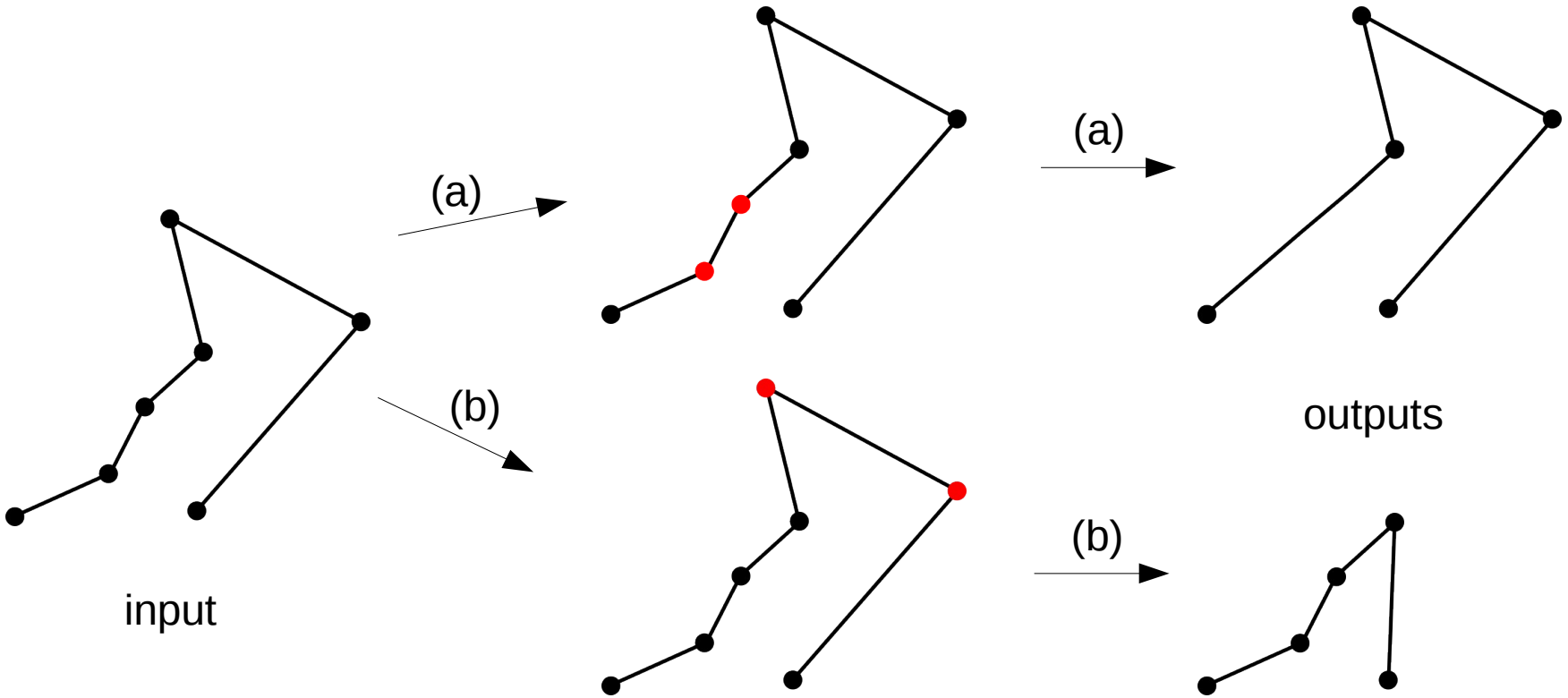
Rensselaer Polytechnic Institute
Universidade Federal de Viçosa

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”.

