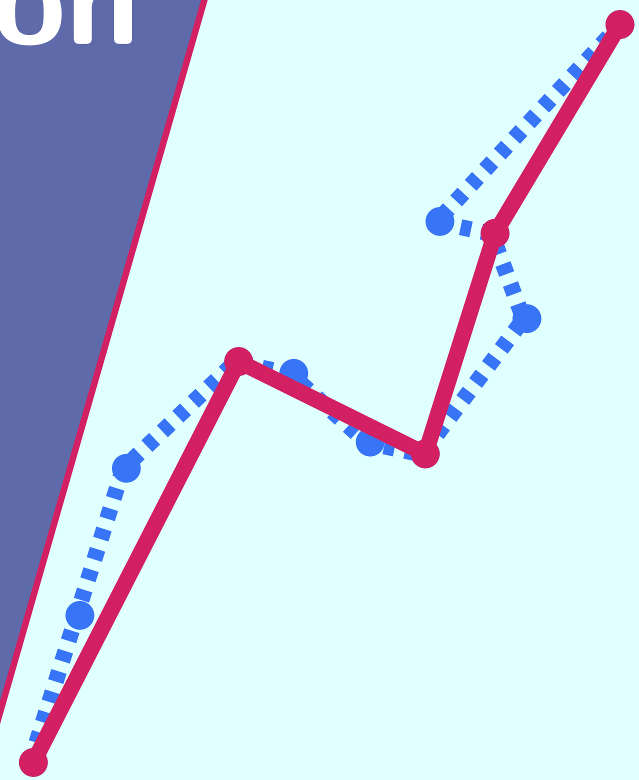
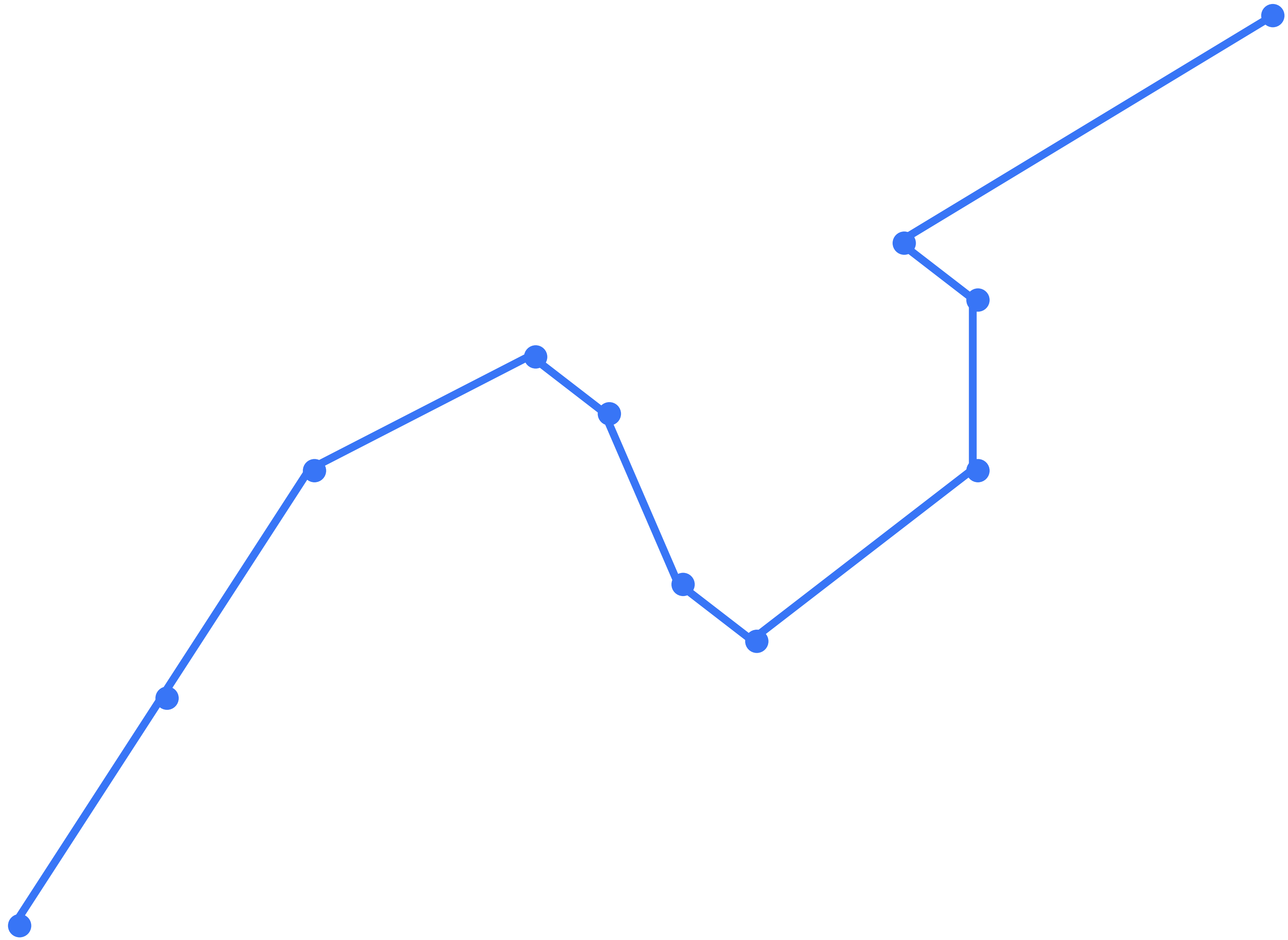
# Progressive Simplification of Polygonal Curves



Kevin Buchin
*Maximilian Konzack*
Wim Reddingius

Technische Universiteit
**Eindhoven**
University of Technology

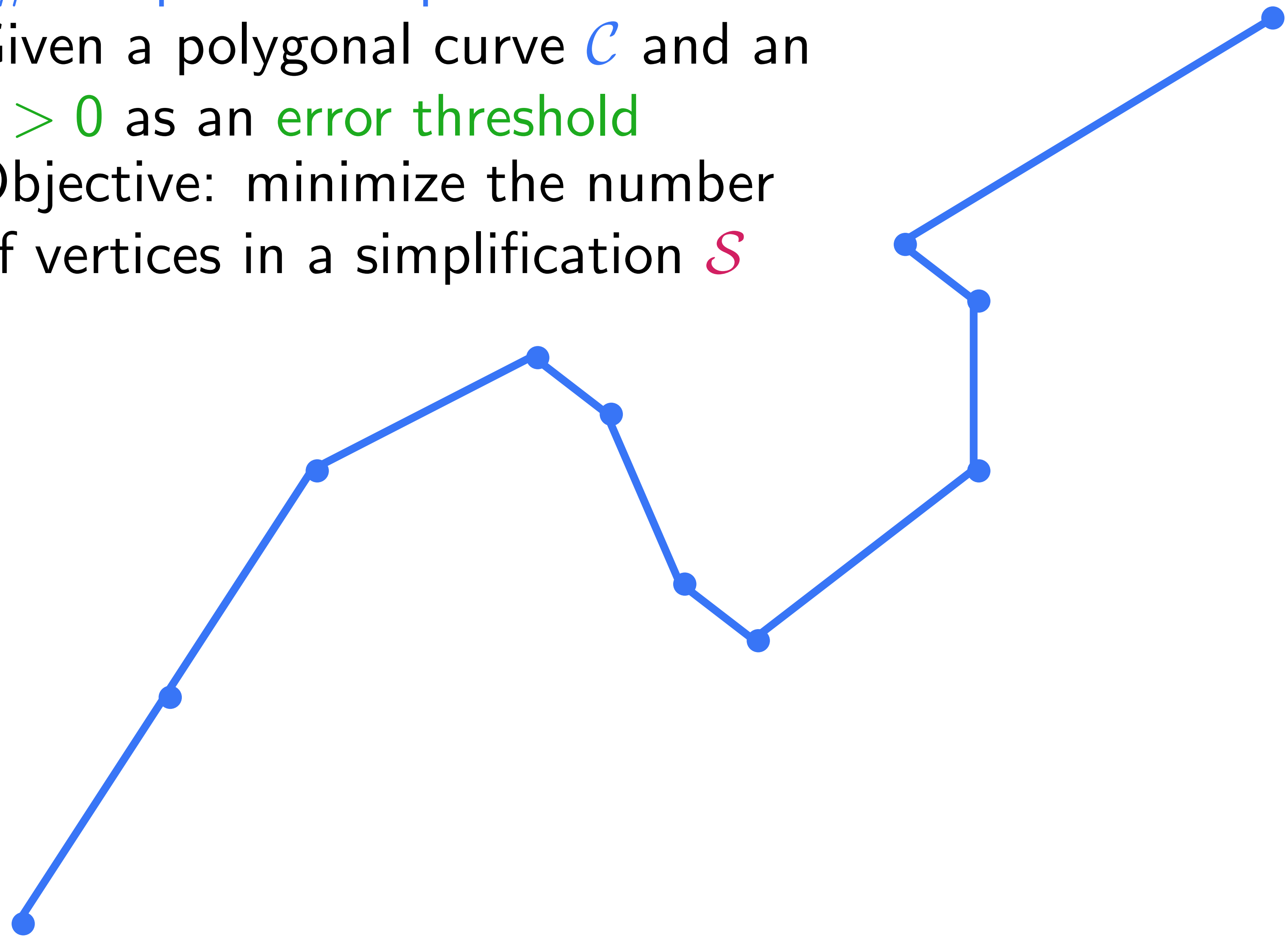
# Curve Simplification



# Curve Simplification

min-$\#$ Simplification problem:

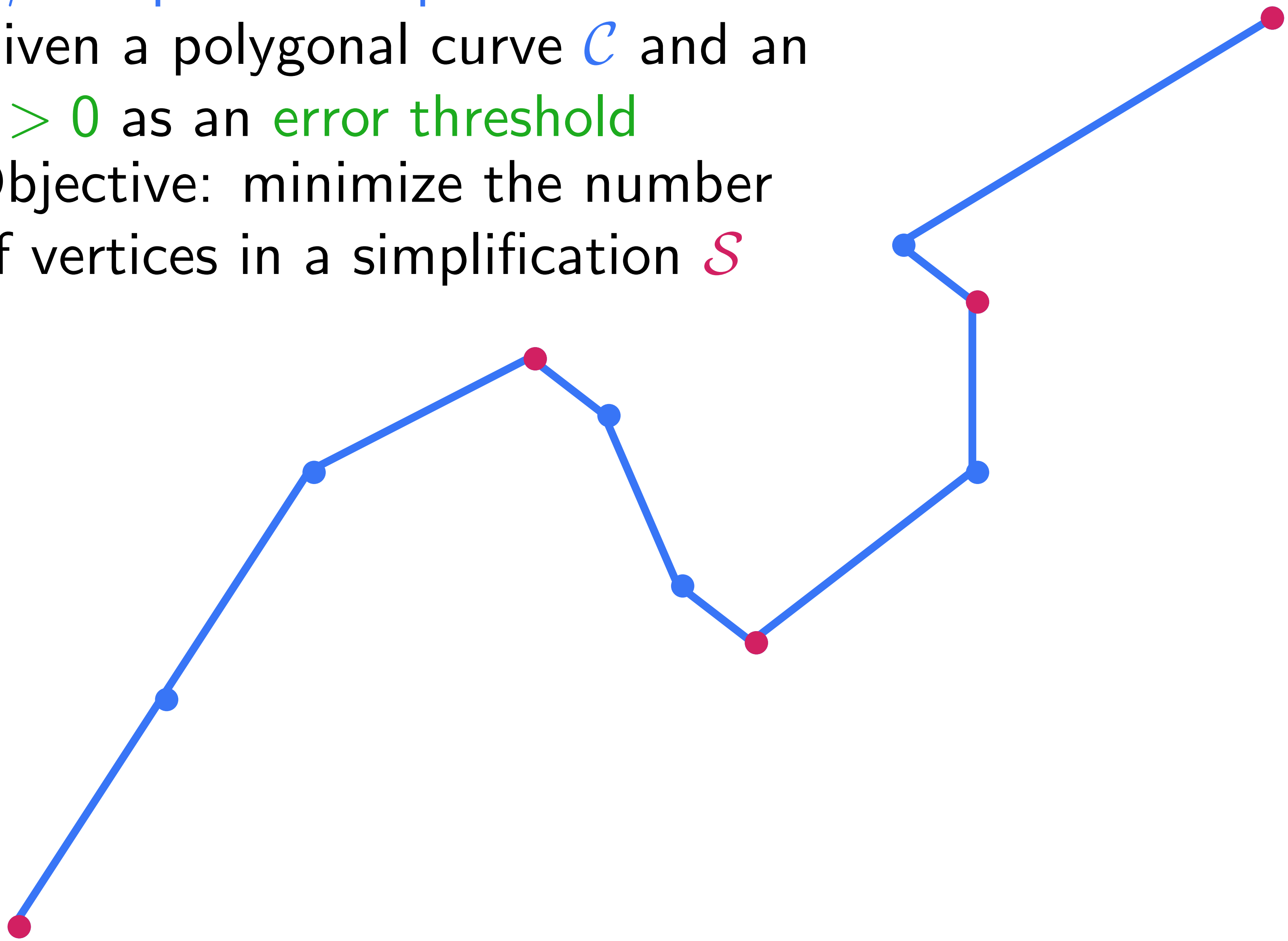
- Given a polygonal curve $\mathcal{C}$ and an $\varepsilon > 0$ as an error threshold
- Objective: minimize the number of vertices in a simplification $\mathcal{S}$



# Curve Simplification

min-# Simplification problem:

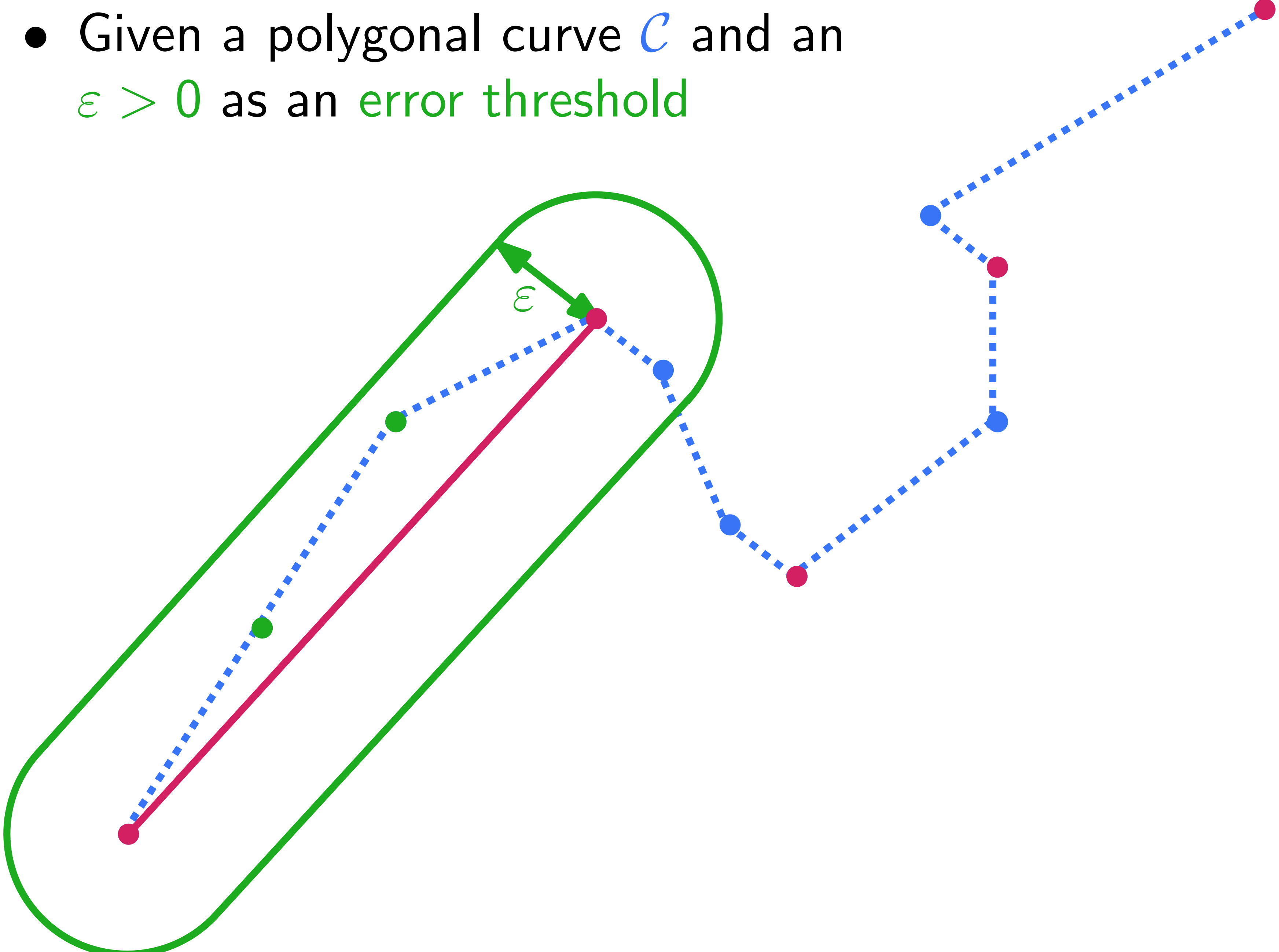
- Given a polygonal curve $\mathcal{C}$ and an $\varepsilon > 0$ as an error threshold
- Objective: minimize the number of vertices in a simplification $\mathcal{S}$



# Curve Simplification

min-# Simplification problem:

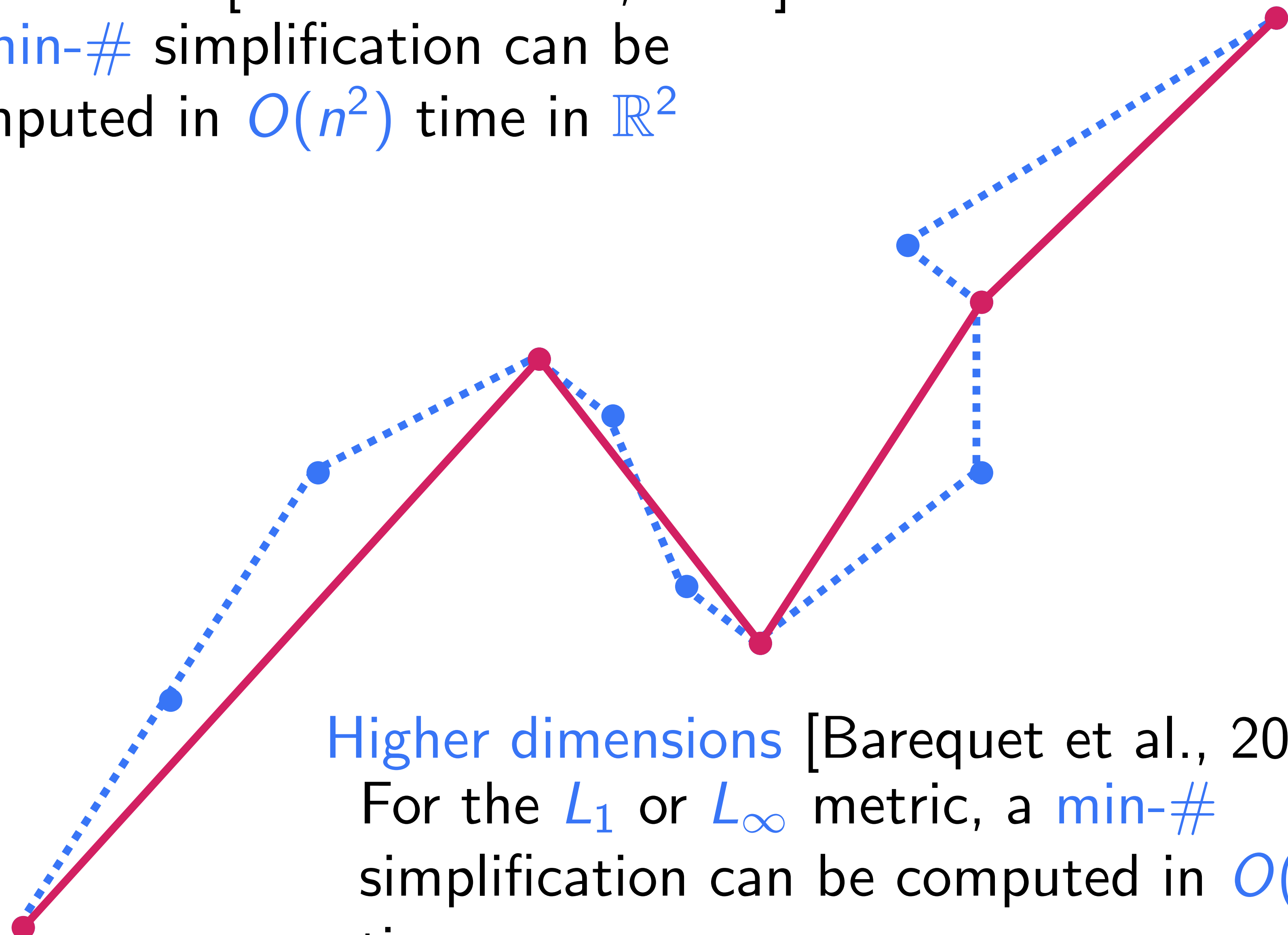
- Given a polygonal curve $\mathcal{C}$ and an $\varepsilon > 0$ as an error threshold



# Curve Simplification

Upper bound [Chan and Chin, 1996]

A min-# simplification can be computed in $O(n^2)$ time in $\mathbb{R}^2$

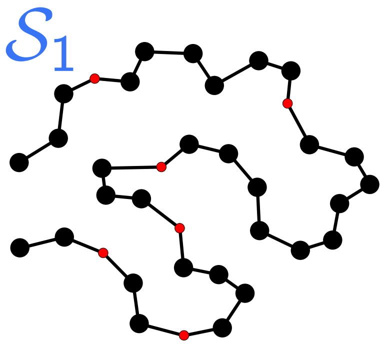


Higher dimensions [Barequet et al., 2002]

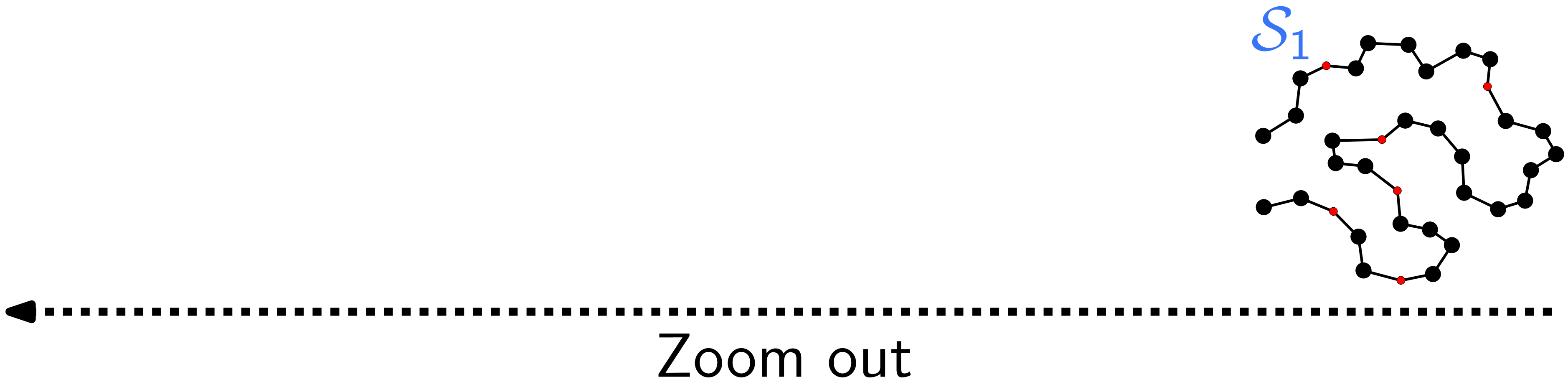
For the $L_1$ or $L_\infty$ metric, a min-# simplification can be computed in $O(n^2)$ time

