Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Chair for
INFORMATICS I
Efficient Algorithms and
Knowledge-Based Systems

Institute for Informatics

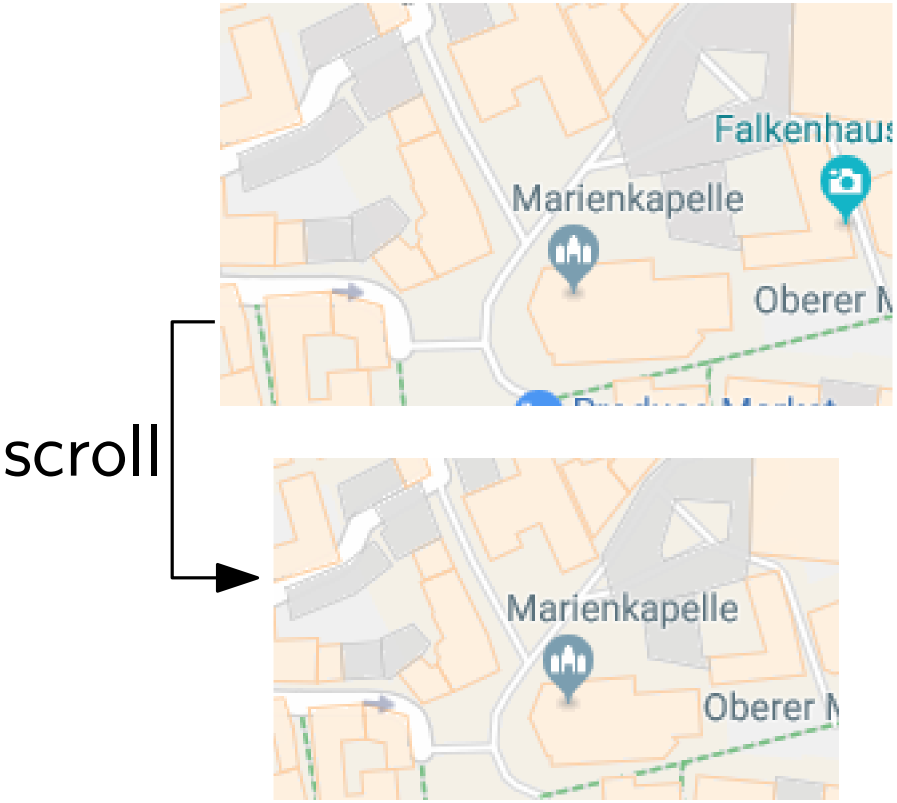
An Optimization-Based Approach for Continuous Map Generalization

Dongliang Peng

Chair of Computer Science I, University of Würzburg, Germany

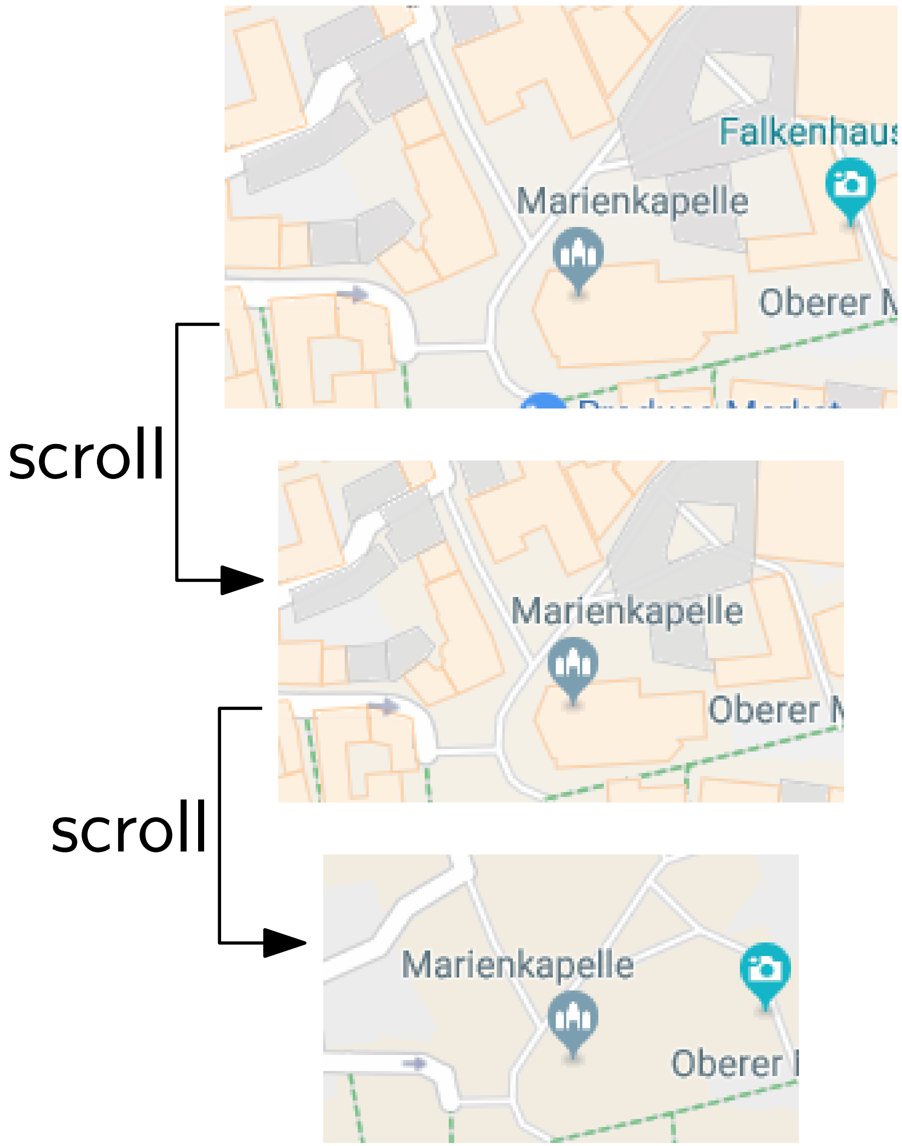
Marienkapelle
[source: Wikipedia]

[Google Maps]



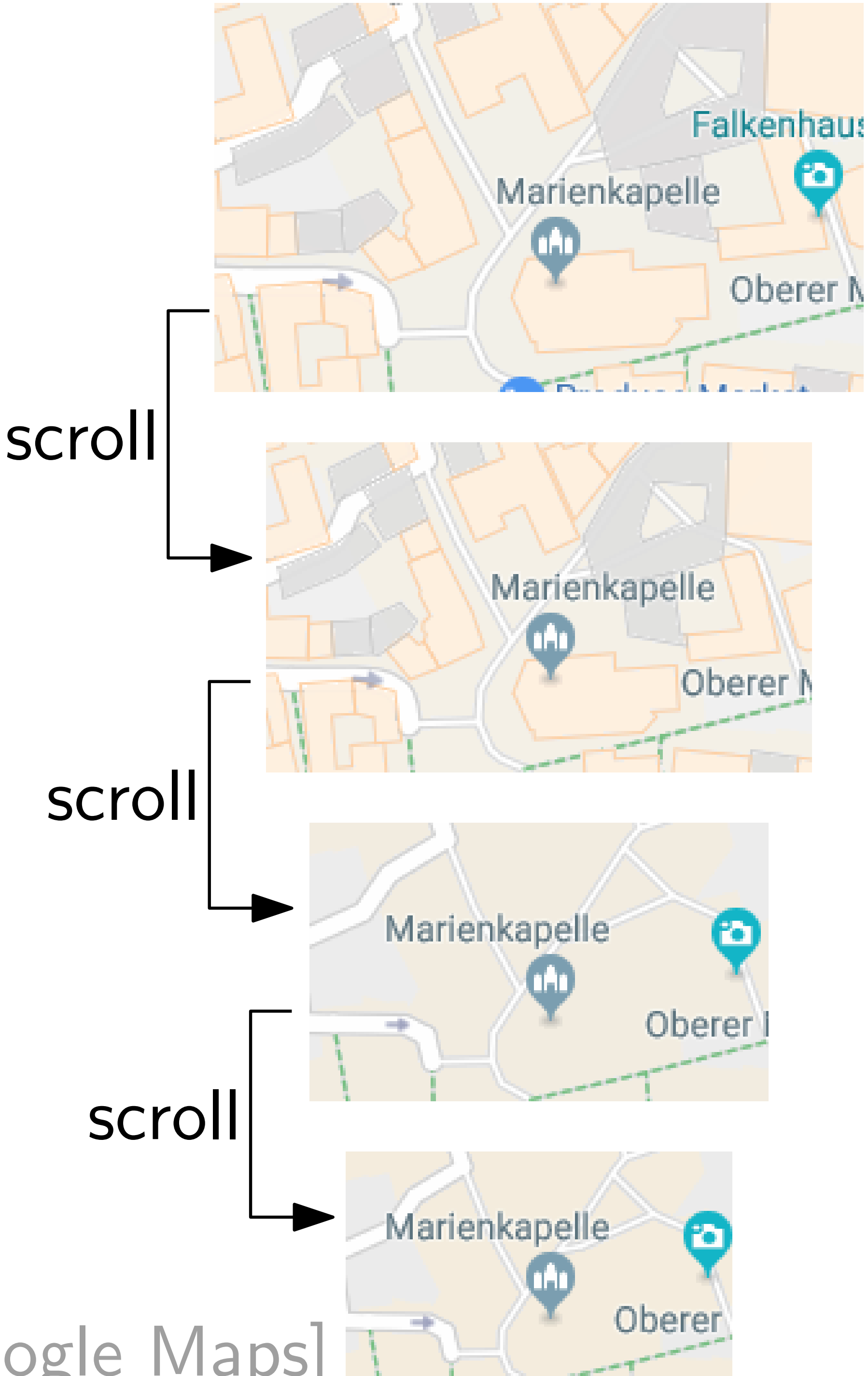
Marienkapelle
[source: Wikipedia]

[Google Maps]



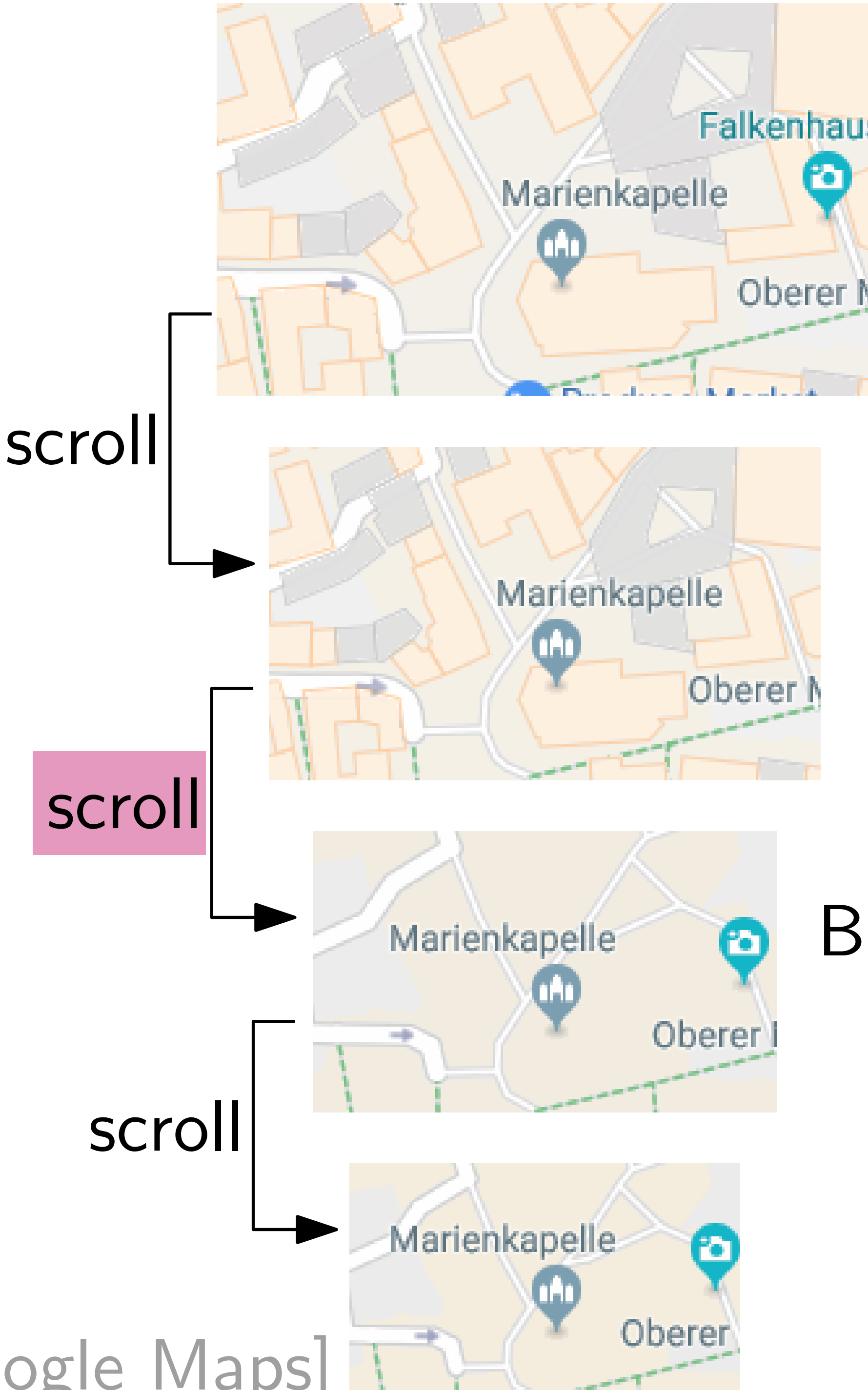
Marienkapelle
[source: Wikipedia]

[Google Maps]



[Google Maps]

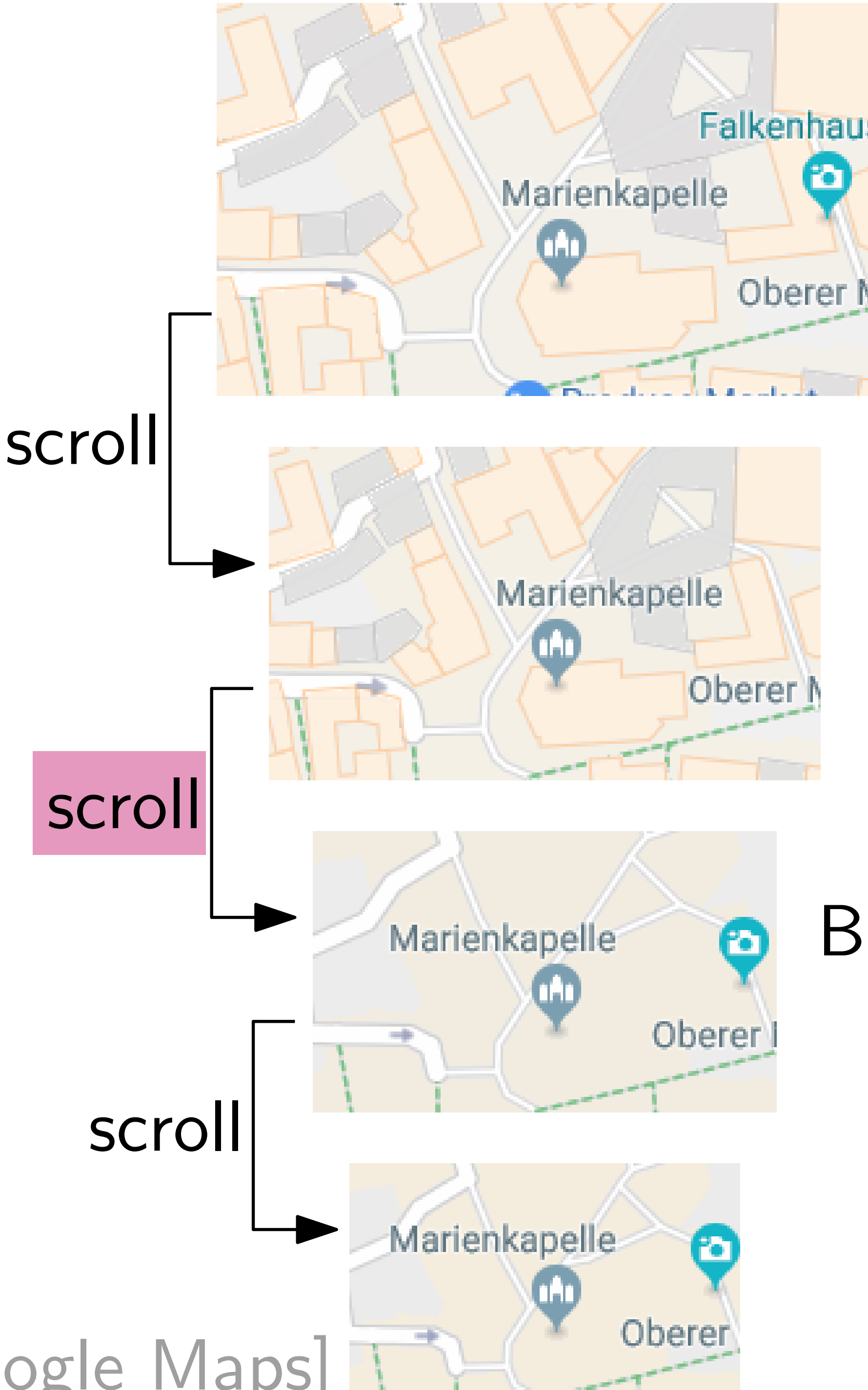
Marienkapelle
[source: Wikipedia]



[Google Maps]

Marienkapelle
[source: Wikipedia]

Buildings disappear suddenly!



[Google Maps]

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. . .

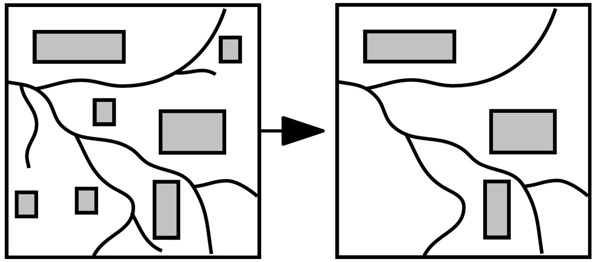
. . . is about deriving a smaller-scale map from an existing map.

Typical generalization operators [ESRI 1996]:

Map Generalization. . .

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

Typical generalization operators [ESRI 1996]:

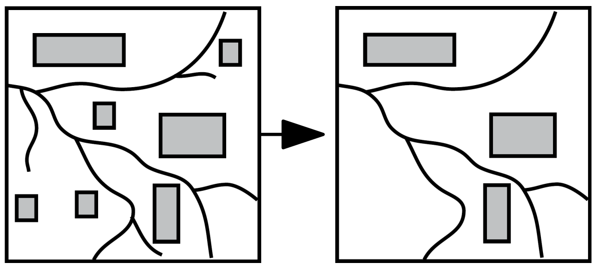


Elimination

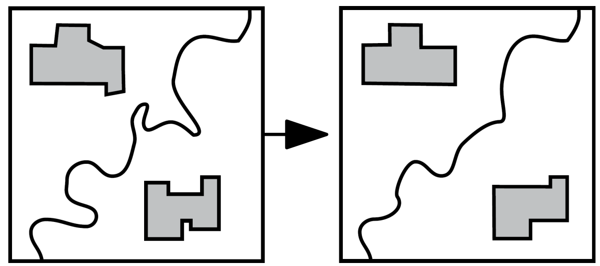
Map Generalization. . .

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

Typical generalization operators [ESRI 1996]:



Elimination

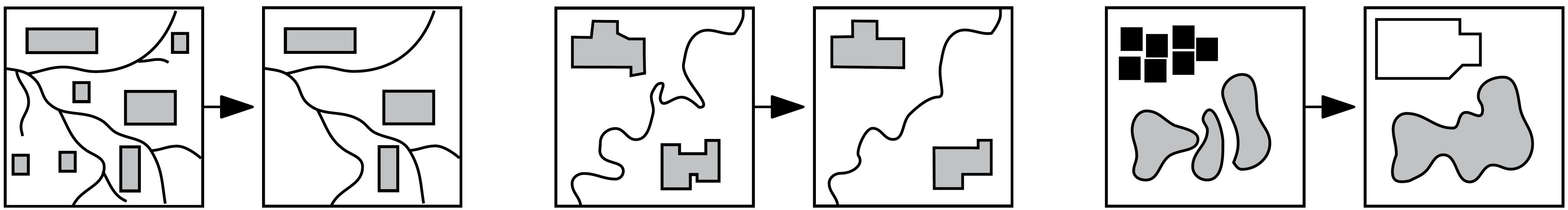


Simplification

Map Generalization. . .

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

Typical generalization operators [ESRI 1996]:

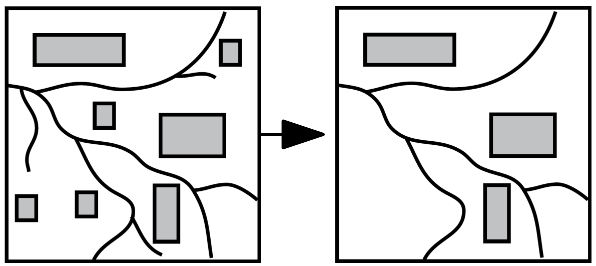


Elimination Simplification Aggregation

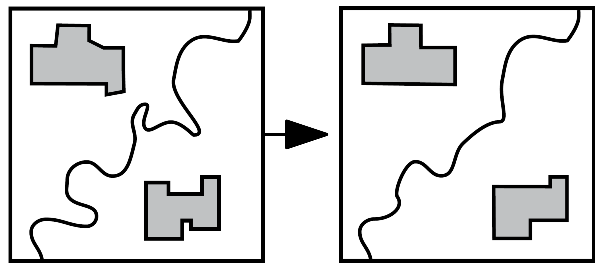
Map Generalization. . .

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

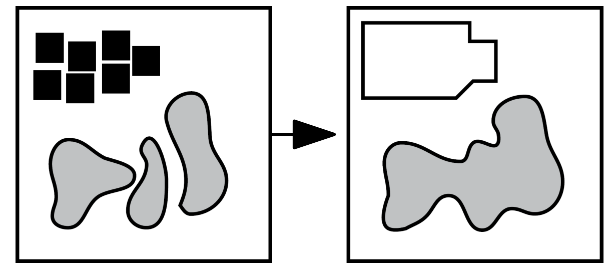
Typical generalization operators [ESRI 1996]:



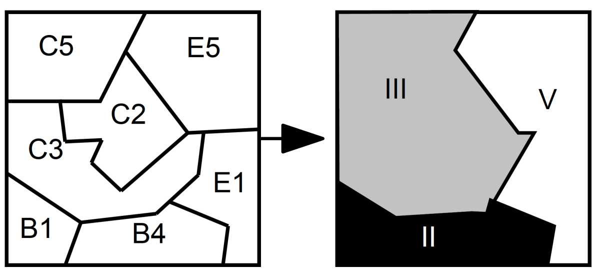
Elimination



Simplification



Aggregation

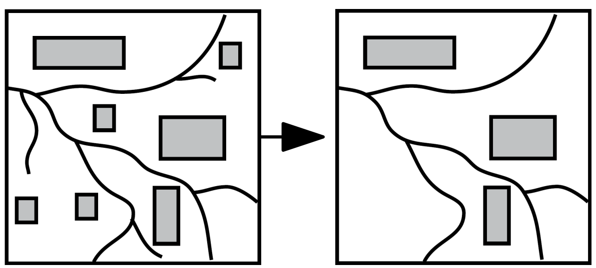


Classifi. & Symboli.

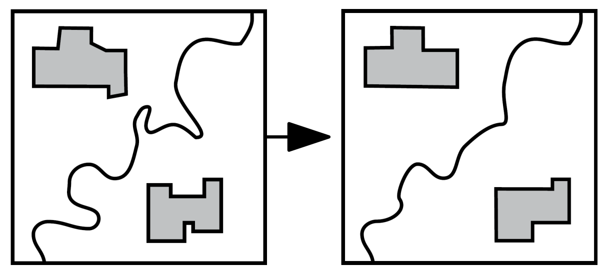
Map Generalization. . .

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

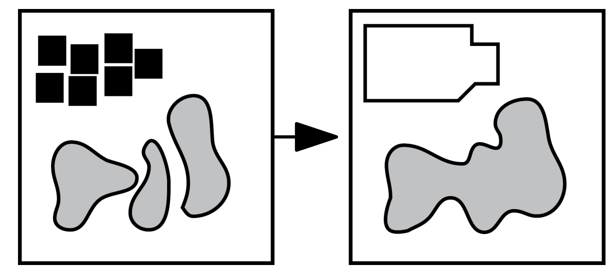
Typical generalization operators [ESRI 1996]:



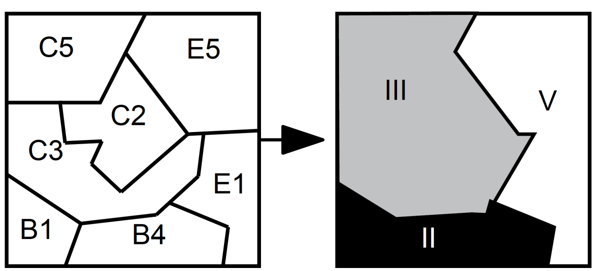
Elimination



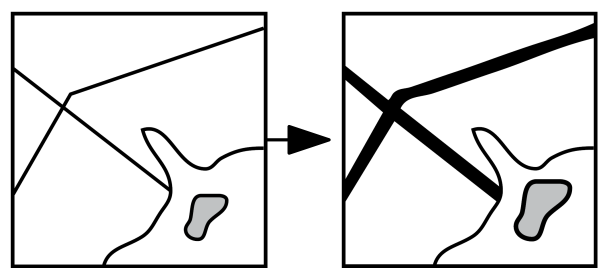
Simplification



Aggregation



Classifi. & Symboli.

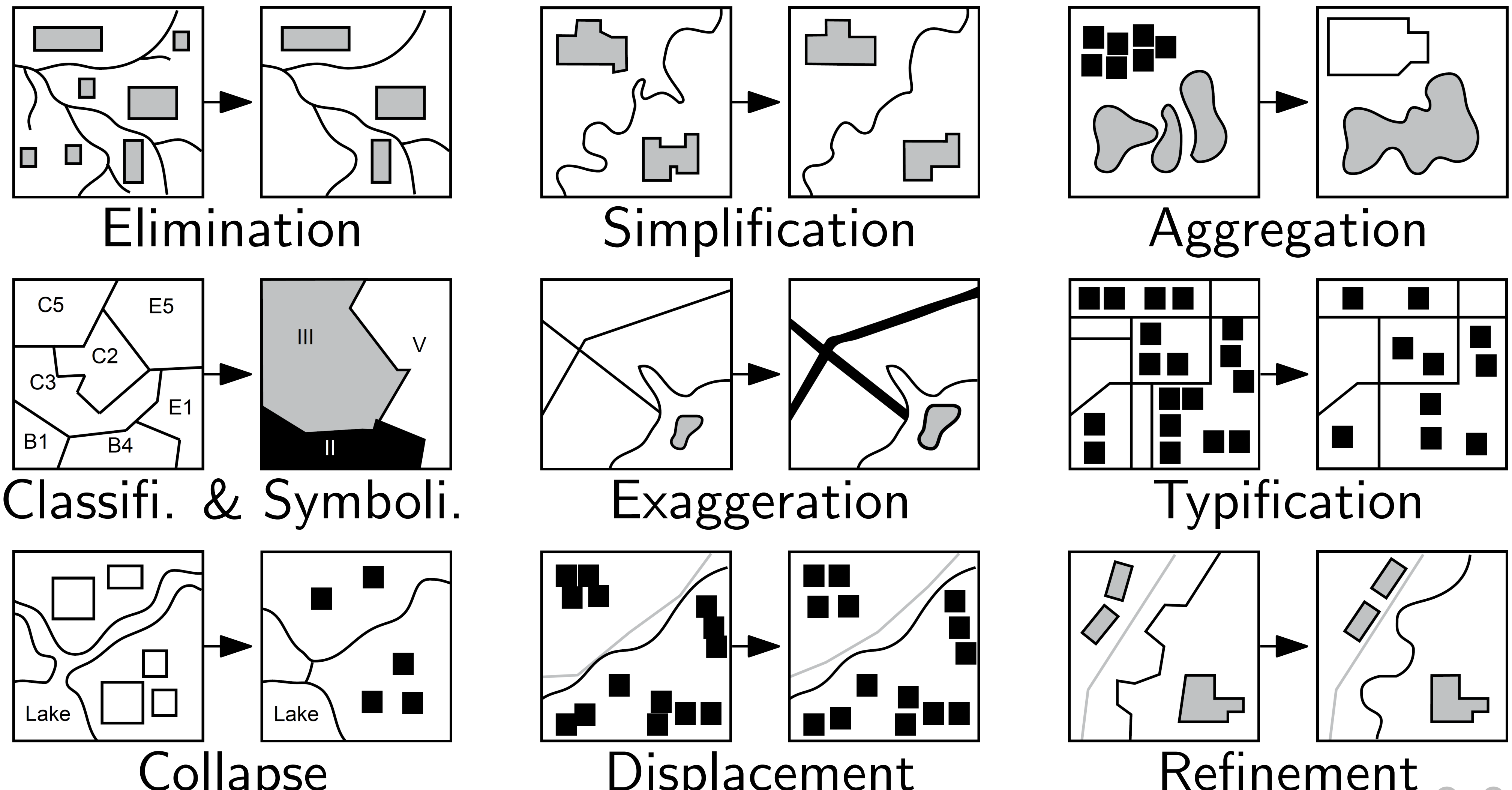


Exaggeration

Map Generalization. . .

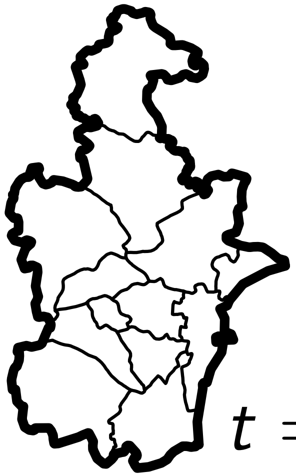
. . . is about deriving a smaller-scale map from an existing map.

Typical generalization operators [ESRI 1996]:



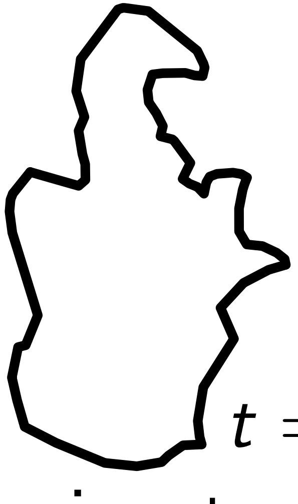
Continuous Map Generalization...

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



$t = 0$

input

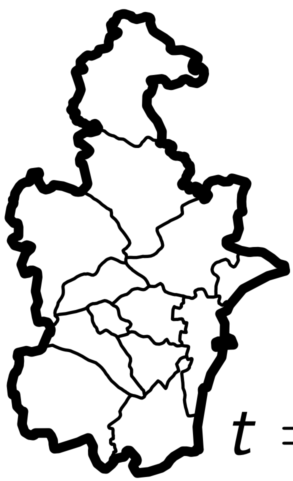


$t = 1$

input

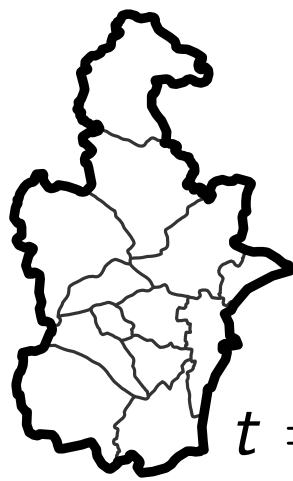
Continuous Map Generalization...

