An Optimization-Based Approach for Continuous Map Generalization

Dongliang Peng

Chair of Computer Science I, University of Würzburg, Germany
Marienkapelle
[source: Wikipedia]
Marienkapelle
[source: Wikipedia]
Marienkapelle
[source: Wikipedia]
Buildings disappear suddenly!

Marienkapelle
[source: Wikipedia]
Buildings disappear suddenly!

Smooth changes provide better experience!
Map Generalization...

...is about deriving a smaller-scale map from an existing map.
Map Generalization . . .

. . . is about deriving a smaller-scale map from an existing map. Typical generalization operators [ESRI 1996]:

Map Generalization...

...is about deriving a smaller-scale map from an existing map.

Typical generalization operators [ESRI 1996]:

Elimination
Map Generalization...

...is about *deriving a smaller-scale map* from an existing map.

Typical generalization operators [ESRI 1996]:

- **Elimination**
- **Simplification**
Map Generalization...

...is about deriving a smaller-scale map from an existing map.

Typical generalization operators [ESRI 1996]:

- Elimination
- Simplification
- Aggregation
Map Generalization...

...is about deriving a smaller-scale map from an existing map.

Typical generalization operators [ESRI 1996]:

- **Elimination**
- **Simplification**
- **Aggregation**

Classifi. & Symboli.
Map Generalization...

...is about deriving a smaller-scale map from an existing map.

Typical generalization operators [ESRI 1996]:

- Elimination
- Simplification
- Aggregation
- Classifi. & Symboli.
- Exaggeration
Map Generalization . . .

. . . is about deriving a smaller-scale map from an existing map.

Typical generalization operators [ESRI 1996]:

- **Elimination**
- **Simplification**
- **Aggregation**
- **Classifi. & Symboli.**
- **Exaggeration**
- **Typification**
- **Collapse**
- **Displacement**
- **Refinement**
Continuous Map Generalization...

...is to derive a series of maps with **smooth changes**.
Continuous Map Generalization...

...is to derive a series of maps with smooth changes.
Continuous Map Generalization…

…is to derive a series of maps with smooth changes.

$\text{input}$

$t = 0$

$t = 0.2$

$t = 0.4$

$t = 1$
Continuous Map Generalization...

...is to derive a series of maps with smooth changes.

![Maps at different times](t=0, t=0.2, t=0.4, t=0.6, t=1)
Continuous Map Generalization...

...is to derive a series of maps with smooth changes.
Related Work

- **Morph** between polylines [Nöllenburg et al. 2008]
Related Work

• **Morph** between polylines  
  [Nöllenburg et al. 2008]

• Generate a **good sequence** of maps  
  [Chimani et al. 2014]
Related Work

- **Morph** between polylines [Nöllenburg et al. 2008]
- Generate a **good sequence** of maps [Chimani et al. 2014]
- Data structure for continuous generalization [van Oosterom et al. 2014]

---

space scale cube (SSC)
Related Work

- **Morph** between polylines [Nöllenburg et al. 2008]
- Generate a **good sequence** of maps [Chimani et al. 2014]
- Data structure for continuous generalization [van Oosterom et al. 2014]

space scale cube (SSC)
<table>
<thead>
<tr>
<th>Contents of Thesis</th>
<th>Optim.</th>
<th>Related Generalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents of Thesis</td>
<td>Optim.</td>
<td>Related Generalization</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Optimal sequence for aggregation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Diagram of optimal sequence for aggregation]
<table>
<thead>
<tr>
<th>Contents of Thesis</th>
<th>Optim.</th>
<th>Related Generalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal sequence for aggregation</td>
<td>A*</td>
<td>ILP</td>
</tr>
<tr>
<td>Contents of Thesis</td>
<td>Optim.</td>
<td>Related Generalization</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Optimal sequence for aggregation</td>
<td>A* ILP</td>
<td>Aggregation Classification</td>
</tr>
</tbody>
</table>

- **A***: Optimality
- **ILP**: Integer Linear Programming
Contents of Thesis

Optimal sequence for aggregation

Administrative boundaries

Optim.

A*

ILP

Related Generalization

Aggregation

Classification
Contents of Thesis

Optimal sequence for aggregation

Administrative boundaries

Optim.