# Progressive Simplification



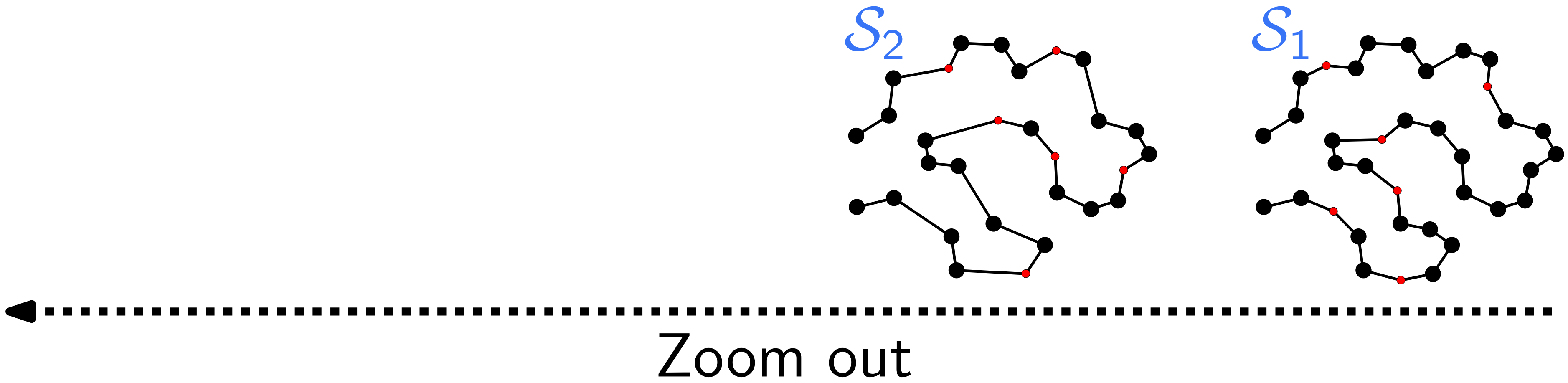


# Progressive Simplification

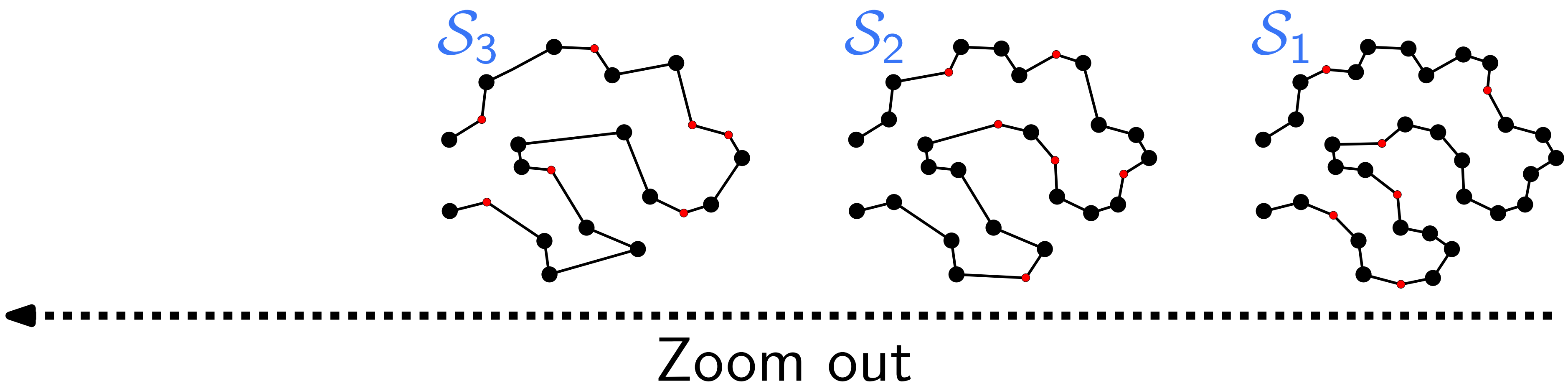


# Progressive Simplification



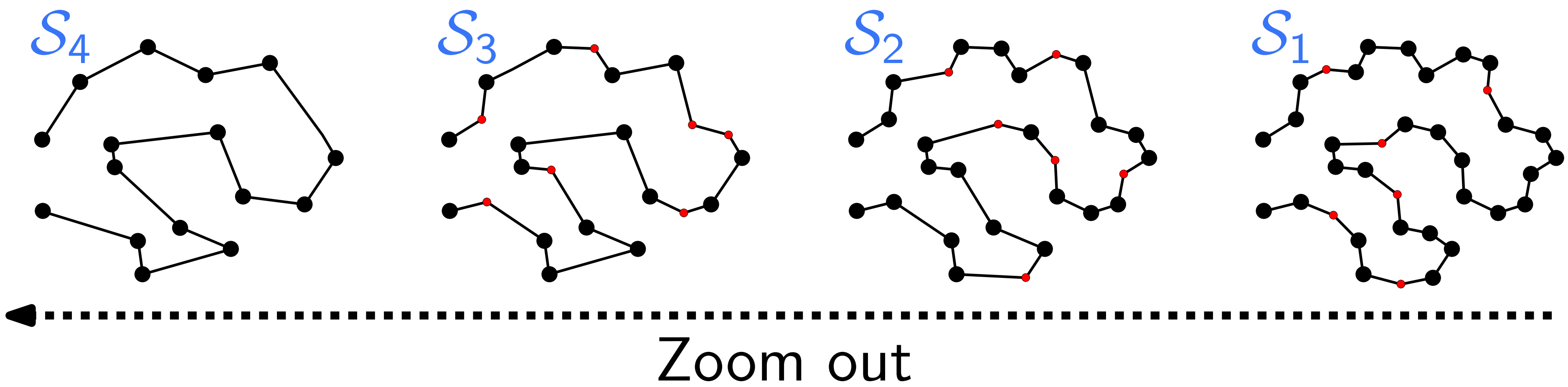


# Progressive Simplification

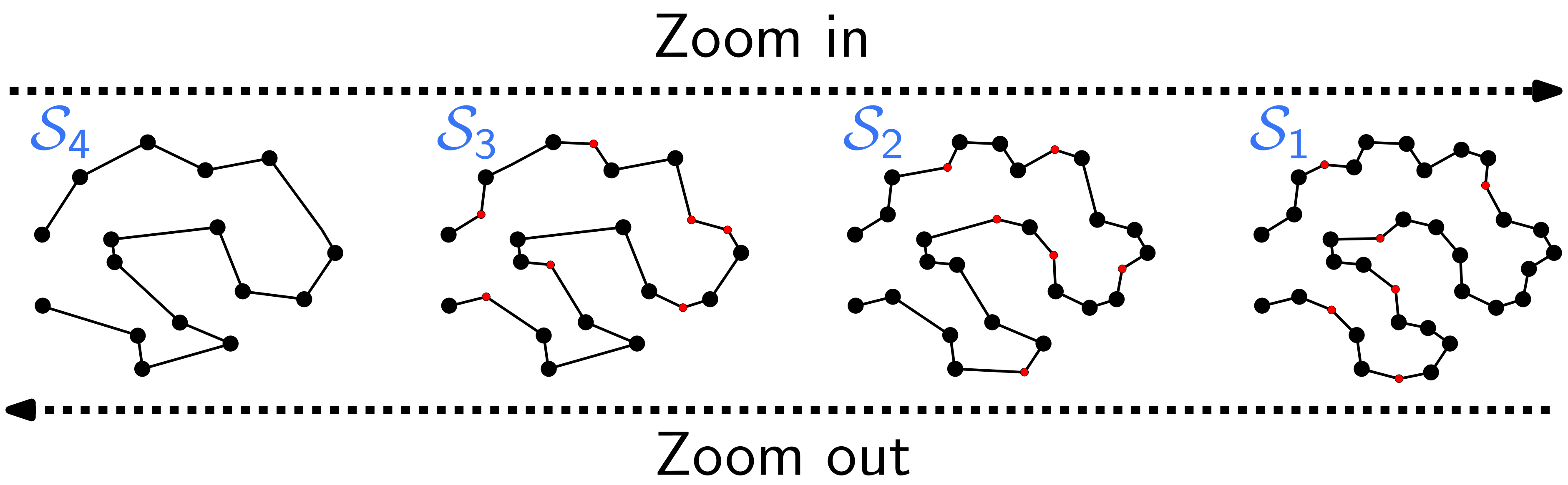


# Progressive Simplification

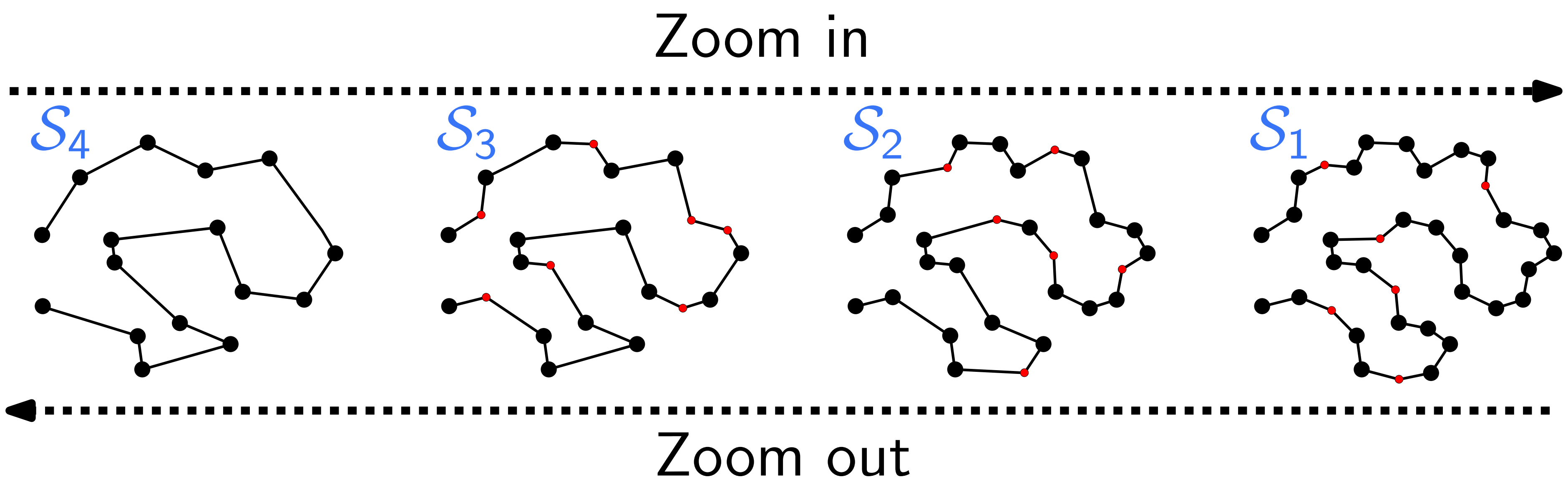




# Progressive Simplification



# Progressive Simplification



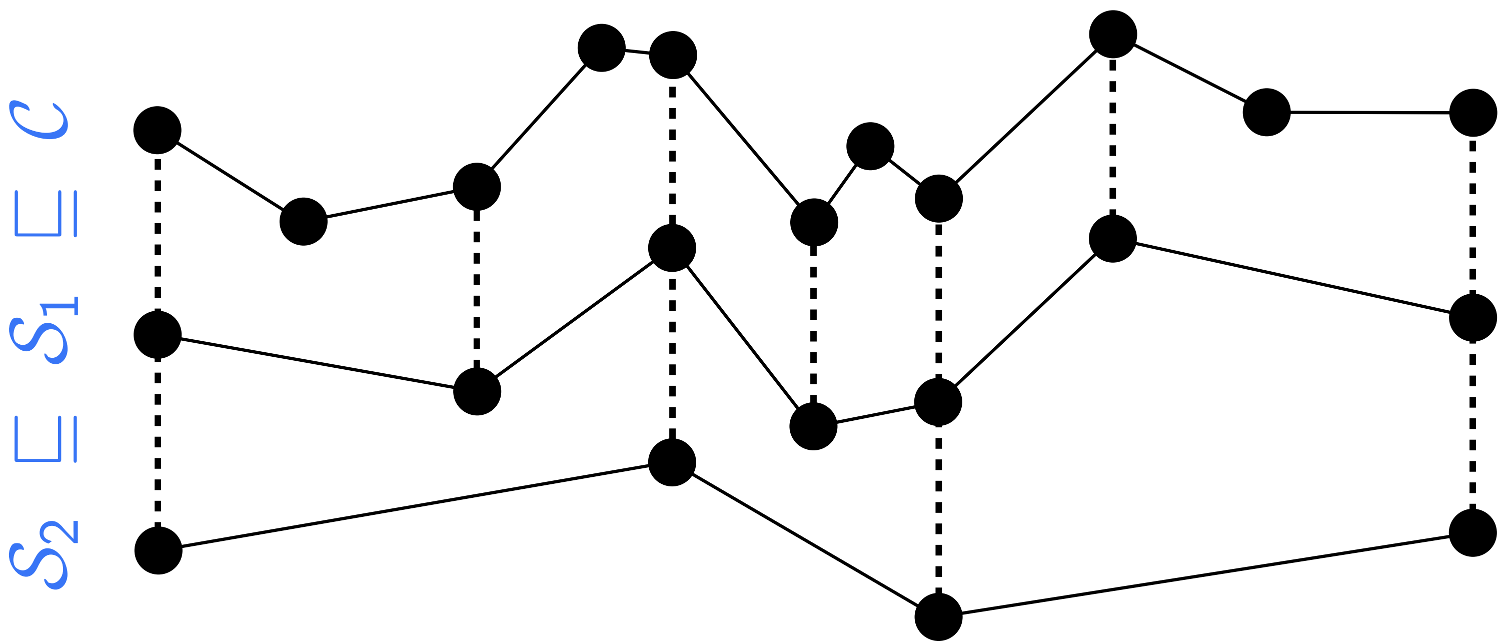

## Impose Consistency Across Many Scales

- Zoom in and out without flickering
- A sequence of $m$ scales: $0 < \varepsilon_1 < \cdots < \varepsilon_m$