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

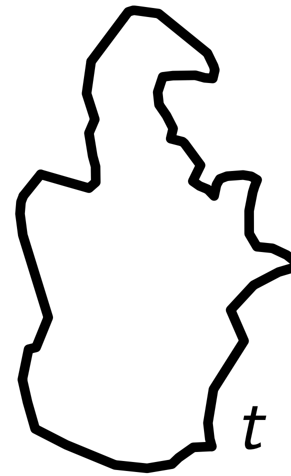


$t = 0$

input



$t = 0.2$

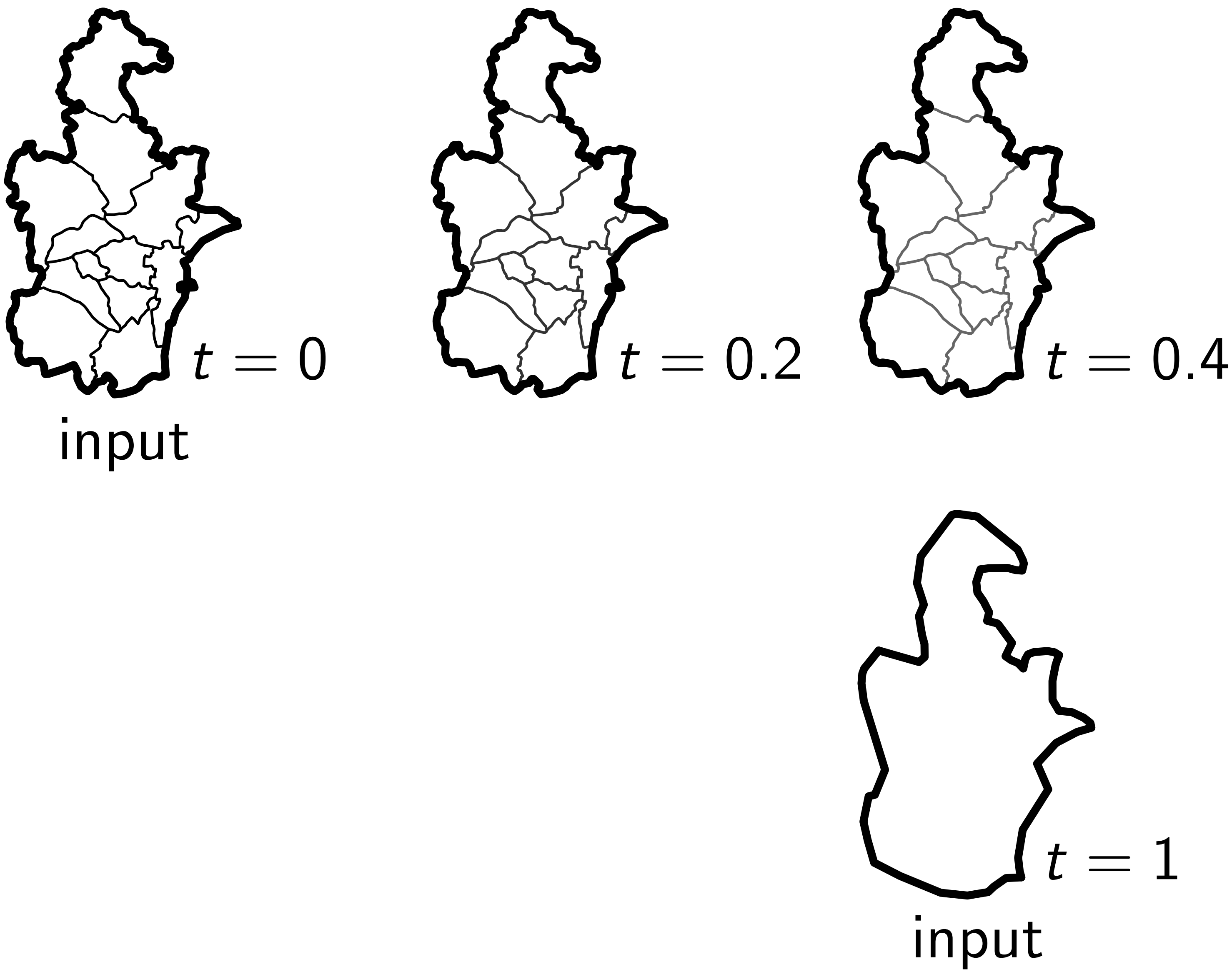


$t = 1$

input

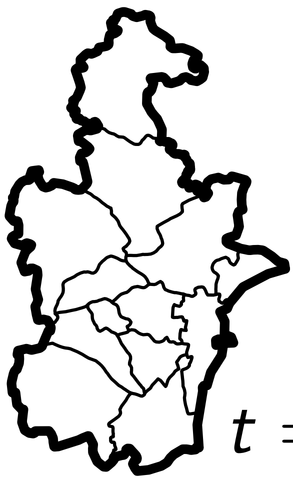
Continuous Map Generalization...

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



Continuous Map Generalization...

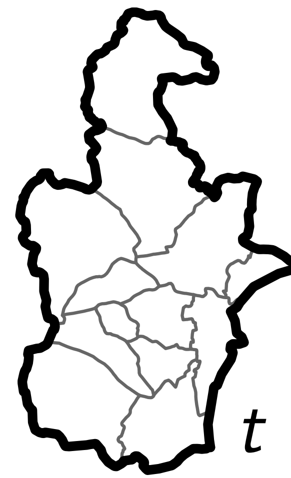
. . . is to derive a series of maps with smooth changes.



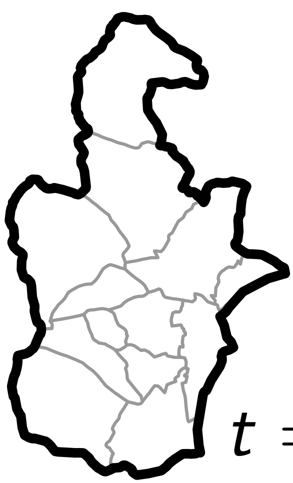
$t = 0$

input

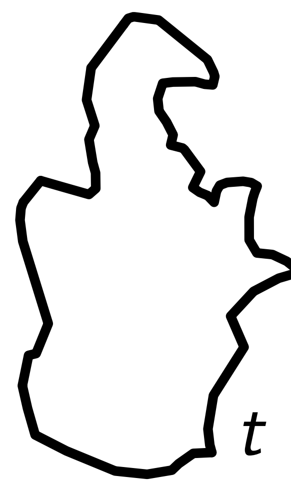
$t = 0.2$



$t = 0.4$



$t = 0.6$

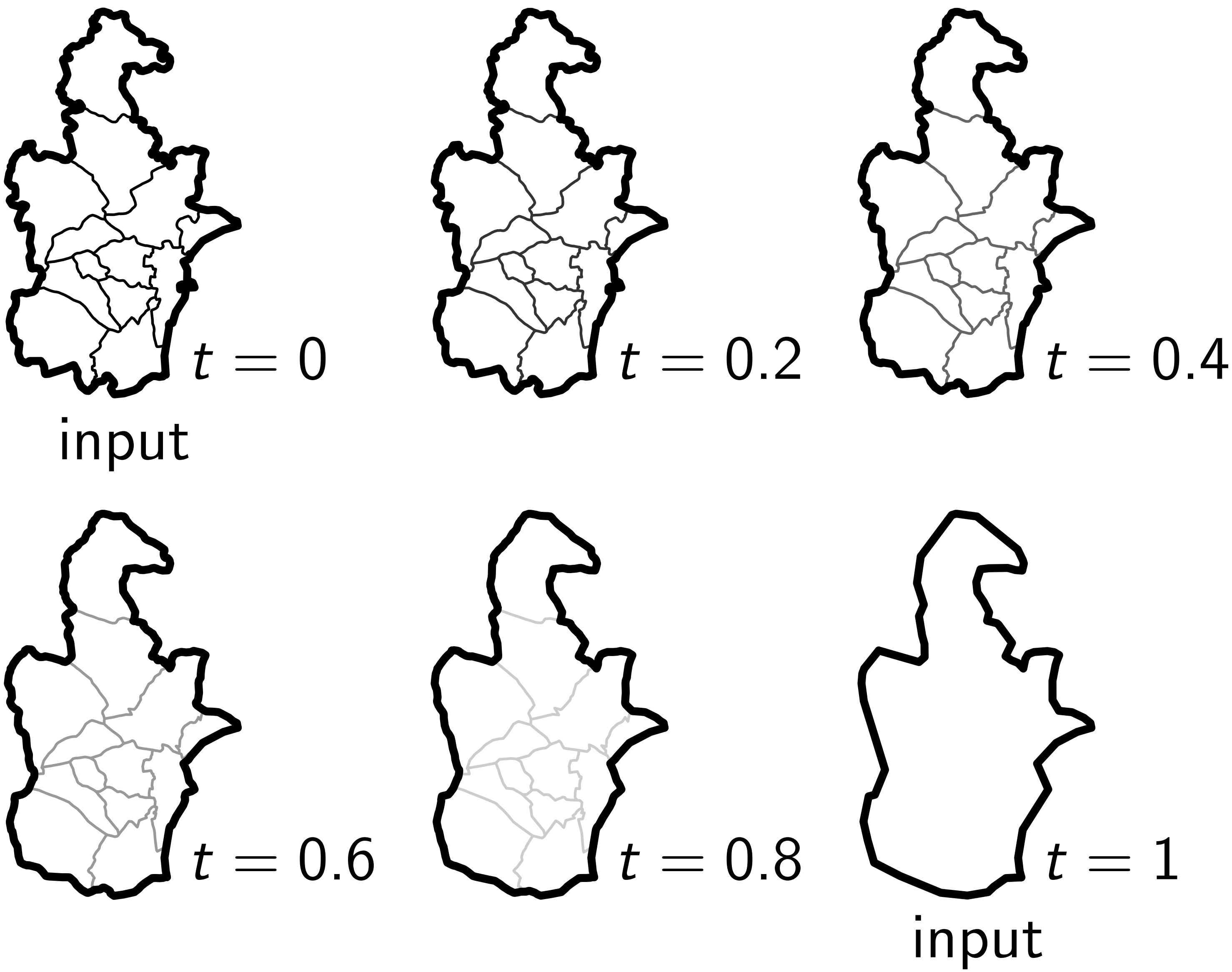


$t = 1$

input

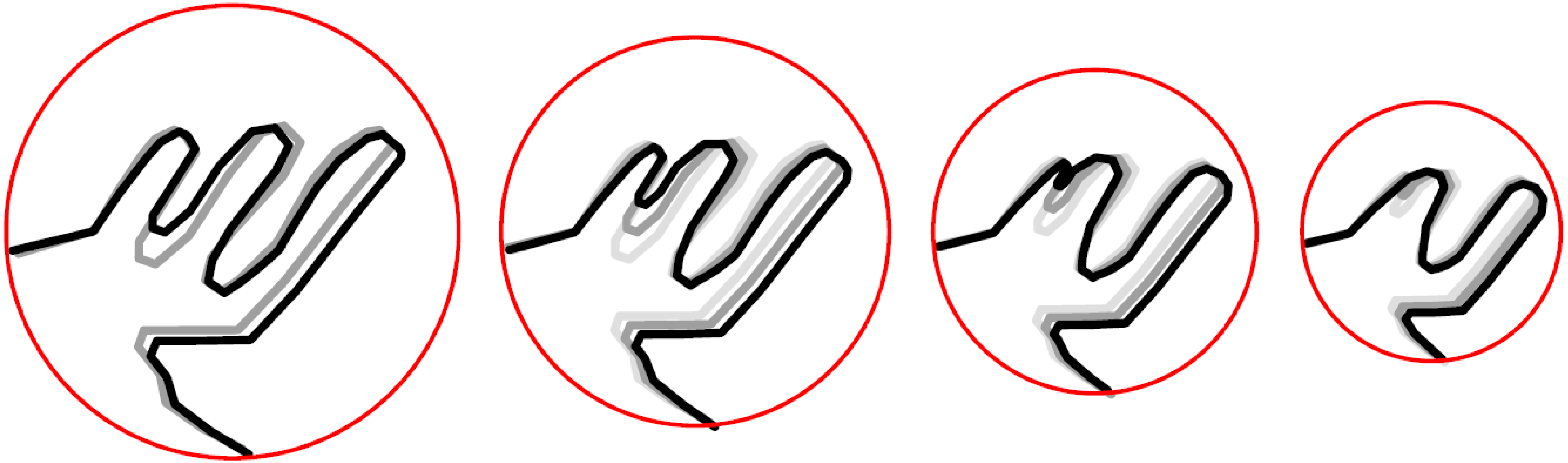
Continuous Map Generalization...

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



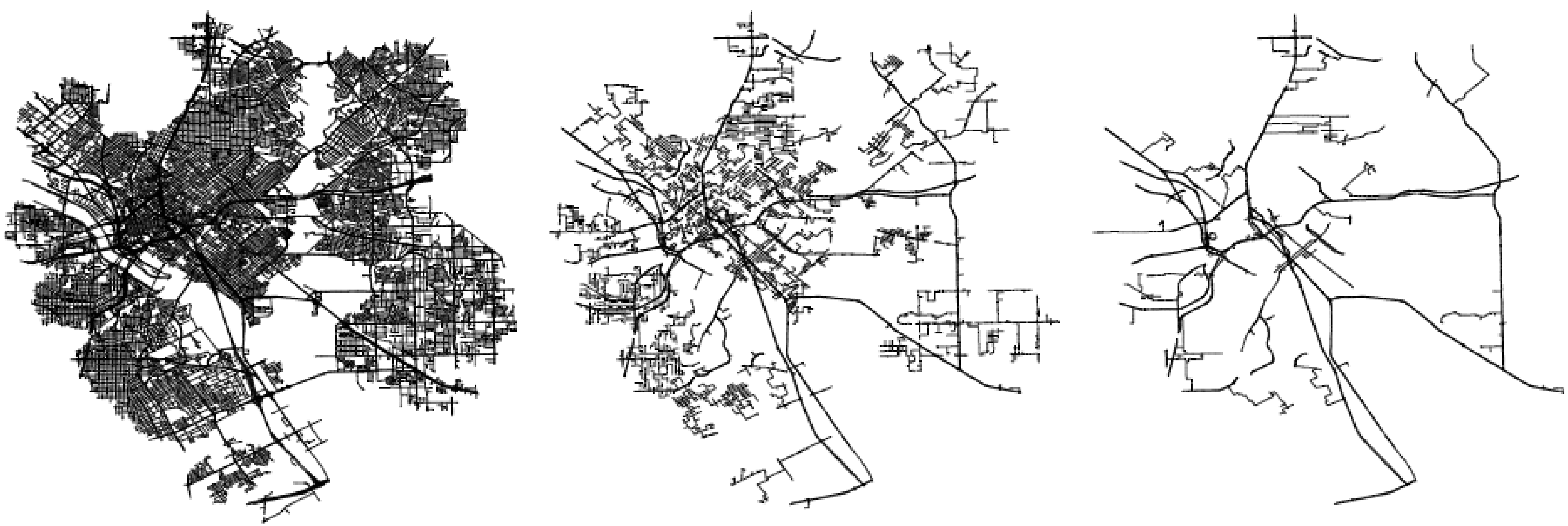
Related Work

- Morph between polylines [Nöllenburg et al. 2008]



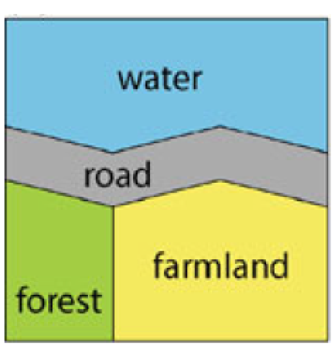
Related Work

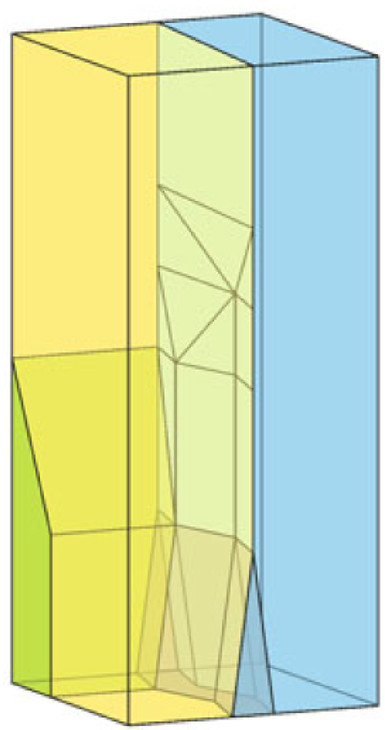
- Morph between polylines [Nöllenburg et al. 2008]
- Generate a good sequence of maps [Chimani et al. 2014]



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]

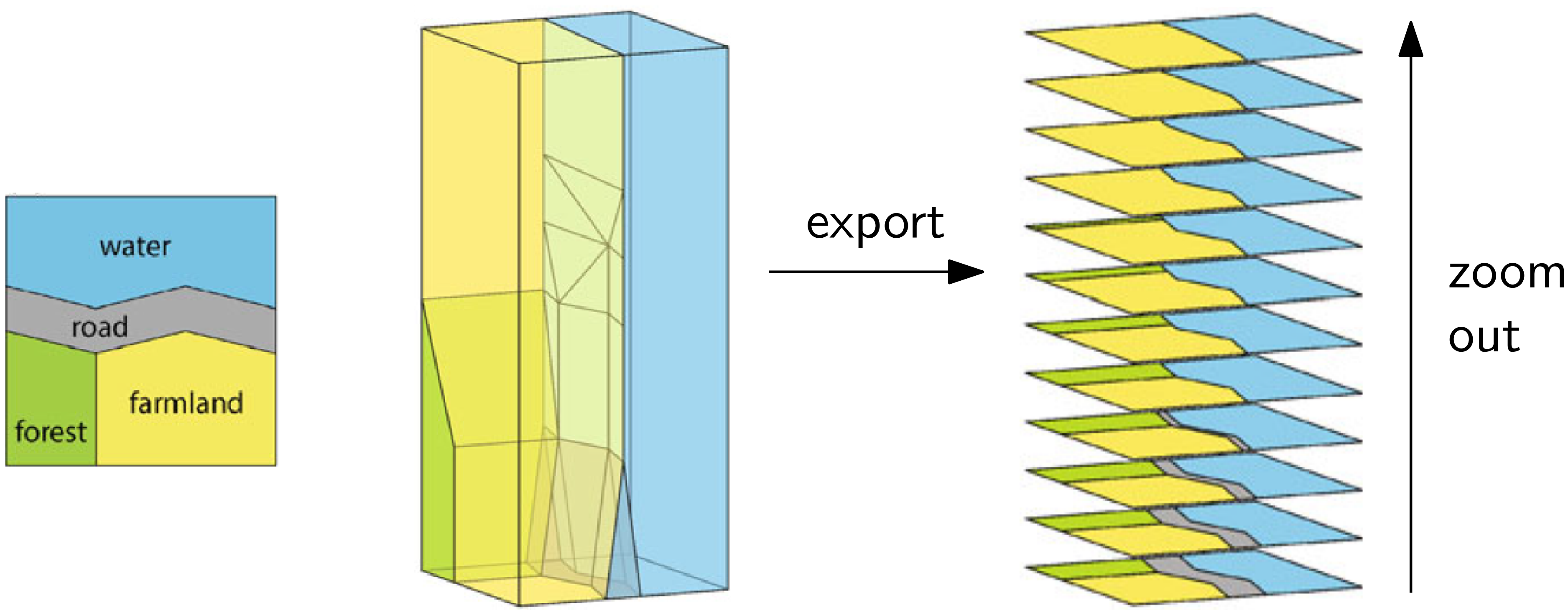




space scale cube (SSC)

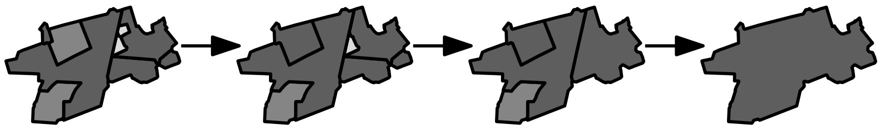
Related Work

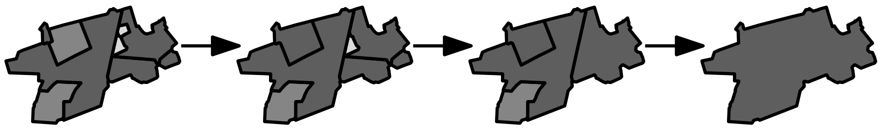
- Morph between polylines [Nöllenburg et al. 2008]
- Generate a good sequence of maps [Chimani et al. 2014]
- Data structure for continuous generalization [van Oosterom et al. 2014]

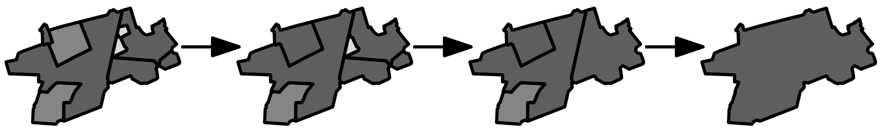
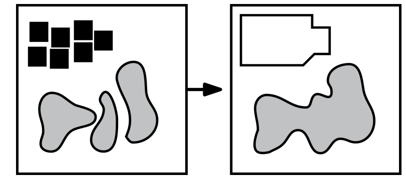
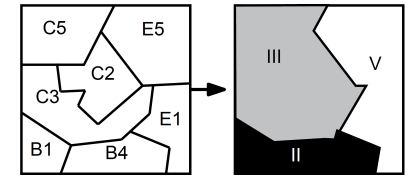


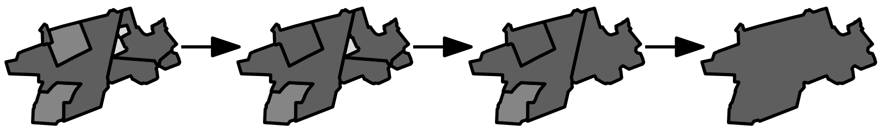
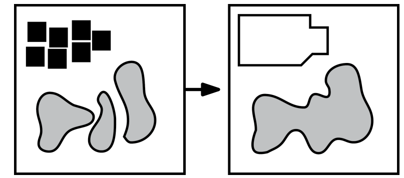
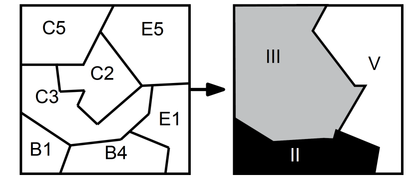
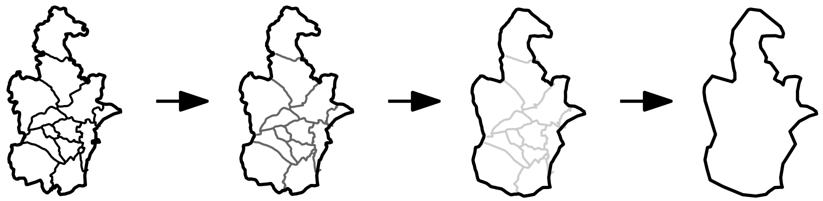
space scale cube (SSC)

Optimal sequence for aggregation



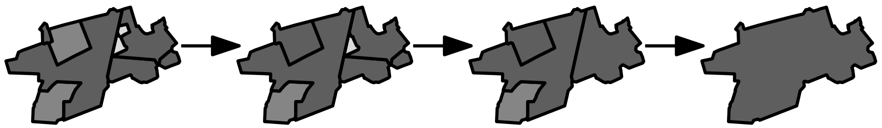
Contents of Thesis	Optim.	Related Generalization
Optimal sequence for aggregation 	$A^\star$ ILP	

Contents of Thesis	Optim.	Related Generalization	
Optimal sequence for aggregation 	$A^\star$ ILP	 Aggregation	 Classification

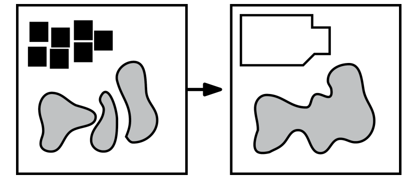
Contents of Thesis	Optim.	Related Generalization
Optimal sequence for aggregation 	A* ILP	 Aggregation  Classification
Administrative boundaires 		

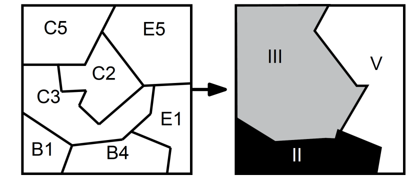
Contents of Thesis	Optim.	Related Generalization

Optimal sequence for aggregation



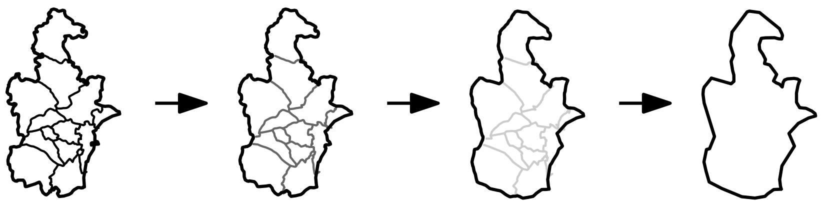
A⋆
ILP



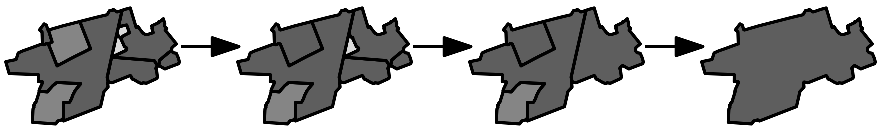
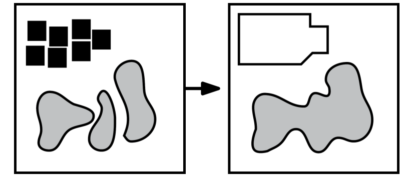
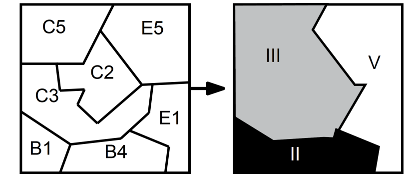
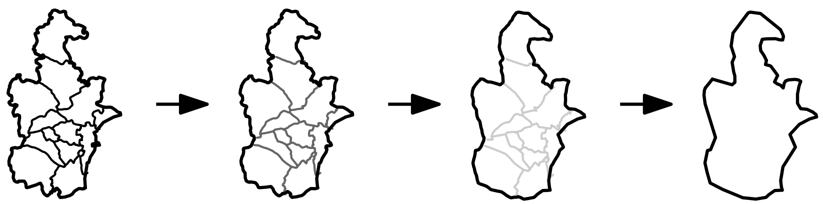
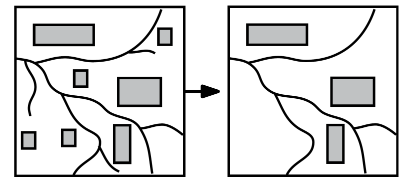
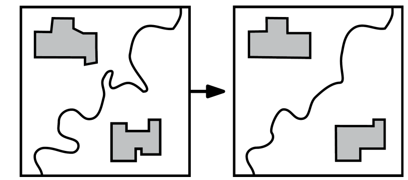


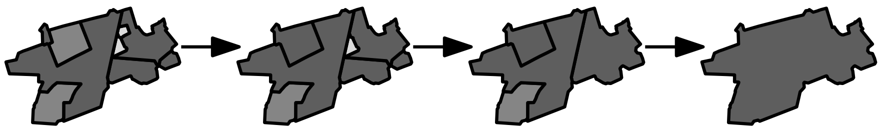
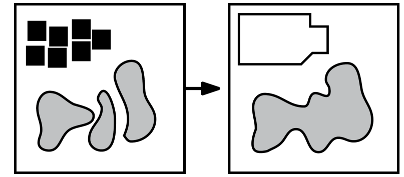
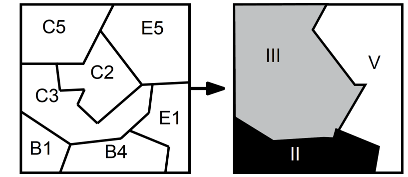
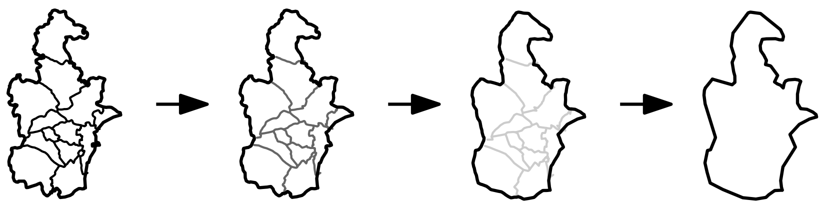
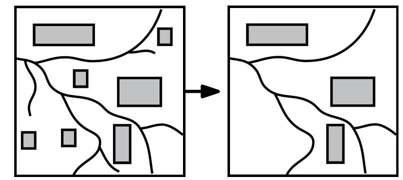
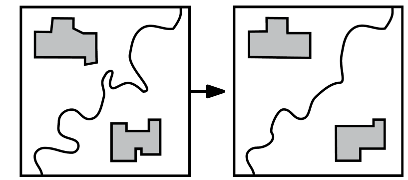
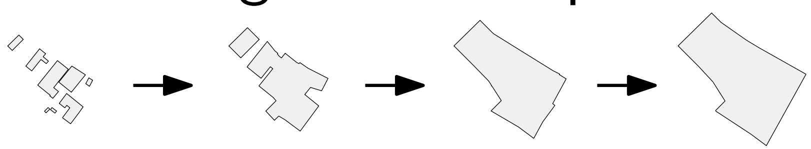
Aggregation Classification

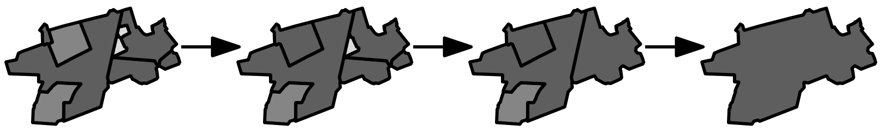
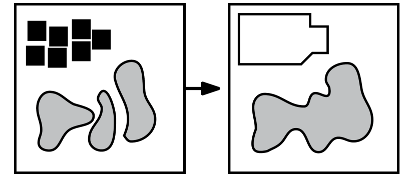
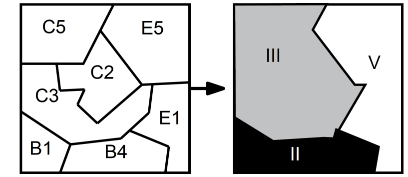
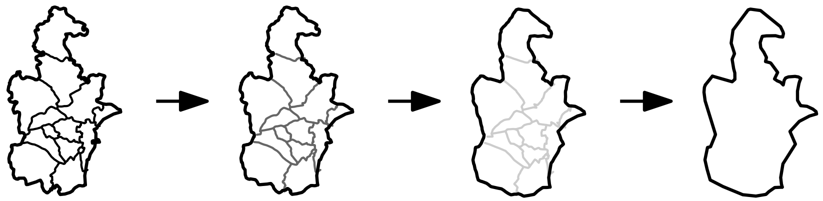
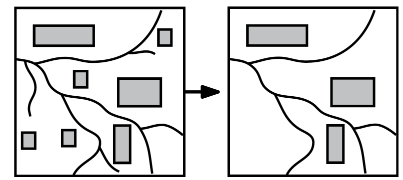
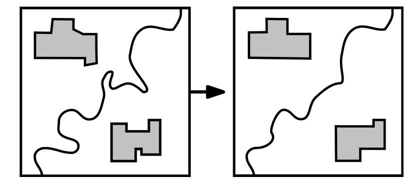
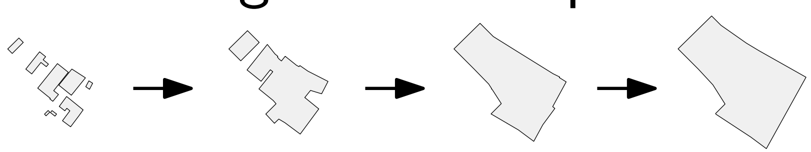
Administrative boundaires



DP

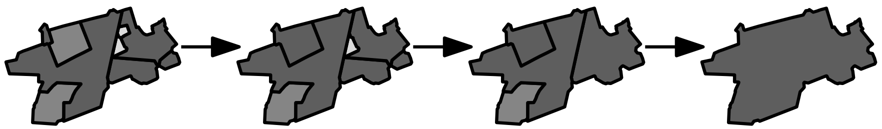
Contents of Thesis	Optim.	Related Generalization
Optimal sequence for aggregation 	A* ILP	 Aggregation  Classification
Administrative boundaires 	DP	 Elimination  Simplification

Contents of Thesis	Optim.	Related Generalization
Optimal sequence for aggregation 	A* ILP	 Aggregation  Classification
Administrative boundaires 	DP	 Elimination  Simplification
Buildings to built-up areas 		

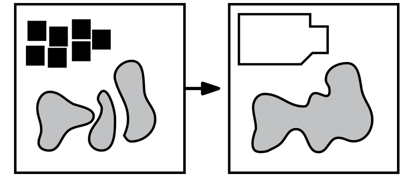
Contents of Thesis	Optim.	Related Generalization
Optimal sequence for aggregation 	A* ILP	 Aggregation  Classification
Administrative boundaires 	DP	 Elimination  Simplification
Buildings to built-up areas 	MST	