A*

ILP

Related Generalization

Aggregation

Classification

DP
Contents of Thesis

Optimal sequence for aggregation

Administrative boundaires

Optim.

A*

ILP

DP

Related Generalization

Aggregation

Classification

Elimination

Simplification
<table>
<thead>
<tr>
<th>Contents of Thesis</th>
<th>Optim.</th>
<th>Related Generalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal sequence for aggregation</td>
<td>A*</td>
<td>Aggregation</td>
</tr>
<tr>
<td></td>
<td>ILP</td>
<td></td>
</tr>
<tr>
<td>Administrative boundaires</td>
<td>DP</td>
<td>Elimination</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Simplification</td>
</tr>
<tr>
<td>Buildings to built-up areas</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Simplification
- Elimination
- Aggregation
- A*
- ILP
- DP

6-8
Contents of Thesis

Optimal sequence for aggregation

Optim.

A*

ILP

Related Generalization

Administrative boundaries

Aggregation

DP

Elimination

Buildings to built-up areas

Simplification

MST

Classification

A⋆
Contents of Thesis

Optimal sequence for aggregation

A*  ILP

Optims.

Administrative boundaries

DP

Buildings to built-up areas

MST

Exaggeration

Related Generalization

Aggregation

Classification

Elimination

Simplification

Aggregation, Simplification, Elimination
Contents of Thesis

Optimal sequence for aggregation

Administrative boundaires

Buildings to built-up areas

Morphing polylines

Optim.

A*

ILP

Related Generalization

Aggregation

Classification

Aggregation

Elimination

Simplification

Exaggeration

Aggregation, Simplification, Elimination
Contents of Thesis

Optimal sequence for aggregation

Optim.

Related Generalization

A*

ILP

Administrative boundary

Aggregation

Classification

DP

Elimination

Simplification

MST

Exaggeration

Aggregation, Simplification, Elimination

LSA

DP

Administrative boundaries

Buildings to built-up areas

Morphing polylines

Classification

Aggregation

Exaggeration
## Contents of Thesis

<table>
<thead>
<tr>
<th>Optimal sequence for aggregation</th>
<th>Optim.</th>
<th>Related Generalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrative boundaries</td>
<td>$A^*$</td>
<td>Aggregation</td>
</tr>
<tr>
<td>Buildings to built-up areas</td>
<td>ILP</td>
<td>Classification</td>
</tr>
<tr>
<td>Morphing polylines</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Optim.
- $A^*$
- ILP

### Related Generalization
- Aggregation
- Classification
- Elimination
- Simplification
- Exaggeration
- Aggregation, Simplification, Elimination
- Simplification
Contents of Thesis

Optimal sequence for aggregation

Administrative boundaires

Buildings to built-up areas

Morphing polylines

Choosing right data structures

Optim.

Related Generalization

A* ILP

A* ILP

Aggregation

Classification

DP

Elimination

Simplification

MST

Exaggeration

Aggregation, Simplification, Elimination

LSA DP

Simplification

ε

p

SortedDictionary, SortedSet, ...
### Contents of Thesis

**Optimal sequence for aggregation**

- **Optim.**
  - A*  
  - ILP

**Related Generalization**

- Aggregation
- Classification

<table>
<thead>
<tr>
<th>Optim.</th>
<th>Related Generalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>A*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>ILP</td>
<td>Classification</td>
</tr>
</tbody>
</table>

**Administrative boundaries**

- **DP**

**Buildings to built-up areas**

- **MST**

**Morphing polylines**

- **LSA**

- **DP**

**Choosing right data structures**

- SortedDictionary, SortedSet, ...
<table>
<thead>
<tr>
<th>Contents of Thesis</th>
<th>Optim.</th>
<th>Related Generalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal sequence for aggregation</td>
<td>A*</td>
<td>Aggregation</td>
</tr>
<tr>
<td></td>
<td>ILP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Classification</td>
</tr>
</tbody>
</table>

