Progressive Simplification of Polygonal Curves

Kevin Buchin
Maximilian Konzack
Wim Reddingius
Curve Simplification
min-# Simplification problem:
- Given a polygonal curve $C$ and an error threshold $\varepsilon > 0$
- Objective: minimize the number of vertices in a simplification $S$
min-# Simplification problem:

- Given a polygonal curve $C$ and an error threshold $\epsilon > 0$
- Objective: minimize the number of vertices in a simplification $S$
min-\# Simplification problem:

- Given a polygonal curve $C$ and an $\varepsilon > 0$ as an error threshold
Curve Simplification

Upper bound [Chan and Chin, 1996]
A min-# simplification can be computed in $O(n^2)$ time in $\mathbb{R}^2$

Higher dimensions [Barequet et al., 2002]
For the $L_1$ or $L_\infty$ metric, a min-# simplification can be computed in $O(n^2)$ time
Progressive Simplification

$S_1$
Progressive Simplification

Zoom out

$S_1$
Progressive Simplification

\[ S_2 \]

\[ S_1 \]
Progressive Simplification

$S_3$

$S_2$

$S_1$

Zoom out
Progressive Simplification

$S_4$

$S_3$

$S_2$

$S_1$
Progressive Simplification

$S_4$

$S_3$

$S_2$

$S_1$
Progressive Simplification

Impose Consistency Across Many Scales

- Zoom in and out without flickering
- A sequence of $m$ scales: $0 < \varepsilon_1 < \cdots < \varepsilon_m$
- Require *monotonicity*: $S_m \sqsubseteq S_{m-1} \sqsubseteq \cdots \sqsubseteq C$
- Minimize $\sum_{k=1}^{m} |S_k|$ (*optimality*)
• An $O(n^3 m)$ time algorithm for the progressive simplification problem

• works with various distance measures such as Hausdorff, Fréchet and area-based distances

• enables simplification for continuous scaling in $O(n^5)$ time
Shortcut:
• Given a polygonal curve \( C \), a shortcut \((p_i, p_j)\) is an ordered pair of vertices

Validity:
• Given \( C \) and an \( \varepsilon > 0 \), \((p_i, p_j)\) is valid if \( \varepsilon(p_i, p_j) \leq \varepsilon \)
Shortcut Graph

$C$
• Given a curve $C$ and an $\varepsilon > 0$, the shortcut graph $G(C, \varepsilon)$ captures all valid shortcuts.
Given a curve $C$ and an $\varepsilon > 0$, the shortcut graph $G(C, \varepsilon)$ captures all valid shortcuts.

Minimum-link path in $G(C, \varepsilon)$ is an optimal simplification $S$. 
Dynamic Programming

- Assign a cost value $c_{i,j}^k \in \mathbb{N}$ for each shortcut $(p_i, p_j) \in G(C, \varepsilon_k)$ at scale $\varepsilon_k$

- $c_{i,j}^k$ relates to the cost of including $(p_i, p_j)$ in $S_k$
Dynamic Programming

- Assign a cost value $c_{i,j}^k \in \mathbb{N}$ for each shortcut $(p_i, p_j) \in G(C, \varepsilon_k)$ at scale $\varepsilon_k$
- $c_{i,j}^k$ relates to the cost of including $(p_i, p_j)$ in $S_k$
- Example: $\varepsilon_1 < \varepsilon_2 < \varepsilon_3$

![Diagram showing a graph with edges and numbers labeled 1]
Dynamic Programming

- Assign a cost value $c_{i,j}^k \in \mathbb{N}$ for each shortcut $(p_i, p_j) \in G(C, \varepsilon_k)$ at scale $\varepsilon_k$
- $c_{i,j}^k$ relates to the cost of including $(p_i, p_j)$ in $S_k$
- Example: $\varepsilon_1 < \varepsilon_2 < \varepsilon_3$

$G(C, \varepsilon_1)$

$p_1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad p_n$

$S_1$

$1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1$
Minimal Progressive Simplification

Dynamic Programming

- Assign a cost value $c_{i,j}^k \in \mathbb{N}$ for each shortcut $(p_i, p_j) \in G(C, \varepsilon_k)$ at scale $\varepsilon_k$

- $c_{i,j}^k$ relates to the cost of including $(p_i, p_j)$ in $S_k$

- Example: $\varepsilon_1 < \varepsilon_2 < \varepsilon_3$

$$G(C, \varepsilon_1) \subseteq G(C, \varepsilon_2)$$
Dynamic Programming

- Assign a cost value $c_{i,j}^k \in \mathbb{N}$ for each shortcut $(p_i, p_j) \in G(C, \varepsilon_k)$ at scale $\varepsilon_k$
- $c_{i,j}^k$ relates to the cost of including $(p_i, p_j)$ in $S_k$
- Example: $\varepsilon_1 < \varepsilon_2 < \varepsilon_3$

$G(C, \varepsilon_1) \subseteq G(C, \varepsilon_2)$
Minimal Progressive Simplification

Dynamic Programming

- Assign a cost value $c^k_{i,j} \in \mathbb{N}$ for each shortcut $(p_i, p_j) \in G(C, \varepsilon_k)$ at scale $\varepsilon_k$

- $c^k_{i,j}$ relates to the cost of including $(p_i, p_j)$ in $S_k$

- Example: $\varepsilon_1 < \varepsilon_2 < \varepsilon_3$

$$G(C, \varepsilon_1) \subseteq G(C, \varepsilon_2)$$

$$S_1 \equiv S_2$$
Dynamic Programming

- Assign a cost value $c_{i,j}^k \in \mathbb{N}$ for each shortcut $(p_i, p_j) \in G(C, \varepsilon_k)$ at scale $\varepsilon_k$
- $c_{i,j}^k$ relates to the cost of including $(p_i, p_j)$ in $S_k$
- Example: $\varepsilon_1 < \varepsilon_2 < \varepsilon_3$

