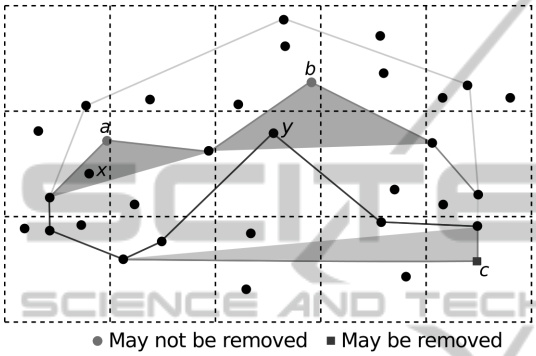

Figure 10: Example of a 3×5 uniform grid superimposed over a map.

vance" to the map shape and performes the map simplification iteratively removing the least "relevant" point. More precisely, the points are ranked using a strategy similar to Visvalingam-Whyatt (Visvalingam and Whyatt, 1993) algorithm's and, a simplification process similar to *Grid-Gen* is performed in the map, processing the points in an order based on their rank.

3.2.1 Visvalingam-Whyatt Algorithm

A method for line generalization proposed by Visvalingam and Whyatt (Visvalingam and Whyatt, 1993) uses the *effective area* of a point to define the point elimination sequence. This method is suitable for both minimal generalization and caricatural generalization. The *Effective area* of a point v_i is defined as the area of the triangle formed by v_i and its neighbours, that is, the triangle whose vertices are v_{i-1}, v_i and v_{i+1}. This represents the areal displacement created if point v_i is removed from the line. The algorithm for Visvalingam-Whyatt method is as follows (Algorithm 2):

As seen in Algorithm 2, it removes points from an input line L and adds these points to a list R ranked by the effective area of each point. This can be used to only include the most important points in the resulting line, as determined by a cut-off factor. This method has shown better results than other simplification algorithms such as the ones proposed by Ramer-Douglas-Peucker (Douglas and Peucker, 1973), Roberge (Robergé, 1985), and Dettori and Falcidieno (Dettori and Falcidieno, 1982).

3.2.2 The Method TopoVW

While the ranking point 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. The method *TopoVW* consists of integrating both strategies.

Given a polyline point v_i, the rank of v_i is defined based on the *effective area* of v_i. As shown by Visvalingam-Whyatt (Visvalingam and Whyatt, 1993), points with higher effective areas are usually "more relevant" than points with smaller areas for preserving the original shape and, thus, during the simplification step, points with smaller effective areas have higher priority for deletion. In order to preserve topological consistency, the simplifications step uses an approach similar to *Grid-Gen*˜s for point deletion.

For efficiency purposes, *TopoVW* initially preprocesses 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*. Since the interior points of all polylines are kept in the same priority queue, *TopoVW* simplifies the polylines globally, that is, in different iterations points from different polylines may be removed. When a point v_i is deleted from the map, this may change only the *effective area* of its two adjacent points v_{i-1} and v_{i+1} (in L's polyline) and, thus, these two areas are recomputed and the new values are used to update the priority of v_{i-1} and v_{i+1} in the queue.

4 EXPERIMENTAL EVALUATION

TopoVW was tested on a computer with the following configuration: Intel® Core™ 2 Duo E7500 at

1596 MHz, 4GB of RAM, Seagate ST2000DM001-1CH1 7200 RPM SATA hard-drive, and Linux Mint 17 Qiana.

The tests were performed on the same datasets previously used by Magalhães et al. (Magalhães et al., 2014): the first 5 datasets were used by the ACM GIS-CUP 2014 (ACM SIGSPATIAL Cup, 2014) organizers to evaluate the algorithms submitted to the GIS-CUP contest. Since datasets 1 and 2 were too small (they contained less than 2000 polyline points and less than 200 control points), test results for both datasets were not included in this paper.

The polygons in dataset 6 were obtained from *IBGE*'s (the Brazilian geography agency) website (IBGE: Instituto Brasileiro de Geografia e Estatistica, 2014) and represents the Brazilian county divisions. Dataset 7 represents the United States continental county division and was obtained from the United States Census website (United States Census, 2014). The control points in datasets 6 and 7 were selected in random positions in the maps. Table 1 presents the number of control and polyline points in each map.