Contents of Thesis	Optim.	Related Generalization

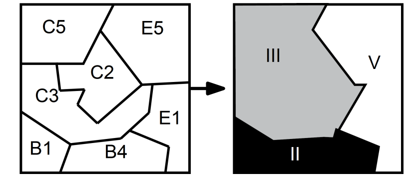
Optimal sequence for aggregation



A⋆
ILP

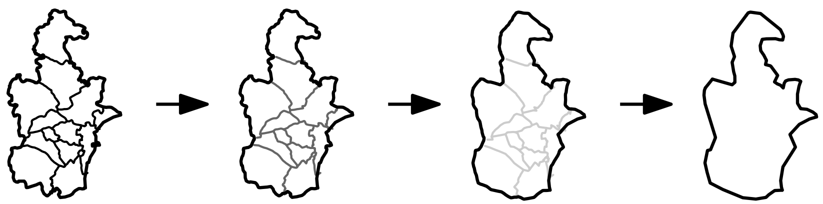


Aggregation

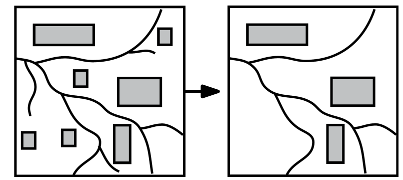


Classification

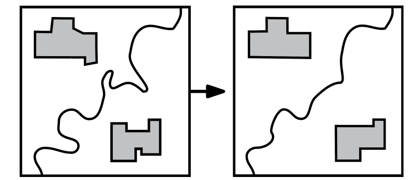
Administrative boundaires



DP

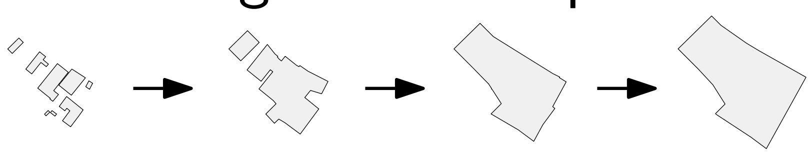


Elimination

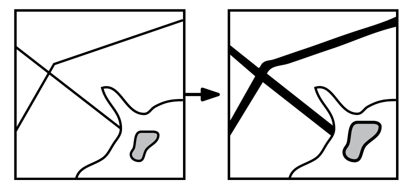


Simplification

Buildings to built-up areas

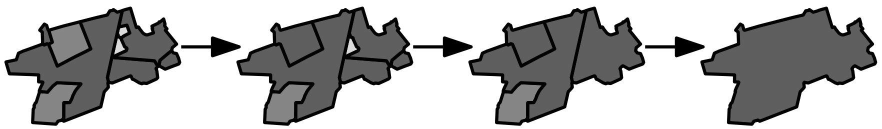
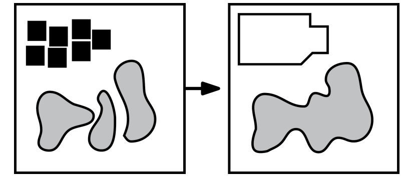
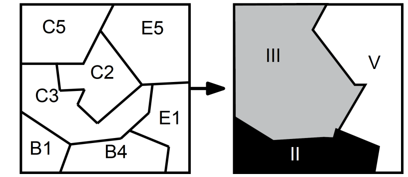
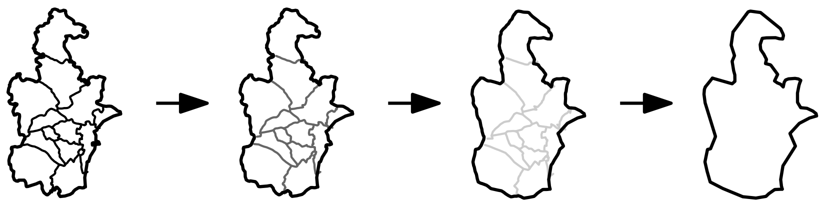
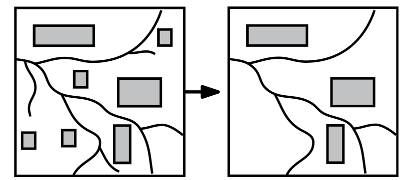
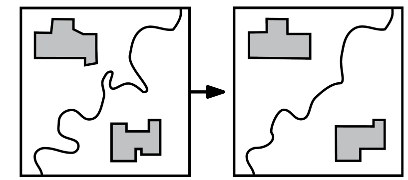
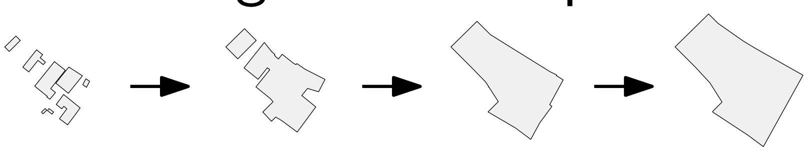
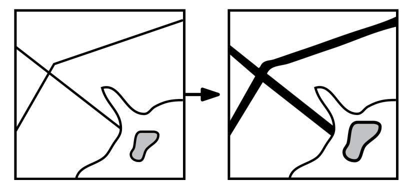
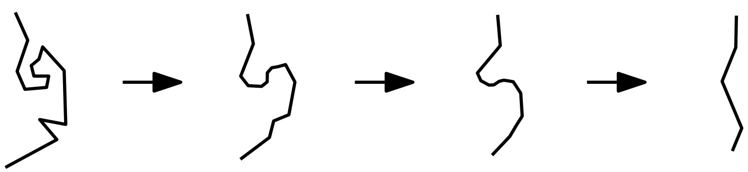


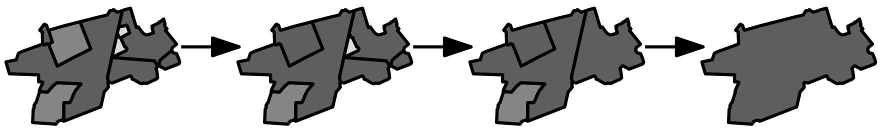
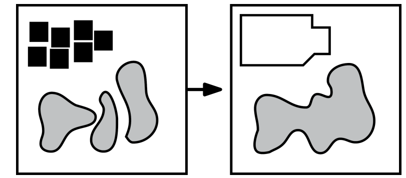
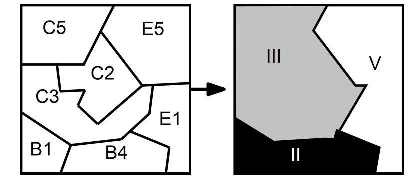
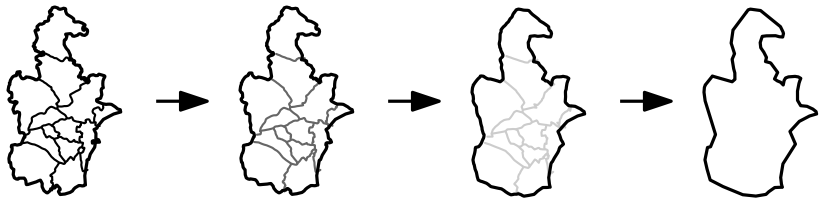
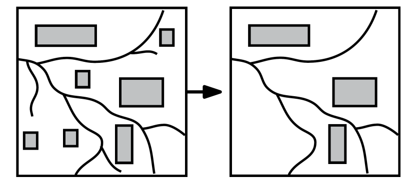
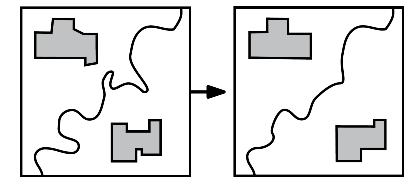
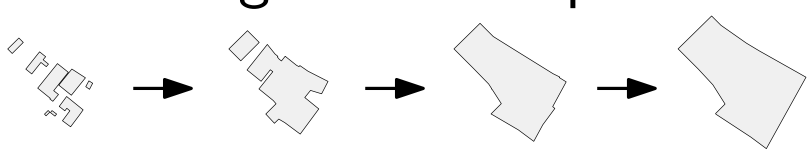
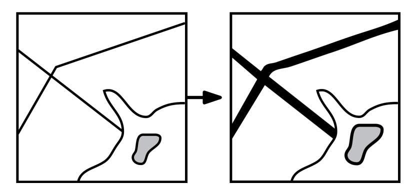
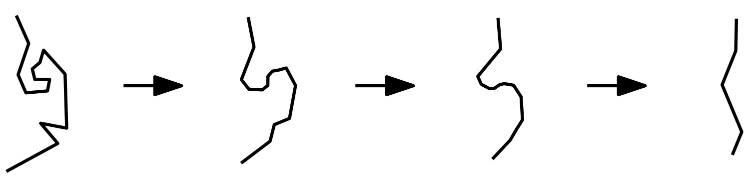
MST

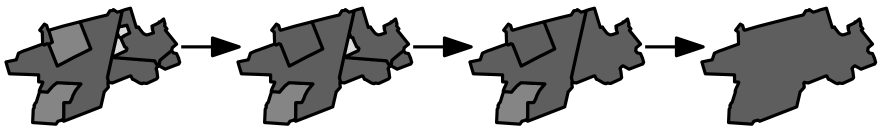
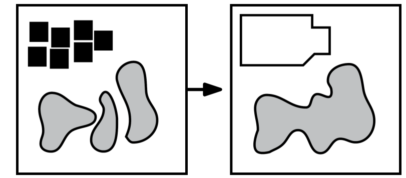
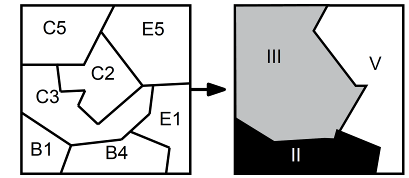
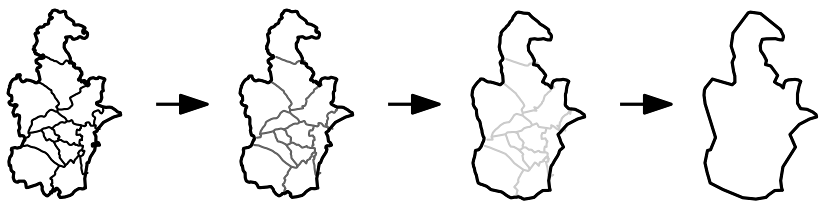
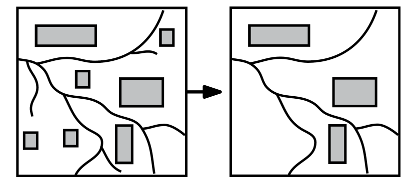
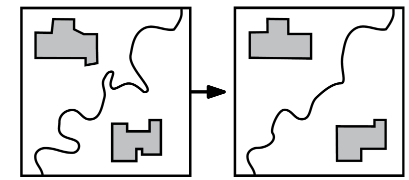
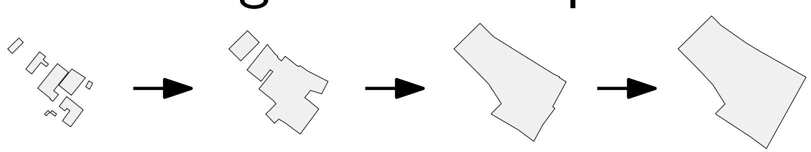
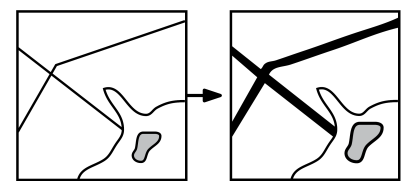
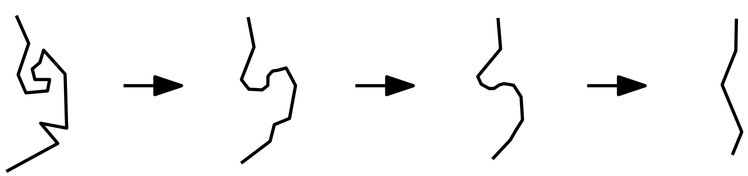
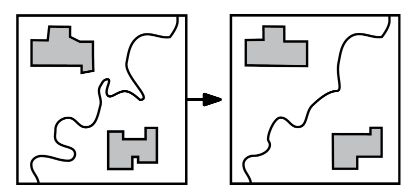


Exaggeration

Aggregation,
Simplification,
Elimination

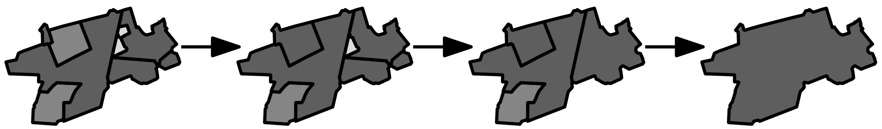
Contents of Thesis	Optim.	Related Generalization	
Optimal sequence for aggregation 	$A^{\star}$ ILP	 Aggregation	 Classification
Administrative boundaires 	DP	 Elimination	 Simplification
Buildings to built-up areas 	MST	 Exaggeration	Aggregation, Simplification, Elimination
Morphing polylines 			

Contents of Thesis	Optim.	Related Generalization	
Optimal sequence for aggregation 	A* ILP	 Aggregation	 Classification
Administrative boundaires 	DP	 Elimination	 Simplification
Buildings to built-up areas 	MST	 Exaggeration	Aggregation, Simplification, Elimination
Morphing polylines 	LSA DP		

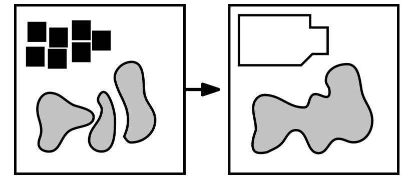
Contents of Thesis	Optim.	Related Generalization	
Optimal sequence for aggregation 	$A^\star$ ILP	 Aggregation	 Classification
Administrative boundaires 	DP	 Elimination	 Simplification
Buildings to built-up areas 	MST	 Exaggeration	Aggregation, Simplification, Elimination
Morphing polylines 	LSA DP	 Simplification	

Contents of Thesis	Optim.	Related Generalization

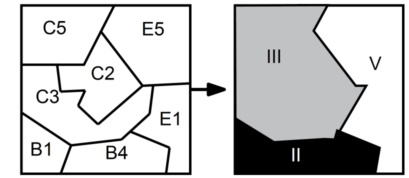
Optimal sequence for aggregation



A* ILP

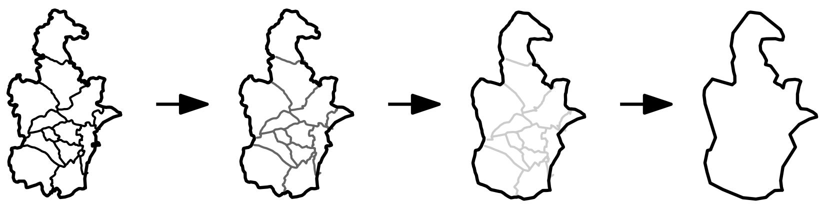


Aggregation

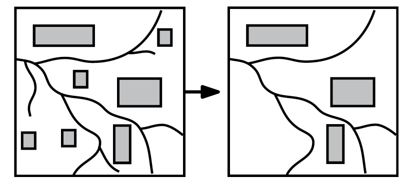


Classification

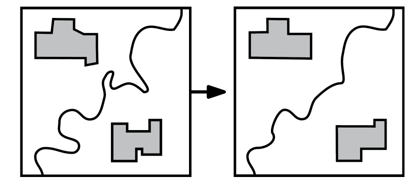
Administrative boundaires



DP

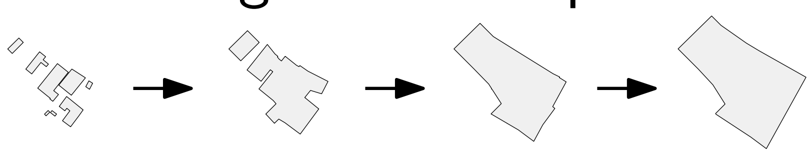


Elimination

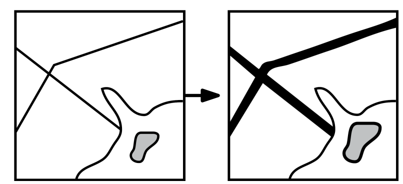


Simplification

Buildings to built-up areas



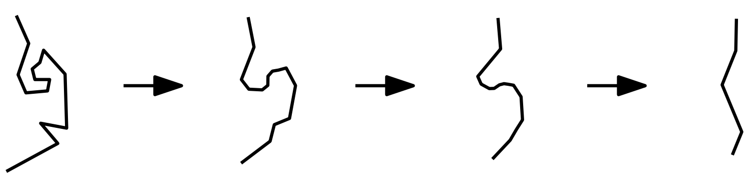
MST



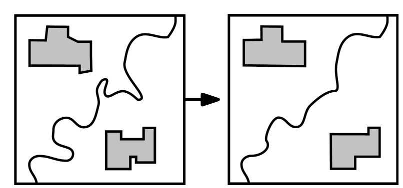
Exaggeration

Aggregation, Simplification, Elimination

Morphing polylines

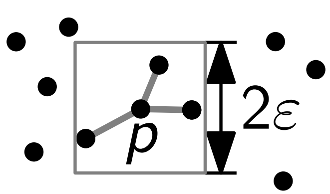


LSA DP

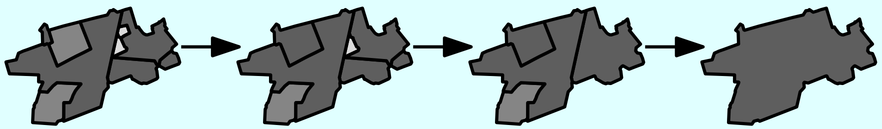
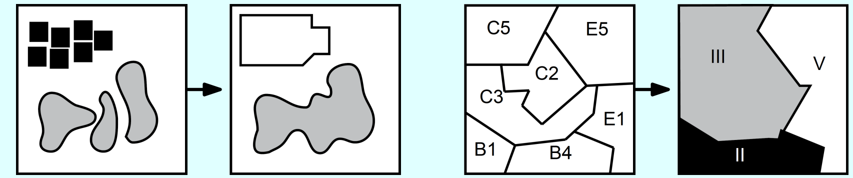
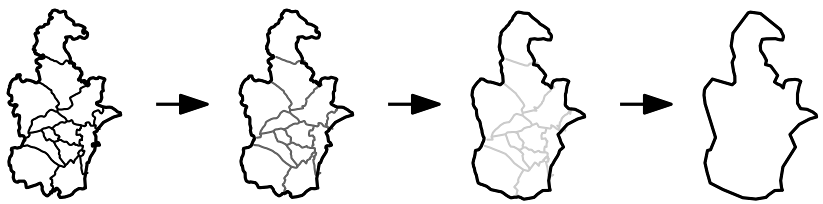
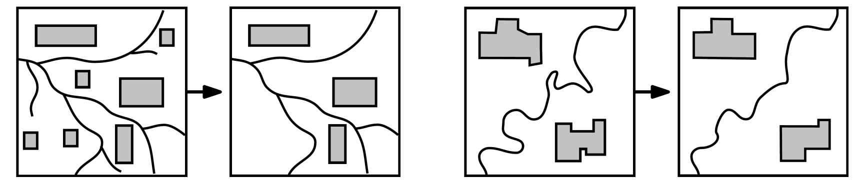
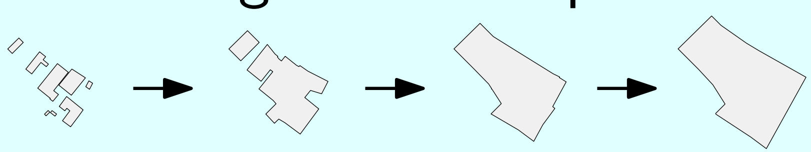
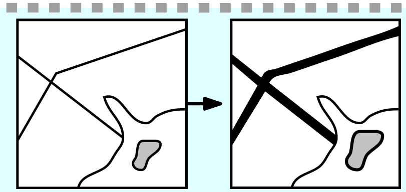
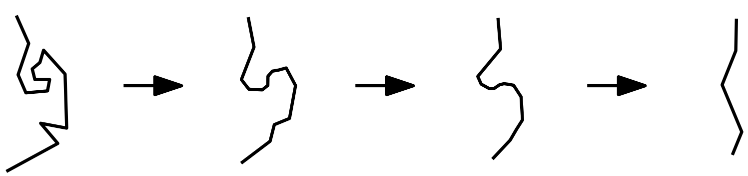
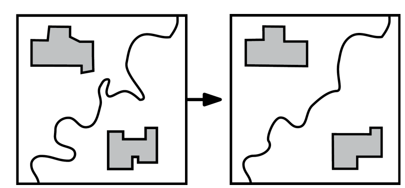
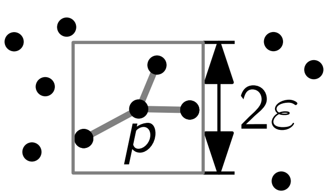


Simplification

Choosing right data structures

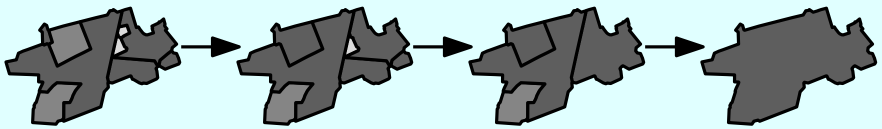


SortedDictionary, SortedSet, ...

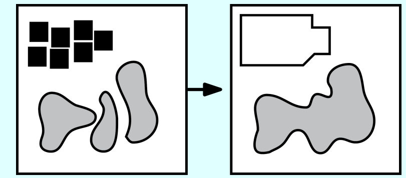
Contents of Thesis	Optim.	Related Generalization
Optimal sequence for aggregation 	$A^\star$ ILP	 Aggregation Classification
Administrative boundaires 	DP	 Elimination Simplification
Buildings to built-up areas 	MST	 Exaggeration Aggregation, Simplification, Elimination
Morphing polylines 	LSA DP	 Simplification
Choosing right data structures 	SortedDictionary, SortedSet, ...	

Contents of Thesis	Optim.	Related Generalization

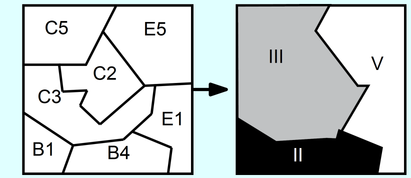
Optimal sequence for aggregation



$A^\star$
ILP

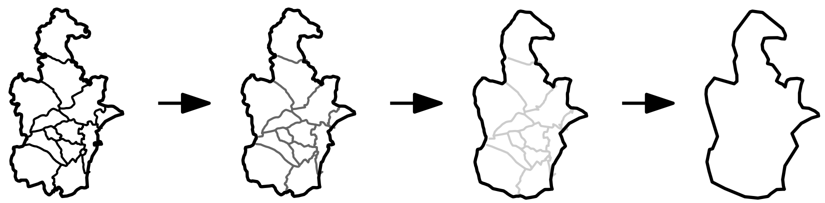


Aggregation

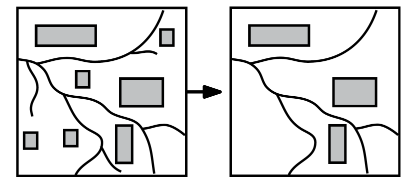


Classification

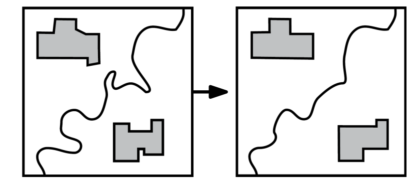
Administrative boundaires



DP

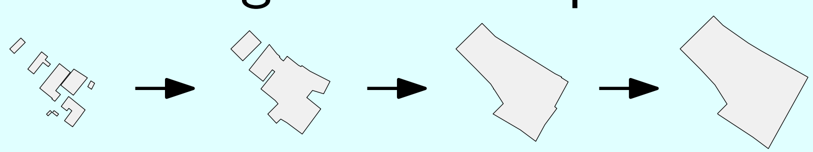


Elimination

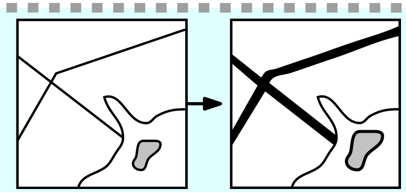


Simplification

Buildings to built-up areas



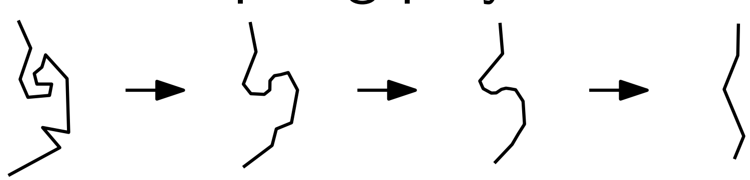
MST



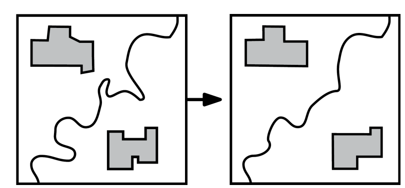
Exaggeration

Aggregation,
Simplification,
Elimination

Morphing polylines

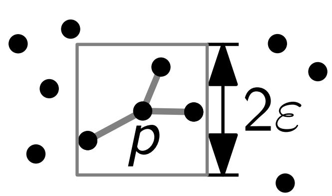


LSA
DP



Simplification

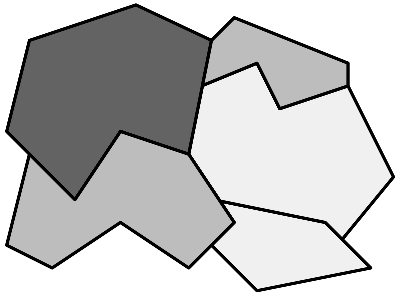
Choosing right data structures

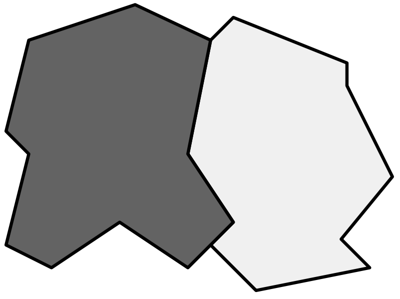


SortedDictionary,
SortedSet, . . .