- **Administrative boundaries**
  - DP

- **Buildings to built-up areas**
  - MST

- **Morphing polylines**
  - LSA

- **Choosing right data structures**
  - SortedDictionary, SortedSet, ...
Research Problem

input

- swamp
- sport facility
- village, town, city
Research Problem

- input

- swamp
- sport facility
- village, town, city

input
Research Problem

input

- swamp
- sport facility
- village, town, city
Research Problem

input

swamp
sport facility
village, town, city

input
Research Problem

input

- Swamp
- Sport facility
- Village, town, city
Research Problem

- swamp
- sport facility
- village, town, city

input

- ...
Research Problem

input

swamp
sport facility
village, town, city

input
Research Problem

- input
- swamp
- sport facility
- village, town, city
Research Problem

Given aggregation costs:
What is an optimal sequence?

- swamp
- sport facility
- village, town, city
Preliminaries

start map

goal map
Preliminaries

Polygon $p_i$: area on start map
Preliminaries

Polygon $p_i$: area on start map

Patch $u_i$: connected set of areas
Preliminaries

Polygon $p_i$: area on start map

Patch $u_i$: connected set of areas

Region $R_i$: area on the goal map
Preliminaries

Polygon $p_i$: area on start map

Patch $u_i$: connected set of areas

Region $R_i$: area on the goal map

Aggregate the smallest patch with its neighbour
Interleave Aggregation Sequences
Interleave Aggregation Sequences

Compute a sequence for each region

input → output → input

input → output → input
Interleave Aggregation Sequences

Compute a sequence for each region

Interleave according to order of smallest areas (as merge sort)
Interleave Aggregation Sequences

Compute a sequence for each region

Interleave according to order of smallest areas (as merge sort)
Subdivision

Subdivision $P_{t,i}$: patches subdividing a region
Subdivision

Subdivision $P_{t,i}$: patches subdividing a region

Size $n$: #polygons on start map

$P_{\text{start}} = P_{1,1}$

$n = 4$

$P_{\text{goal}} = P_{4,1}$
Subdivision

Subdivision $P_{t,i}$: patches subdividing a region

Size $n$: #polygons on start map

#subdivisions is exponential in $n$. 

$n = 4$

$P_{start} = P_{1,1}$

$P_{goal} = P_{4,1}$
Formalizing a Pathfinding Problem

Diagram: A graph with nodes labeled 'start' and 'goal'. The 'start' node is connected to three other nodes, each of which is connected to the 'goal' node.
Formalizing a Pathfinding Problem

- Each **subdivision** is represented as a **node**

![Diagram of formalized pathfinding problem]

- Start
- Nodes
- Goal
Formalizing a Pathfinding Problem

- Each subdivision is represented as a node
- Find a shortest path w.r.t. cost functions
Cost Function

- Type change: \( f_{\text{type}}(P_{s,i}, P_{s+1,j}) \)

We wish to aggregate patches with similar types.
Cost Function

- Type change: \( f_{\text{type}}(P_{s,i}, P_{s+1,j}) \)

We wish to aggregate patches with similar types.
Cost Function

- **Type change:** $f_{\text{type}}(P_{s,i}, P_{s+1,j})$
  We wish to aggregate patches with similar types

- **Interior length:** $f_{\text{length}}(P_{s,k})$
  Less length, easier to perceive
Cost Function

- **Type change**: $f_{\text{type}}(P_s, i, P_{s+1}, j)$
  
  We wish to aggregate patches with similar types

- **Interior length**: $f_{\text{length}}(P_s, k)$
  
  Less length, easier to perceive

\[ \ell_{\text{int}}(P_s, k) = 19.5 \]
Cost Function

- Path $\Pi = (P_{1,i_1}, P_{2,i_2}, \ldots, P_{t,i_t})$
Cost Function

• Path $\Pi = (P_{1,i_1}, P_{2,i_2}, \ldots, P_{t,i_t})$

$$g_{\text{type}}(\Pi) = \sum_{s=1}^{t-1} f_{\text{type}}(P_s,i_s, P_{s+1},i_{s+1})$$

$$g_{\text{length}}(\Pi) = \sum_{s=2}^{t-1} f_{\text{length}}(P_s,i_s)$$
Cost Function