Comparisons between the processing times of *TopoVW* and *Grid-Gen* were performed. 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 (Magalhães et al., 2014). Table 1 presents the time that the two methods spent in the simplification process and, also, the I/O time to load the map (in the GIS-CUP GML format) and write the output (the I/O time is the same to the two implementations).

Observe that, in the worst case, the simplification step of *TopoVW* was 7 times slower than the same step in *Grid-Gen*. However, in all scenarios I/O used a considerable amount of the total processing time. Indeed, if the total running time of the algorithms was considered, *TopoVW* was less than 3 times slower than *Grid-Gen* in the worst test.

Table 1: Processing-time (in milliseconds) for simplifying the datasets with control points.

Dataset	3	4	5	6	7
Polyline pts. ($\times 10^3$)	8	30	30	300	4000
Control points	151	256	1607	10^4	10^7
I/O	12	37	53	447	89187
Simp. (*TopoVW*)	9	52	46	1469	95732
Simp. (*Grid-Gen*)	4	16	15	196	19158

Figure 11(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 the top layer. It is easy to see that *TopoVW* maintained the map shape better than does *Grid-Gen*. Indeed, it is difficult to see in this figure regions where the dashed polylines that represents the map simplified by *TopoVW* are visible. Figure 11(b) displays a zoomed region from the map in Figure 11(a) where it is possible to observe the difference between the three maps. Notice that *TopoVW*'s output keeps the similarity with the original map better than does *Grid-Gen*. It is important to mention that, even though both methods were configured to remove the same amount of points from the map, *Grid-Gen* processes each polyline independently trying to remove the maximum amount of points from

Table 2: Processing-time (in milliseconds) for simplifying the datasets without using control points.

Dataset	3	4	5	6	7
Control points	151	256	1607	10^4	10^7
I/O	9	32	37	490	3856
Simp. (*TopoVW*)	8	49	42	1400	89439
Simp. (*Grid-Gen*)	3	14	12	195	16892
Simp. (VW)	6	24	23	900	21062

(a)

(b)

Figure 11: (a) Region of a simplified map representing part of dataset 3. The polylines in solid, dotted and dashed represent, respectively, the original map and the maps simplified by Grid-Gen and TopoVW (both methods were configured to remove 50% of the polyline points); (b) Zoomed region from the map.

Table 3: Maximum number of points removed by simplification.

Dataset	1	2	3	4	5	6	7
No. of points	992	1564	8531	28014	28323	342738	3645559
Visvalingam-Whyatt	938	1472	7579	25308	23661	309648	3627135
Grid-Gen	938	1463	7578	25298	23639	309445	3627008
TopoVW	938	1463	7578	25297	23640	309424	3626993
Grid-Gen w/ Control Pts.	928	1435	7545	25212	23411	308992	3609484
TopoVW w/ Control Pts.	927	1430	7545	25207	23390	308935	3627008

each one (until the total number of points removed reach the target value) and, thus, some polylines may be too simplified (while no points are removed from other lines). *TopoVW*, on the other hand, performs the simplification globally (removing, in each step, the less significant point considering all the polylines) and, therefore, the balance in the number of points removed from each polyline is better.

We also compared the processing time of *TopoVW*, *Grid-Gen*, and the Visvalingam-Whyatt algorithm (*VW*) when they were configured to remove 50% of the map points. The times, in milliseconds, are presented in Table 2. Since *VW* does not support the use of control points, the control points from the datasets were removed during these tests. In the worst test scenario, the simplification time of *TopoVW* was only 4 times slower than *VW*. This suggests that the overhead associated with the topological verifications performed by *TopoVW* does not add a big impact in the performance of the algorithm.

Another comparison was made between a Visvalingam-Whyatt algorithm implementation (*VW*), *Grid-Gen* and *TopoVW* regarding the maximum amount of points that these methods can remove due to the simplification.