Research Problem

input





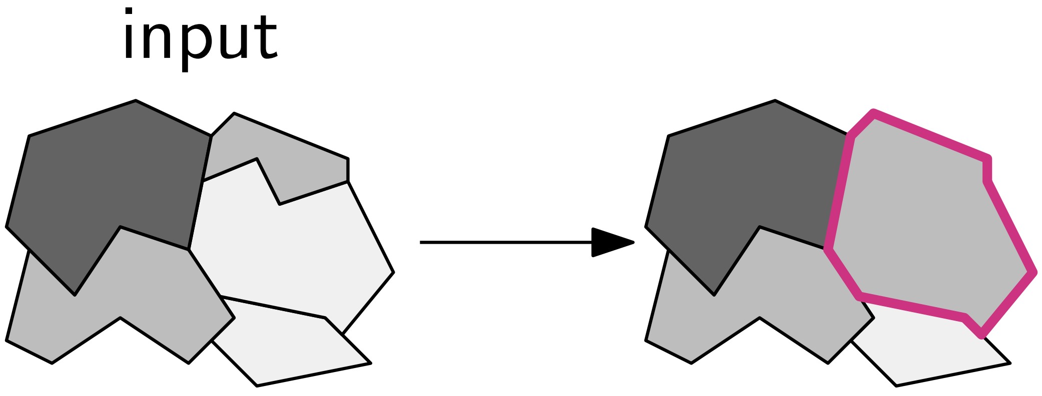
input

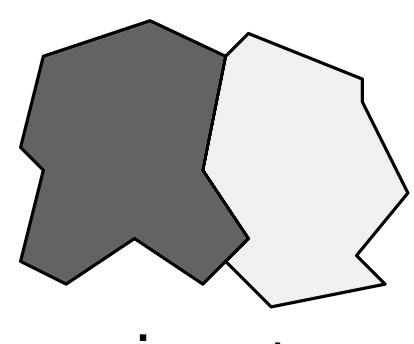
swamp

sport facility

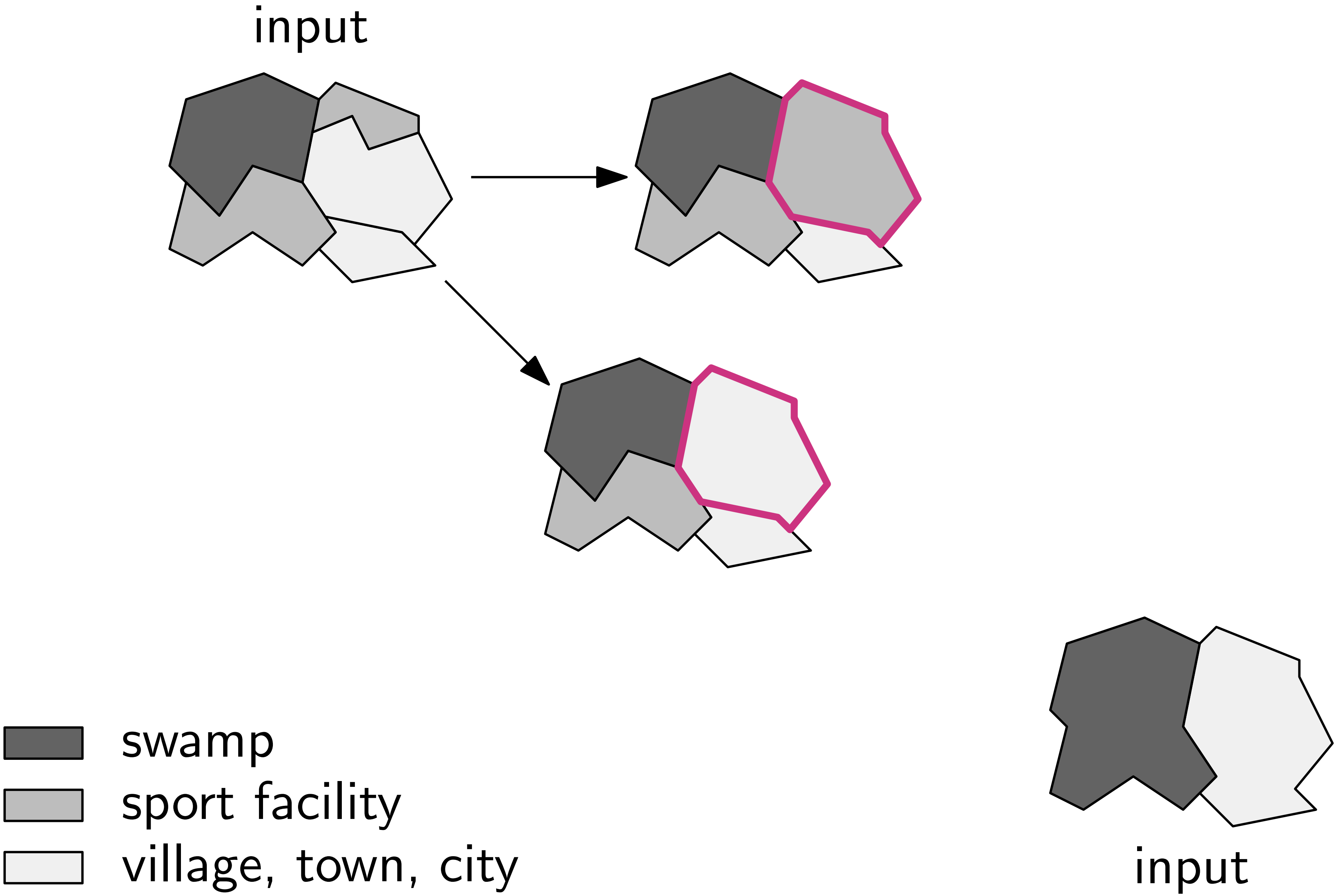
village, town, city

Research Problem

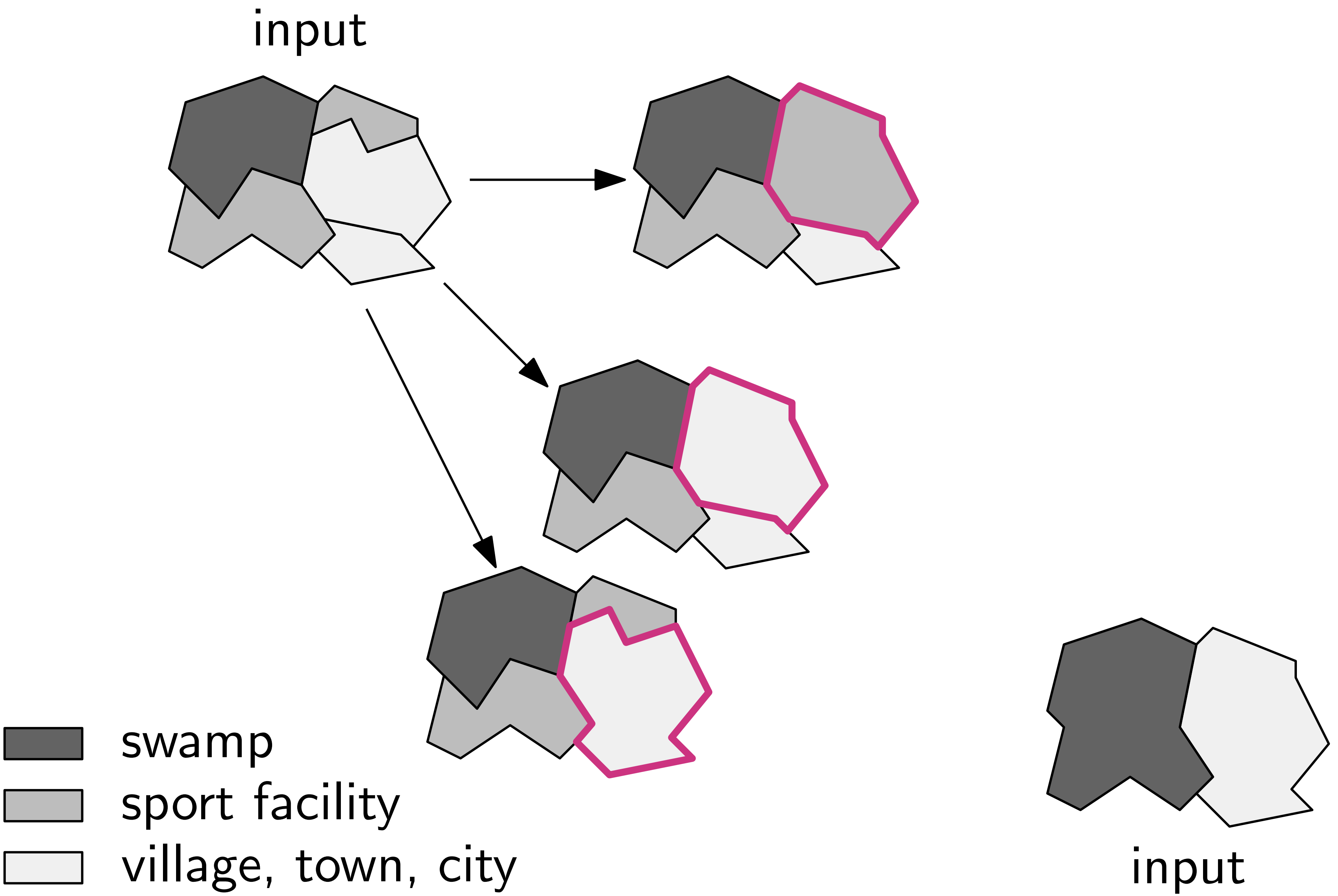




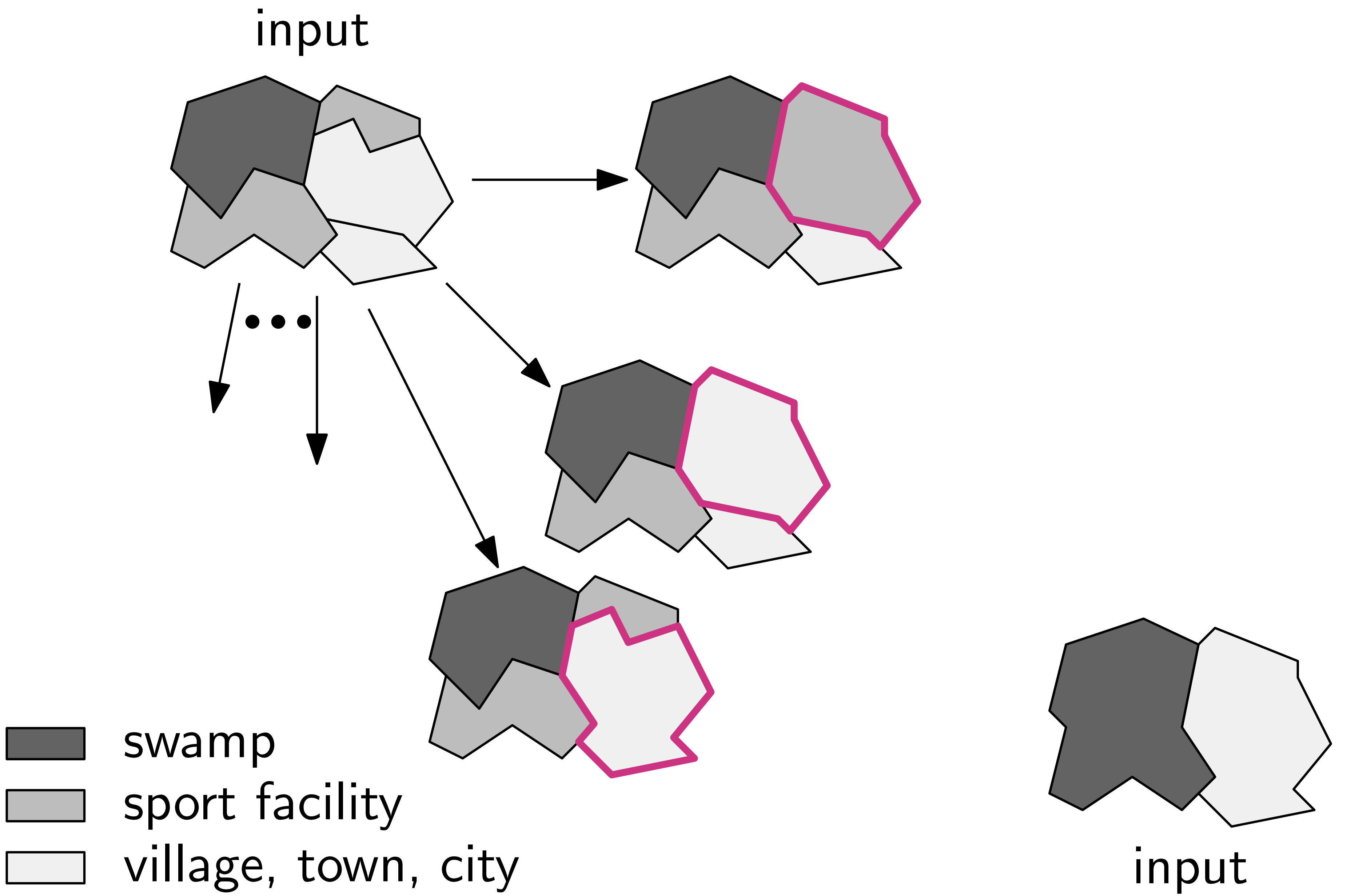
Research Problem



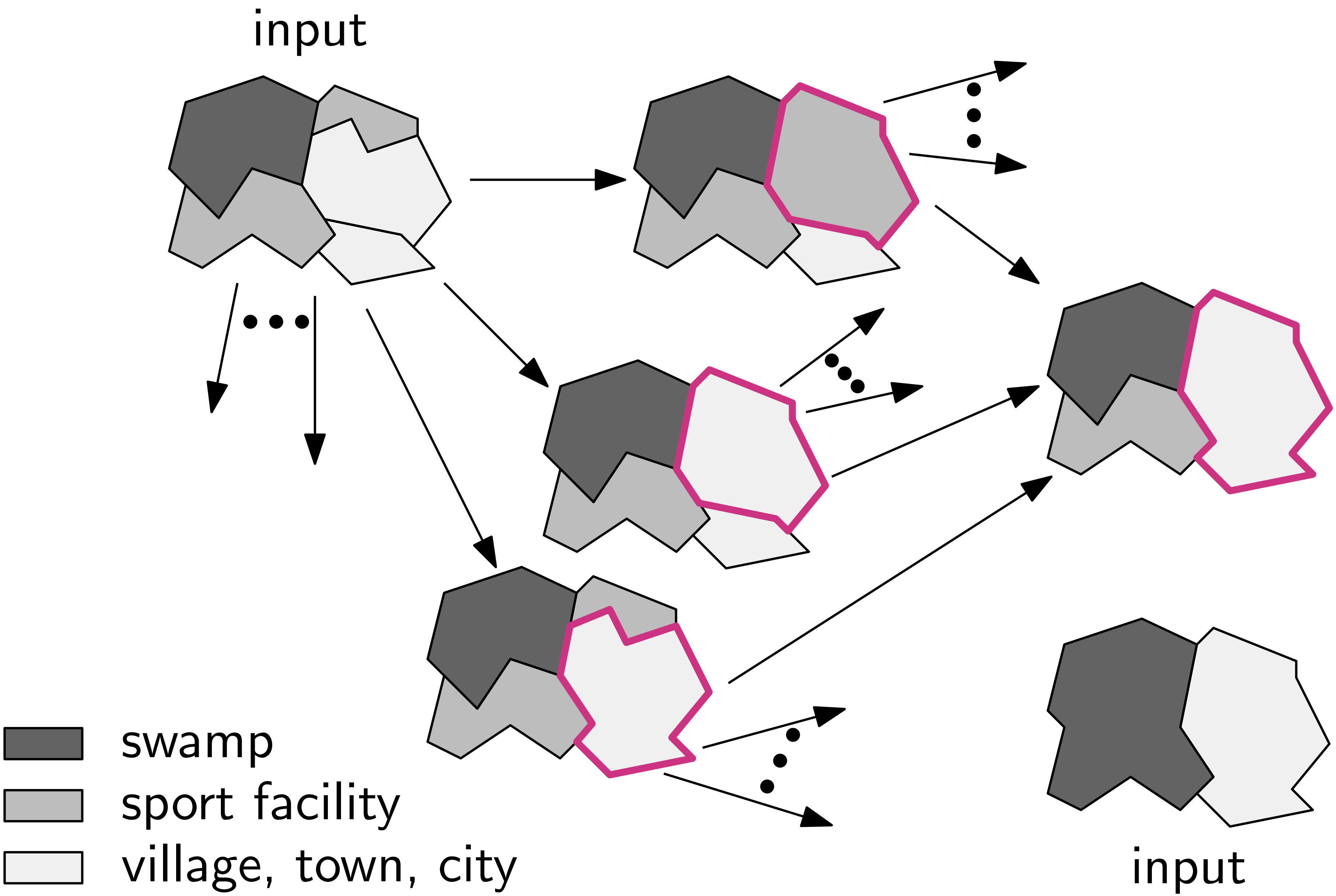
Research Problem



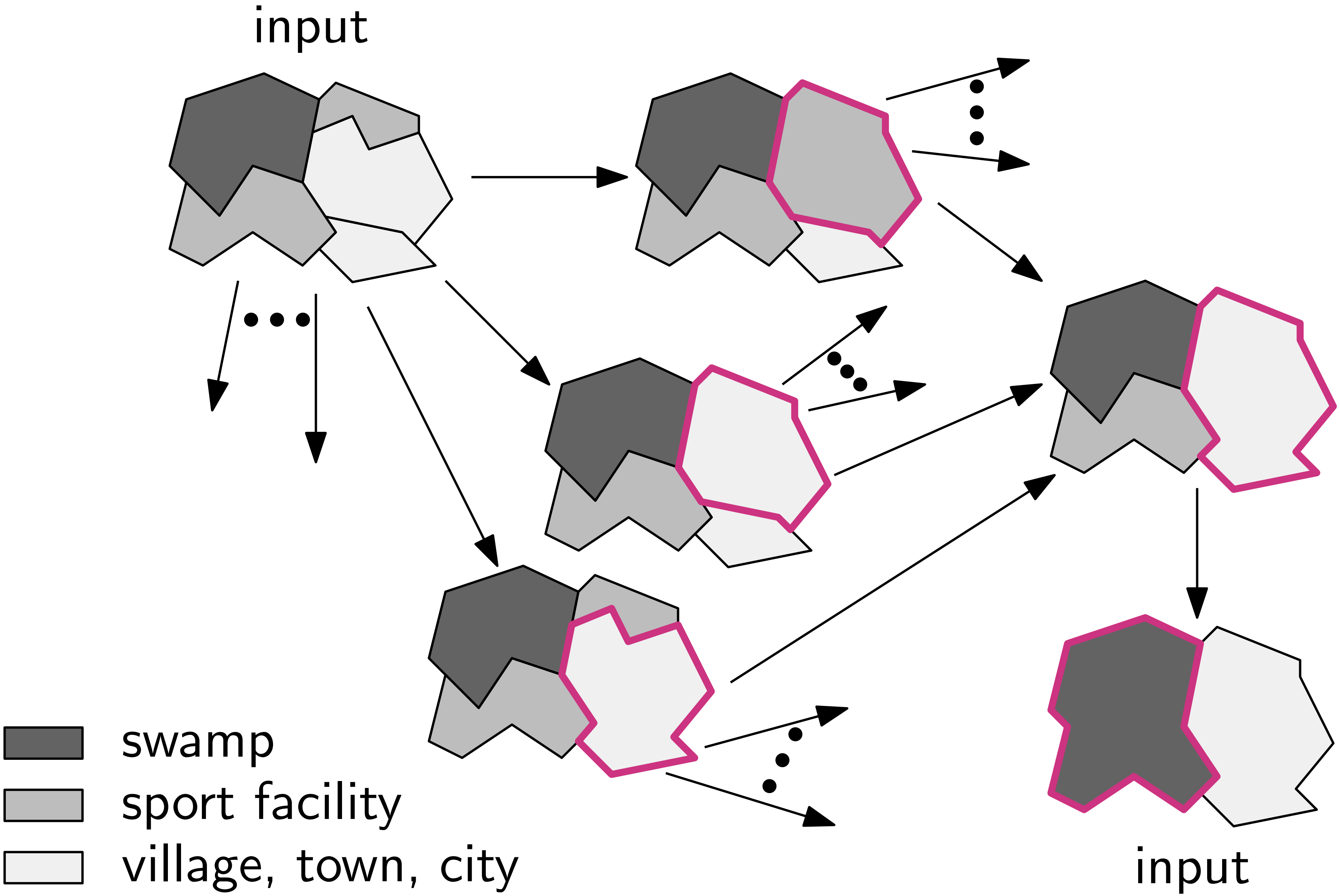
Research Problem



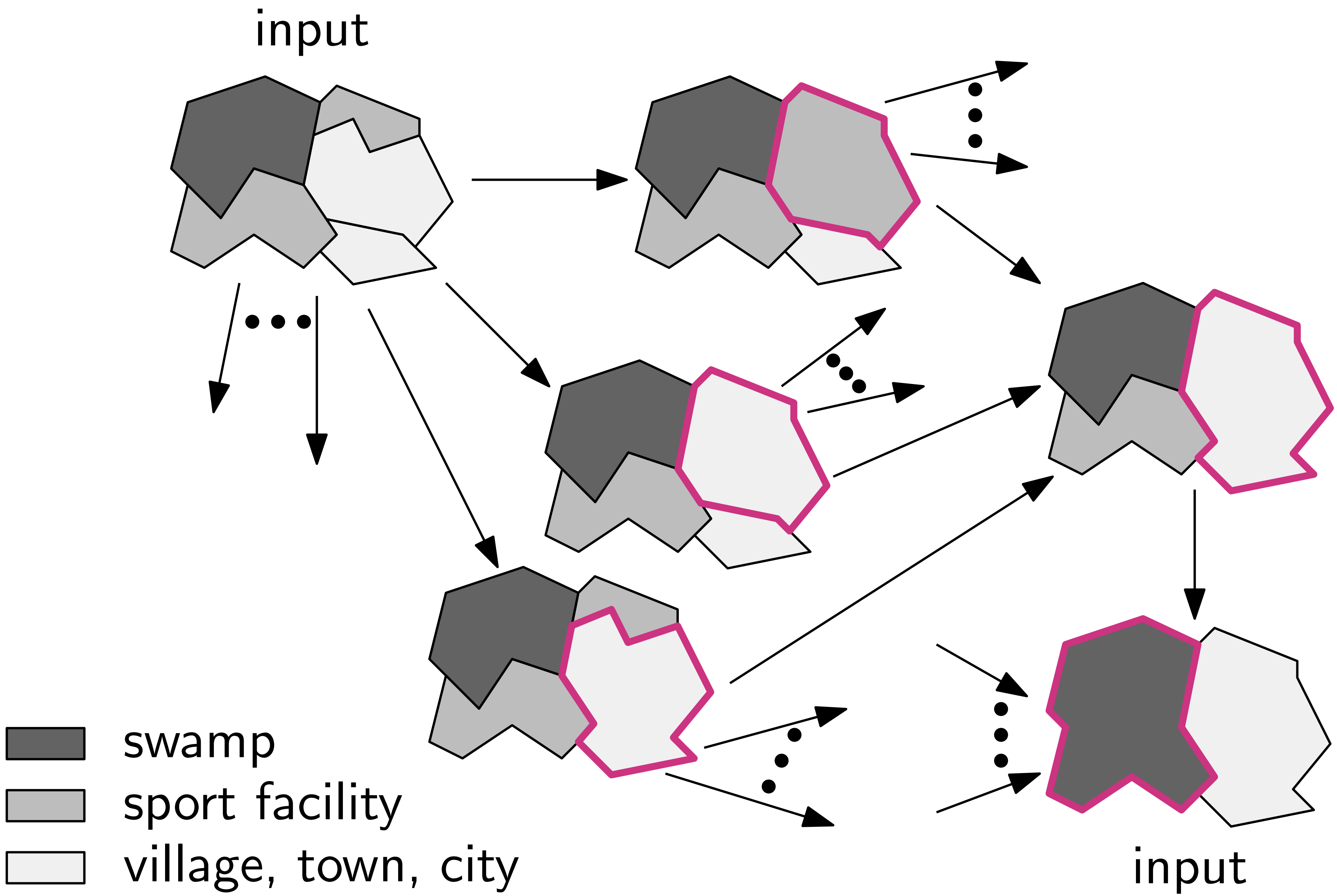
Research Problem



Research Problem

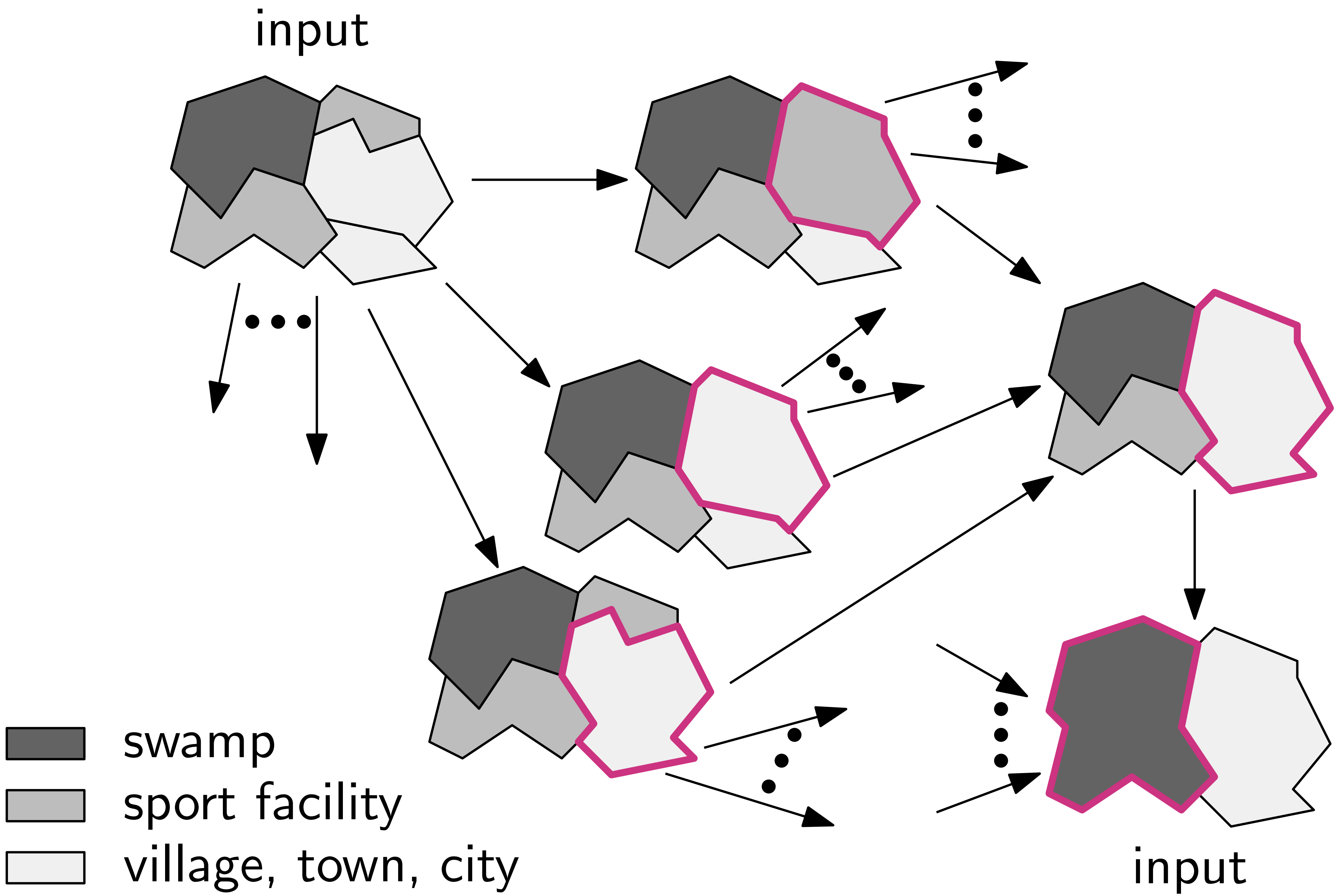


Research Problem

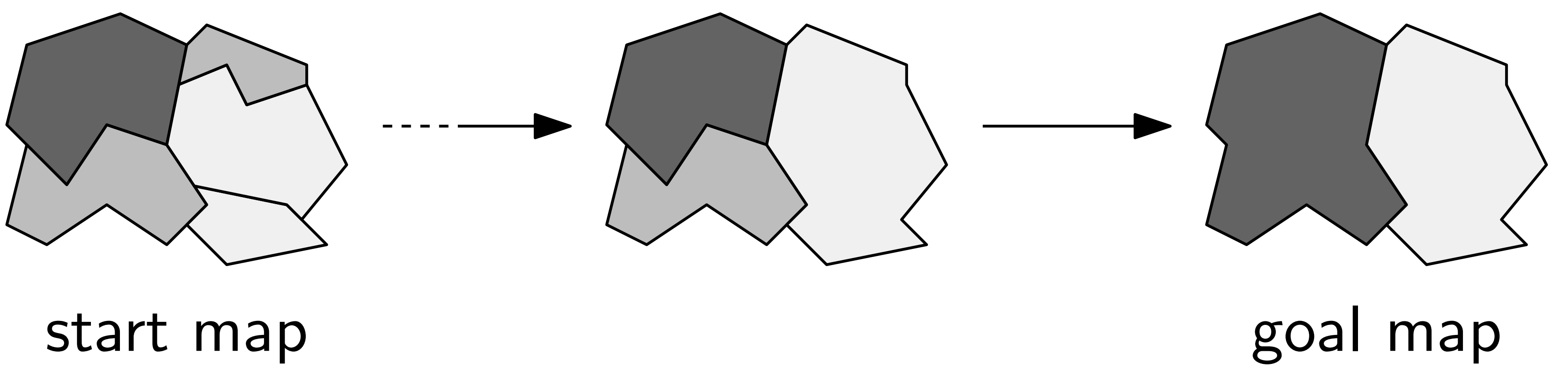


Research Problem

Given aggregation costs:
What is an optimal sequence?