• Path $\Pi = (P_{1,i_1}, P_{2,i_2}, \ldots, P_{t,i_t})$

$$g_{type}(\Pi) = \sum_{s=1}^{t-1} f_{type}(P_{s,i_s}, P_{s+1,i_{s+1}})$$

$$g_{length}(\Pi) = \sum_{s=2}^{t-1} f_{length}(P_{s,i_s})$$

• Combination of the two costs:

$$g(\Pi) = (1 - \lambda)g_{type}(\Pi) + \lambda g_{length}(\Pi)$$
Cost Function

• Path $\Pi = (P_{1,i_1}, P_{2,i_2}, \ldots, P_{t,i_t})$

\[
g_{\text{type}}(\Pi) = \sum_{s=1}^{t-1} f_{\text{type}}(P_{s,i_s}, P_{s+1,i_{s+1}})
\]

\[
g_{\text{length}}(\Pi) = \sum_{s=2}^{t-1} f_{\text{length}}(P_{s,i_s})
\]

• Combination of the two costs:

\[
g(\Pi) = (1 - \lambda)g_{\text{type}}(\Pi) + \lambda g_{\text{length}}(\Pi)
\]

$\lambda = 0.5$
A* Algorithm

- A best-first search algorithm. Find a path from $s$ to $t$
A* Algorithm

- A best-first search algorithm. Find a path from $s$ to $t$
- Cost function: $F(u) = g(u) + h(u)$
  - $g(u)$: exact cost of $s$-$u$ path
  - $h(u)$: estimated cost of shortest $u$-$t$ path
A* Algorithm

- A best-first search algorithm. Find a path from \( s \) to \( t \)

- Cost function: \( F(u) = g(u) + h(u) \)
  - \( g(u) \): exact cost of \( s-u \) path
  - \( h(u) \): estimated cost of shortest \( u-t \) path

- Guarantees a shortest path if \( h(u) \) is smaller than real cost

\[ g(u) \]
\[ h(u) \]
A* Algorithm

• A best-first search algorithm. Find a path from $s$ to $t$

• Cost function: $F(u) = g(u) + h(u)$
  – $g(u)$: exact cost of $s$-$u$ path
  – $h(u)$: estimated cost of shortest $u$-$t$ path

• Guarantees a shortest path if $h(u)$ is smaller than real cost

• Helps ignore some paths
Estimating Cost

• $h_{\text{type}}(P_{t,i}) = \sum_{s=t}^{n-1} f_{\text{type}}(P_{s,i_s}, P_{s+1,i_{s+1}})$

We assume: Each patch immediately gets the target type.
Estimating Cost

- \( h_{\text{type}}(P_{t,i}) = \sum_{s=t}^{n-1} f_{\text{type}}(P_{s,i_s}, P_{s+1,i_{s+1}}) \)

  We assume: Each patch immediately gets the target type.

- \( h_{\text{length}}(P_{t,i}) \)
Overestimation

- Try finding a path by exploring at most $M = 200,000$ nodes. If fail, try again but increasing estimated costs.
Overestimation

- Try finding a path by exploring at most $M = 200,000$ nodes. If fail, try again but increasing estimated costs.
- A path seems more expensive, thus may be ignored.
Overestimation

- Try finding a path by exploring at most $M = 200,000$ nodes. If fail, try again but increasing estimated costs.

- A path *seems more expensive*, thus may be ignored.

- **Not optimal** anymore once increasing estimated costs.

![Diagram showing start, intermediate nodes, and goal with paths highlighted.]

16-3
Integer Linear Programming

Form of an integer linear program (ILP)

\[
\begin{align*}
\text{minimize} & \quad C^T x \\
\text{subject to} & \quad E x \leq H, \\
& \quad x \geq 0, \\
\text{and} & \quad x \in \mathbb{Z}^l,
\end{align*}
\]
Integer Linear Programming

Form of an integer linear program (ILP)

\[
\begin{align*}
\text{minimize} & \quad C^T x \\
\text{subject to} & \quad E x \leq H, \\
& \quad x \geq 0, \\
& \quad x \in \mathbb{Z}^I,
\end{align*}
\]

Given variables \( x \),
minimize a cost subject to some constraints.
Integer Linear Programming