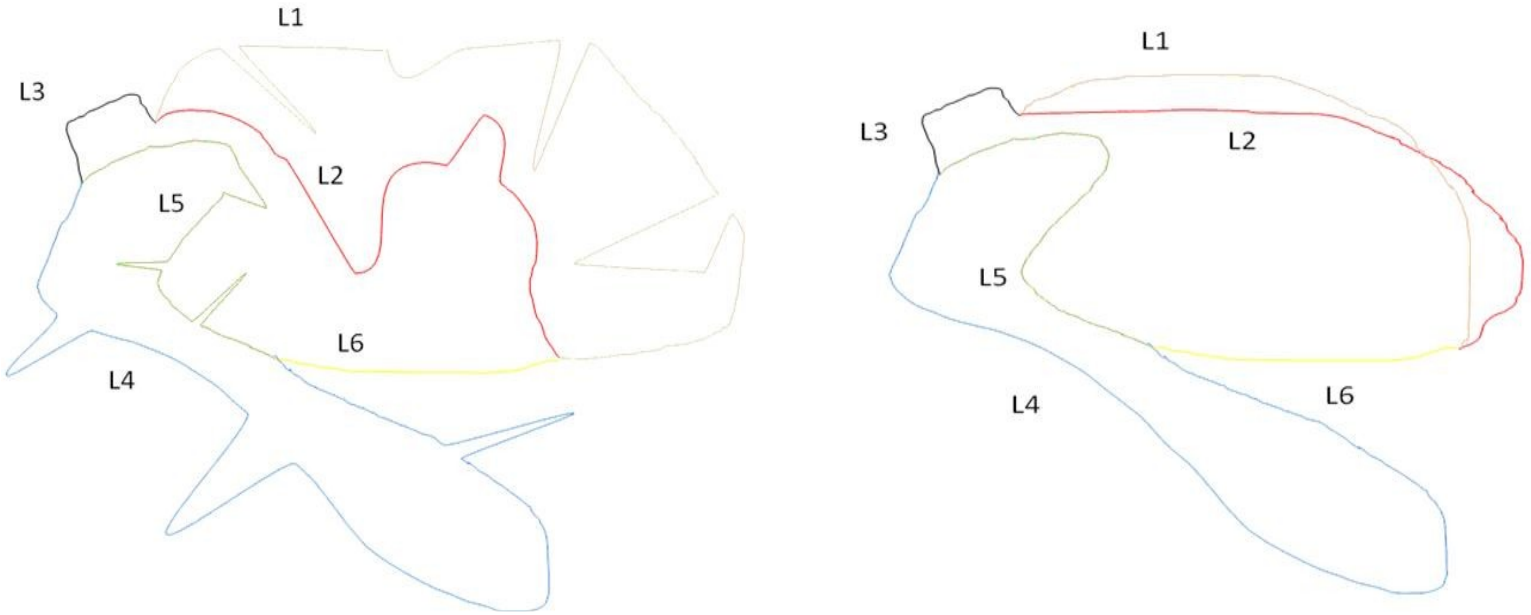


Curve generalization

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

Map generalization

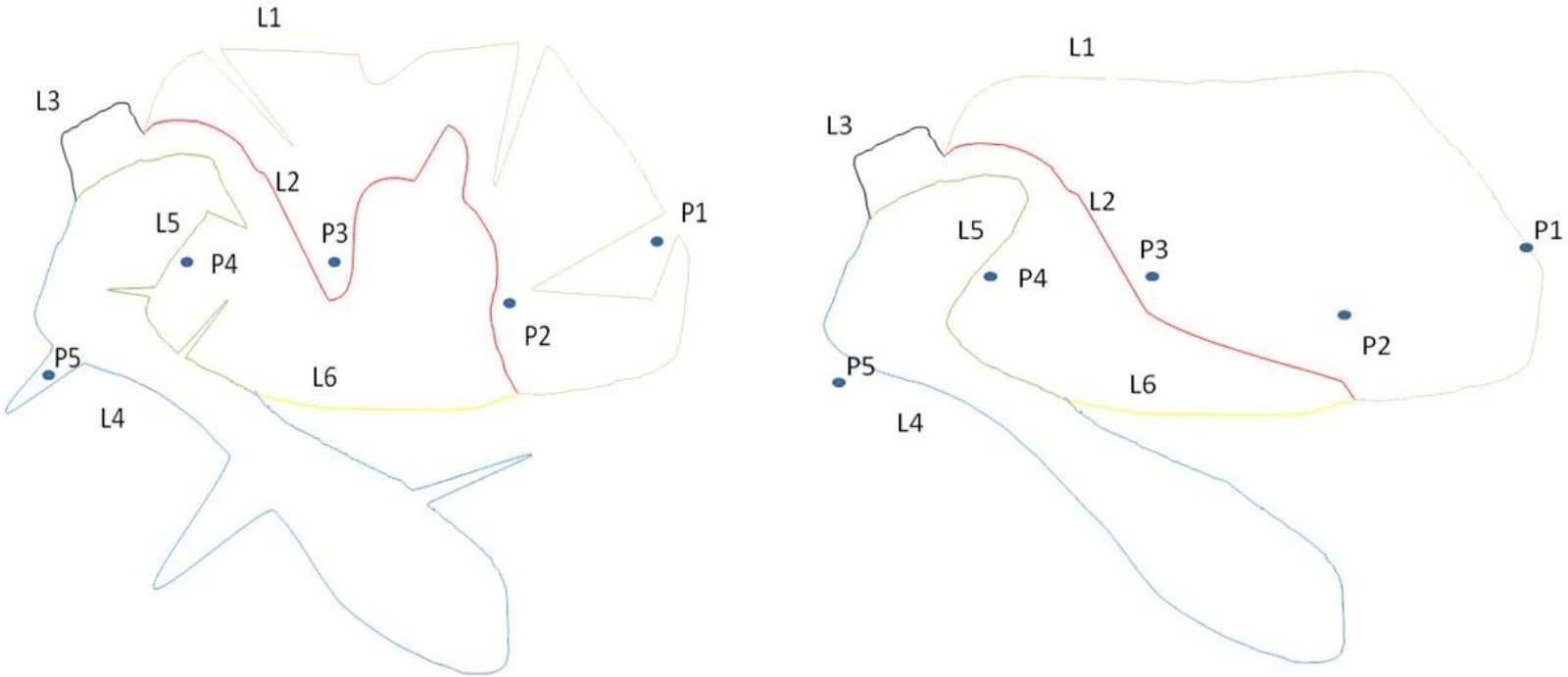
- Simplify polylines in a map.
- Remove points (except endpoints)
- Challenge: topological problems.