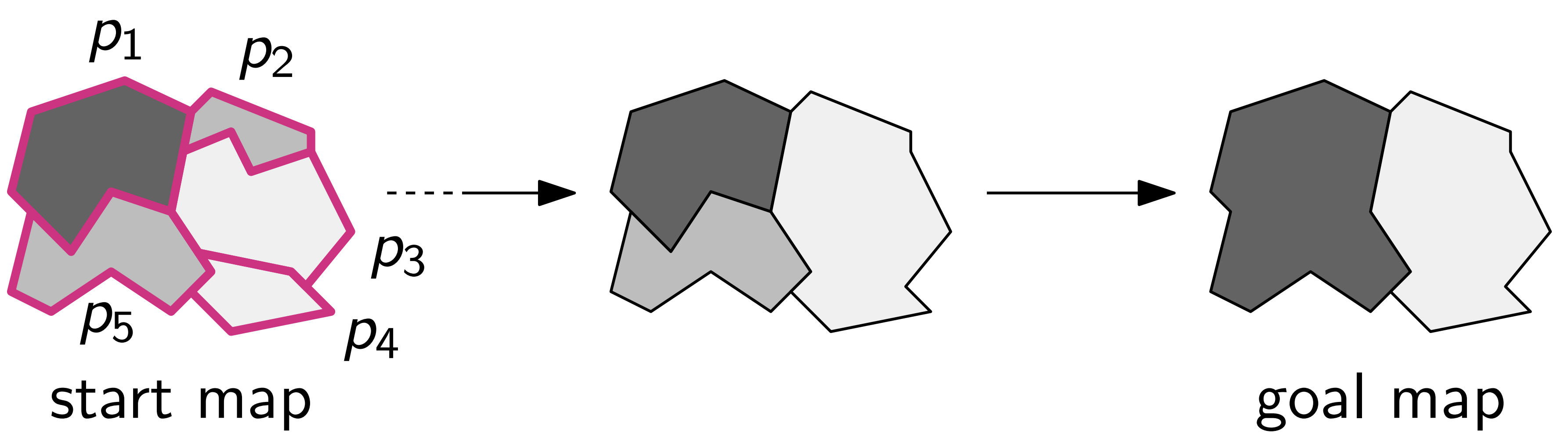


Preliminaries



Preliminaries

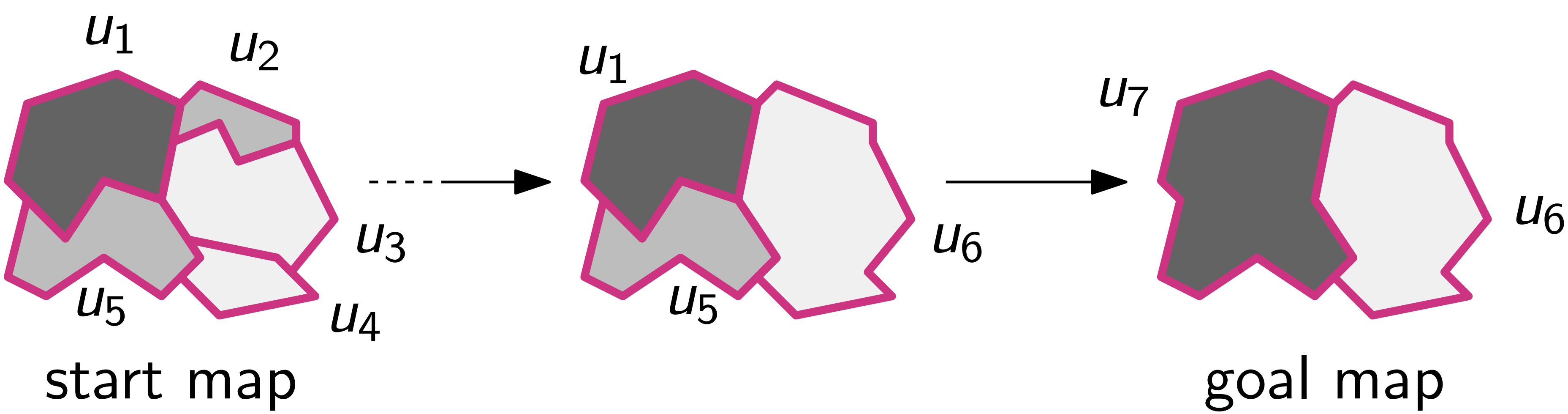
Polygon p_i : area on start map



Preliminaries

Polygon p_i : area on start map

Patch u_i : connected set of areas

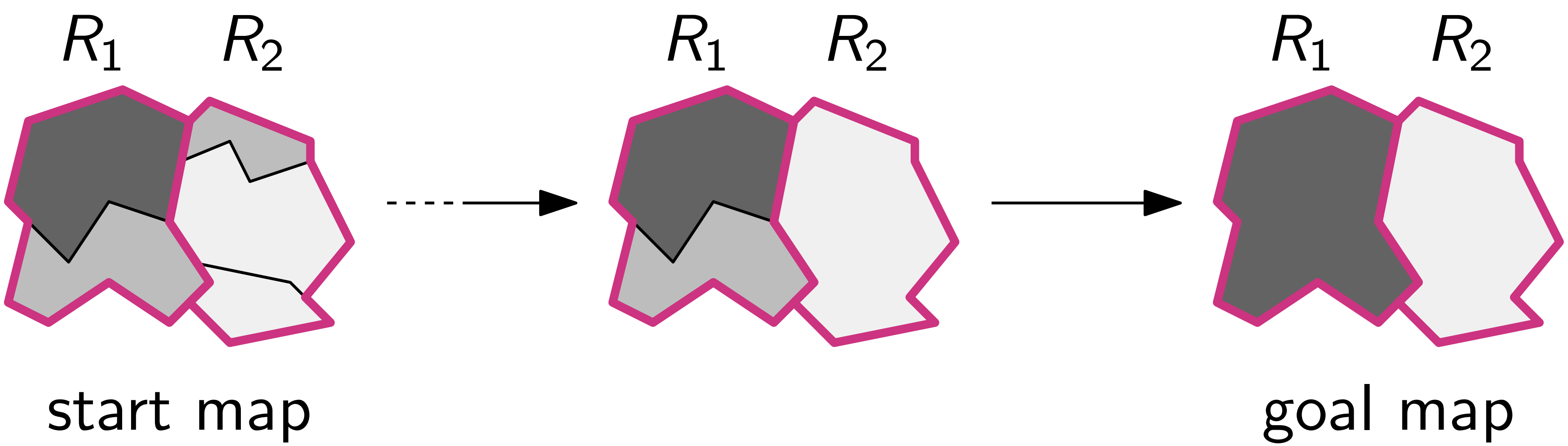


Preliminaries

Polygon p_i : area on start map

Patch u_i : connected set of areas

Region R_i : area on the goal map



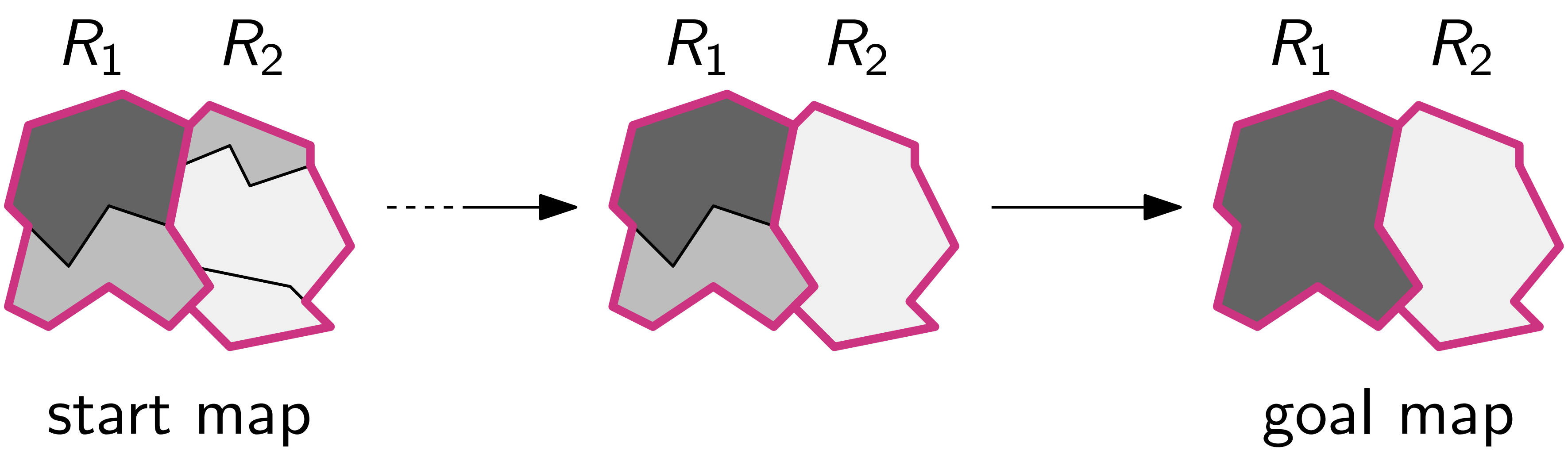
Preliminaries

Polygon p_i : area on start map

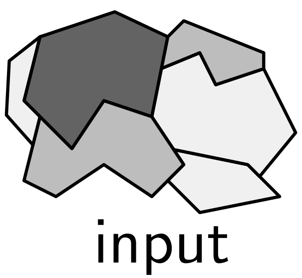
Patch u_i : connected set of areas

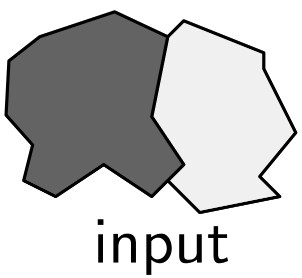
Region R_i : area on the goal map

Aggregate the smallest patch with its neighbour



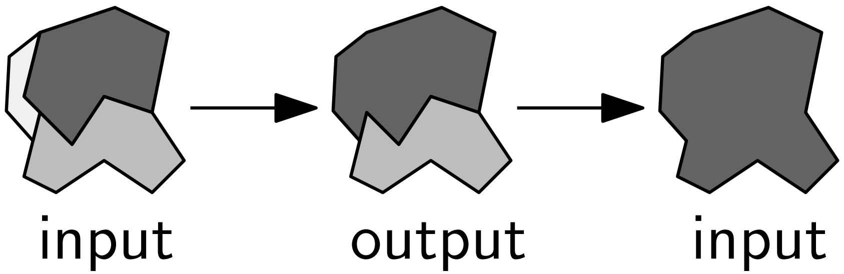
Interleave Aggregation Sequences

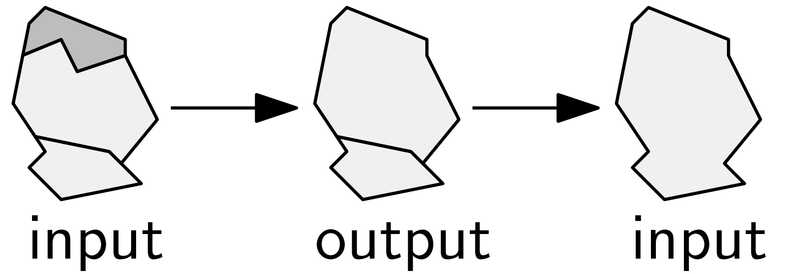


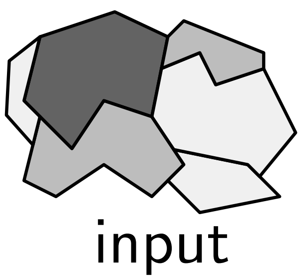


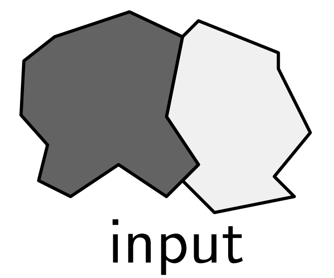
Interleave Aggregation Sequences

Compute a sequence for each region



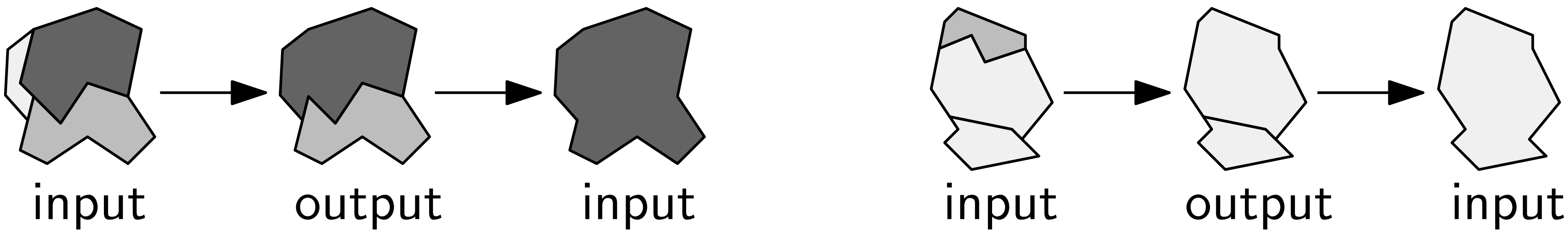




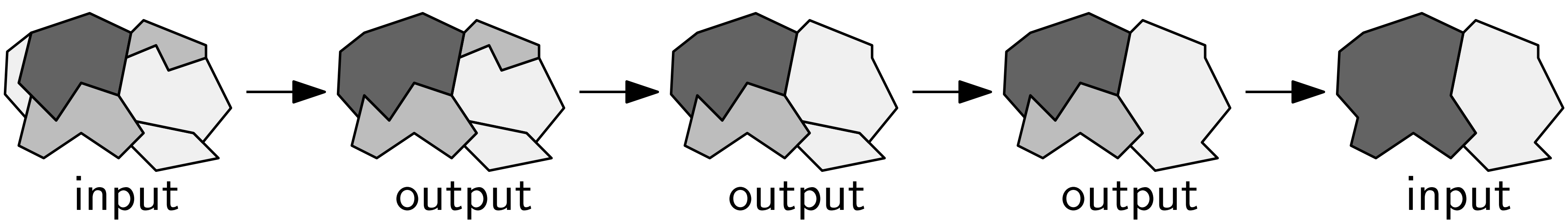


Interleave Aggregation Sequences

Compute a sequence for each region

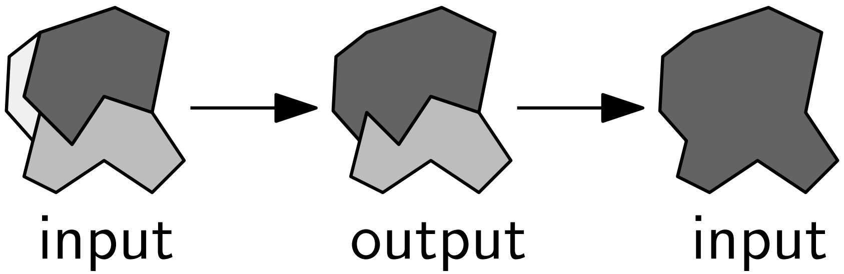


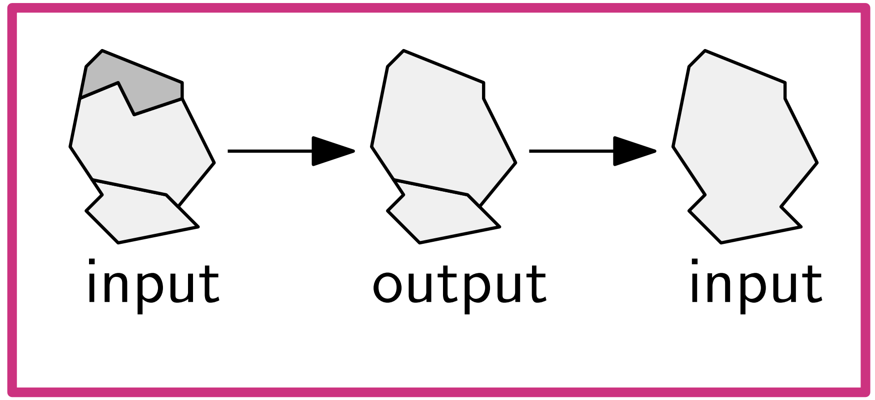
Interleave according to order of smallest areas (as merge sort)



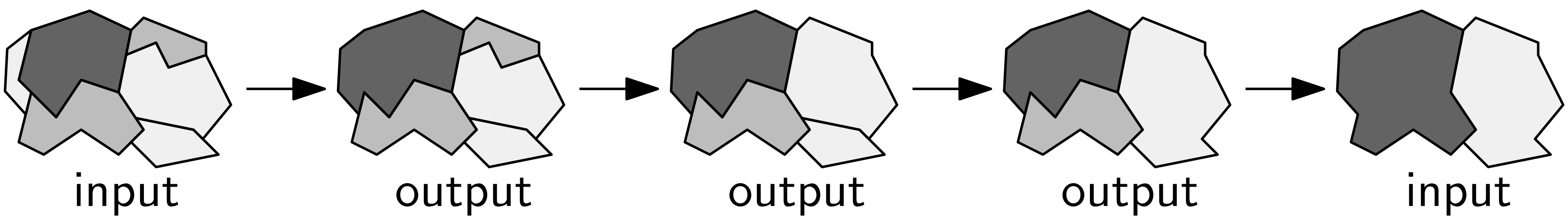
Interleave Aggregation Sequences

Compute a sequence for each region



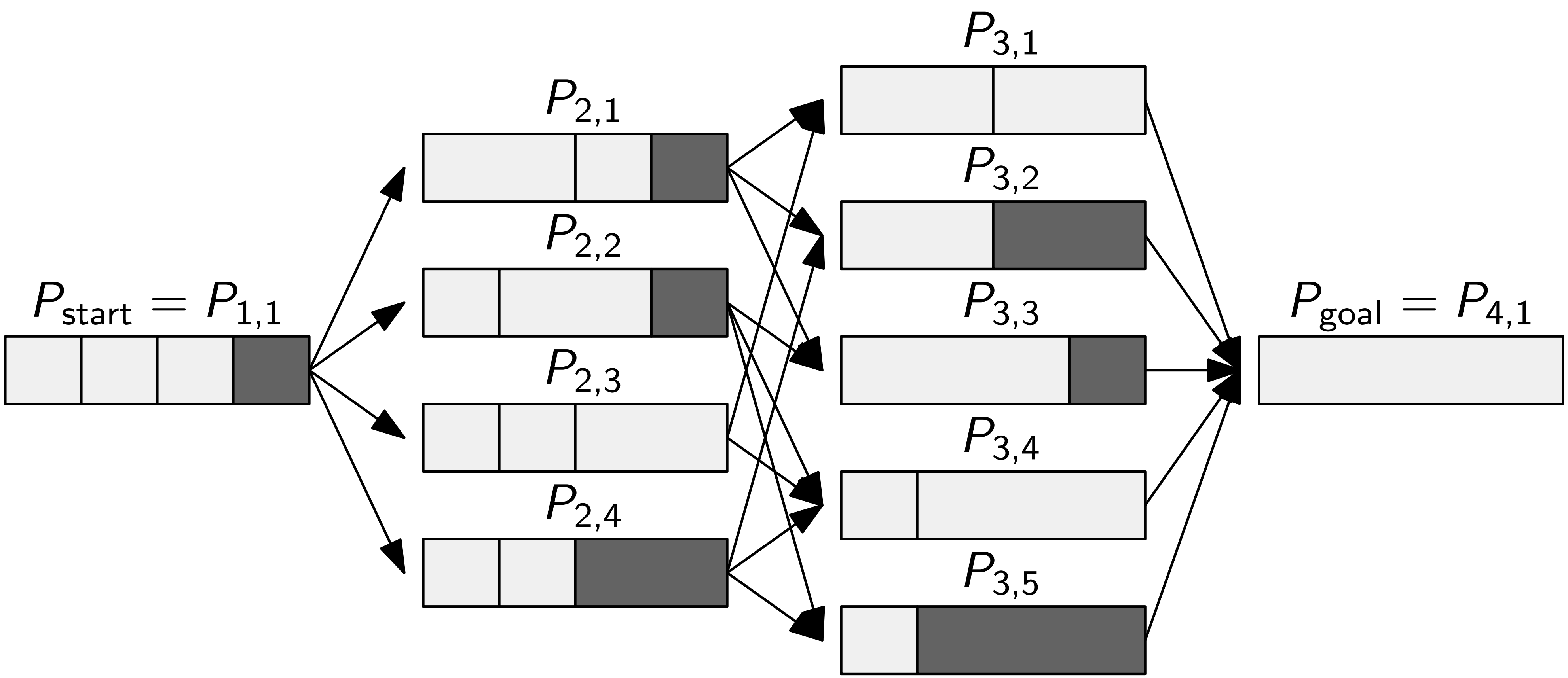


Interleave according to order of smallest areas (as merge sort)



Subdivision

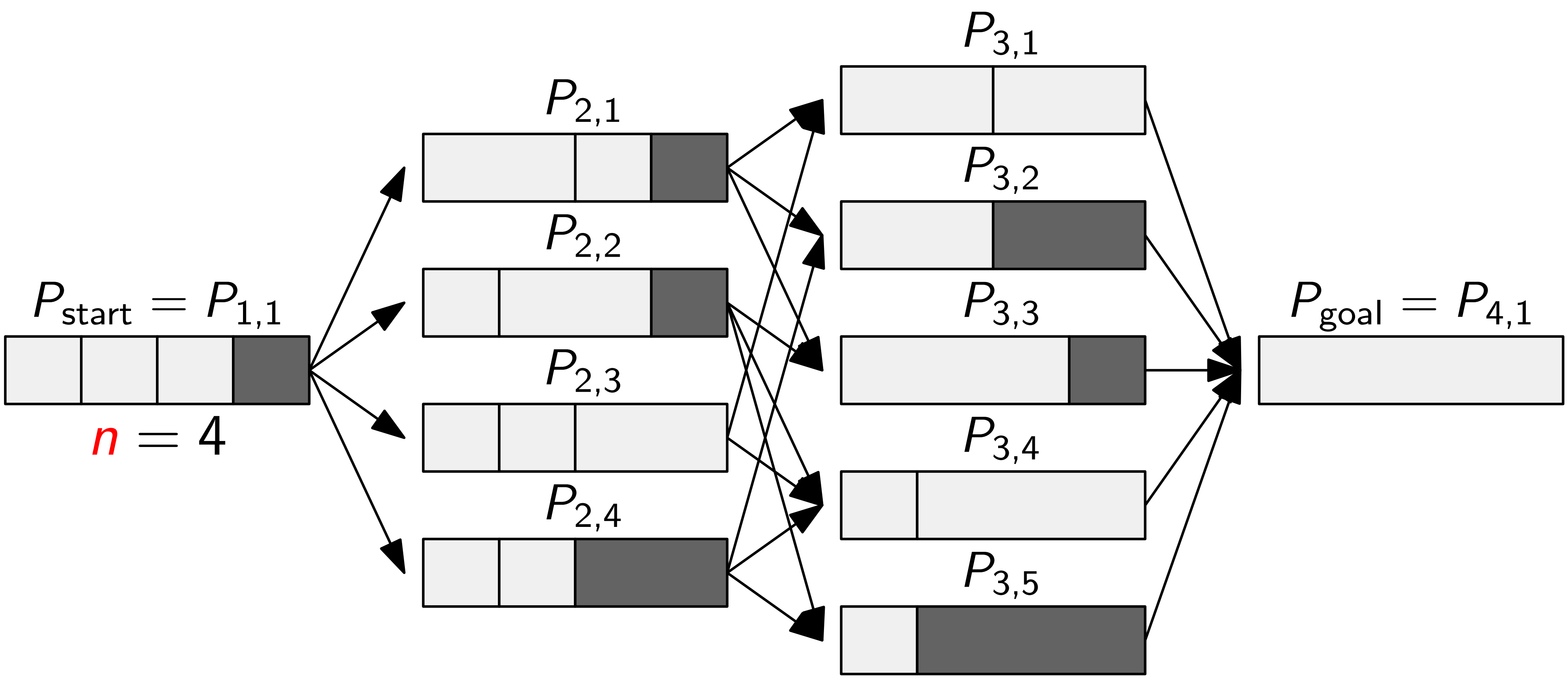
Subdivision $P_{t,i}$: patches subdividing a region



Subdivision

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

Size n : #polygons on start map

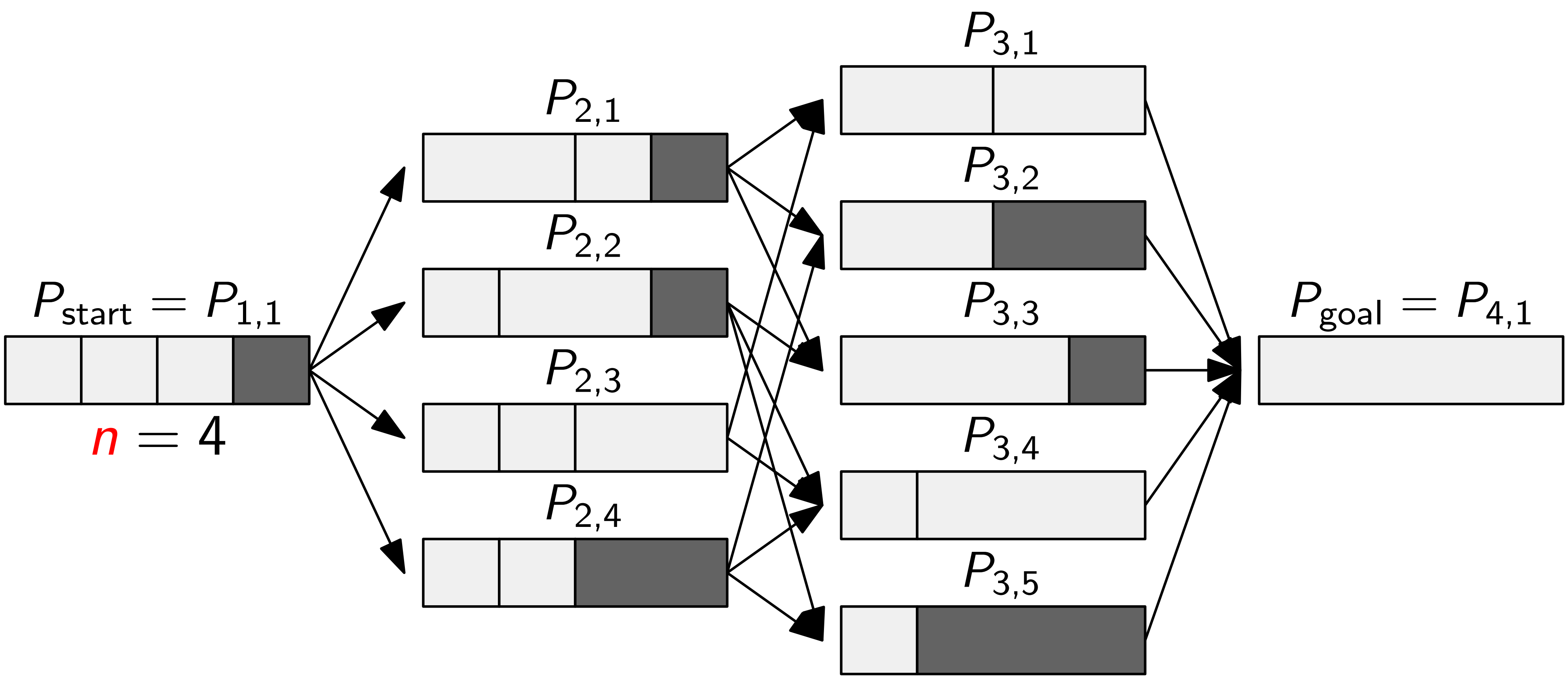


Subdivision

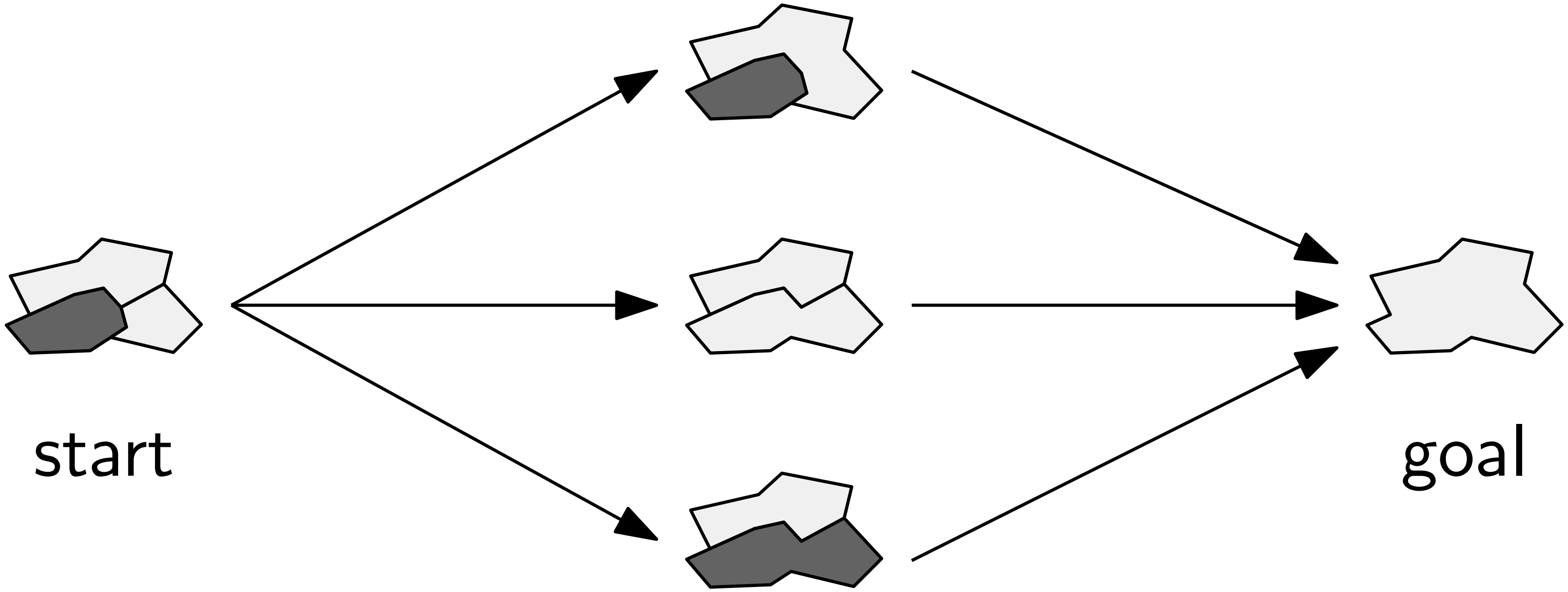
Subdivision $P_{t,i}$: patches subdividing a region

Size n : #polygons on start map

#subdivions is exponential in n.

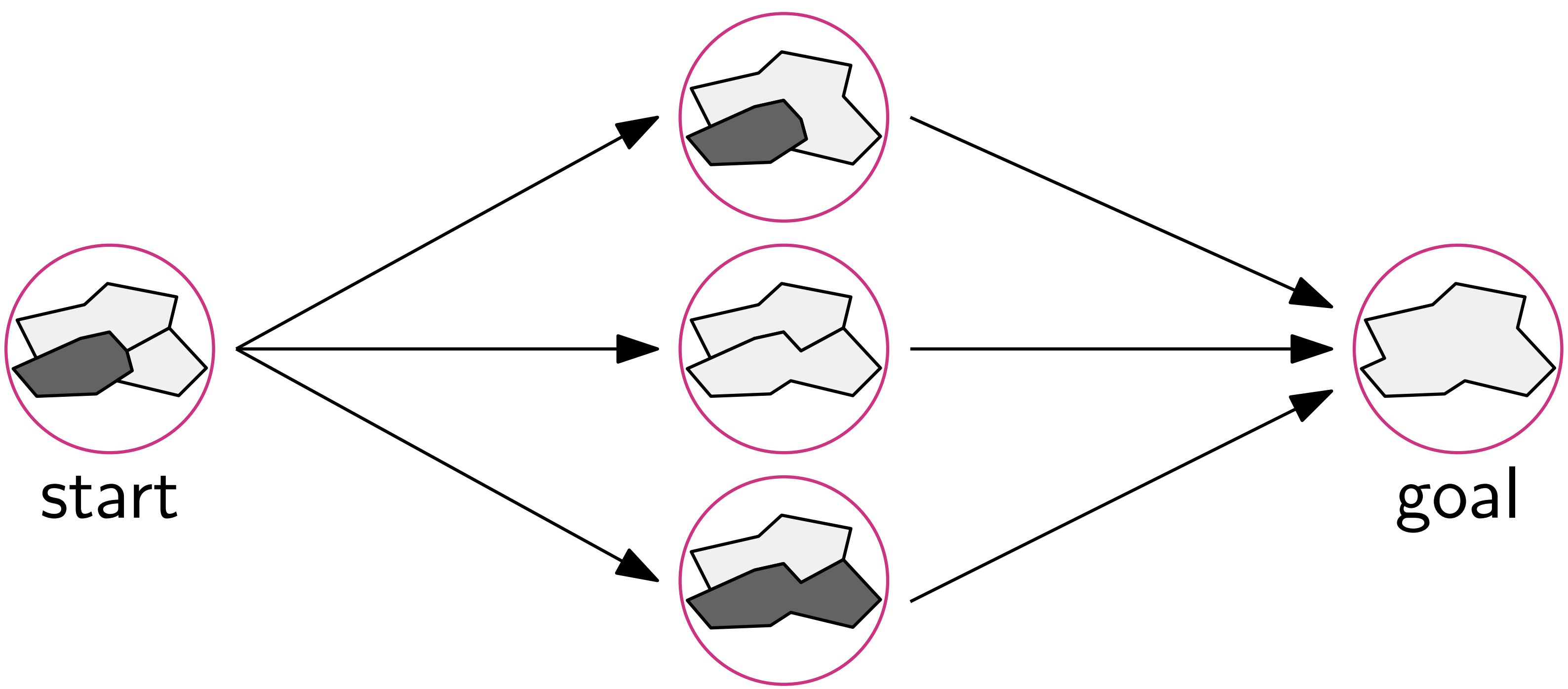


Formalizing a Pathfinding Problem



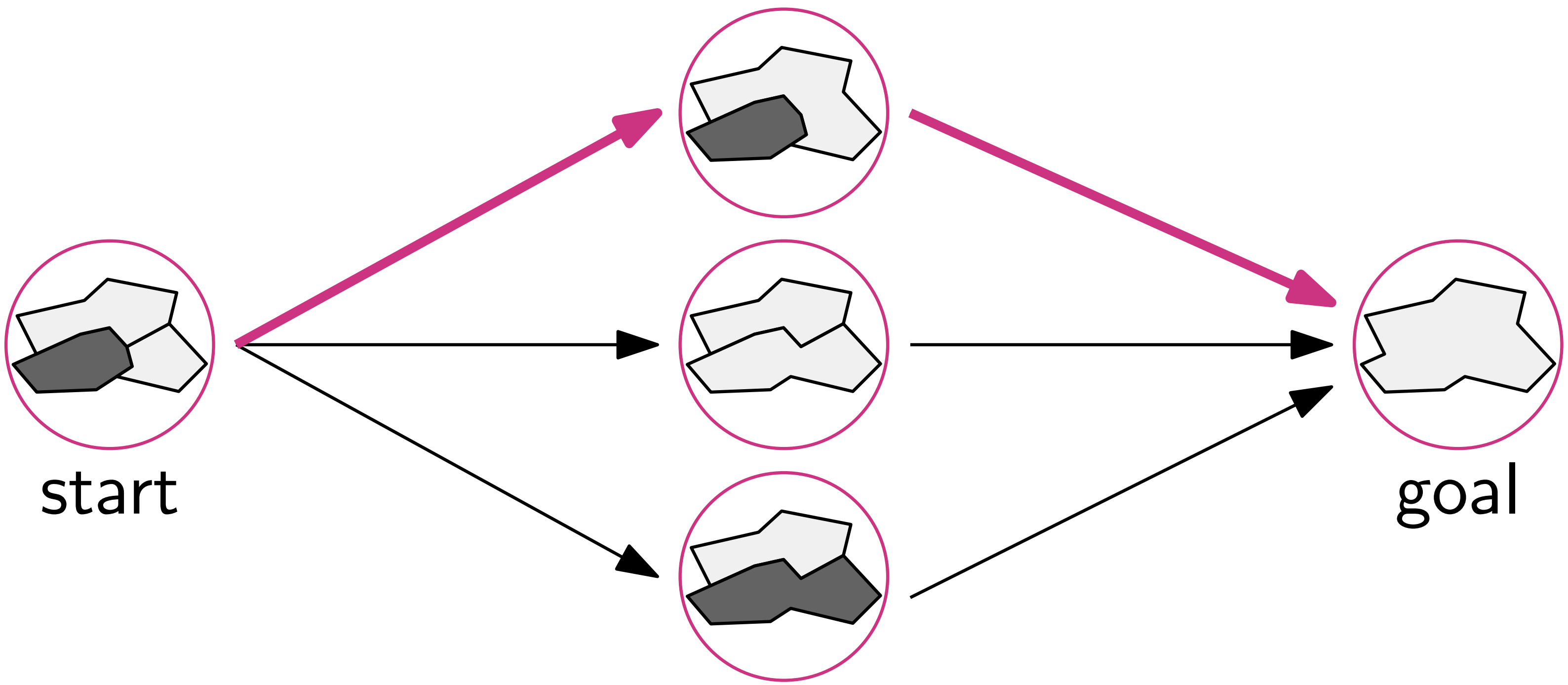
Formalizing a Pathfinding Problem

- Each subdivision is represented as a node



Formalizing a Pathfinding Problem

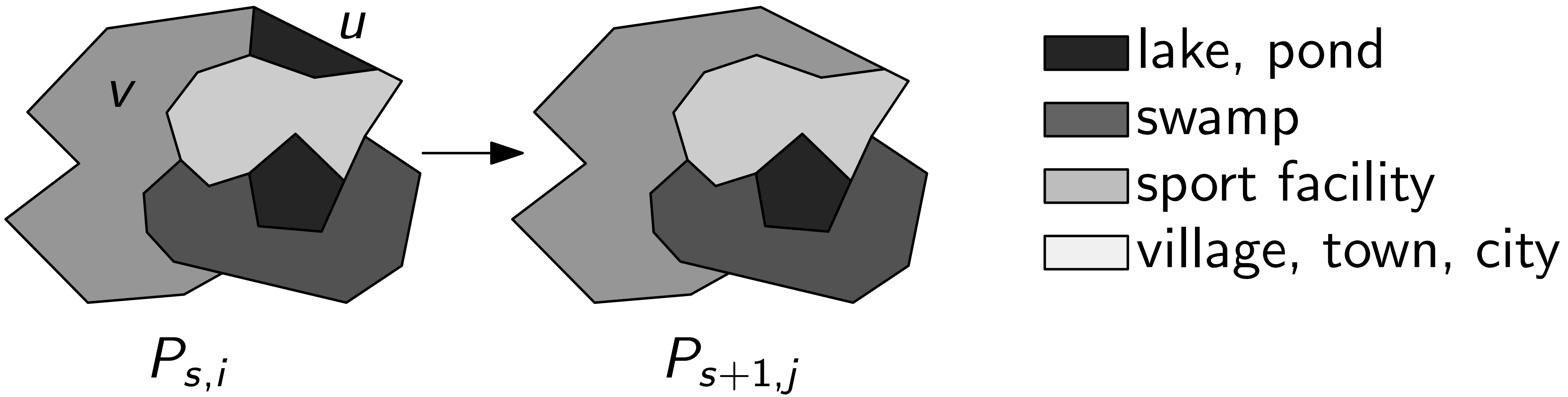
- Each subdivision is represented as a node
- Find a shortest path w.r.t. cost functions



Cost Function

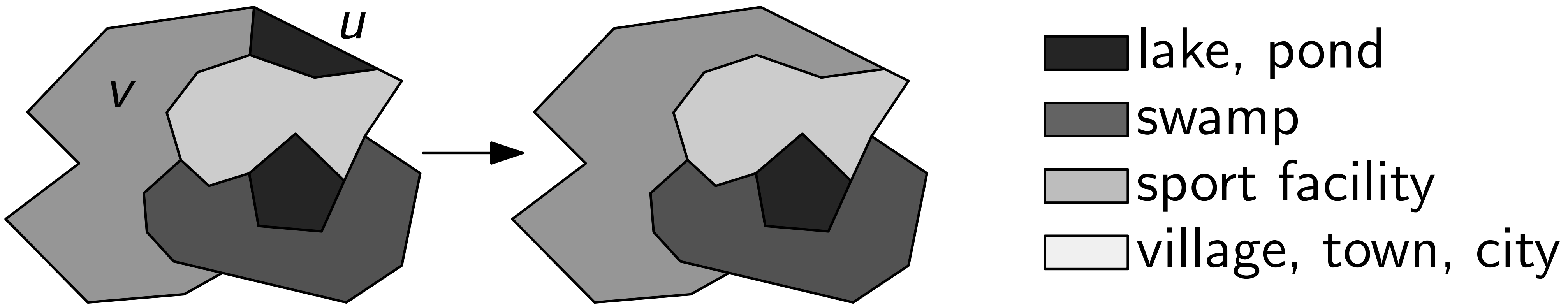
- Type change: $f_{\text{type}}(P_{s,i}, P_{s+1,j})$

  We wish to aggregate patches with similar types



Cost Function

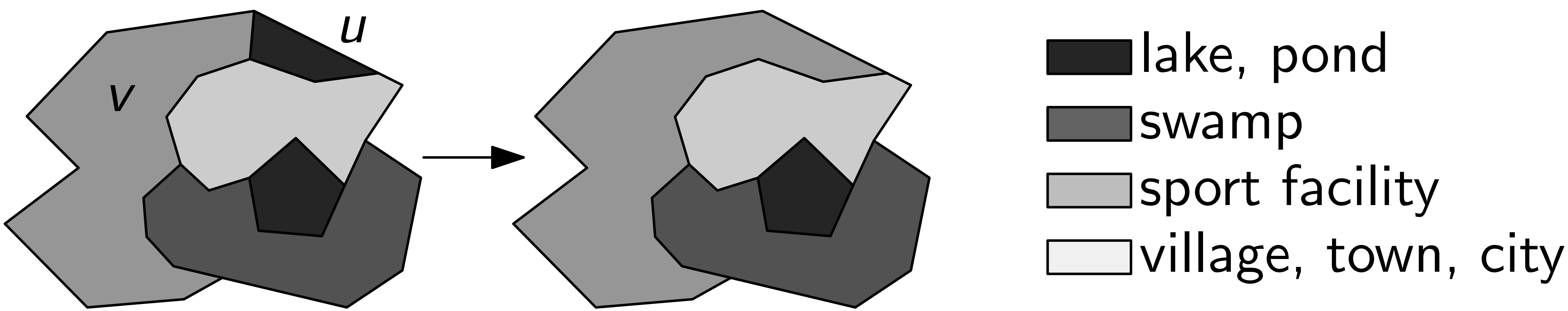
- Type change: $f_{\text{type}}(P_{s,i}, P_{s+1,j})$
  We wish to aggregate patches with similar types