Form of an integer linear program (ILP)

\[
\text{minimize } C^T x \quad \text{subject to } E x \leq H, \\
\quad x \geq 0, \\
\quad \text{and } x \in \mathbb{Z}^I,
\]

Given variables \( x \), minimize a \textbf{cost} subject to some \textbf{constraints}. 
Using Integer Linear Programming

- Model complete graph by setting variables and constraints
Using Integer Linear Programming

- Model complete graph by setting variables and constraints
- Solve ILP with minimizing total cost
Using Integer Linear Programming

- Model **complete graph** by setting variables and constraints
- Solve ILP with **minimizing** total cost
- Define path according to values of variables, known from solution
Example Variable and Constraints

- Variable: \( x_{t,p,r} \in \{0, 1\} \quad \forall t \in T, \forall p, r \in P \)
  
  \( x_{t,p,r} = 1 \iff p \) is assigned to \( r \) at time \( t \).
Example Variable and Constraints

- Variable: $x_{t,p,r} \in \{0, 1\}$ $\forall t \in T, \forall p, r \in P$
  $x_{t,p,r} = 1 \iff p$ is assigned to $r$ at time $t$.  

\[x_{1,p,r} = 0\]
\[x_{1,r,r} = 1\]
\[x_{1,q,r} = 0\]
Example Variable and Constraints

- Variable: \( x_{t,p,r} \in \{0, 1\} \quad \forall t \in T, \forall p, r \in P \)
  \( x_{t,p,r} = 1 \iff p \text{ is assigned to } r \text{ at time } t. \)
Example Variable and Constraints

• Variable: \( x_{t,p,r} \in \{0, 1\} \quad \forall t \in T, \forall p, r \in P \)
  \[ x_{t,p,r} = 1 \iff p \text{ is assigned to } r \text{ at time } t. \]
Example Variable and Constraints

- Variable: $x_{t,p,r} \in \{0, 1\}$ $\forall t \in T, \forall p, r \in P$
  $x_{t,p,r} = 1 \iff p$ is assigned to $r$ at time $t$.

- Constraints:
  $p$ is assigned to only one polygon: $\sum_{r \in P} x_{t,p,r} = 1$

\begin{align*}
  x_{1,p,r} &= 0 \\
  x_{1,r,r} &= 1 \\
  x_{1,q,r} &= 0 \\
  x_{2,p,r} &= 1 \\
  x_{2,r,r} &= 1 \\
  x_{2,q,r} &= 0 \\
  x_{3,p,r} &= 1 \\
  x_{3,r,r} &= 1 \\
  x_{3,q,r} &= 1
\end{align*}
Example Variable and Constraints

- Variable: \( x_{t,p,r} \in \{0, 1\} \quad \forall t \in T, \forall p, r \in P \)
  \[ x_{t,p,r} = 1 \iff p \text{ is assigned to } r \text{ at time } t. \]

- Constraints:
  \( p \) is assigned to only one polygon: \( \sum_{r \in P} x_{t,p,r} = 1 \)
  Enforce aggregation:
  \[ \sum_{r \in P} x_{t,r,r} = n - t + 1 \]
Example Variable and Constraints

- **Variable:** \( x_{t,p,r} \in \{0, 1\} \quad \forall t \in T, \forall p, r \in P \)
  \[ x_{t,p,r} = 1 \iff p \text{ is assigned to } r \text{ at time } t. \]

- **Constraints:**
  - \( p \) is assigned to only one polygon:
    \[ \sum_{r \in P} x_{t,p,r} = 1 \]
  - Enforce aggregation:
    \[ \sum_{r \in P} x_{t,r,r} = n - t + 1 \]

- In total,
  - 5 sets of variables
  - 17 sets of constraints

\[ x_{1,p,r} = 0 \]
\[ x_{1,r,r} = 1 \]
\[ x_{1,q,r} = 0 \]
\[ x_{2,p,r} = 1 \]
\[ x_{2,r,r} = 1 \]
\[ x_{2,q,r} = 0 \]
\[ x_{3,p,r} = 1 \]
\[ x_{3,r,r} = 1 \]
\[ x_{3,q,r} = 1 \]
Case Study