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

The problem

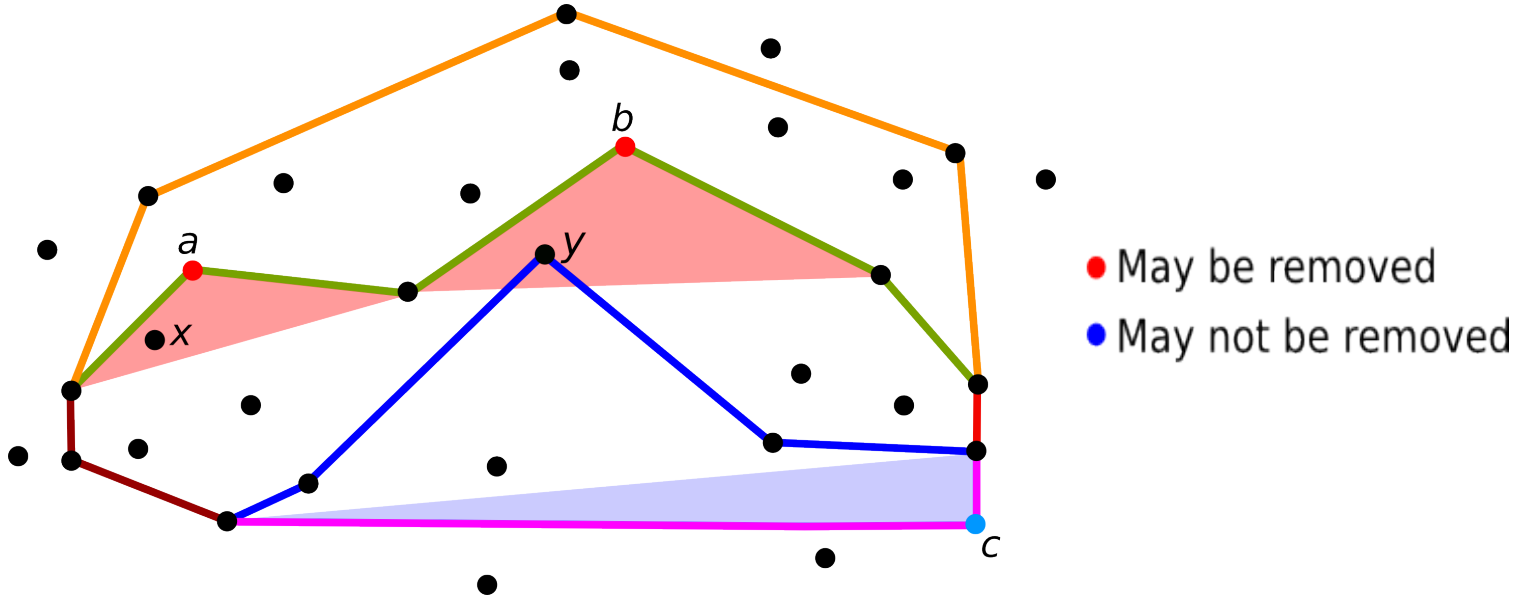
- Map generalization with control points.
  - Avoid topological changes in polylines & in control points.
- Example: city cannot lie in the wrong state.



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

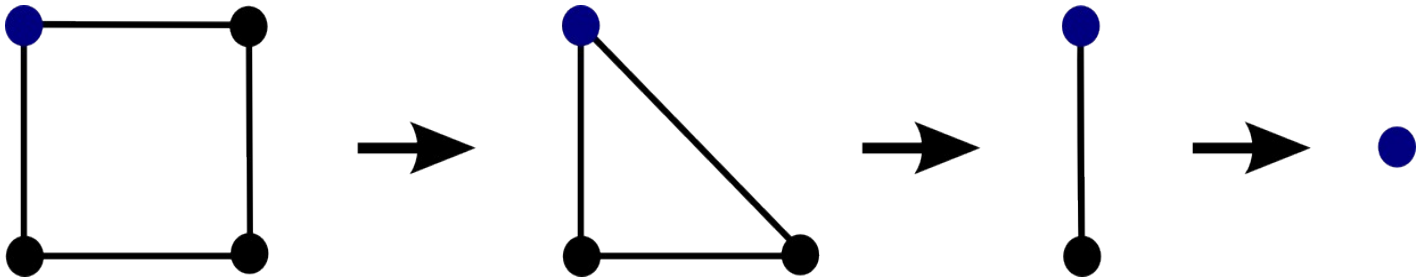
Our solution

- Grid-Gen (ACM GISCUP)
  - Process polylines independently.
  - Remove polyline point ↔ no topological problem.
  - No topological problem ↔ no point in triangle!