$P_{s,i}$ $\quad$ $P_{s+1,j}$

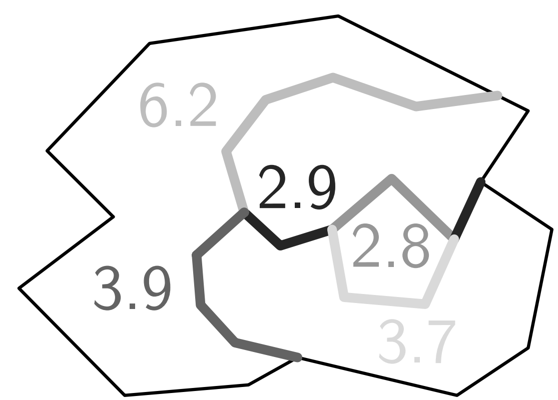
Cost Function

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



$P_{s,i}$ $P_{s+1,j}$

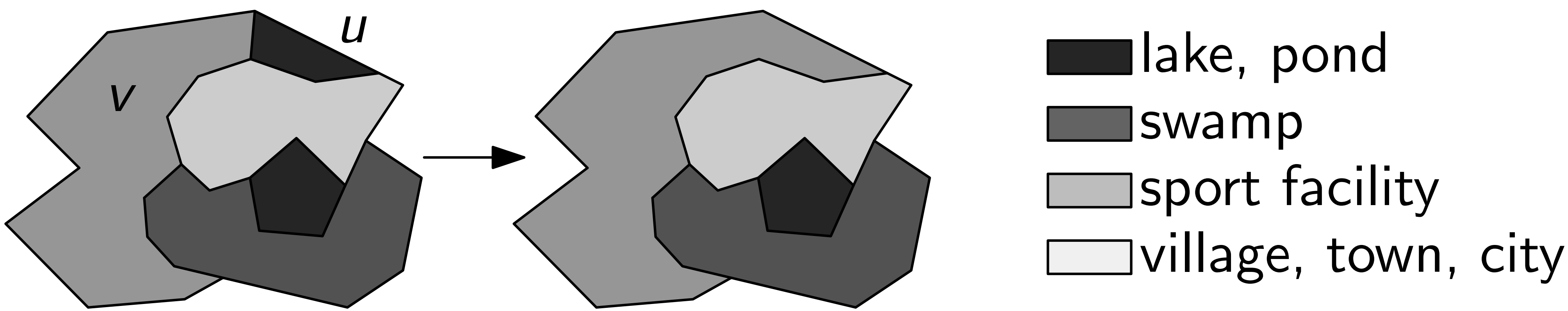
- Interior length: $f_{\text{length}}(P_{s,k})$
  Less length, easier to perceive



$\ell_{\text{int}}(P_{s,k}) = 19.5$

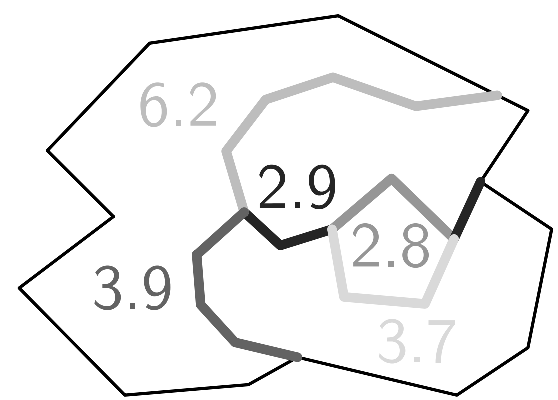
Cost Function

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



$P_{s,i}$ $\qquad\qquad$ $P_{s+1,j}$

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

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

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

- Combination of the two costs:
$g(\Pi) = (1-\lambda) g_{\text{type}}(\Pi) + \lambda g_{\text{length}}(\Pi)$

Cost Function

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

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

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

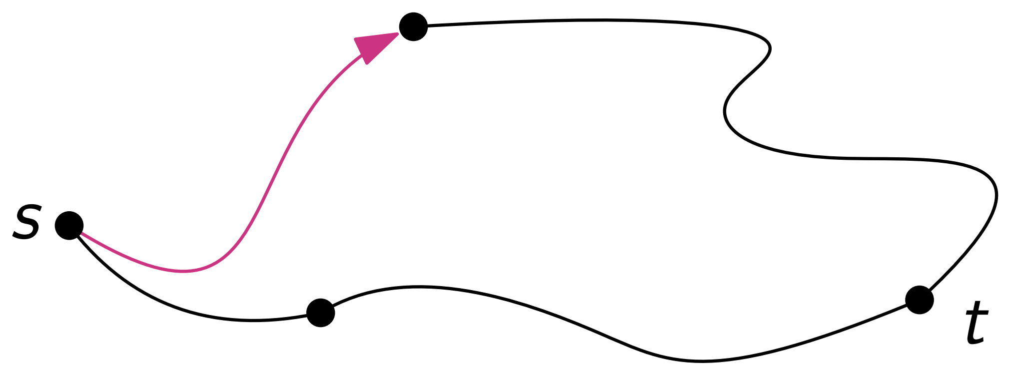
- Combination of the two costs:

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

$$\lambda = 0.5$$

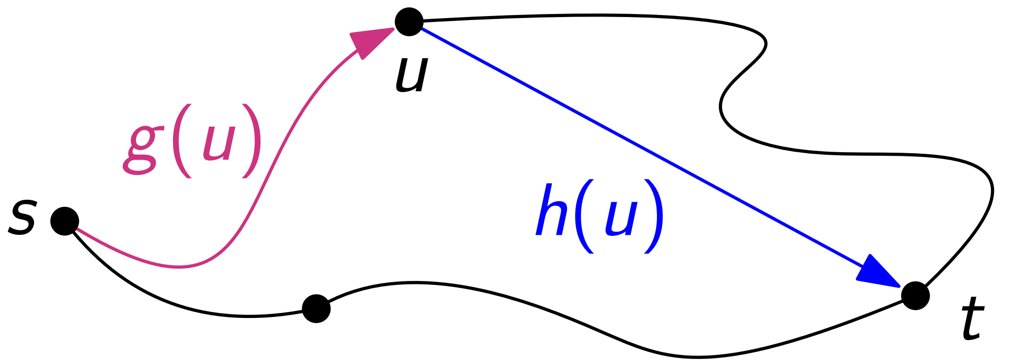
A* Algorithm

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



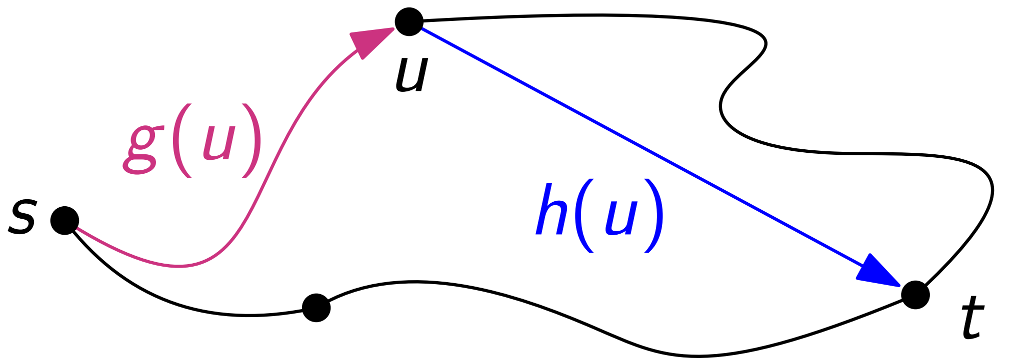
A* Algorithm

- 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

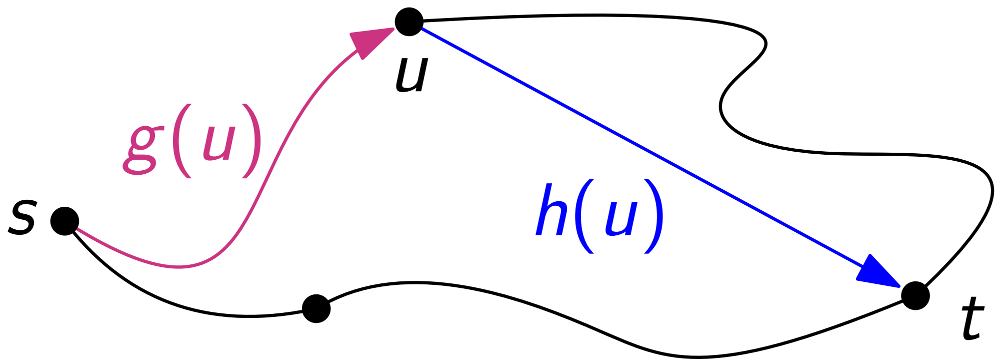
- A best-first search algorithm. Find a path from s to t
- Cost function: $F(u) = g(u) + h(u)$
  - $g(u)$: exact cost of s-u path
  - $h(u)$: estimated cost of shortest u-t path



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

A* Algorithm

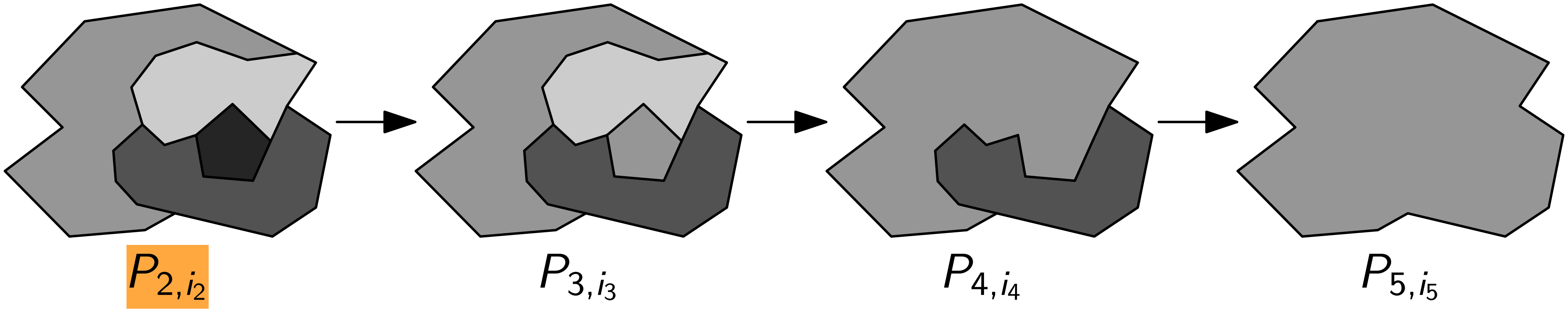
- A best-first search algorithm. Find a path from s to t
- Cost function: $F(u) = g(u) + h(u)$
  - $g(u)$: exact cost of s-u path
  - $h(u)$: estimated cost of shortest u-t path

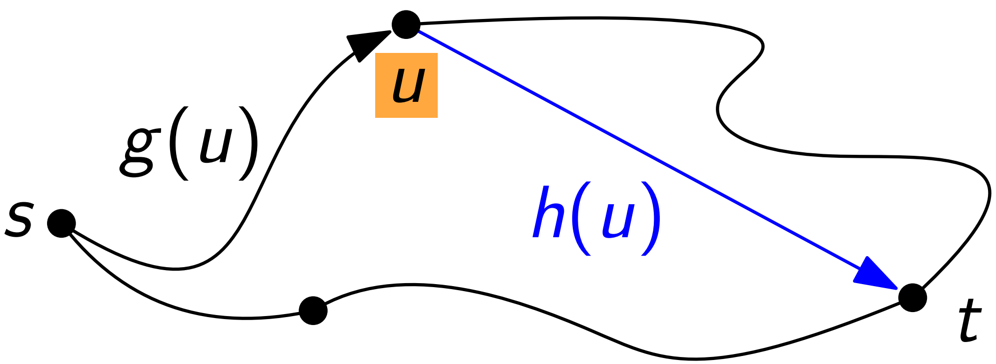


- Guarantees a shortest path if $h(u)$ is smaller than real cost
- Helps ignore some paths

Estimating Cost

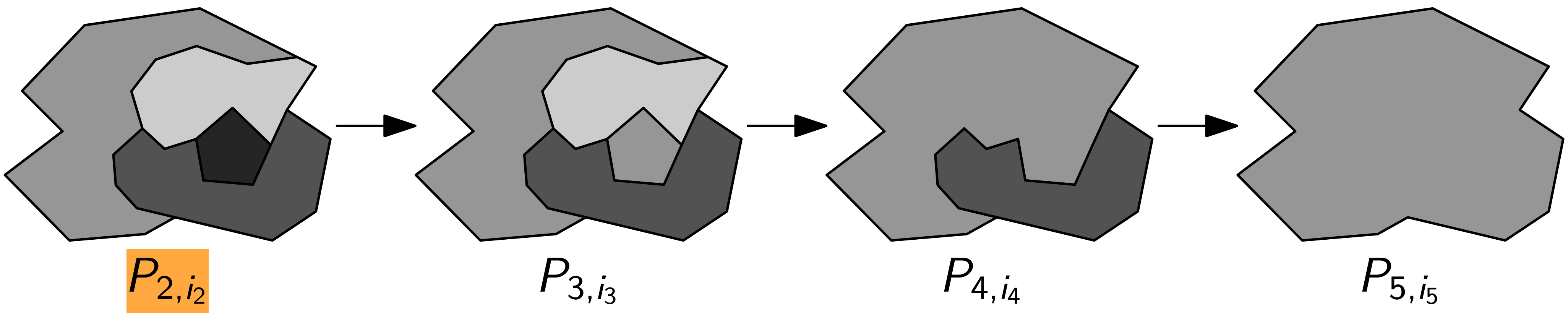
- $h_{\text{type}}(\boxed{P_{t,i}}) = \sum_{s=t}^{n-1} f_{\text{type}}(P_{s,i_s}, P_{s+1,i_{s+1}})$
  We assume: Each patch immediately gets the target type.



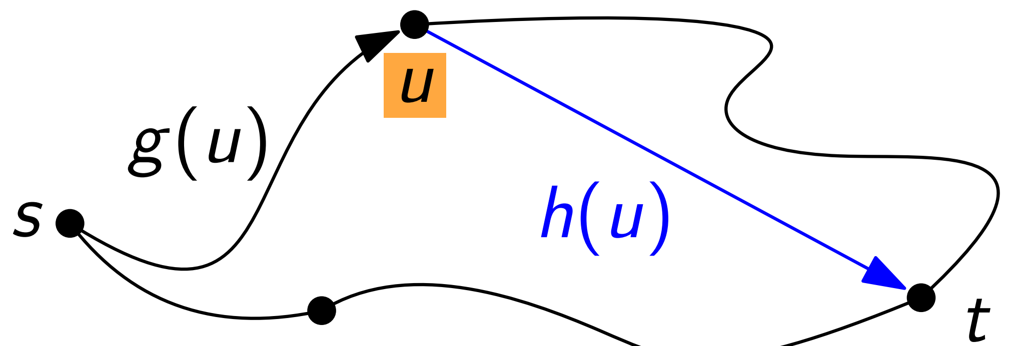


Estimating Cost

- $h_{\text{type}}(\boxed{P_{t,i}}) = \sum_{s=t}^{n-1} f_{\text{type}}(P_{s,i_s}, P_{s+1,i_{s+1}})$
  We assume: Each patch immediately gets the target type.

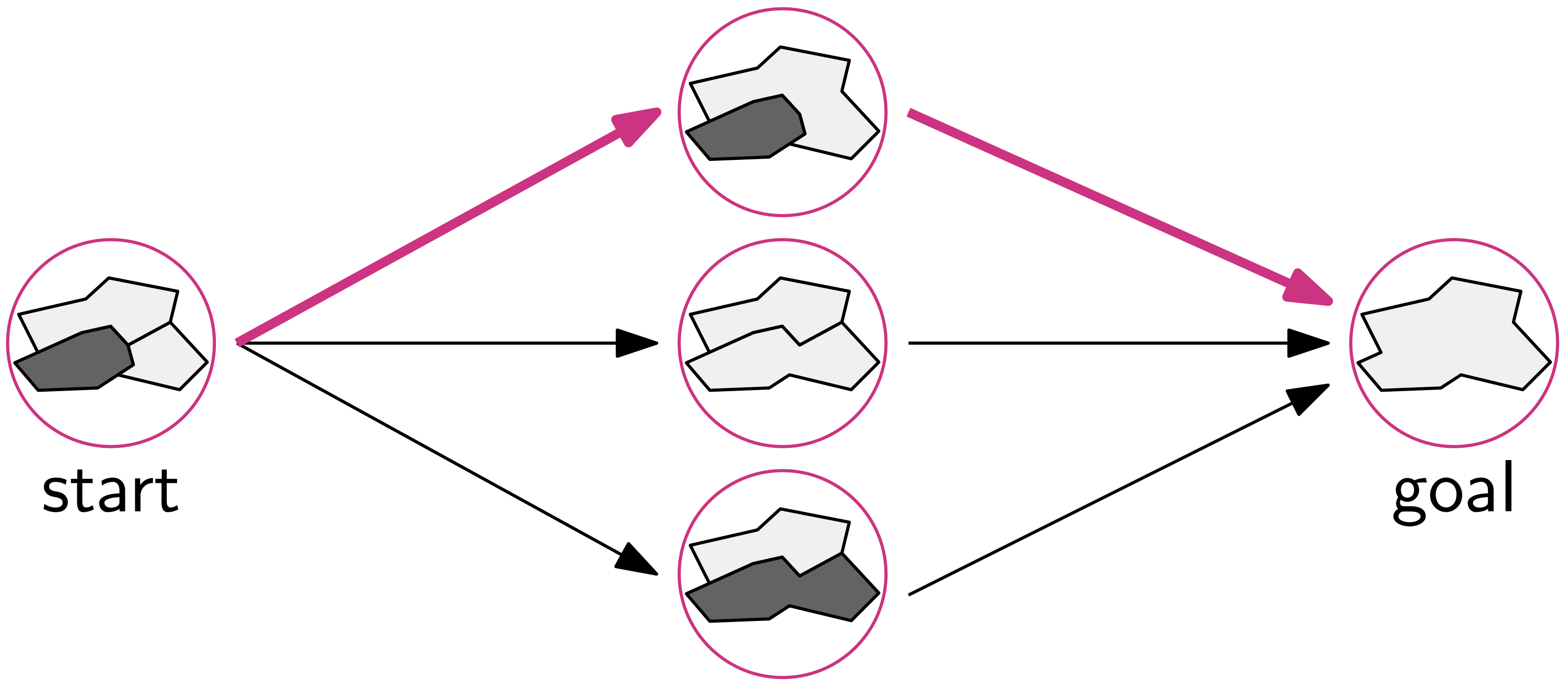


- $h_{\text{length}}(\boxed{P_{t,i}})$



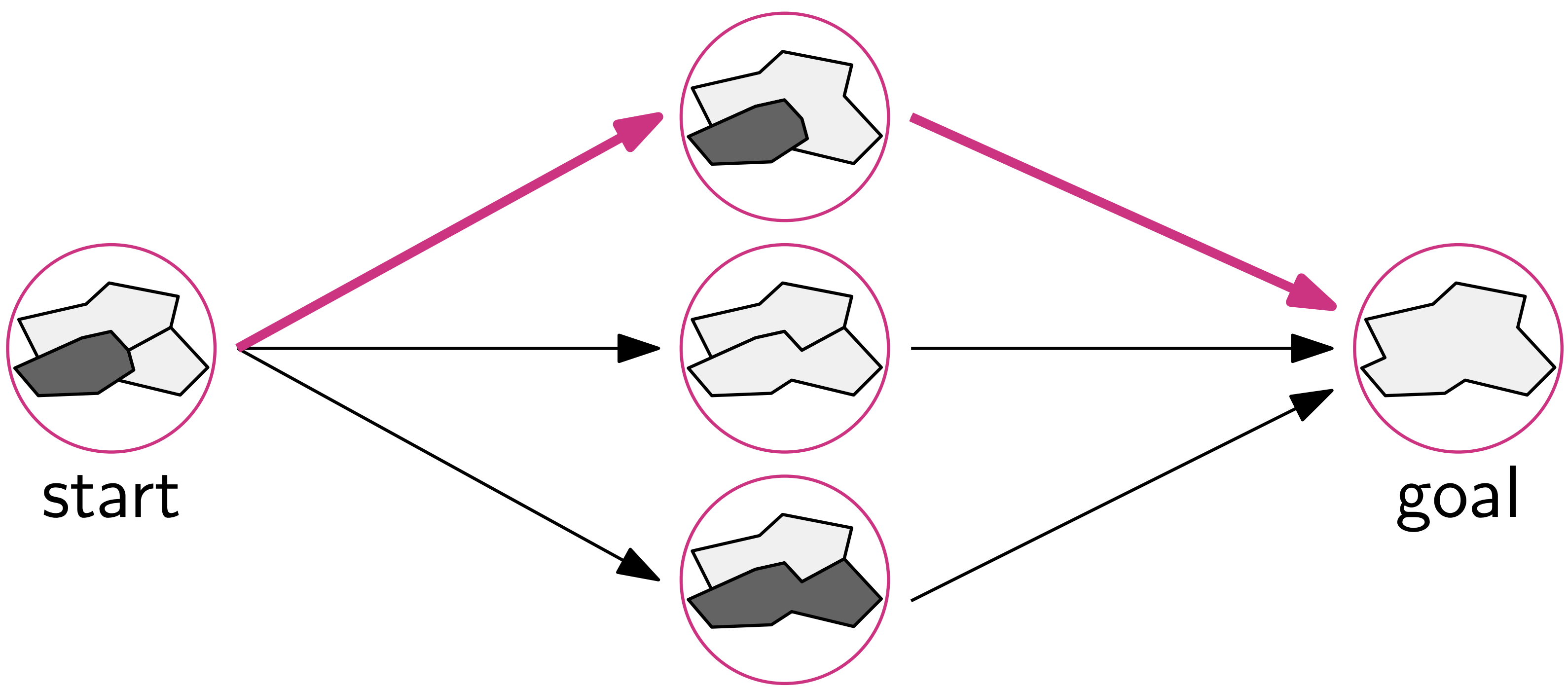
Overestimation

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



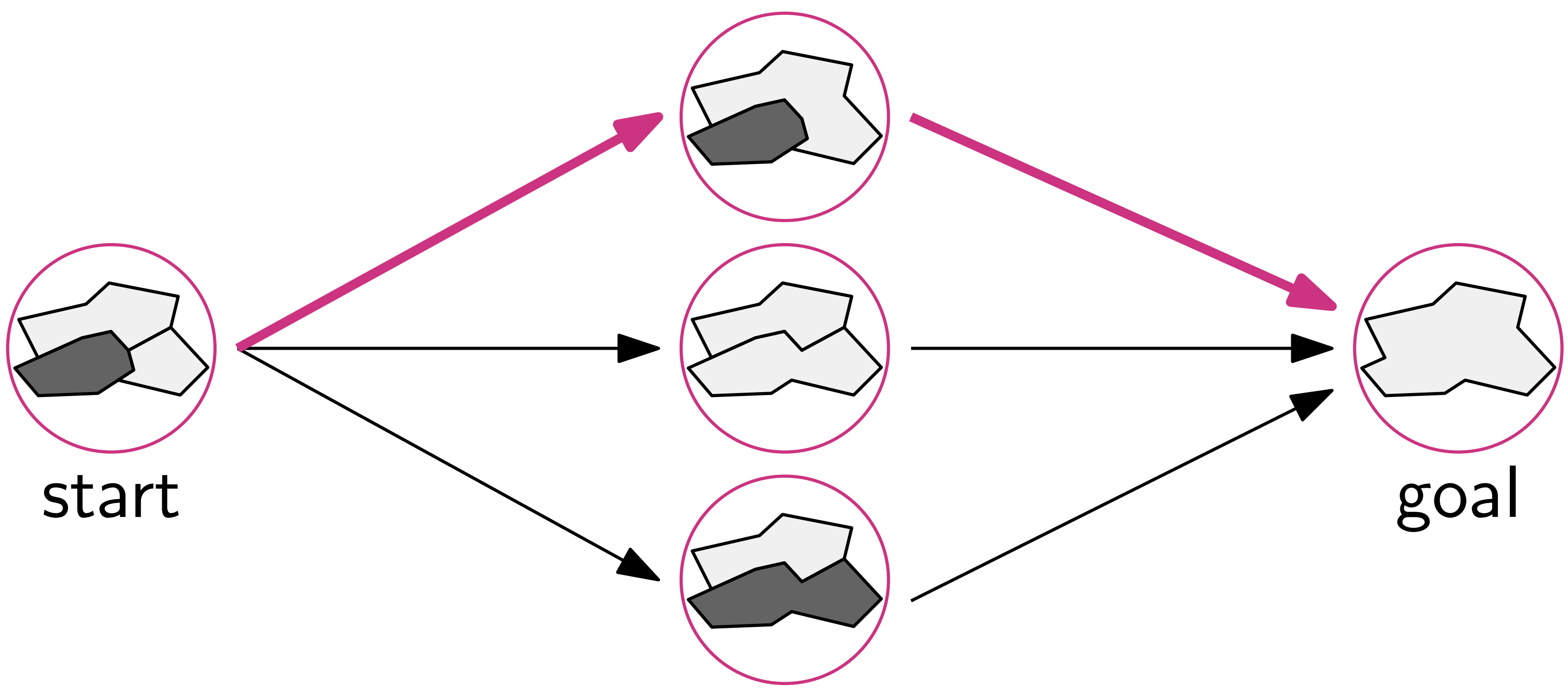
Overestimation

- 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



Integer Linear Programming

Form of an integer linear program (ILP)

$$\begin{aligned} \text{minimize} \quad & C^{\mathsf{T}} x \\ \text{subject to} \quad & Ex \leq H, \\ & x \geq \mathbf{0}, \\ \text{and} \quad & x \in \mathbb{Z}^{l}, \end{aligned}$$

Integer Linear Programming

Form of an integer linear program (ILP)

$$\begin{aligned} \text{minimize} \quad & C^{\mathsf{T}} x \\ \text{subject to} \quad & Ex \leq H, \\ & x \geq \mathbf{0}, \\ \text{and} \quad & x \in \mathbb{Z}^{l}, \end{aligned}$$

Given variables x,
minimize a cost subject to some constraints.

Integer Linear Programming

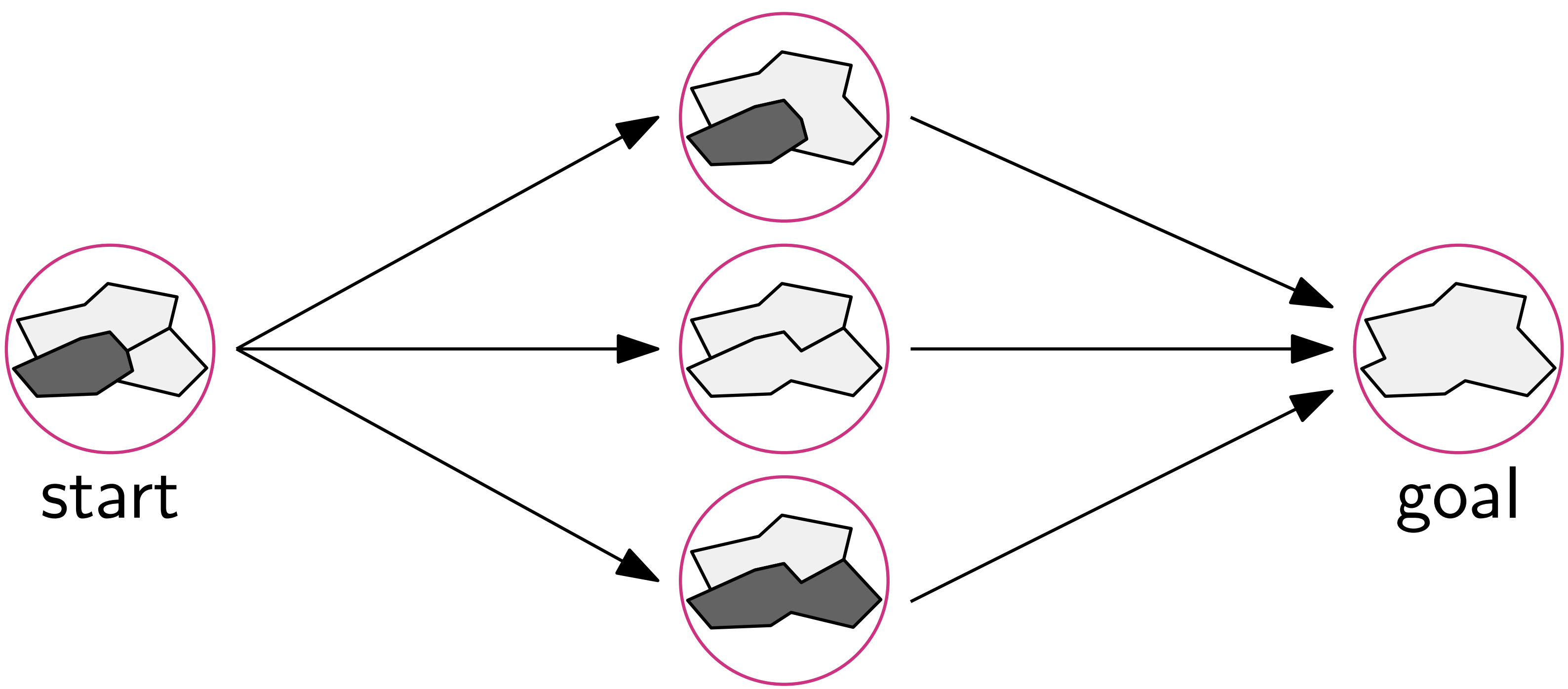
Form of an integer linear program (ILP)

$$\begin{aligned} \text{minimize} \quad & \boxed{c^\mathsf{T} x} \\ \text{subject to} \quad & Ex \leq H, \\ & x \geq \mathbf{0}, \\ \text{and} \quad & x \in \mathbb{Z}^l, \end{aligned}$$

Given variables x,
minimize a cost subject to some constraints.