- Require *monotonicity*: $\mathcal{S}_m \sqsubseteq \mathcal{S}_{m-1} \sqsubseteq \cdots \sqsubseteq \mathcal{C}$
- Minimize $\sum_{k=1}^{m} |\mathcal{S}_k|$ (*optimality*)

# Results



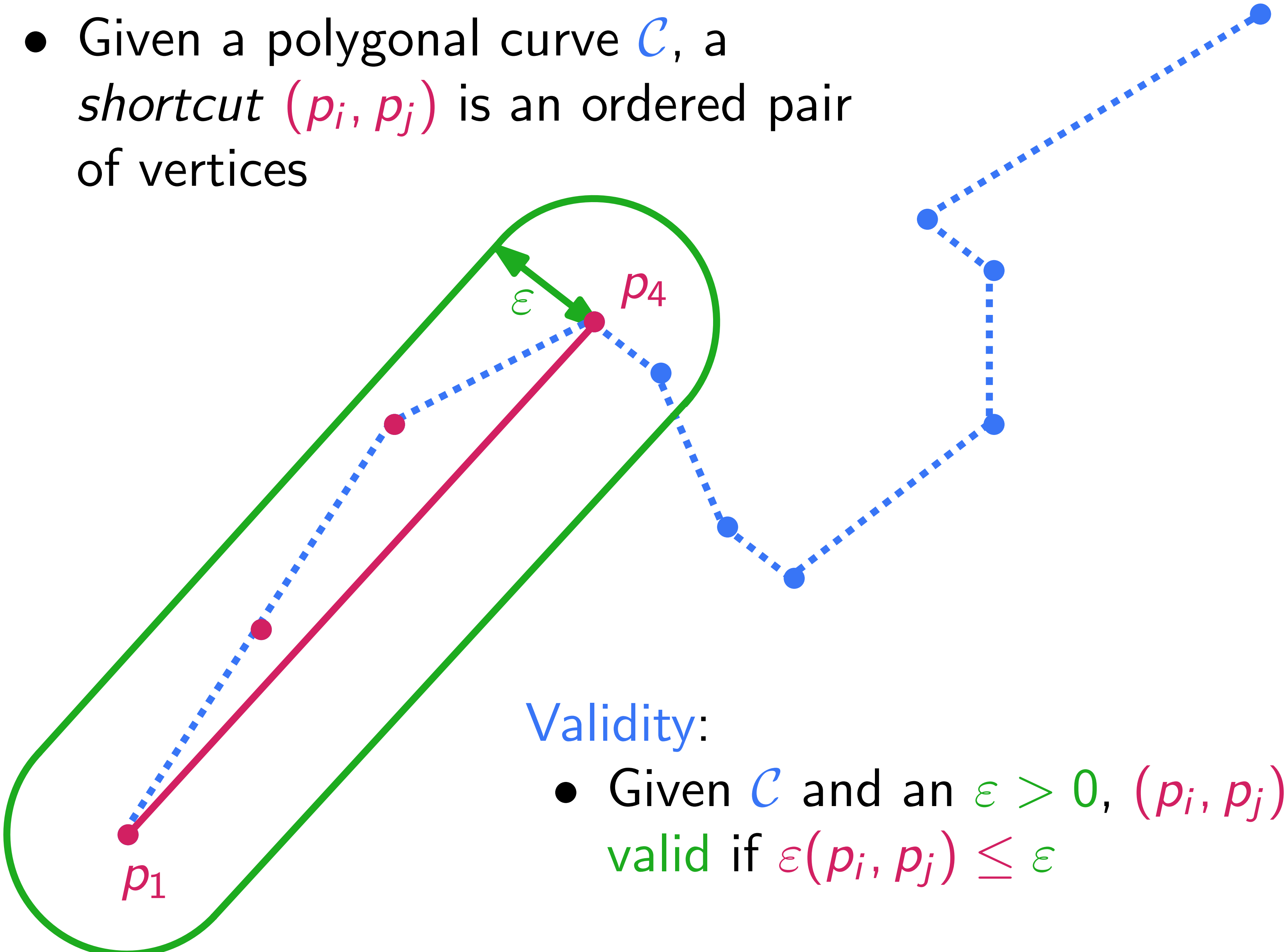
- An $O(n^3 m)$ time algorithm for the progressive simplification problem
- works with various distance measures such as Hausdorff, Fréchet and area-based distances
- enables simplification for continuous scaling in $O(n^5)$ time



# Shortcut Graph

Shortcut:

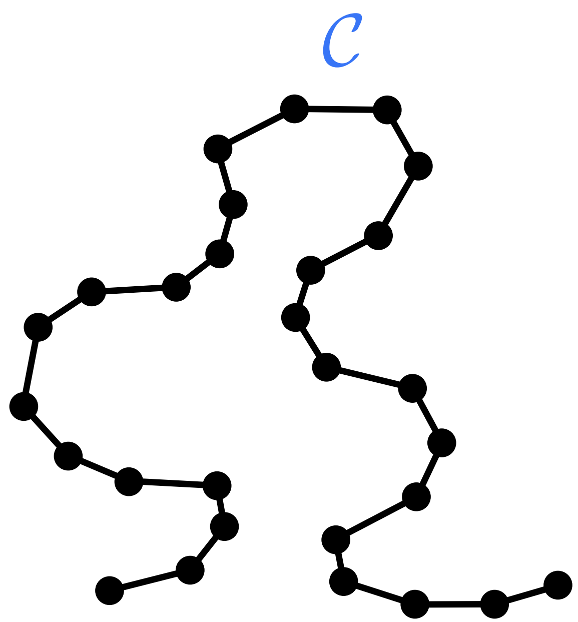
- Given a polygonal curve $\mathcal{C}$, a *shortcut* $(p_i, p_j)$ is an ordered pair of vertices




Validity:

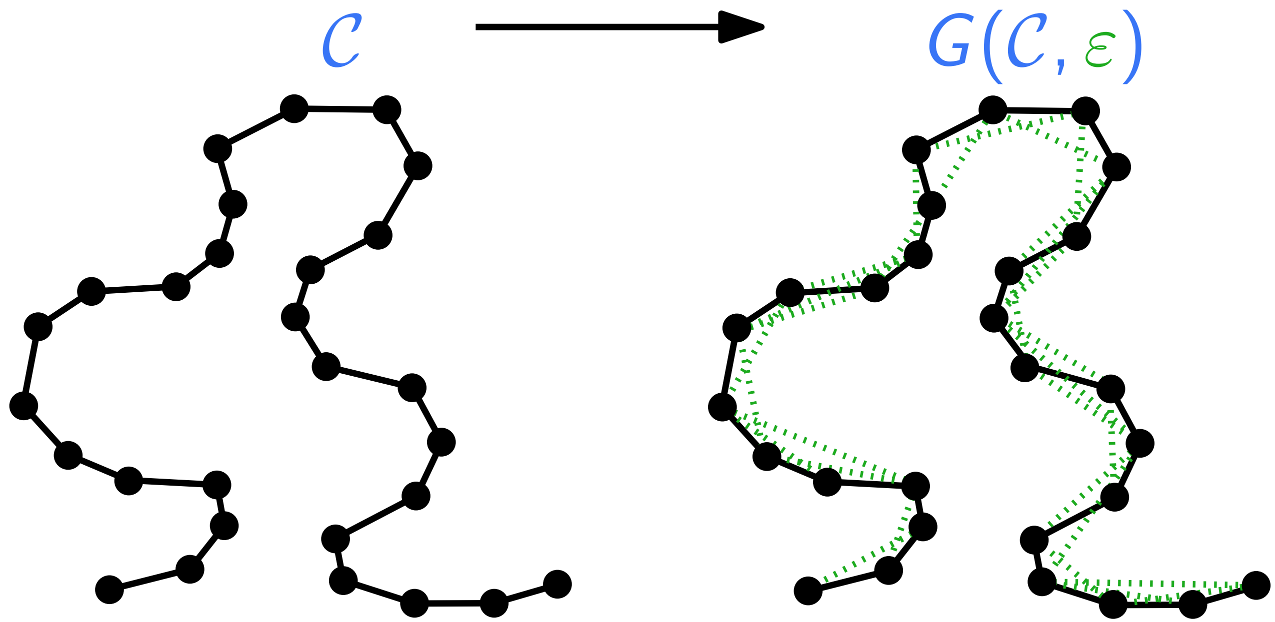
- Given $\mathcal{C}$ and an $\varepsilon > 0$, $(p_i, p_j)$ is valid if $\varepsilon(p_i, p_j) \leq \varepsilon$

# Shortcut Graph



# Shortcut Graph

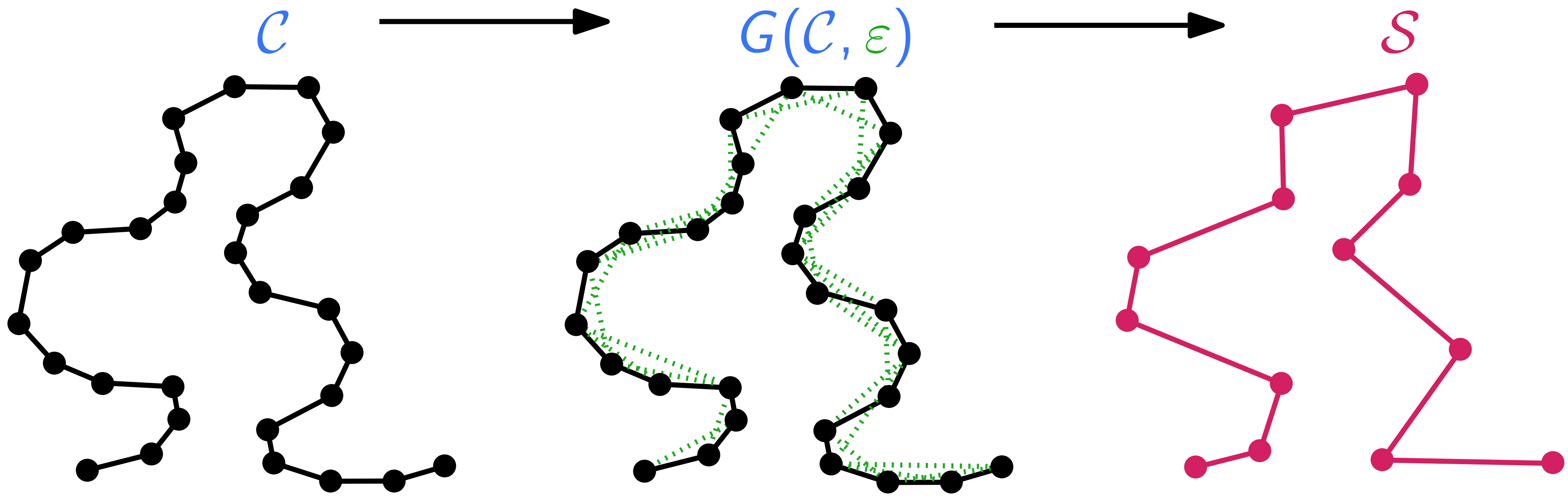
- Given a curve $\mathcal{C}$ and an $\varepsilon > 0$, the *shortcut graph* $G(\mathcal{C}, \varepsilon)$ captures all valid shortcuts



# Shortcut Graph



- Given a curve $\mathcal{C}$ and an $\varepsilon > 0$, the *shortcut graph* $G(\mathcal{C}, \varepsilon)$ captures all valid shortcuts




- Minimum-link path in $G(\mathcal{C}, \varepsilon)$ is an optimal simplification $\mathcal{S}$

# Minimal Progressive Simplification

## Dynamic Programming

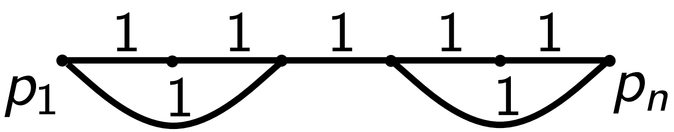
- Assign a cost value $c_{i,j}^k \in \mathbb{N}$ for each shortcut $(p_i, p_j) \in G(\mathcal{C}, \varepsilon_k)$ at scale $\varepsilon_k$
- $c_{i,j}^k$ relates to the cost of including $(p_i, p_j)$ in $\mathcal{S}_k$

# Minimal Progressive Simplification

## Dynamic Programming

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

$G(\mathcal{C}, \varepsilon_1)$



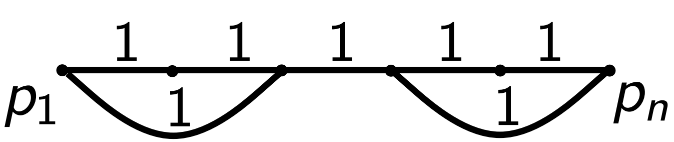
# Minimal Progressive Simplification

Technische Universiteit Eindhoven University of Technology

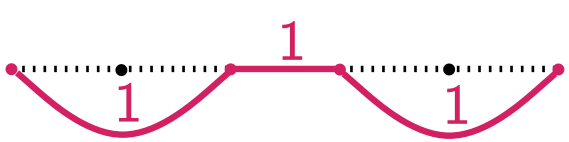
## Dynamic Programming

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

$G(\mathcal{C}, \varepsilon_1)$



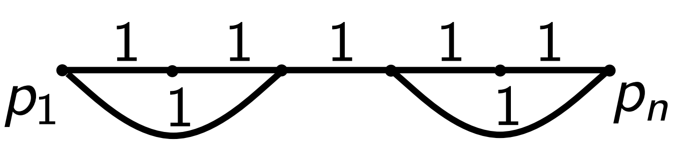

$\mathcal{S}_1$

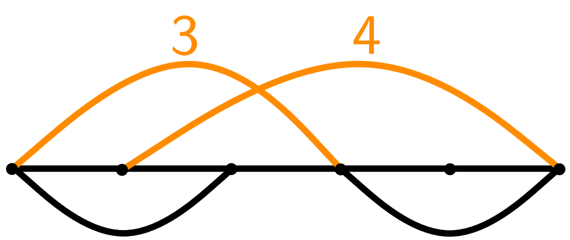


## Dynamic Programming

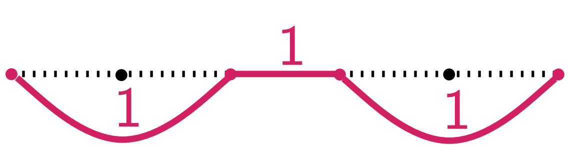
- Assign a cost value $c_{i,j}^{k} \in \mathbb{N}$ for each shortcut $(p_i, p_j) \in G(\mathcal{C}, \varepsilon_k)$ at scale $\varepsilon_k$
- $c_{i,j}^{k}$ relates to the cost of including $(p_i, p_j)$ in $\mathcal{S}_k$
- Example: $\varepsilon_1 < \varepsilon_2 < \varepsilon_3$