- Environment: C#, CPLEX
Case Study

- Environment: C#, CPLEX
- Data

5,448 patches
scale 1 : 50 k

734 patches (regions)
scale 1 : 250 k
Comparison of $A^*$ and ILP

percentage of regions that were found **optimal** solutions

\[ n: \text{number of polygons} \]
An Optimal Sequence by A*
An Optimal Sequence by A*
An Optimal Sequence by A*
An Optimal Sequence by A*
An Optimal Sequence by $A^*$

start ($n = 17$)

goal
An Optimal Sequence by A*
An Optimal Sequence by A*
An Optimal Sequence by $A^*$

start ($n = 17$)

goal
An Optimal Sequence by A*
An Optimal Sequence by A*
An Optimal Sequence by A*
An Optimal Sequence by A*
An Optimal Sequence by A*
An Optimal Sequence by A*
An Optimal Sequence by $A^*$

- Start ($n = 17$)
- Goal
An Optimal Sequence by A*
An Optimal Sequence by A*
An Optimal Sequence by A*
## Contents of Thesis

<table>
<thead>
<tr>
<th>Optimal sequence for aggregation</th>
<th>Optim.</th>
<th>Related Generalization</th>
</tr>
</thead>
</table>
| ![Optimal sequence for aggregation](image1) | A*  
ILP | Aggregation  
Classification |
| ![Administrative boundaires](image2) | DP | Elimination  
Simplification |
| ![Buildings to built-up areas](image3) | MST | Aggregation,  
Simplification,  
Elimination |
| ![Morphing polylines](image4) | LSA 
DP | Simplification |
| ![Choosing right data structures](image5) | SortedDictionary,  
SortedSet, ... |

- **Optimization**
  - A*
  - ILP

- **Related Generalization**
  - Aggregation
  - Elimination
  - Simplification
  - Exaggeration
  - Aggregation, Simplification, Elimination

- **Data Structures**
  - SortedDictionary
  - SortedSet
  - ...
Generalizing Buildings to Built-up Areas

Input: buildings
Generalizing Buildings to Built-up Areas

Input: buildings
Aggregate and Grow

- Aggregate buildings that are too close, when zooming out

original buildings
Aggregate and Grow

- Aggregate buildings that are too close, when zooming out
- Bridges and buildings constitute a minimum spanning tree (MST)

original buildings
Aggregate and Grow

- Aggregate buildings that are too close, when zooming out
- Bridges and buildings constitute a minimum spanning tree (MST)

\[ t = 0 \]

original buildings

\[ t \text{ increases} \]
Aggregate and Grow

- Aggregate buildings that are too close, when zooming out
- Bridges and buildings constitute a minimum spanning tree (MST)

<table>
<thead>
<tr>
<th>Original buildings</th>
<th>Add bridge</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t = 0$</td>
<td>$t = 0.4$</td>
</tr>
</tbody>
</table>

zoom out: $t$ increases
Aggregate and Grow

- Aggregate buildings that are too close, when zooming out
- Bridges and buildings constitute a minimum spanning tree (MST)

Original buildings

\[ t = 0 \]

Add bridge

\[ t = 0.4 \]