$G(C, \varepsilon_1) \subseteq G(C, \varepsilon_2) \subseteq G(C, \varepsilon_3)$

$S_1 \cong S_2$
Minimal Progressive Simplification

Dynamic Programming

- Assign a cost value $c_{i,j}^k \in \mathbb{N}$ for each shortcut $(p_i, p_j) \in G(C, \varepsilon_k)$ at scale $\varepsilon_k$
- $c_{i,j}^k$ relates to the cost of including $(p_i, p_j)$ in $S_k$
- Example: $\varepsilon_1 < \varepsilon_2 < \varepsilon_3$
Dynamic Program

\[
c_{i,j}^k = \begin{cases} 
1 & \text{if } k = 1 \\
1 + \min_{\pi \in \prod_{i,j}^{k-1}} \sum_{(p_x,p_y) \in \pi} c_{x,y}^{k-1} & \text{if } 1 < k \leq m
\end{cases}
\]

\(\prod_{i,j}^{k}\) denotes the set of all paths in \(G(C, \varepsilon_k)\) from \(p_i\) to \(p_j\)
Minimal Progressive Simplification

Dynamic Program

\[ c_{i,j}^k = \begin{cases} 
1 & \text{if } k = 1 \\
1 + \min_{\pi \in \prod_{i,j}^{k-1}} \sum_{(p_x,p_y) \in \pi} c_{x,y}^{k-1} & \text{if } 1 < k \leq m 
\end{cases} \]

\( \prod_{i,j}^k \) denotes the set of all paths in \( G(C, \varepsilon_k) \) from \( p_i \) to \( p_j \)

Algorithm

Construct simplifications from \( S_m \) down to \( S_1 \)

1. Compute costs \( c_{i,j}^k \) at scale \( \varepsilon_k \)

2. Compute shortest path \( P \) from \( p_i \) to \( p_j \) in \( G(C, \varepsilon_k) \) for all \( (p_i, p_j) \in S_{k+1} \)

3. Link \( P \) to obtain \( S_k \)
Algorithm
Construct simplifications from $S_m$ down to $S_1$

1. Compute costs $c_{i,j}^k$ at scale $\varepsilon_k$
2. Compute shortest path $P$ from $p_i$ to $p_j$ in $G(C, \varepsilon_k)$ for all $(p_i, p_j) \in S_{k+1}$
3. Link $P$ to obtain $S_k$

Running time $m$ times
Minimal Progressive Simplification

- Employ the algorithm by [Chan and Chin, 1996]
- Runs in $O(n^2)$ time in the plane

Algorithm
Construct simplifications from $S_m$ down to $S_1$

1. Compute costs $c_{i,j}^k$ at scale $\varepsilon_k$
2. Compute shortest path $P$ from $p_i$ to $p_j$ in $G(C, \varepsilon_k)$ for all $(p_i, p_j) \in S_{k+1}$
3. Link $P$ to obtain $S_k$

Running time
$m$ times $O(n^2)$
Minimal Progressive Simplification

Algorithm

Construct simplifications from $S_m$ down to $S_1$

1. Compute costs $c_{i,j}^k$ at scale $\varepsilon_k$

2. Compute shortest path $P$ from $p_i$ to $p_j$ in $G(C, \varepsilon_k)$ for all $(p_i, p_j) \in S_{k+1}$

3. Link $P$ to obtain $S_k$

Running time

$m$ times

$O(n^2)$

$O(n^3)$

- Run Dijkstra’s algorithm on $O(n)$ nodes of $G(C, \varepsilon_k)$
- Dijkstra’s algorithm runs in $O(n^2)$ time on $G$ with integer weights
Minimal Progressive Simplification

• Employ the algorithm by [Chan and Chin, 1996]

Algorithm
Construct simplifications from $S_m$ down to $S_1$

1. Compute costs $c_{i,j}^k$ at scale $\varepsilon_k$

2. Compute shortest path $P$ from $p_i$ to $p_j$ in $G(C, \varepsilon_k)$ for all $(p_i, p_j) \in S_{k+1}$

3. Link $P$ to obtain $S_k$

Running time
$m$ times
$O(n^2)$
$O(n^3)$
$O(n)$
Minimal Progressive Simplification

Total Running Time

- Optimal progressive simplification computable in $O(n^3m)$ time
- Takes $O(n^5)$ time for continuous scaling

Algorithm

Construct simplifications from $S_m$ down to $S_1$

1. Compute costs $c_{i,j}^k$ at scale $\varepsilon_k$
2. Compute shortest path $P$ from $p_i$ to $p_j$ in $G(C, \varepsilon_k)$ for all $(p_i, p_j) \in S_{k+1}$
3. Link $P$ to obtain $S_k$

Running time

- $m$ times $O(n^2)$
- $O(n^3)$
- $O(n)$
Conclusion

- An $O(n^3 m)$ time algorithm for the progressive simplification problem
- works with various distance measures such as Hausdorff, Fréchet and area-based distances
- enables simplification for continuous scaling in $O(n^5)$ time
Conclusion

Further Results

• Technique to compute all shortcuts for a fixed $\varepsilon$ in $O(n^2 \log n)$ time instead of $O(n^3)$ time

• Storage-efficient representation of the shortcut graph allowing to find shortest paths in $O(n \log n)$ time

• Experimental evaluation on a trajectory of a migrating vulture

Thank you for your attention.