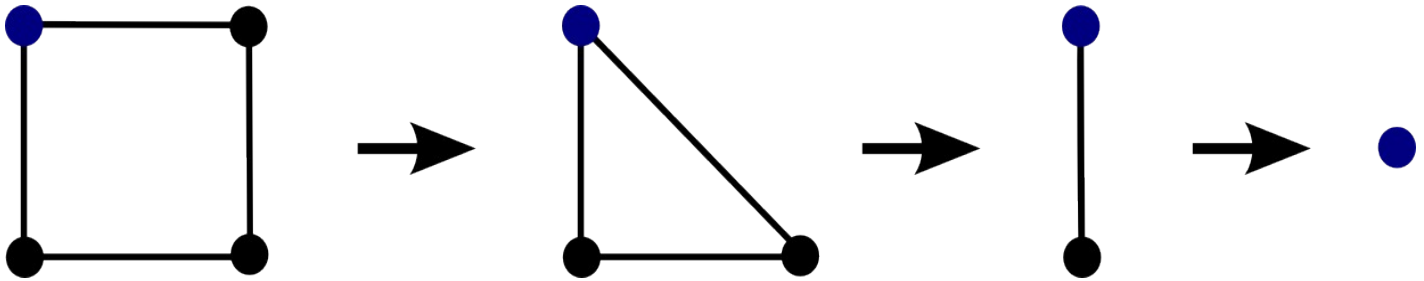
Our solution

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

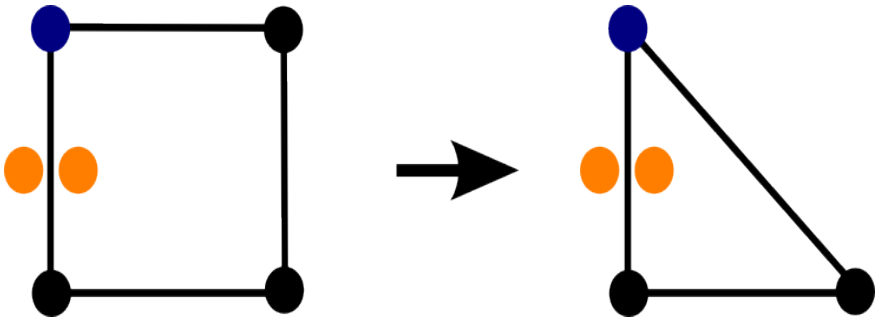


Our solution

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

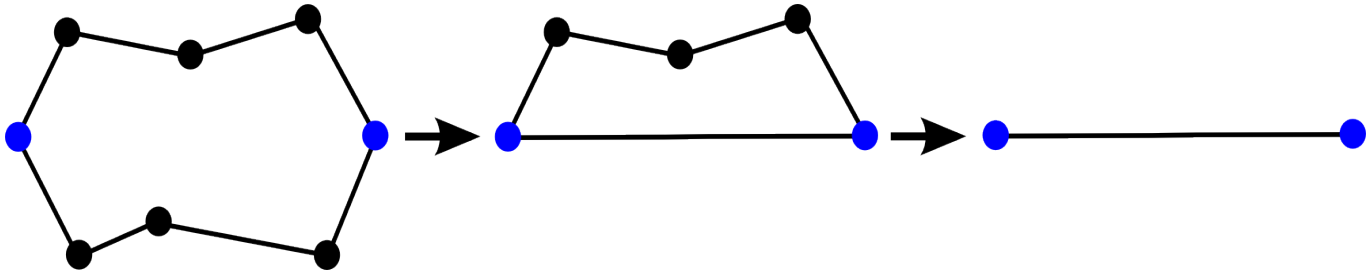


- Solution: dummy points.