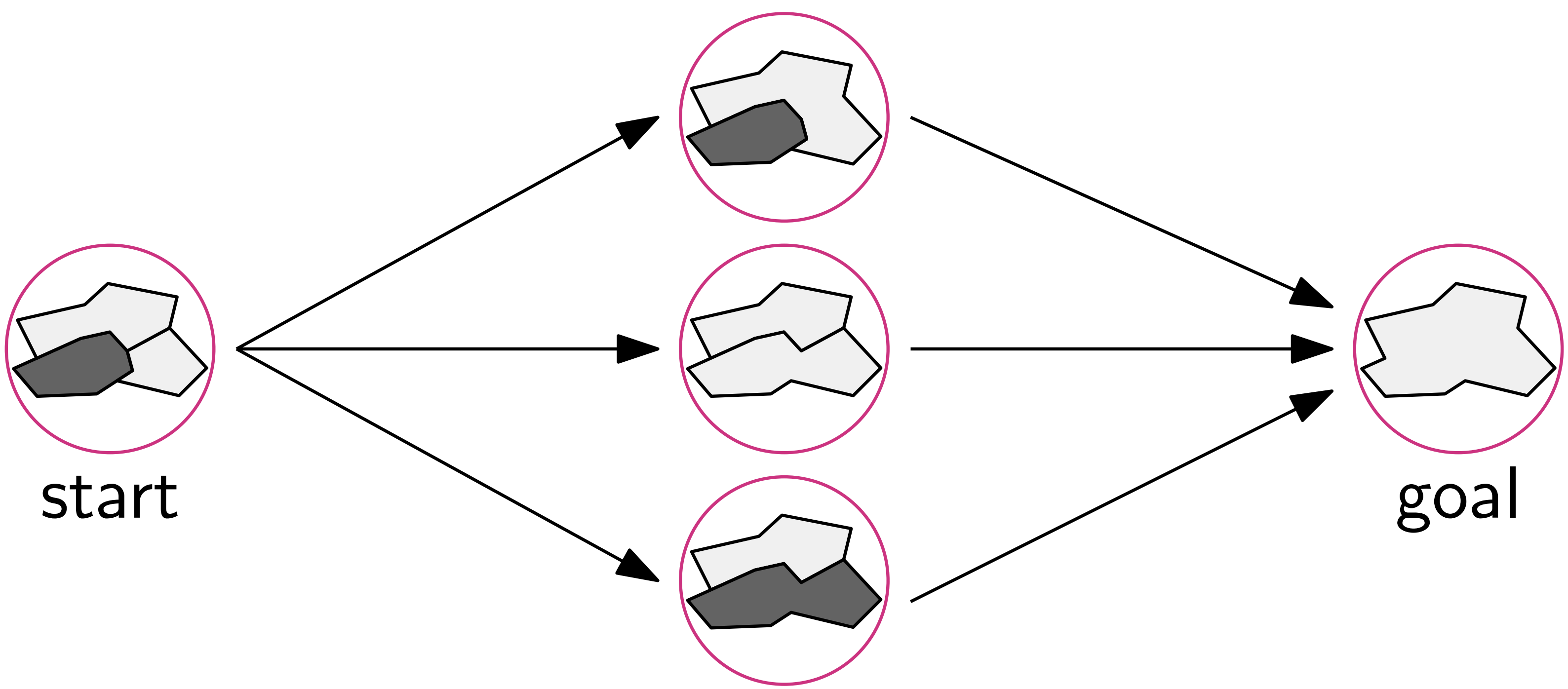
Using Integer Linear Programming

- Model complete graph by setting variables and constraints



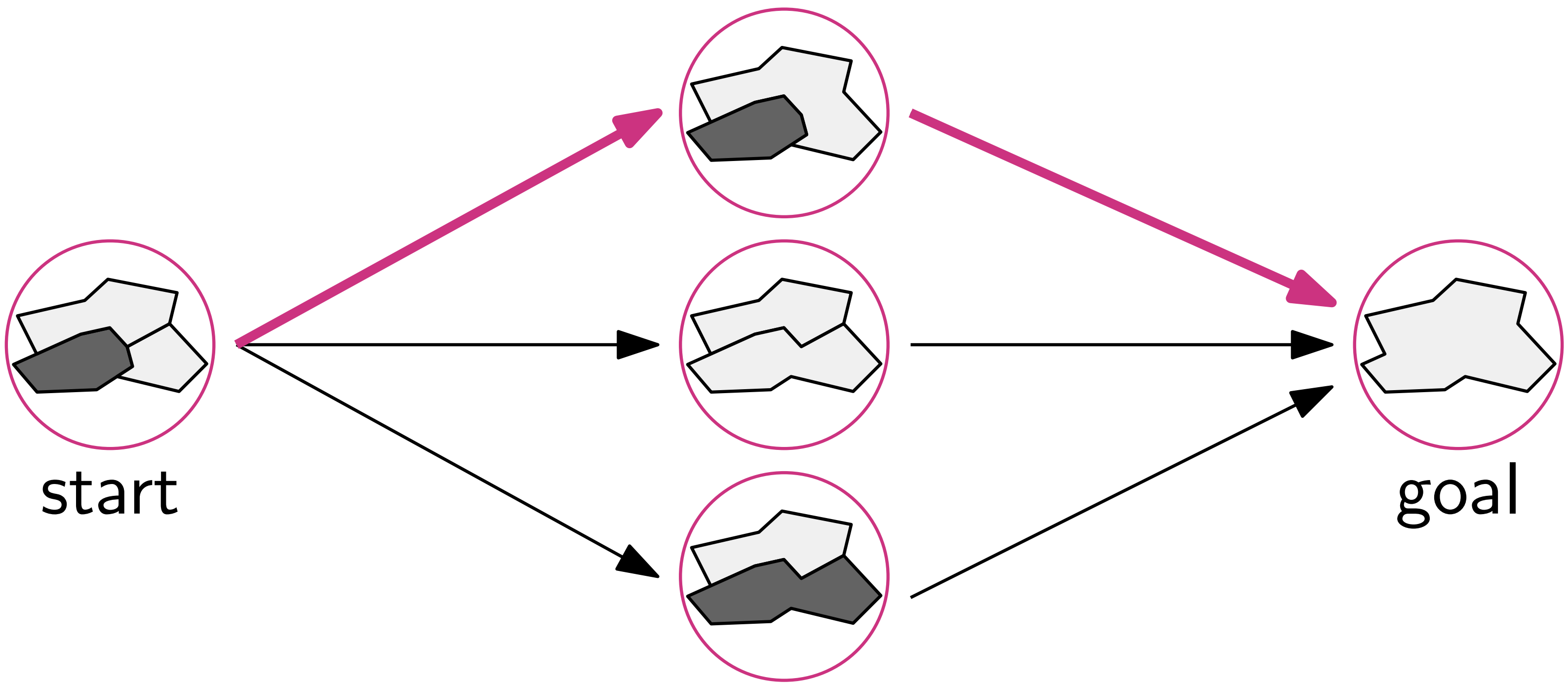
Using Integer Linear Programming

- Model complete graph by setting variables and constraints
- Solve ILP with minimizing total cost



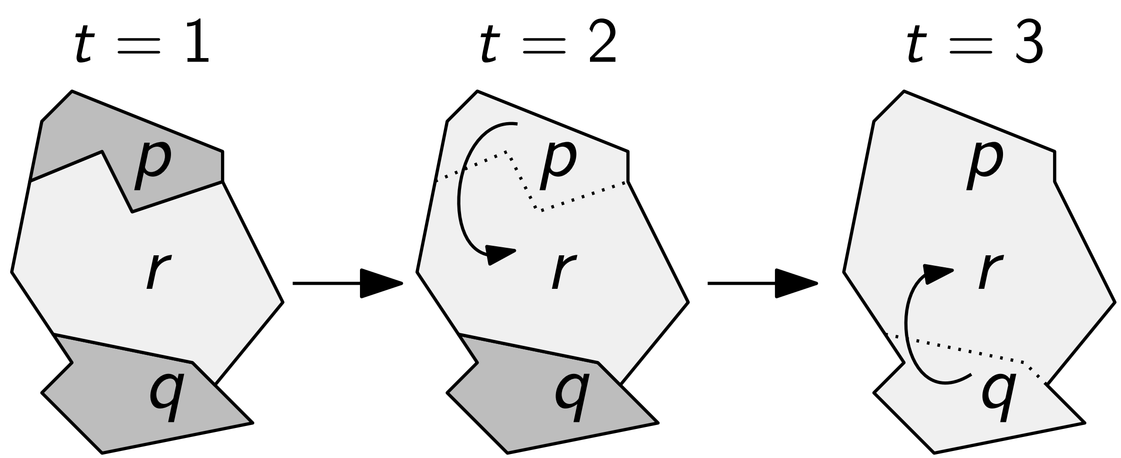
Using Integer Linear Programming

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



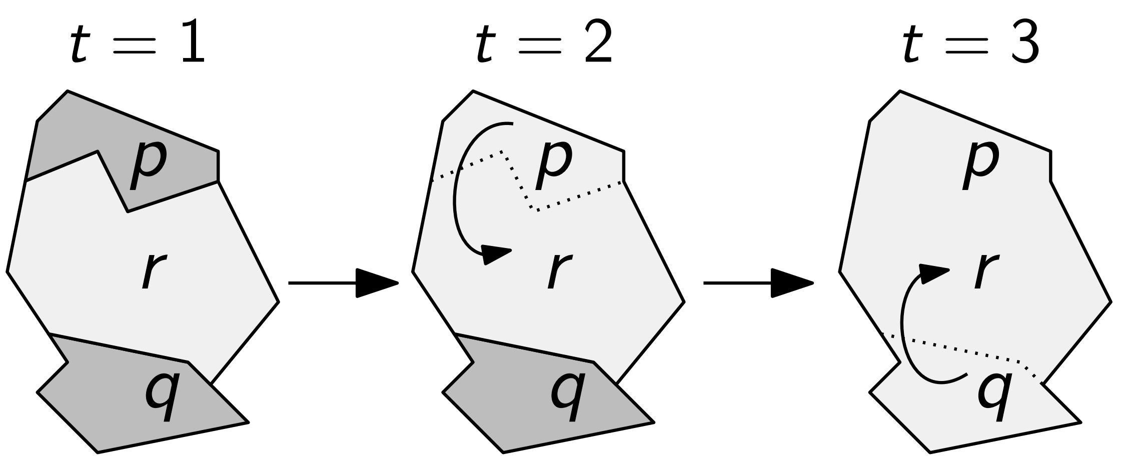
Example Variable and Constraints

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



Example Variable and Constraints

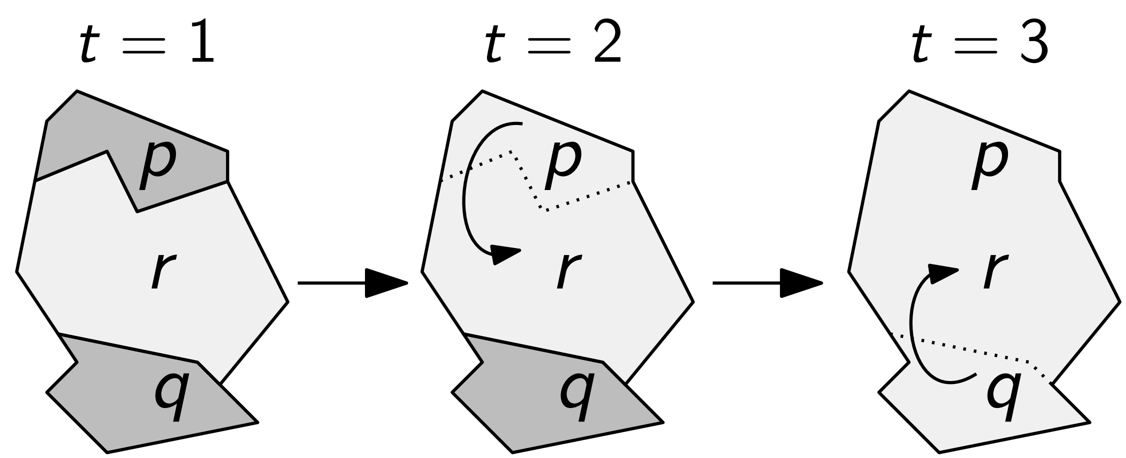
- Variable: $x_{t,p,r} \in \{0,1\} \qquad \forall t \in T, \forall p, r \in P$
  $x_{t,p,r} = 1 \Leftrightarrow p$ is assigned to r at time t.



$x_{1,p,r} = 0$
$x_{1,r,r} = 1$
$x_{1,q,r} = 0$

Example Variable and Constraints

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

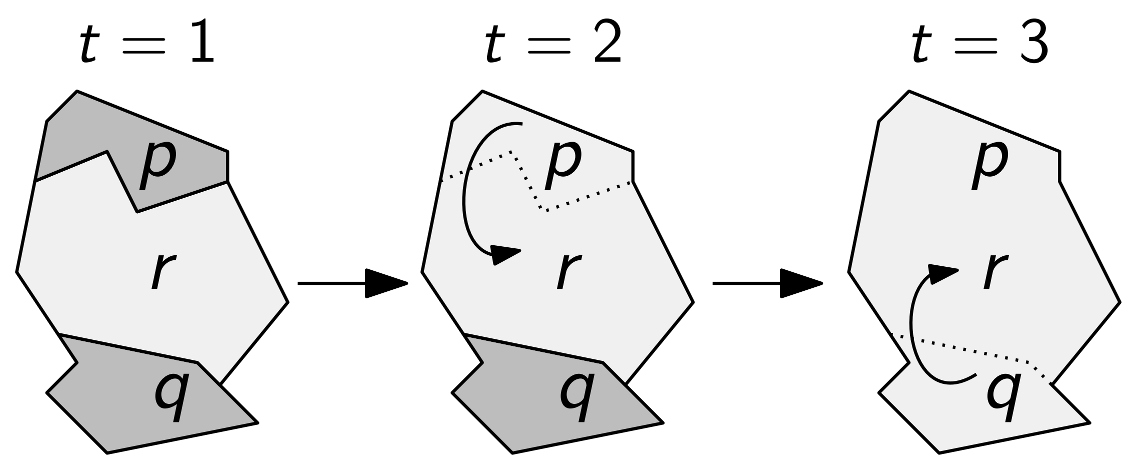


$x_{1,p,r} = 0$
$x_{1,r,r} = 1$
$x_{1,q,r} = 0$

$x_{2,p,r} = 1$
$x_{2,r,r} = 1$
$x_{2,q,r} = 0$

Example Variable and Constraints

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

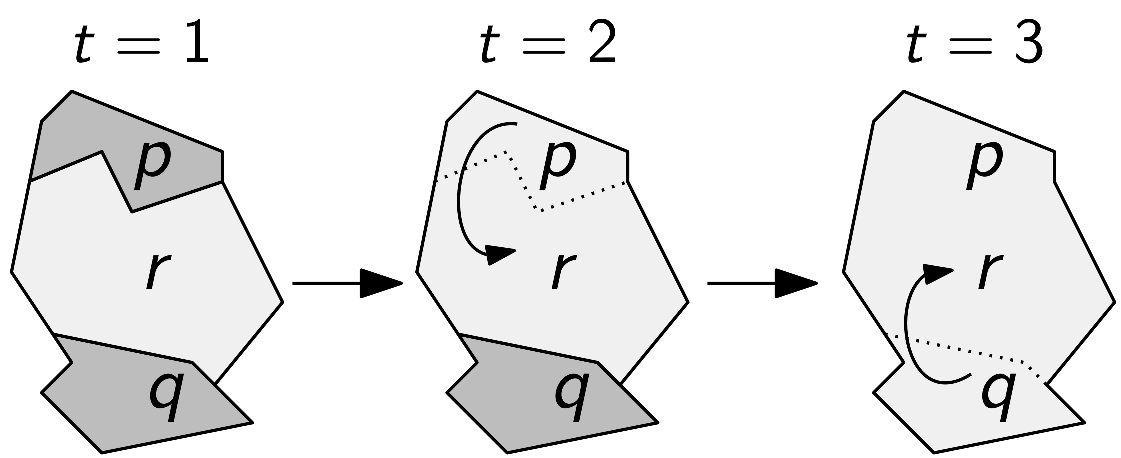


$x_{1,p,r} = 0$	$x_{2,p,r} = 1$	$x_{3,p,r} = 1$
$x_{1,r,r} = 1$	$x_{2,r,r} = 1$	$x_{3,r,r} = 1$
$x_{1,q,r} = 0$	$x_{2,q,r} = 0$	$x_{3,q,r} = 1$

Example Variable and Constraints

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

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



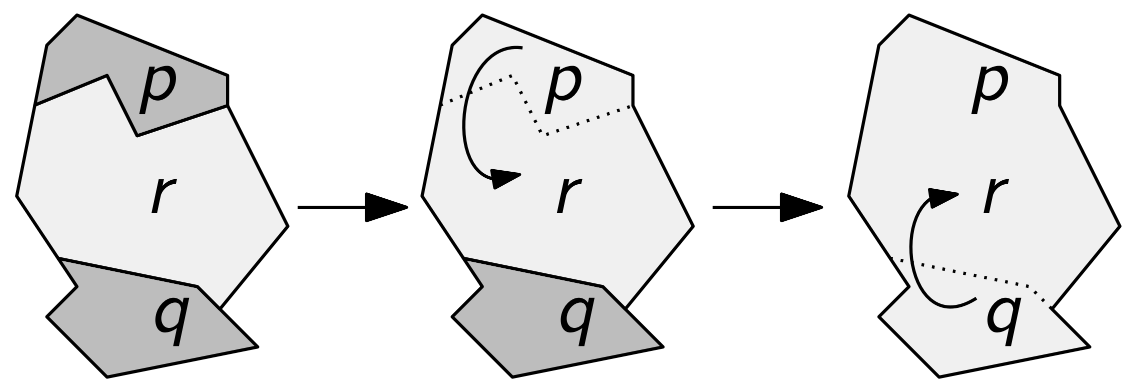
$t=1$	$t=2$	$t=3$
$x_{1,p,r} = 0$	$x_{2,p,r} = 1$	$x_{3,p,r} = 1$
$x_{1,r,r} = 1$	$x_{2,r,r} = 1$	$x_{3,r,r} = 1$
$x_{1,q,r} = 0$	$x_{2,q,r} = 0$	$x_{3,q,r} = 1$

Example Variable and Constraints

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

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

$t = 1 \qquad t = 2 \qquad t = 3$



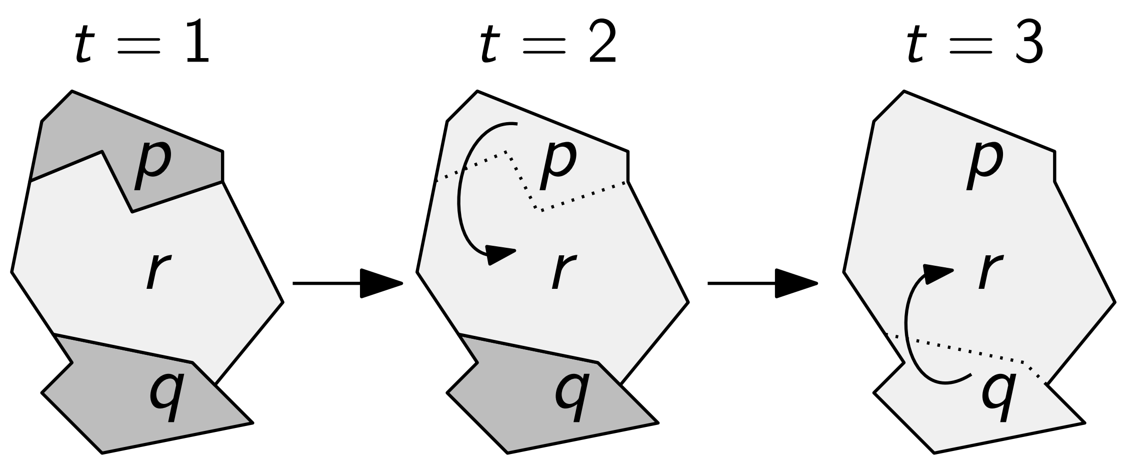
$x_{1,p,r} = 0$	$x_{2,p,r} = 1$	$x_{3,p,r} = 1$
$x_{1,r,r} = 1$	$x_{2,r,r} = 1$	$x_{3,r,r} = 1$
$x_{1,q,r} = 0$	$x_{2,q,r} = 0$	$x_{3,q,r} = 1$

Example Variable and Constraints

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

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

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



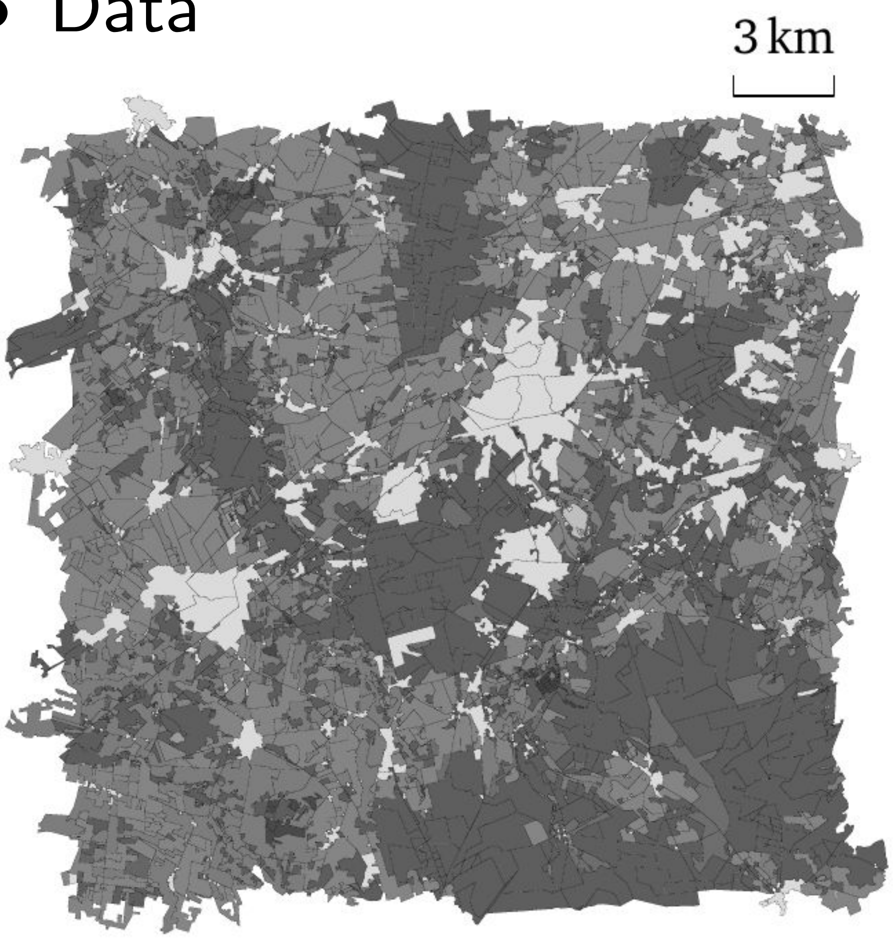
$x_{1,p,r} = 0$ $\quad$ $x_{2,p,r} = 1$ $\quad$ $x_{3,p,r} = 1$
$x_{1,r,r} = 1$ $\quad$ $x_{2,r,r} = 1$ $\quad$ $x_{3,r,r} = 1$
$x_{1,q,r} = 0$ $\quad$ $x_{2,q,r} = 0$ $\quad$ $x_{3,q,r} = 1$

Case Study

- Environment: C#, CPLEX

Case Study

- Environment: C#, CPLEX
- Data

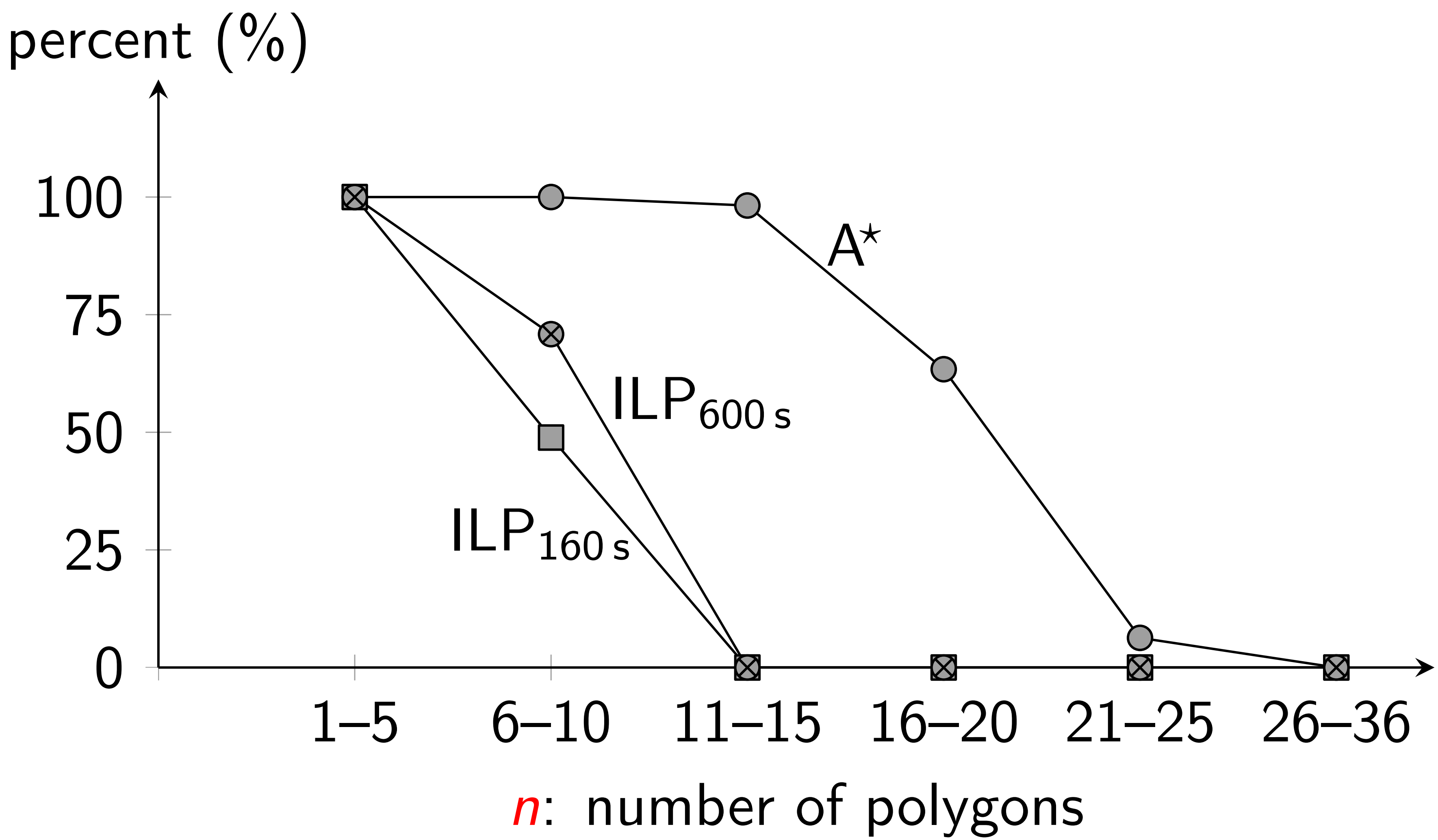


5,448 patches
scale 1 : 50 k

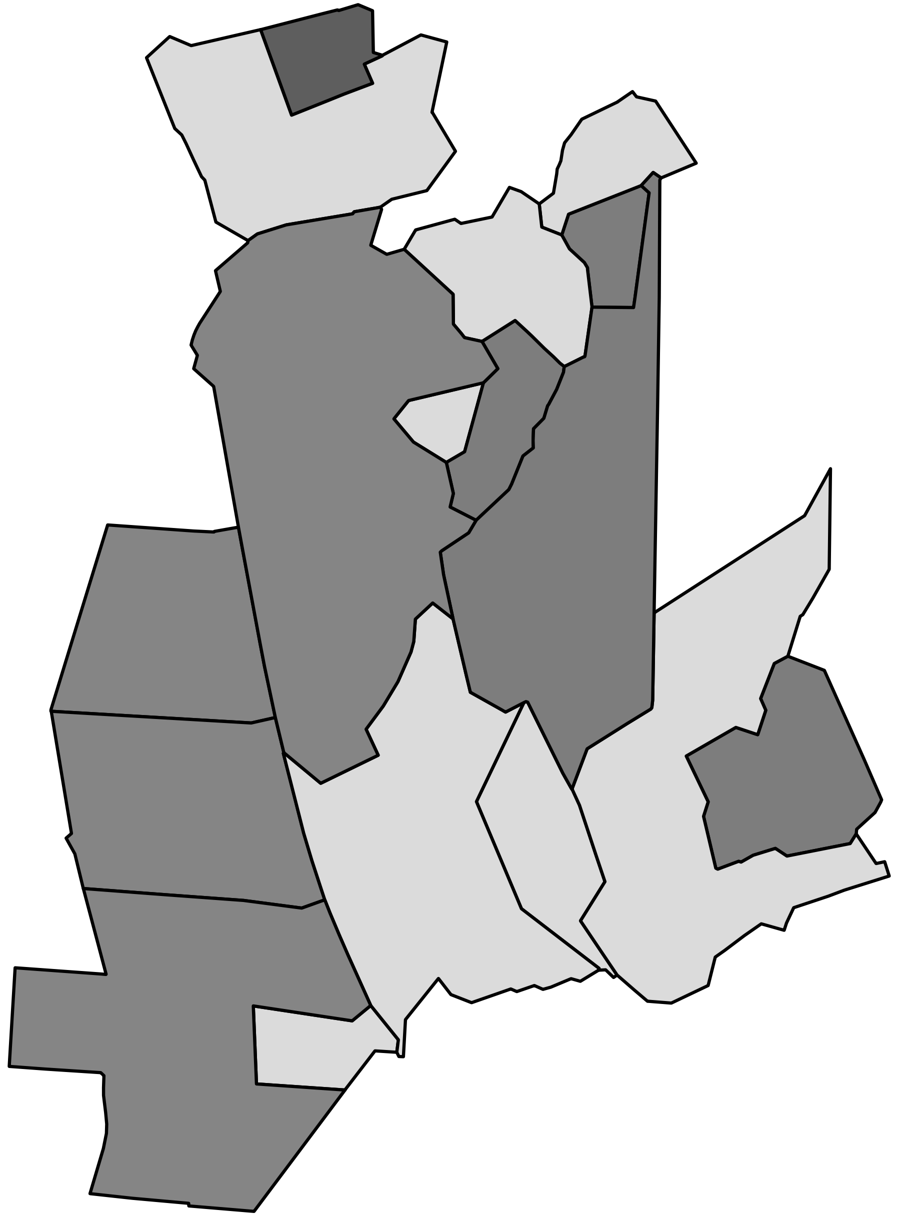
734 patches (regions)
scale 1 : 250 k

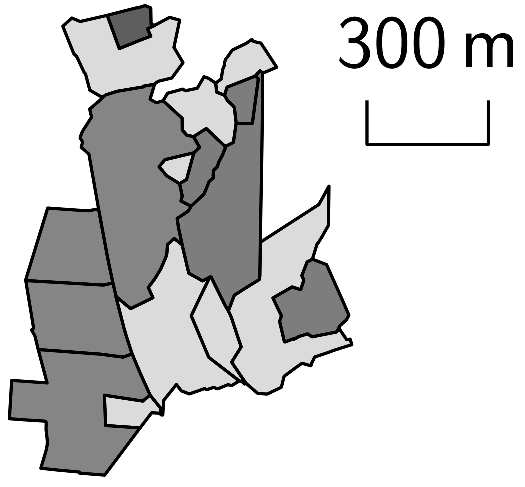
Comparison of A* and ILP

percentage of regions that were found optimal solutions



An Optimal Sequence by A*

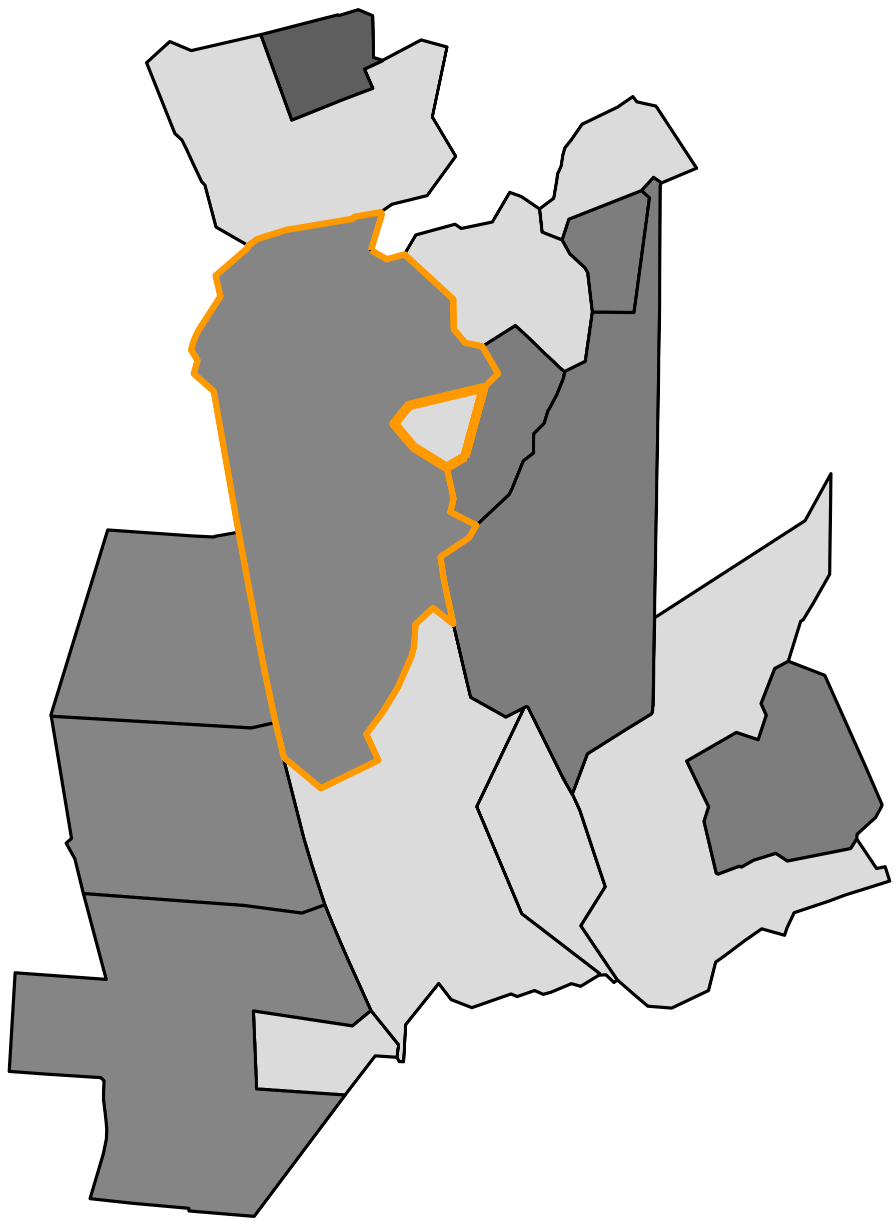


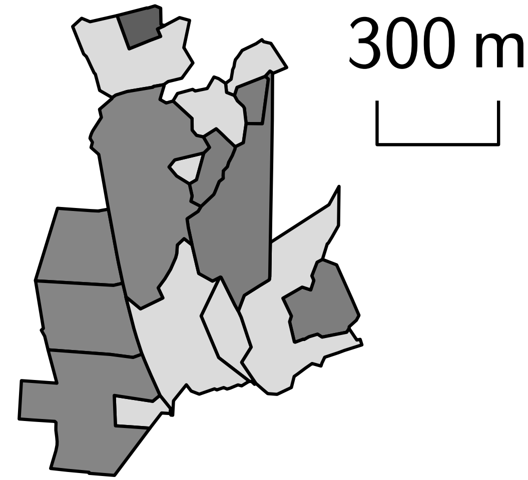


start ($n = 17$)