The three methods were used to simplify maps from the dataset previously used, attempting to remove the maximum amount of points from the original maps. Additionally, both *Grid-Gen* and *TopoVW* were tested with and without control points. The results are shown in Table 3. In no test case *TopoVW* removed more points than *VW*. This happened because *TopoVW* preserves the topological consistency of the map and, thus, some points that the *VW* algorithm removes may not be removed by *TopoVW*. Indeed, since *VW* does not try to satisfy the topological constraints it removes all the interior points from the polylines (keeping only the endpoints) when it is configured to remove the maximum amount of points.

Furthermore, when the methods were tested using control points, the number of points removed by *TopoVW* was smaller than when using no control points. Again, this happens because when there are control points the amount of topological constraints that *TopoVW* needs to satisfy increases.

Figure 12 presents a comparison between the output generated by *VW* (represented by solid lines, in the top layer) and *TopoVW*'s output (represented by dashed lines) when both methods were configured to remove 50% of the polylines points in dataset 6 (since *VW* does not support the use of control points, both methods were executed using no control points). Notice that it is difficult to observe differences between the outputs of the two methods. When they happen, these differences (shown in the zoomed part of the map) are caused due to the fact that *VW* does not avoid topological errors.

In all datasets, we observed few (and small) differences between *VW* and *TopoVW*'s output. These differences appear mainly in the regions where *VW*'s output contains topological errors. Since the *VW* algorithm is suitable for computing map minimal generalization and caricatural generalization with good quality (Visvalingam and Whyatt, 1993) and *TopoVW*'s output is similar to *VW*'s, this suggests that *TopoVW* can also compute these kinds of generalization with good quality while having the advantage of not creating topological errors.

5 CONCLUSIONS AND FUTURE WORKS

It was presented *TopoVW*, a heuristic that uses techniques based on the Visvalingam-Whyatt (*VW*) (Visvalingam and Whyatt, 1993) algorithm to perform map simplification generating maps that not only are topologically consistent but also tries to preserve the similarity with the original datasets. This means that, if applied to a map containing roads, buildings and rivers, *TopoVW*ś simplification will neither produce intersections of such features nor change the topological relationships (for example, if a road is in the left side of a building it will continue in the left side after the simplification). *TopoVW* also supports the use of control points in the generalization process, avoiding changing the topological relationships between the polylines and these control points. Control points may be used in practical applications such as simpli-

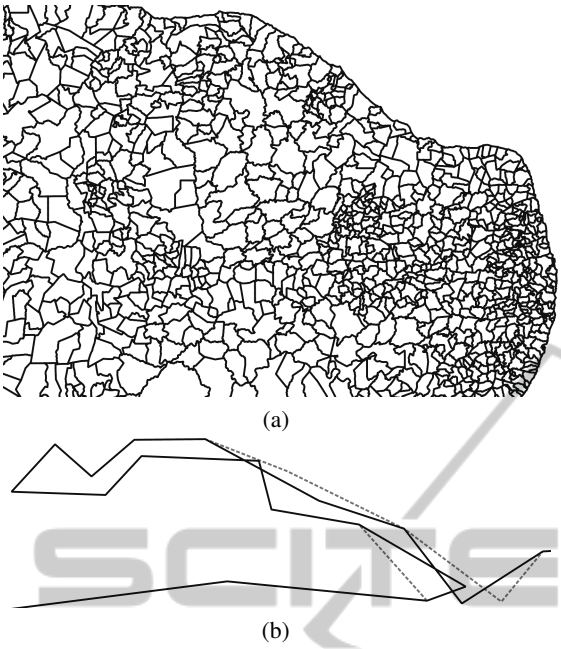

Figure 12: Comparison between the output generated by VW (represented by solid lines, in the top layer) and TopoVW's output (represented by dashed lines) using dataset 6 as input: (a) Part of the Northeast of Brazil. (b) Zoom in a region where it is possible to see a small difference between the outputs.

fying state/county maps, where lines represent county borders and cities may be represented by points (more specifically, control points). If *TopoVW* is applied to this situation, it is guaranteed that no point will be on the wrong side of the lines. Namely, after simplification, every city will stay inside the correct county or state.