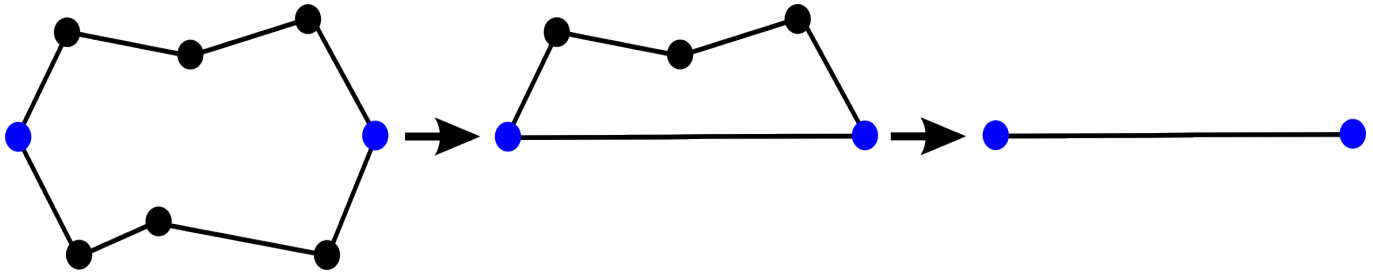
Our solution

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

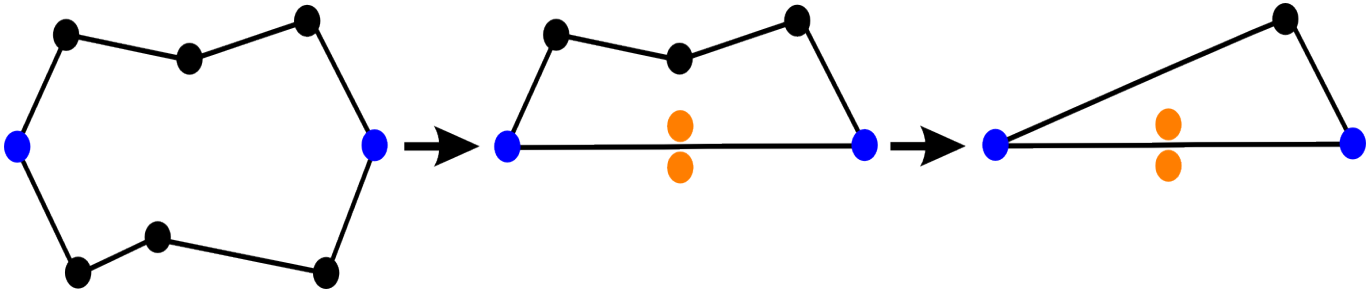


Our solution

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

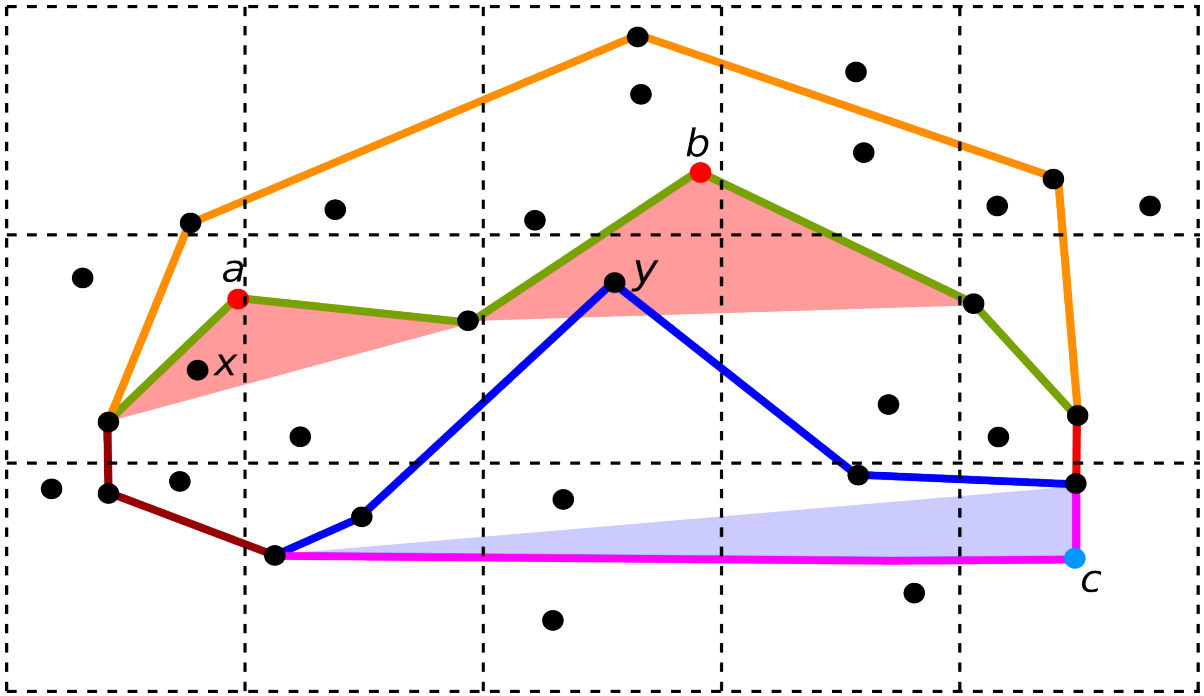


- Also solved with dummy points.