$G(\mathcal{C}, \varepsilon_1) \subseteq G(\mathcal{C}, \varepsilon_2)$



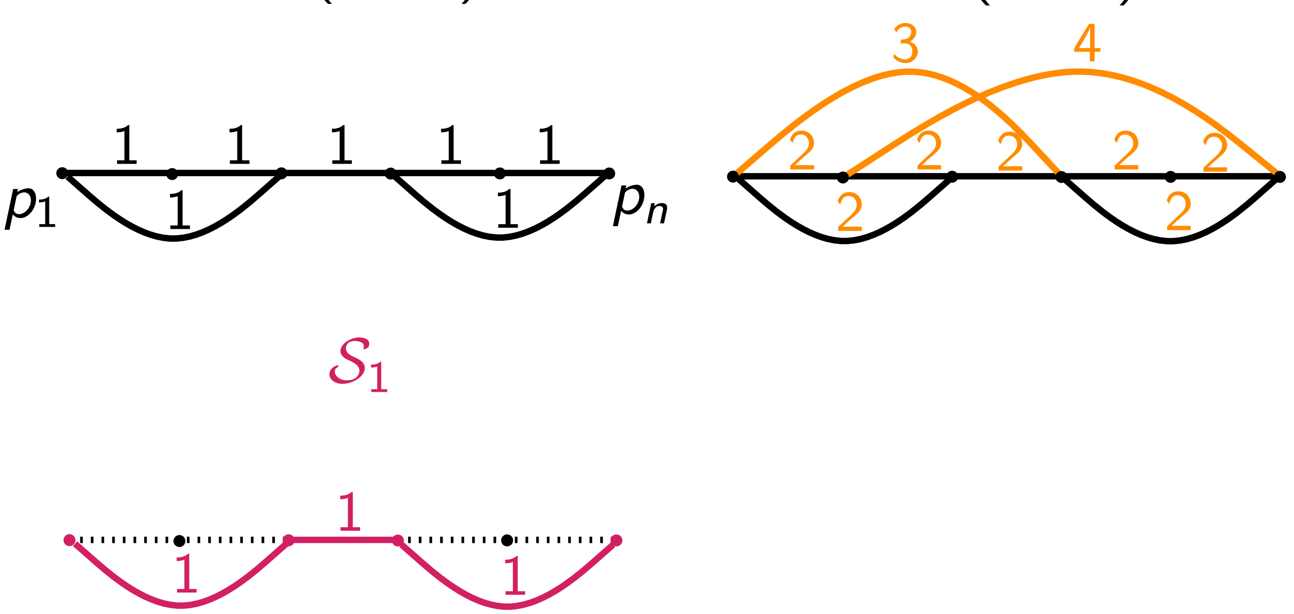

$\mathcal{S}_1$



## Dynamic Programming

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

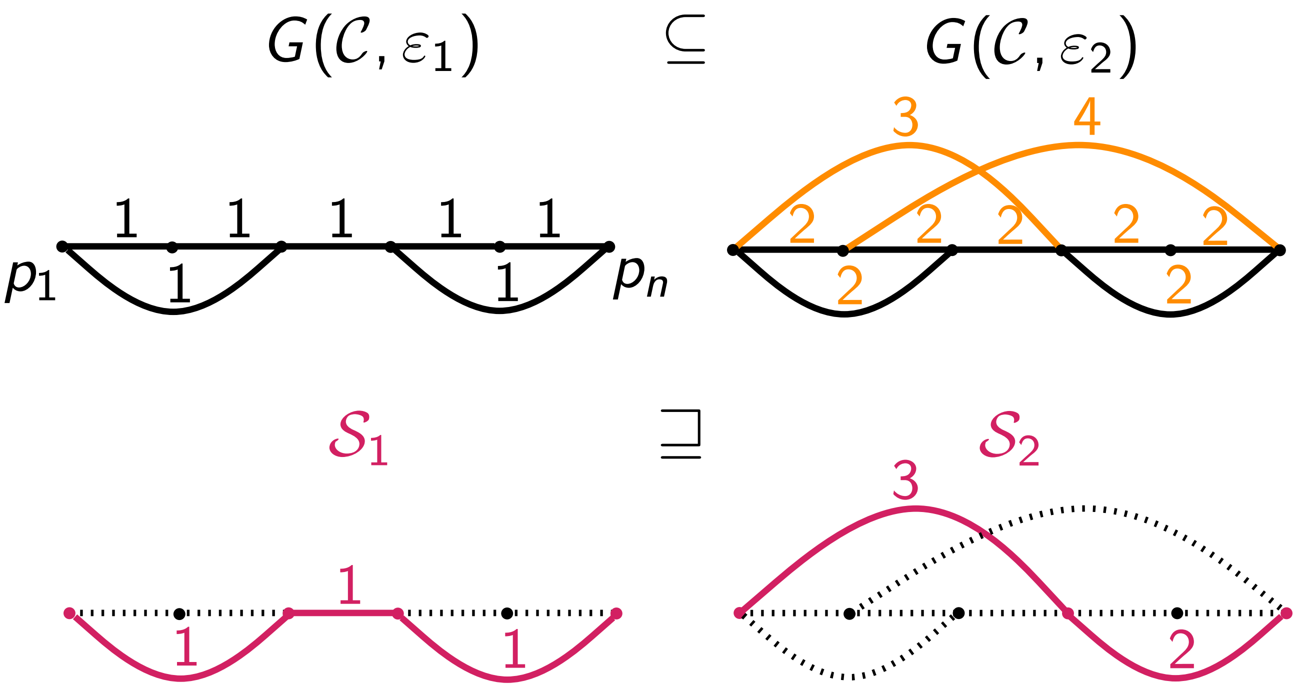
$G(\mathcal{C}, \varepsilon_1) \subseteq G(\mathcal{C}, \varepsilon_2)$