During the experiments, we observed that *TopoVW*'s output is very similar to *VW*'s. In fact, the small differences that we observed between the methods happened in regions were *VW* created topological errors. Since *VW* is suitable for creating map generalizations and caricatures with good quality, this suggests that *TopoVW* also has this feature.

As the tests showed, *TopoVW* is very efficient, being only 7 times slower than *Grid-Gen* a method that avoids topological errors during the simplification but does not try to keep the similarity between the input and the output maps. Furthermore, it adds little overhead to the *VW* algorithm (an algorithm that, despites keeping the similarity between the input and output, does not try to preserve the topological relationships), being only 4 times slower than *VW*. Finally, *TopoVW* uses dummy control points to treat its special cases and, thus, it is very simple to implement.

Future work includes comparing *TopoVW*'s performance and output quality against methods based on the Ramer-Douglas-Peucker algorithm. Furthermore, another extension is to perform more tests in order to evaluate the quality of *TopoVW*'s output not only in maps, but also in vector illustrations. Finally, another interesting work is to adapt *TopoVW* to perform simplification of 3D vector objects.

ACKNOWLEDGEMENTS

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

REFERENCES

ACM SIGSPATIAL Cup (2014). GIS Cup sample data. mypages.iit.edu/∼xzhang22/GISCUP2014/ (accessed on 12/01/2014).

Da Silva, A. C. and Wu, S.-T. (2005). Preserving coincidence and incidence topologies in saalfeld's polyline simplification algorithm. In *Proceedings of Simposio Brasileiro de Geoinformatica (GeoInfo)*, pages 107–121. INPE.

Dettori, G. and Falcidieno, B. (1982). An algorithm for selecting main points on a line. *Computers & Geosciences*, 8(1):3–10.

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

Estkowski, R. and Mitchell, J. S. (2001). Simplifying a polygonal subdivision while keeping it simple. In *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 40–49. ACM.

Franklin, W. R., Narayanaswami, C., Kankanhalli, M., Sun, D., Zhou, M.-C., and Wu, P. Y. (1989). Uniform grids: A technique for intersection detection on serial and parallel machines. In *Proceedings of Auto-Carto*, volume 9, pages 100–109.

IBGE: Instituto Brasileiro de Geografia e Estatistica (2014). Malhas digitais dos municipios Brasileiros. geoftp.ibge.gov.br/malhas_digitais/municipio_2007/ (accessed on 12/01/2014).

João, E. (1998). *Causes and Consequences of map generalization*. CRC Press.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Li, L., Wang, Q., Zhang, X., and Wang, H. (2013). An algorithm for fast topological consistent simplification of face features. *Journal of Computational Information Systems*, 9(2):791–803.

Mackaness, W. A., Ruas, A., and Sarjakoski, L. T. (2011). *Generalisation of geographic information: cartographic modelling and applications*. Elsevier.

Magalhães, S. V. G., Franklin, W. R., Li, W., and Andrade, M. V. A. (2014). Fast map generalization heuristic with a uniform grid. In *ACM SIGSPATIAL*.

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

Robergé, J. (1985). A data reduction algorithm for planar curves. *Computer Vision, Graphics, and Image Processing*, 29(2):168–195.

Ruas, A. and Lagrange, J.-P. (1995). Data knowledge and modelling for generalization. *MULLER, JC, LAGRANGE, JP, WEIBEL, R. GIS and generalization: metodology and practice. GISDATA I, serie editors*, pages 73–90.

Saalfeld, A. (1999). Topologically consistent line simplification with the douglas-peucker algorithm. *Cartography and Geographic Information Science*, 26(1):7–18.

Shea, K. and McMaster, R. B. (1989). Cartographic generalization in a digital environment: When and how to generalize. In *Proceedings of AutoCarto*.

United States Census (2014). US counties Shapefiles. www.census.gov/cgi-bin/geo/shapefiles2013/main (accessed on 12/01/2014).

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

Ware, J. M., Jones, C. B., and Thomas, N. (2003). Automated map generalization with multiple operators: a simulated annealing approach. *International Journal of Geographical Information Science*, 17:743–769.