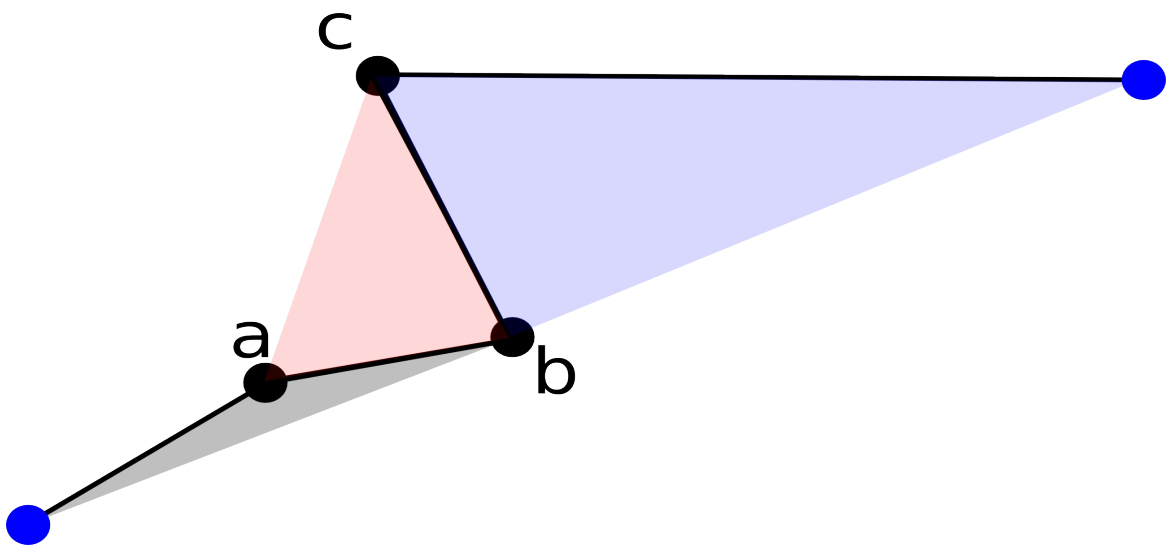
Uniform grid

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



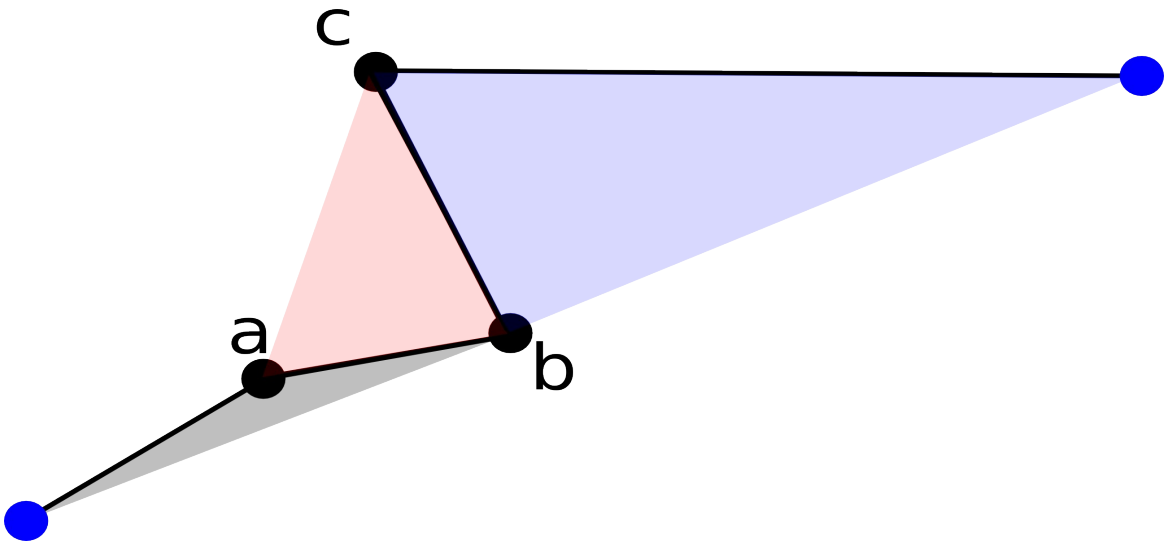
Grid-Gen2

- 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”.



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.