Grow

Zoom out:
\[ t \text{ increases} \]
Aggregate and Grow

- Aggregate buildings that are too close, when zooming out
- Bridges and buildings constitute a minimum spanning tree (MST)

```
• Aggregate buildings that are too close, when zooming out
• Bridges and buildings constitute a minimum spanning tree (MST)
```

```
original buildings

\[
t = 0
\]

add bridge

\[
t = 0.4
\]

grow

\[
t = 0.6
\]

zoom out: \( t \) increases
Aggregate and Grow

- Aggregate buildings that are too close, when zooming out
- Bridges and buildings constitute a minimum spanning tree (MST)

Original buildings $t = 0$

Add bridge grow $t = 0.4$

Add bridge grow $t = 0.6$

Zoom out: $t$ increases
Aggregate and Grow

- Aggregate buildings that are too close, when zooming out
- Bridges and buildings constitute a minimum spanning tree (MST)

![Diagram showing the process of aggregate and grow with different values of t (0, 0.4, 0.6, 1).]

• Zoom out: t increases

original buildings

\[ t = 0 \]

add bridge

\[ t = 0.4 \]

grow

add bridge

\[ t = 0.6 \]

grow

add bridge

\[ t = 1 \]
Aggregate and Grow

- Aggregate buildings that are too close, when zooming out.
- Bridges and buildings constitute a minimum spanning tree (MST).

![Diagram showing aggregate buildings and zooming out](attachment:image.png)

- Original buildings at $t = 0$
- Add bridge at $t = 0.4$
- Grow at $t = 0.6$
- Add bridge at $t = 1$

*zoom out: $t$ increases*
Three Join Types of Buffering

building
Three Join Types of Buffering

mite: keep right angles

building
Three Join Types of Buffering

building

miter: keep right angles

$d_G$
Three Join Types of Buffering

- **building**
  - square: avoid long spikes

- miter:
  - keep right angles
  - $d_G$
Three Join Types of Buffering

miter: keep right angles

square: avoid long spikes
Three Join Types of Buffering

- **miter:** keep right angles
- **square:** avoid long spikes
- **round:** detect if two buildings are too close
Simplifying Based on Dilation and Erosion

d
polygon
Simplifying Based on Dilation and Erosion

dilate with $d$: remove dents
Simplifying Based on Dilation and Erosion

dilate with \( d \): remove dents

erode with \( 2d \): remove bumps
Simplifying Based on Dilation and Erosion

dilate with $d$: remove dents

erode with $2d$: remove bumps

dilate with $d$
Case Study

- Runtime: $O(n^3)$,
  $n$: total number of edges over all input buildings
Case Study

- Runtime: $O(n^3)$,
  $n$: total number of edges over all input buildings
- Environment
  C#, Clipper (for buffering, dilation, erosion, and merge)
Case Study

• Runtime: $O(n^3)$,
  $n$: total number of edges over all input buildings

• Environment
  C#, Clipper (for buffering, dilation, erosion, and merge)

• Data: 2.5 k buildings, $n = 19$ k edges, 1 : 15 k,
  $d_G = 25$ m (IGN)
Case Study

- Runtime: $O(n^3)$, $n$: total number of edges over all input buildings
- Environment
  C#, Clipper (for buffering, dilation, erosion, and merge)
- Data: 2.5 k buildings, $n = 19$ k edges, 1 : 15 k, $d_G = 25$ m (IGN)
- 12 min for computing a sequence of 10 maps
Animation

zooming out

400 m
Animation

zooming out

400 m
Animation

zooming out

400 m
Animation

zooming out

400 m
Animation

zooming out

400 m
Animation

zooming out

400 m
Animation

400 m

zooming out
Animation

zooming out

400 m
Animation

zooming out

400 m
Animation

zooming out

400 m
Contents of Thesis

Optimal sequence for aggregation

Optim.

A*

ILP

Related Generalization

Aggregation

Classification

Administrative boundaires

DP

Buildings to built-up areas

MST

Exaggeration

Aggregation, Simplification, Elimination

Morphing polylines

LSA

DP

Simplification

Choosing right data structures

\(\epsilon\)

\(p\)

SortedDictionary,

SortedSet, \ldots
Conclusion

• Studied four topics of continuous generalization
Conclusion

- Studied four topics of continuous generalization
- Used optimization methods to attain good results
Conclusion

- Studied *four* topics of continuous generalization
- Used optimization methods to attain good results
- Shared experience of using right data structures
Conclusion

• Studied four topics of continuous generalization
• Used optimization methods to attain good results
• Shared experience of using right data structures

Future work

• Improve our methods
Conclusion

- Studied four topics of continuous generalization
- Used optimization methods to attain good results
- Shared experience of using right data structures

Future work

- Improve our methods
- Usability testing
Conclusion

- Studied **four** topics of continuous generalization
- Used optimization methods to attain good results
- Shared experience of using right data structures

Future work

- Improve our methods
- Usability testing
- Work on complete maps (with roads, buildings, ...)

scroll
Conclusion

- Studied four topics of continuous generalization
- Used optimization methods to attain good results
- Shared experience of using right data structures

Future work

- Improve our methods
- Usability testing
- Work on complete maps (with roads, buildings, ...)

Thank you!