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

Salles Viana Gomes de Magalhães, PhD. Student
Prof. Dr. W Randolph Franklin, RPI/Supervisor
Wenli Li, PhD. Student
Prof. Dr. Marcus V. A. Andrade, UFV
Curve generalization

- Curve generalization: represent a polyline using fewer points.
- It is desirable to keep the lines “similar”.

```
input

(a)  (b)

outputs

(a)  (b)
```
Curve generalization

• There are well-known algorithms to solve this problem.
• Examples:
  • Douglas-Peucker:
    • Repeatedly adds points.
  • Visvalingam-Whyatt
    • Repeatedly removes points.
Map generalization

- Simplify polylines in a map.
- Remove points (except endpoints)
- Challenge: topological problems.

Source: http://mypages.iit.edu/~xzhang22/GISCUP2014
The problem

• Map generalization with control points.
  • Avoid topological changes in polylines & in control points.
  • Example: city cannot lie in the wrong state.

Source: http://mypages.iit.edu/~xzhang22/GISCUP2014
Our solution

- Grid-Gen (ACM GISCUP)
  - Process polylines independently.
  - Remove polyline point $\leftrightarrow$ no topological problem.
  - No topological problem $\leftrightarrow$ no point in triangle!
Our solution

- Special cases:
  - Coincident endpoints & no control point inside.
Our solution

• Special cases:
  • Coincident endpoints & no control point inside.

• Solution: dummy points.
Our solution

• Special cases:
  • Two polylines with the same endpoints & no control point inside.
Our solution

- Special cases:
  - Two polylines with the same endpoints & no control point inside.

- Also solved with dummy points.
Uniform grid

• For efficiency: uniform grid.
• Polylines points & control points → grid.
Grid-Gen2

- Grid-Gen: We only try to satisfy the constraints.
- Grid-Gen2:
  - Points ranked based on “effective area” (Visvalingam-Whyatt).
  - Remove first points with small “area”.

![Diagram showing points a, b, c and a triangle]
Grid-Gen2

- Grid-Gen: We only try to satisfy the constraints.
- Grid-Gen2:
  - Points ranked based on “effective area” (Visvalingam-Whyatt).
  - Remove first points with small “area”.
  - Areas of neighbors are updated.
  - For efficiency → priority queue.
Experimental results

- i7-3520M 3.6 GHz processor, 8GB of RAM memory
- Samsung 840 EVO SSD (500 GiB)
- Linux Mint 17
Experimental results

- i7-3520M 3.6 GHz processor, 8GB of RAM memory
- Samsung 840 EVO SSD (500 GiB)
- Linux Mint 17

- Goal: remove 50% of the points.
## Experimental results

- Grid-Gen vs Grid-Gen2
- Time (ms) for each step (only simplification is different).
- Bottleneck: **I/O** and simplification step.
- Simplification: Grid-Gen2 is 8 times slower.

<table>
<thead>
<tr>
<th>Dataset # input points</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></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>Unif. grid init.</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>24</td>
<td>1472</td>
</tr>
<tr>
<td>Simp. (<em>Grid-Gen2</em>)</td>
<td>2</td>
<td>15</td>
<td>13</td>
<td>435</td>
<td>23759</td>
</tr>
<tr>
<td>Simp. (<em>Grid-Gen</em>)</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>
Experimental results

- Example of solution (blue = original, red = Grid-Gen, green = Grid-Gen2)
Experimental results

- Example of solution (blue = original, red = Grid-Gen, green = Grid-Gen2)
Conclusions

- Grid-Gen and Grid-Gen2 are very fast simplification algorithms.
- Grid-Gen2 is only two times slower (total) than Grid-Gen.

- Future work:
  - Compare against other methods.
  - Study the uniform grid size.
An efficient map generalization heuristic based on the Visvalingam-Whyatt algorithm

Thank you!

Acknowledgement:

Contact:
Salles Viana Gomes de Magalhães, vianas2@rpi.edu
W Randolph Franklin, mail@wrfranklin.org
Wenli Li, liw9@rpi.edu
Marcus V. A. Andrade, marcus@ufv.br