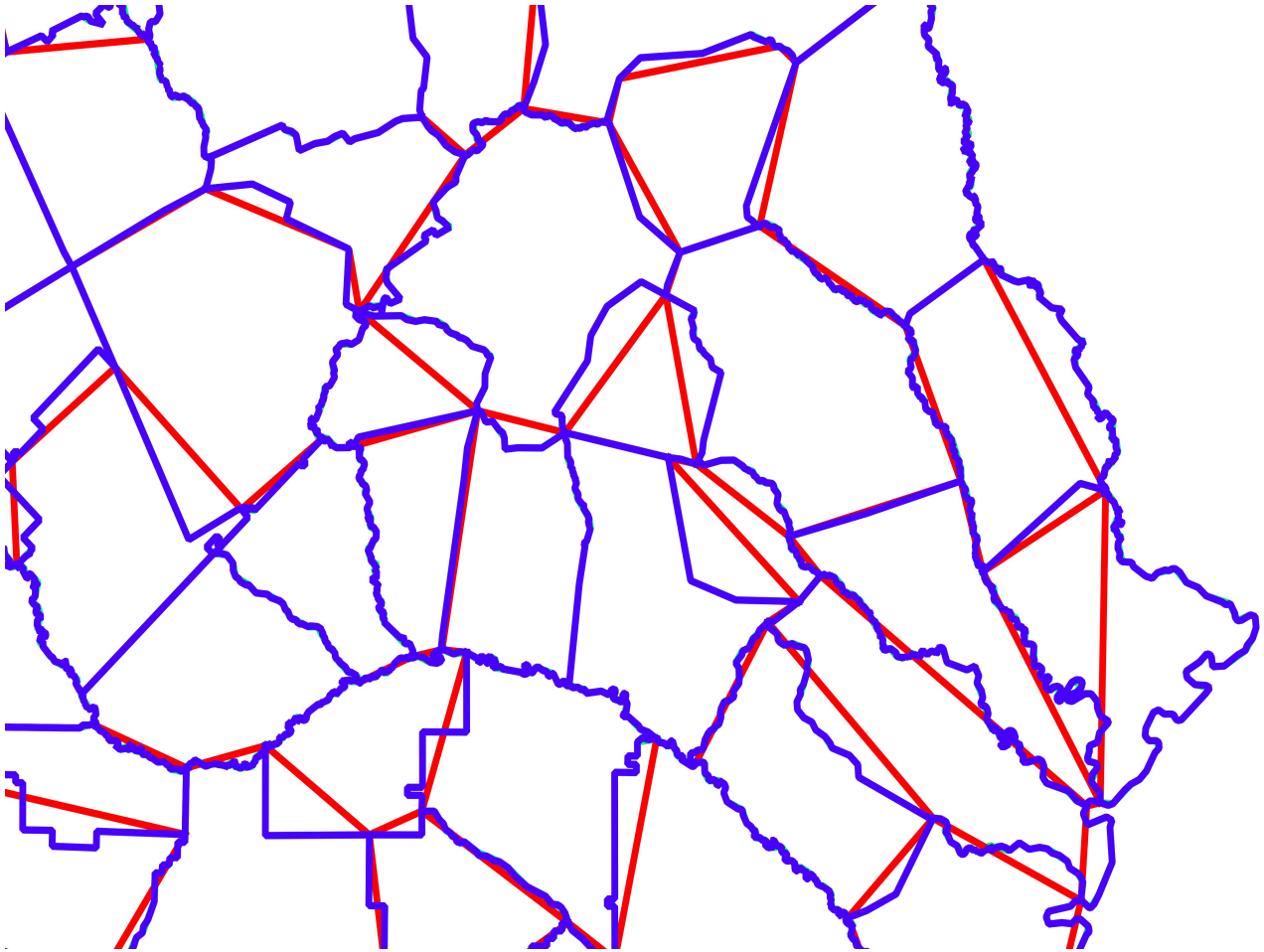
Experimental results

- 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.

Dataset	3	4	5	6	7
# input points	8531	3×10^4	3×10^4	3×10^5	4×10^6
Input reading	10	22	29	257	37092
Unif. grid init.	0	1	1	24	1472
Simp. (*Grid-Gen2*)	2	15	13	435	23759
Simp. (*Grid-Gen*)	1	4	3	54	3481
Output writing	6	21	21	170	1817