# Minimal Progressive Simplification

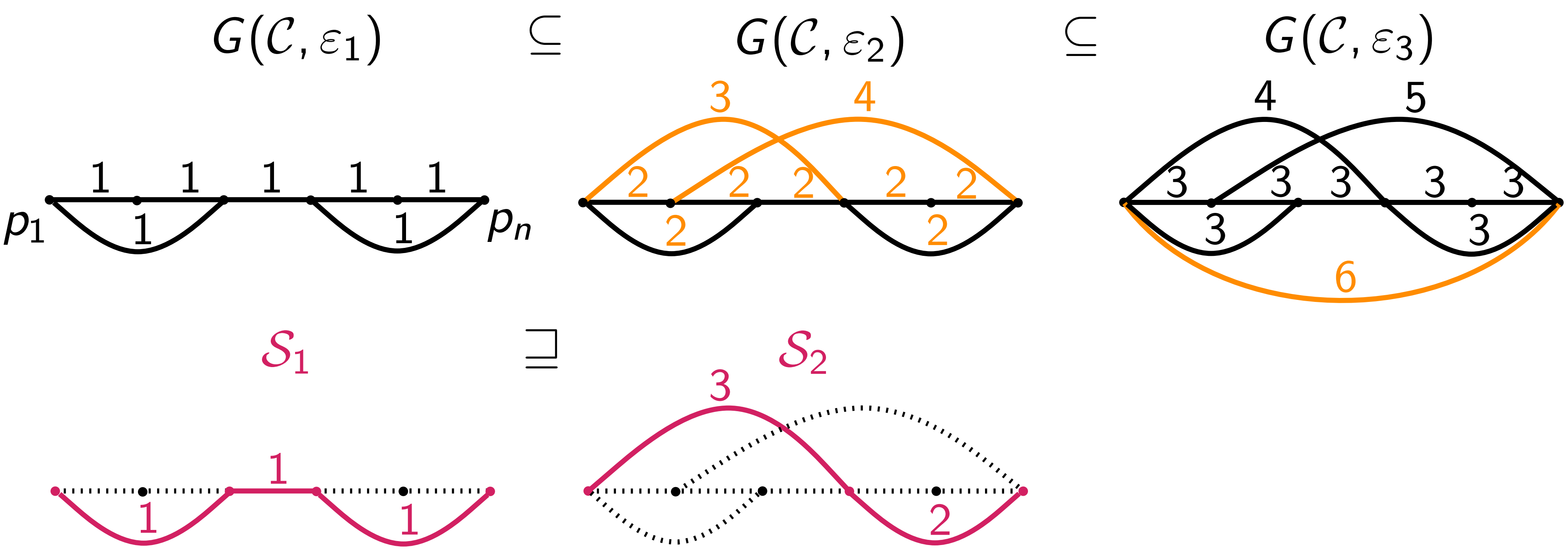
## Dynamic Programming

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



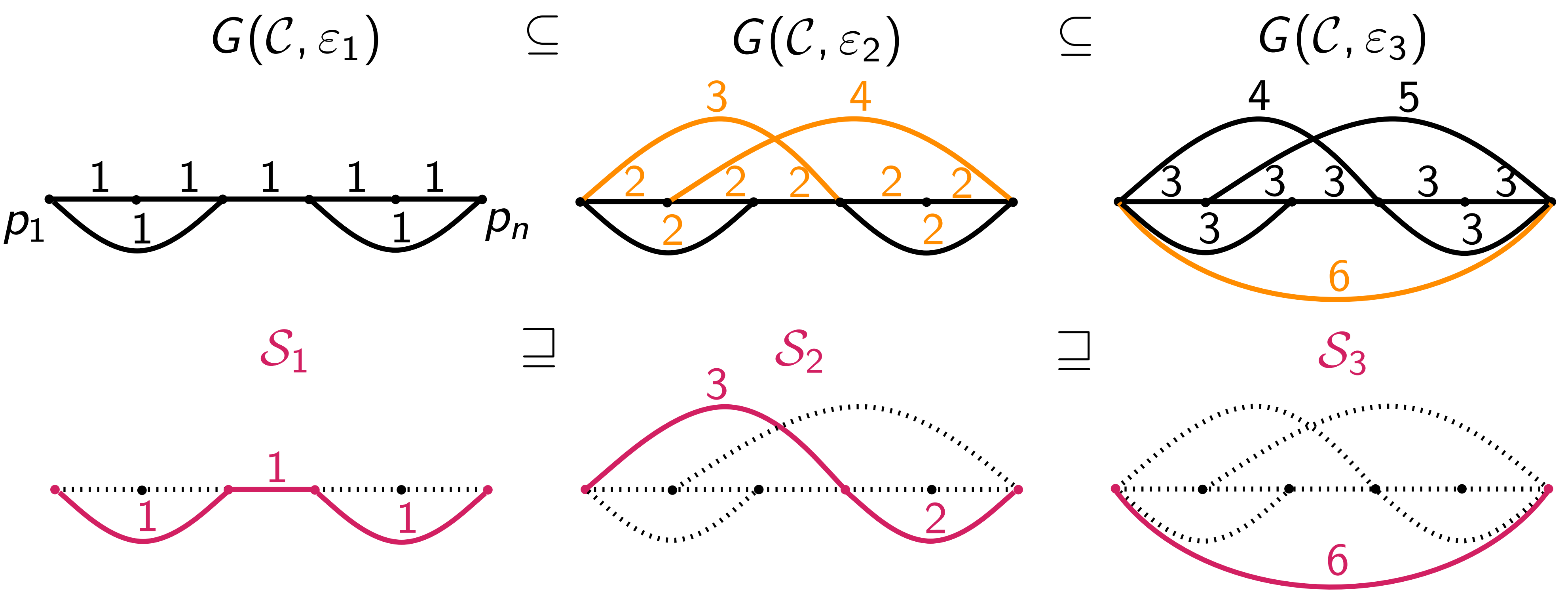
## Dynamic Programming

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



## Dynamic Programming

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



# Minimal Progressive Simplification

## Dynamic Program

$$c_{i,j}^{k} = \begin{cases} 1 & \text{if } k = 1 \\ 1 + \min\limits_{\pi \in \prod_{i,j}^{k-1}} \sum\limits_{(p_x, p_y) \in \pi} c_{x,y}^{k-1} & \text{if } 1 < k \leq m \end{cases}$$

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

# Minimal Progressive Simplification



## Dynamic Program

$$c_{i,j}^{k} = \begin{cases} 1 & \text{if } k = 1 \\ 1 + \min\limits_{\pi \in \prod_{i,j}^{k-1}} \sum\limits_{(p_x, p_y) \in \pi} c_{x,y}^{k-1} & \text{if } 1 < k \leq m \end{cases}$$

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

## Algorithm

Construct simplifications from $\mathcal{S}_m$ down to $\mathcal{S}_1$

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

# Minimal Progressive Simplification

Algorithm

Construct simplifications from $\mathcal{S}_m$ down to $\mathcal{S}_1$

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

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 | Running time |
| --- | --- |
| Construct simplifications from $\mathcal{S}_m$ down to $\mathcal{S}_1$ | $m$ times |
| 1. Compute costs $c_{i,j}^k$ at scale $\varepsilon_k$ | $O(n^2)$ |
| 2. Compute shortest path $P$ from $p_i$ to $p_j$ in $G(\mathcal{C}, \varepsilon_k)$ for all $(p_i, p_j) \in S_{k+1}$ | |
| 3. Link $P$ to obtain $\mathcal{S}_k$ | |

# Minimal Progressive Simplification



- Run Dijkstra's algorithm on $O(n)$ nodes of $G(\mathcal{C}, \varepsilon_k)$
- Dijkstra's algorithm runs in $O(n^2)$ time on $G$ with integer weights

| Algorithm | Running time |
|---|---|
| Construct simplifications from $\mathcal{S}_m$ down to $\mathcal{S}_1$ | $m$ times |
| 1. Compute costs $c^k_{i,j}$ at scale $\varepsilon_k$ | $O(n^2)$ |
| 2. Compute shortest path $P$ from $p_i$ to $p_j$ in $G(\mathcal{C}, \varepsilon_k)$ for all $(p_i, p_j) \in S_{k+1}$ | $O(n^3)$ |
| 3. Link $P$ to obtain $\mathcal{S}_k$ | |

# Minimal Progressive Simplification

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

| Algorithm | Running time |
|---|---|
| Construct simplifications from $\mathcal{S}_m$ down to $\mathcal{S}_1$ | $m$ times |
| 1. Compute costs $c_{i,j}^k$ at scale $\varepsilon_k$ | $O(n^2)$ |
| 2. Compute shortest path $P$ from $p_i$ to $p_j$ in $G(\mathcal{C}, \varepsilon_k)$ for all $(p_i, p_j) \in S_{k+1}$ | $O(n^3)$ |
| 3. Link $P$ to obtain $\mathcal{S}_k$ | $O(n)$ |

# Minimal Progressive Simplification

TU/e Technische Universiteit Eindhoven University of Technology

## Total Running Time

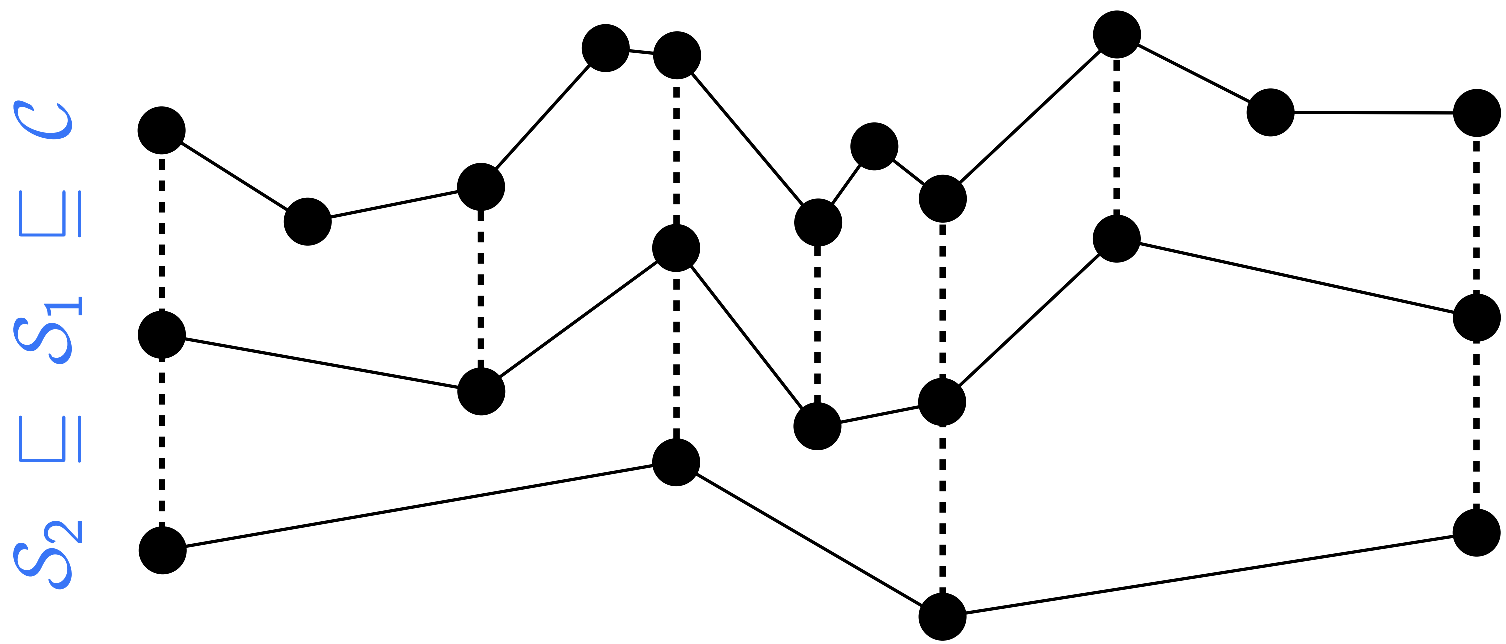
- Optimal progressive simplification computable in $O(n^3 m)$ time
- Takes $O(n^5)$ time for continuous scaling

| Algorithm | Running time |
|---|---|
| Construct simplifications from $\mathcal{S}_m$ down to $\mathcal{S}_1$ | $m$ times |
| 1. Compute costs $c_{i,j}^k$ at scale $\varepsilon_k$ | $O(n^2)$ |
| 2. Compute shortest path $P$ from $p_i$ to $p_j$ in $G(\mathcal{C}, \varepsilon_k)$ for all $(p_i, p_j) \in S_{k+1}$ | $O(n^3)$ |
| 3. Link $P$ to obtain $\mathcal{S}_k$ | $O(n)$ |

# 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.