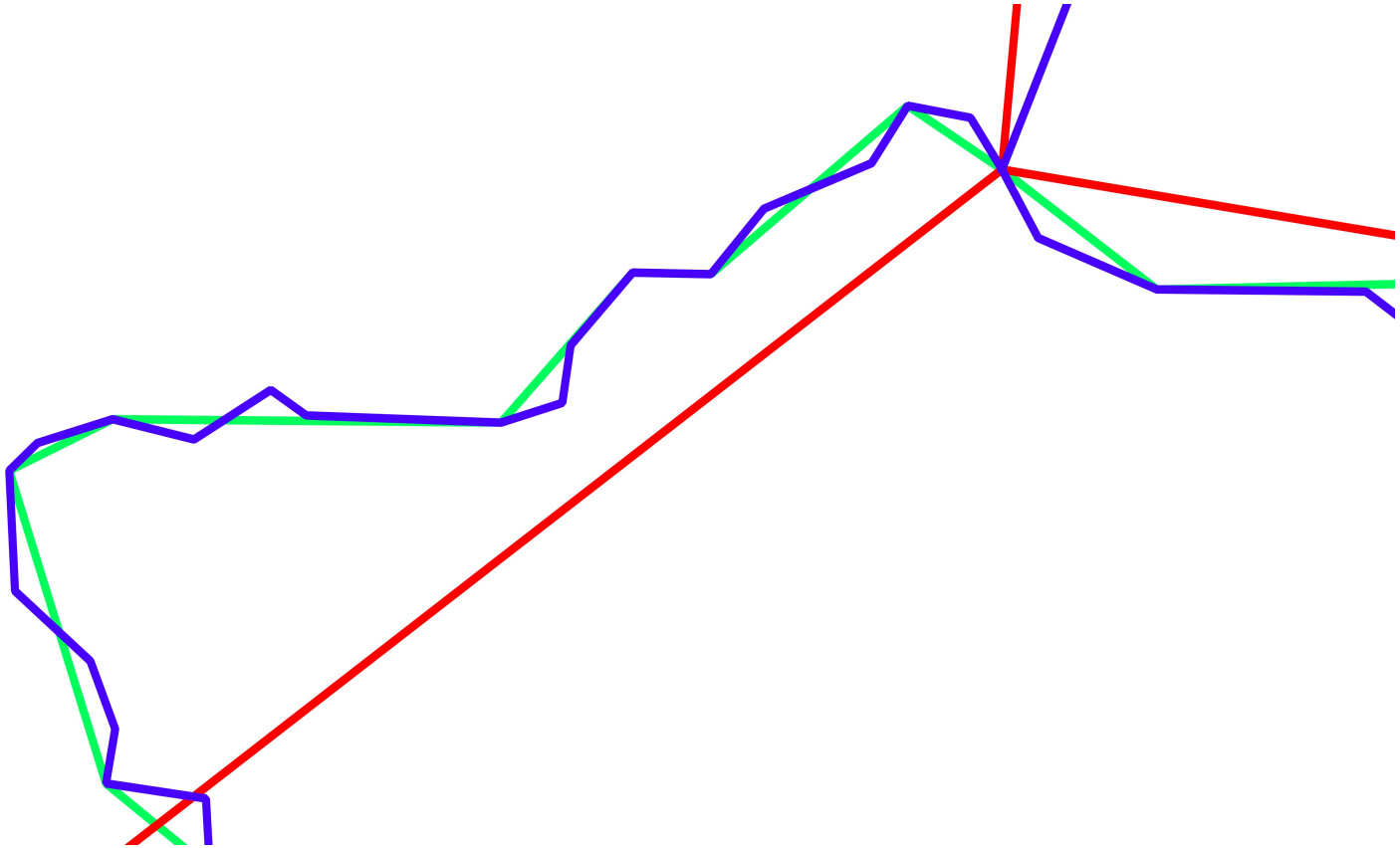
Experimental results

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



Experimental results

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

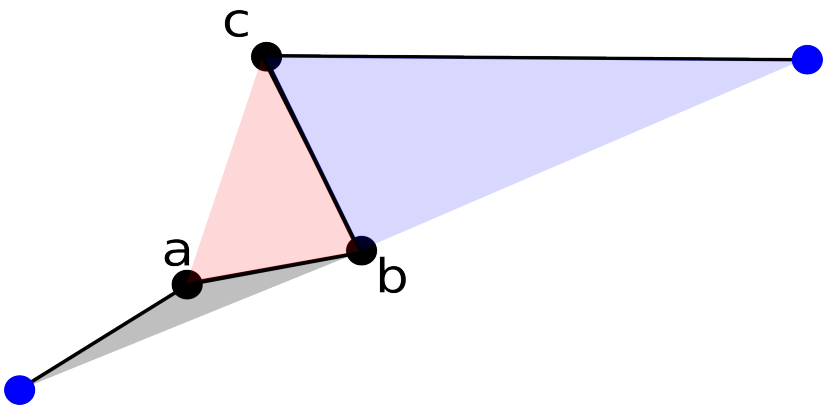


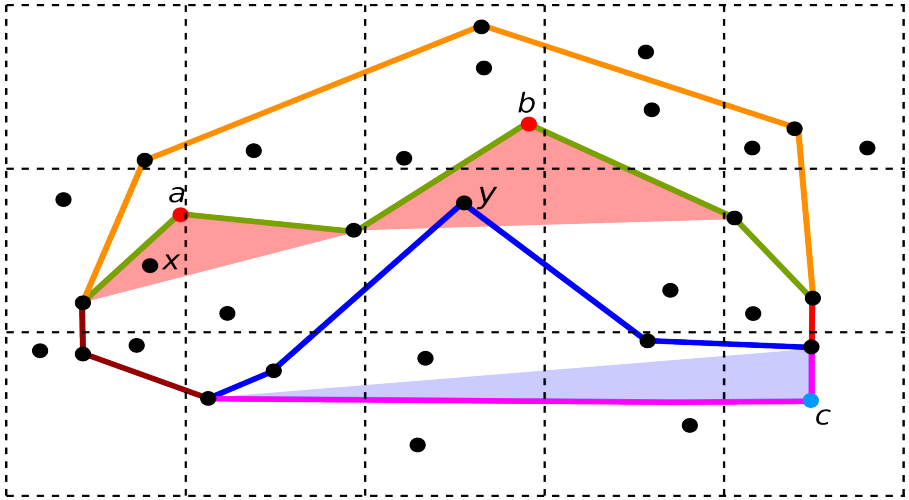
Conclusions

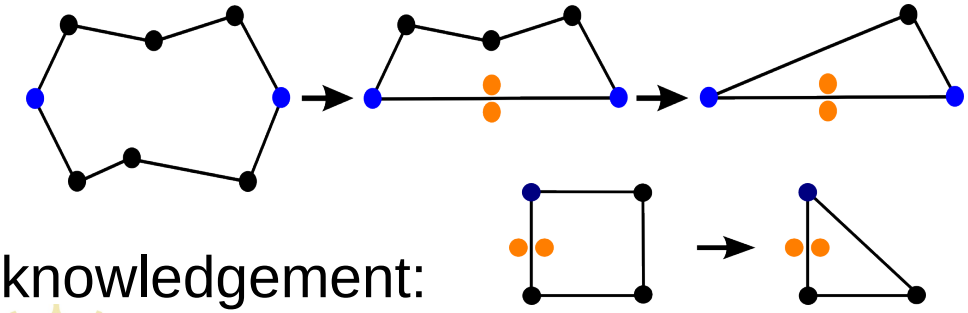
- Grid-Gen and Grid-Gen2 are very fast simplification algorithms.
- Grid-Gen2 is only two times slower (total) than Grid-Gen.

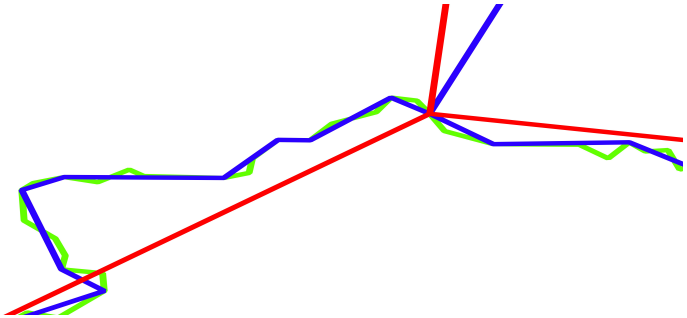
- Future work:
  - Compare against other methods.
  - Study the uniform grid size.

Thank you!









Acknowledgement:

NSF

CAPES

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