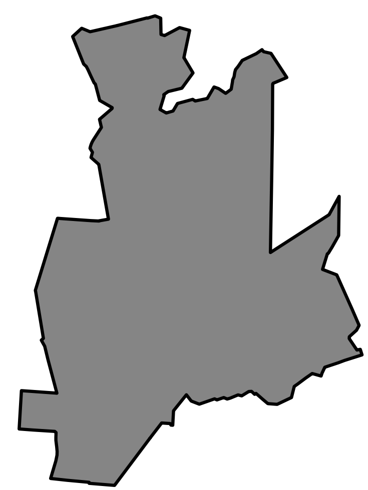
goal

An Optimal Sequence by A*



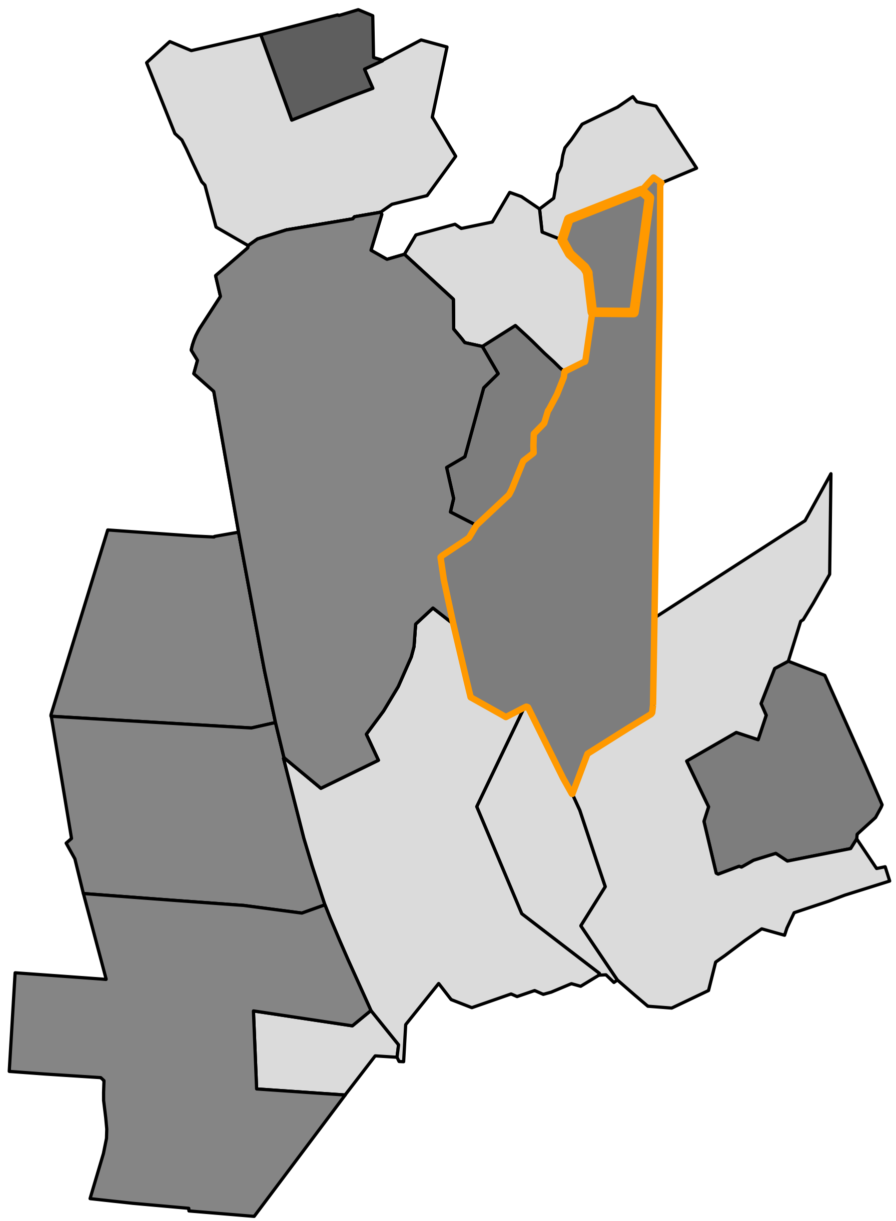


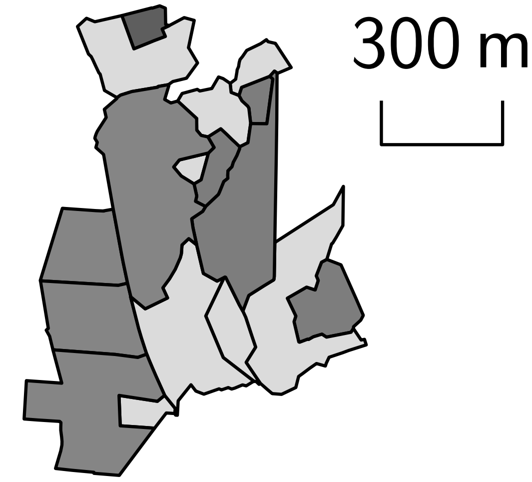
start ($n = 17$)



goal

An Optimal Sequence by A*

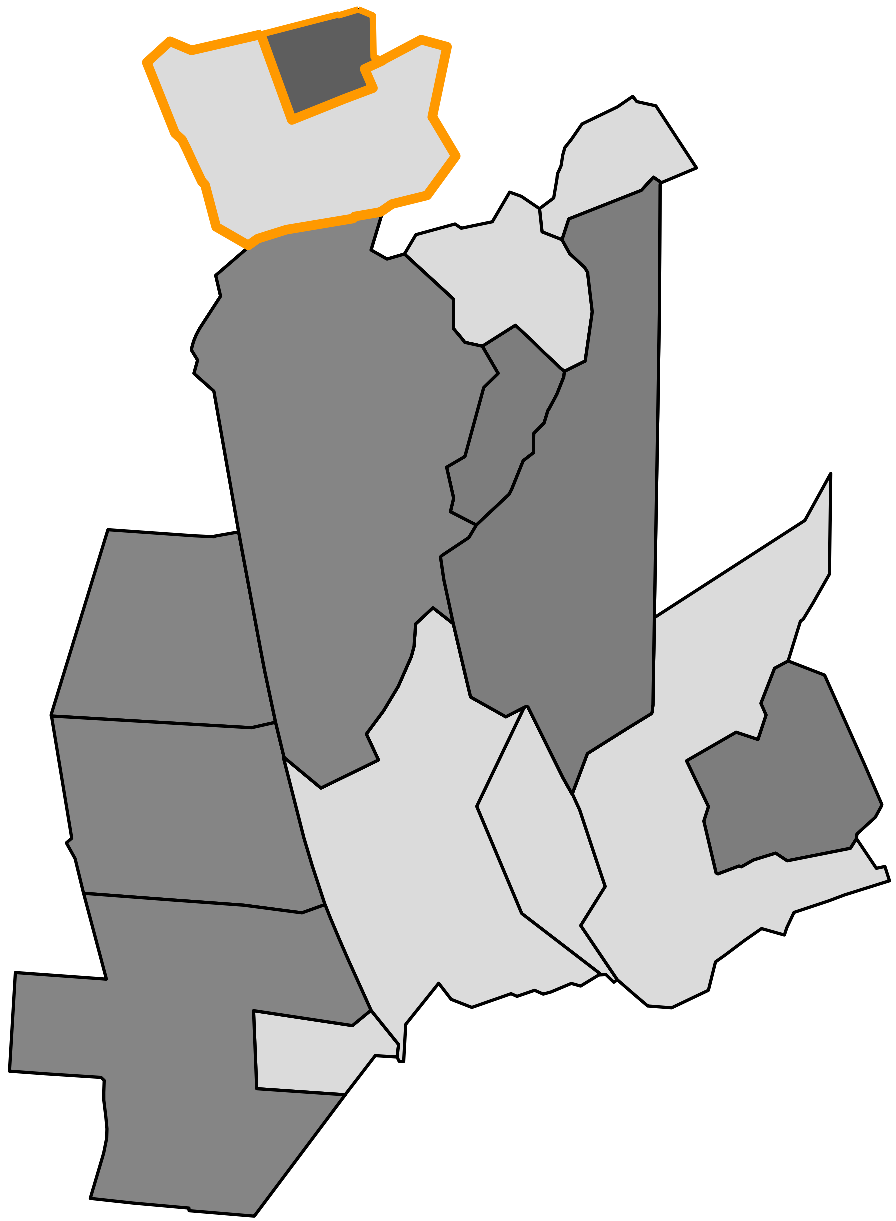


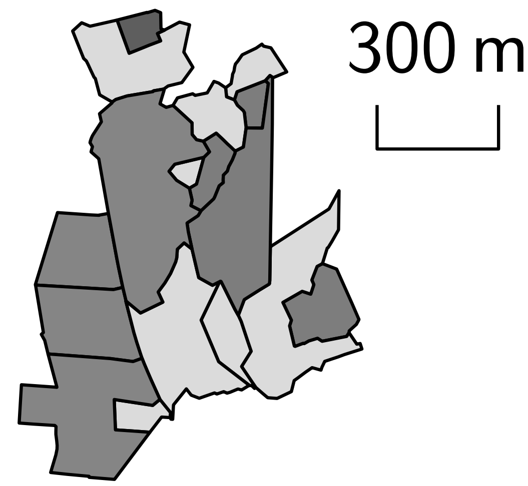


start ($n = 17$)

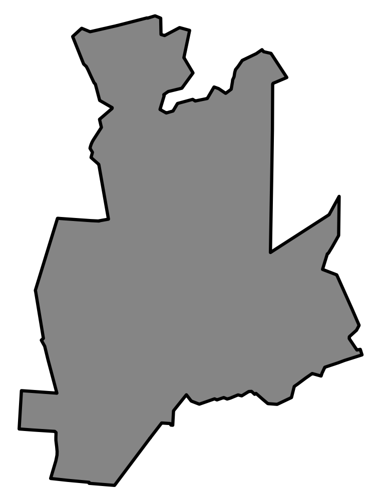
goal

An Optimal Sequence by A*



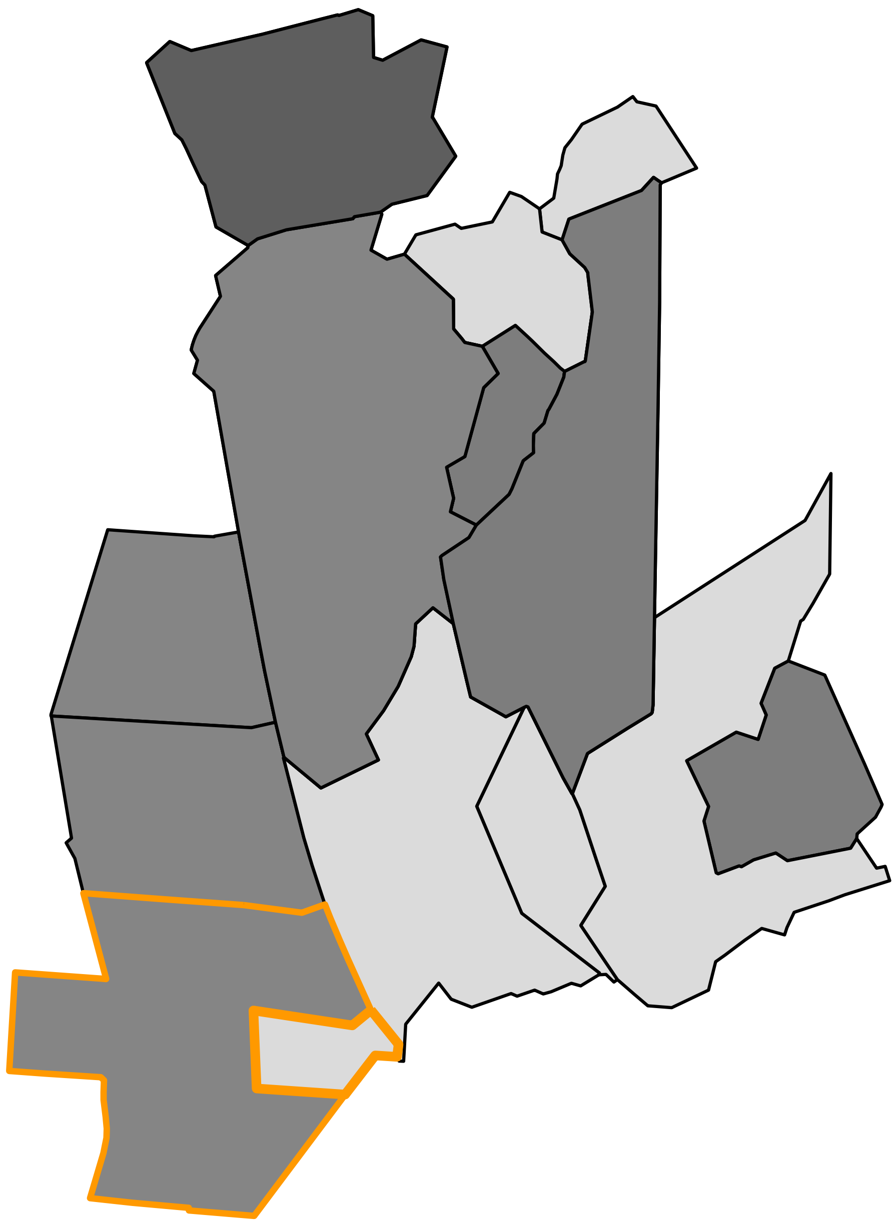


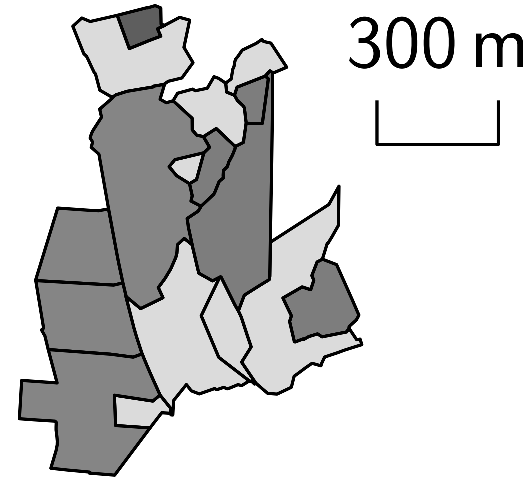
start ($n = 17$)



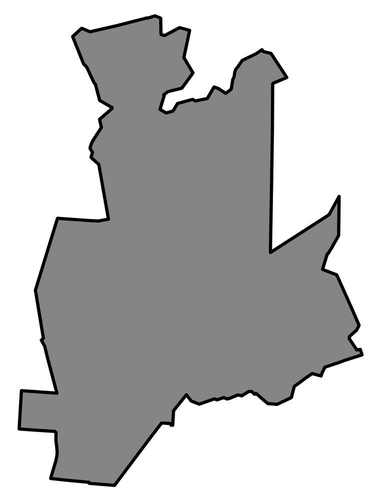
goal

An Optimal Sequence by A*



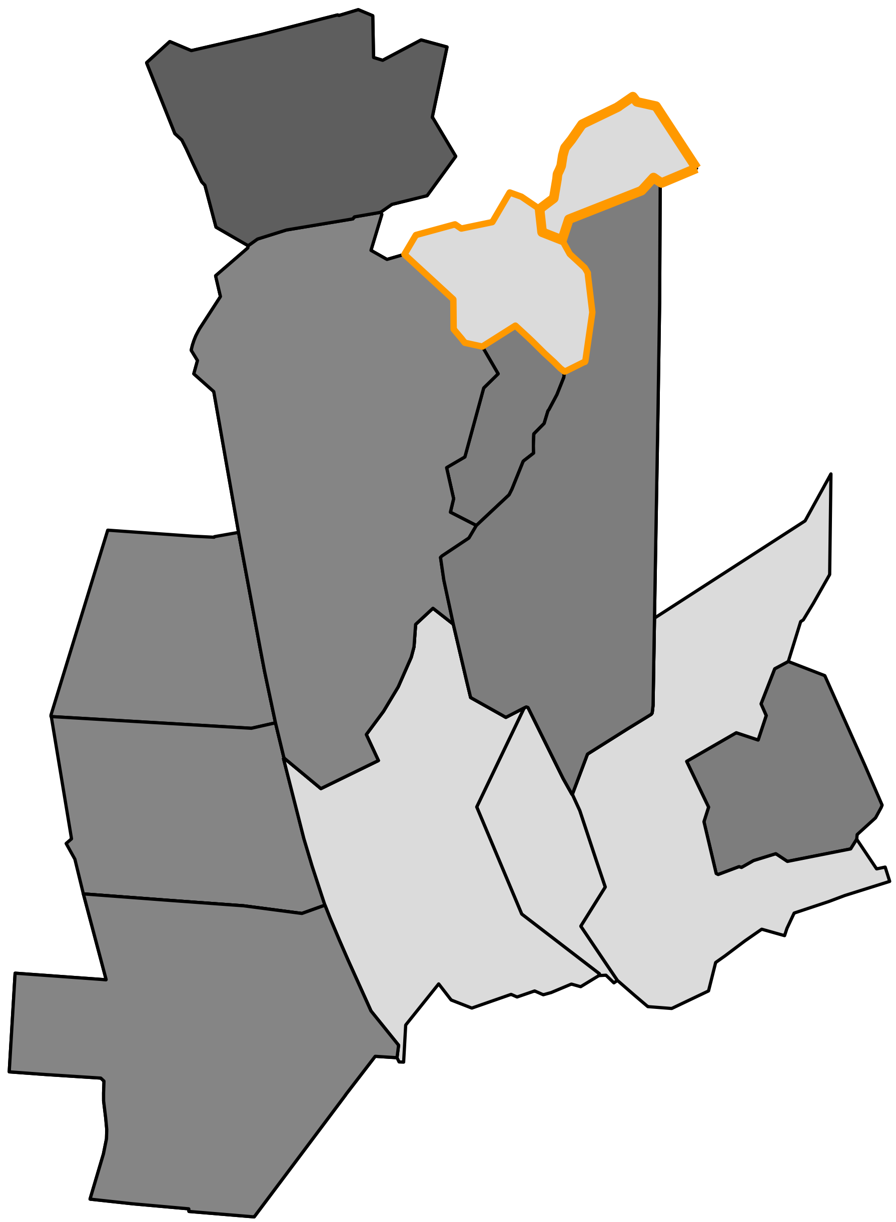


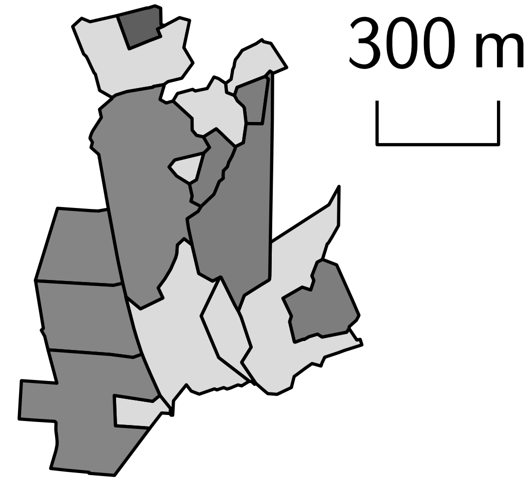
start ($n = 17$)



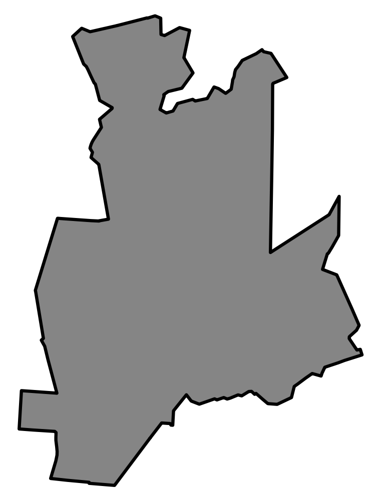
goal

An Optimal Sequence by A*



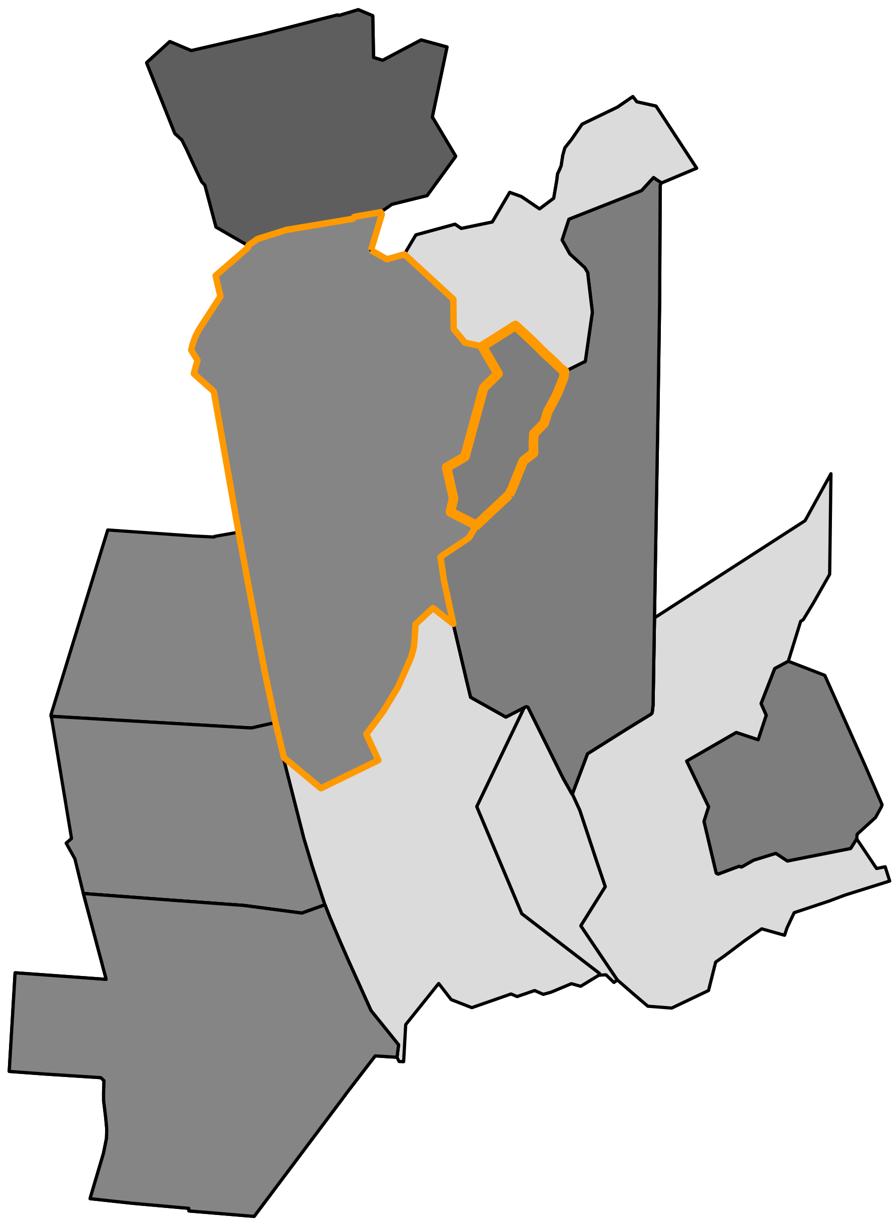


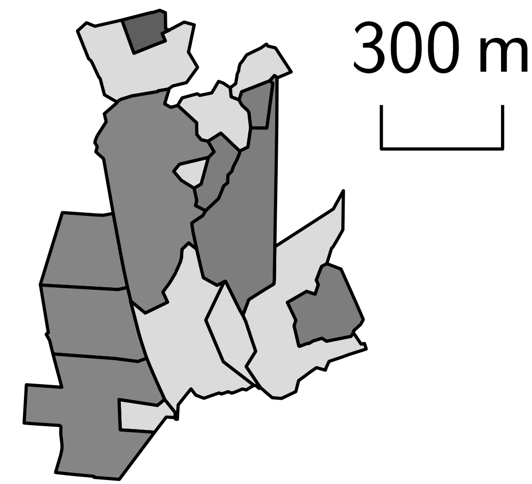
start ($n = 17$)



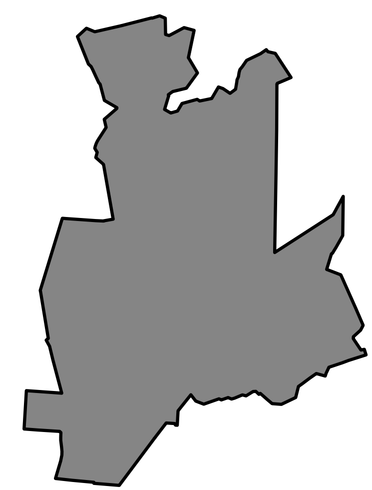
goal

An Optimal Sequence by A*



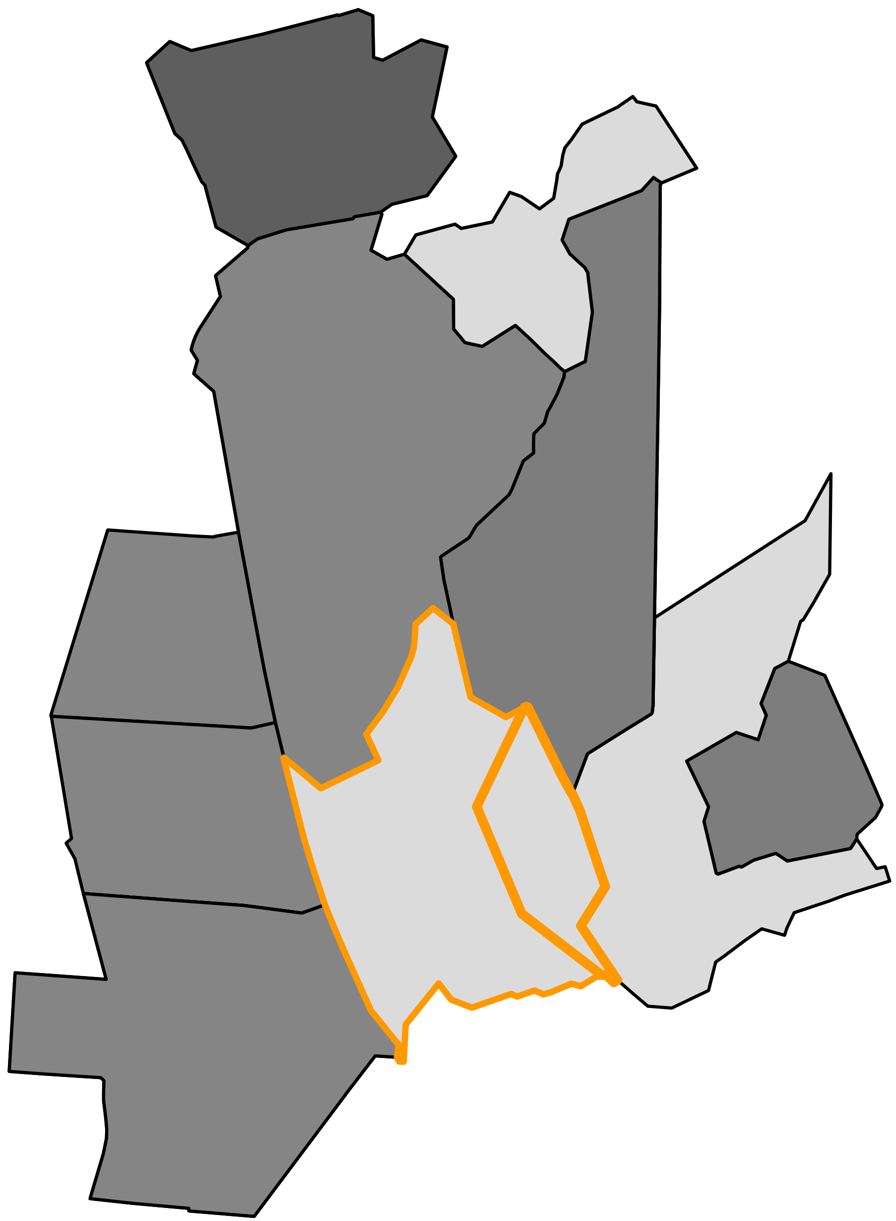


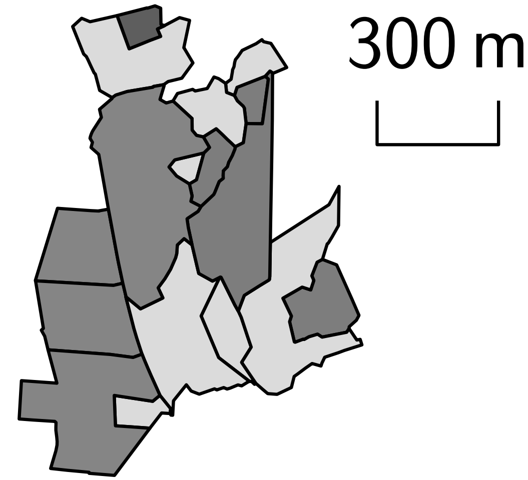
start ($n = 17$)



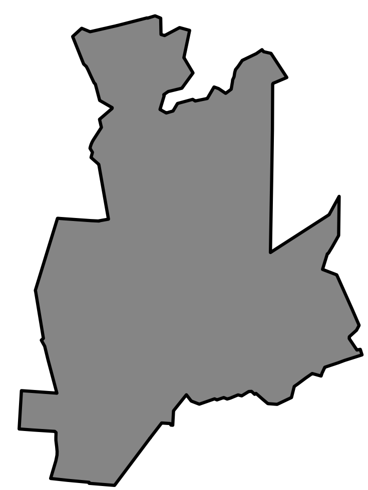
goal

An Optimal Sequence by A*



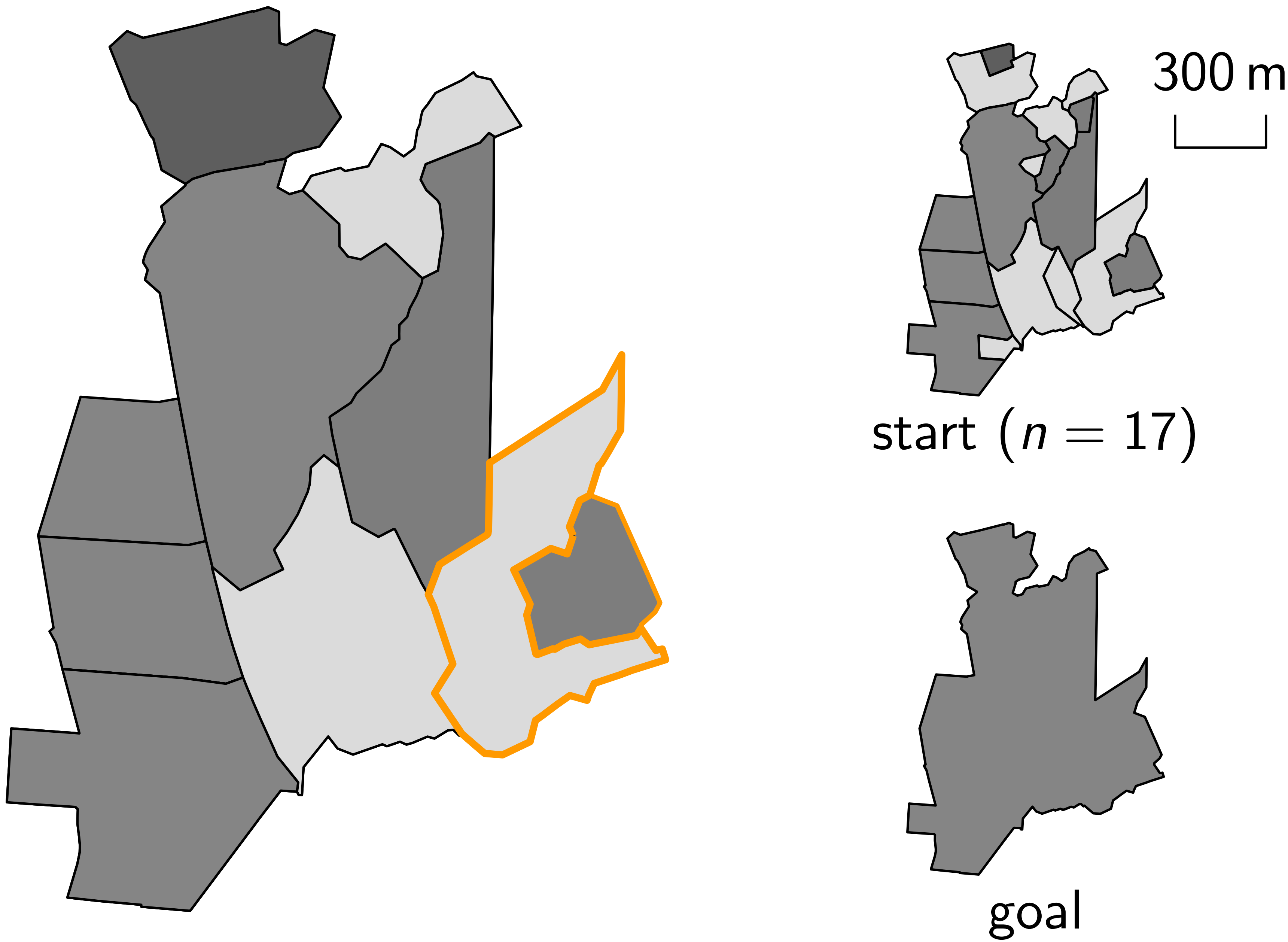


start ($n = 17$)

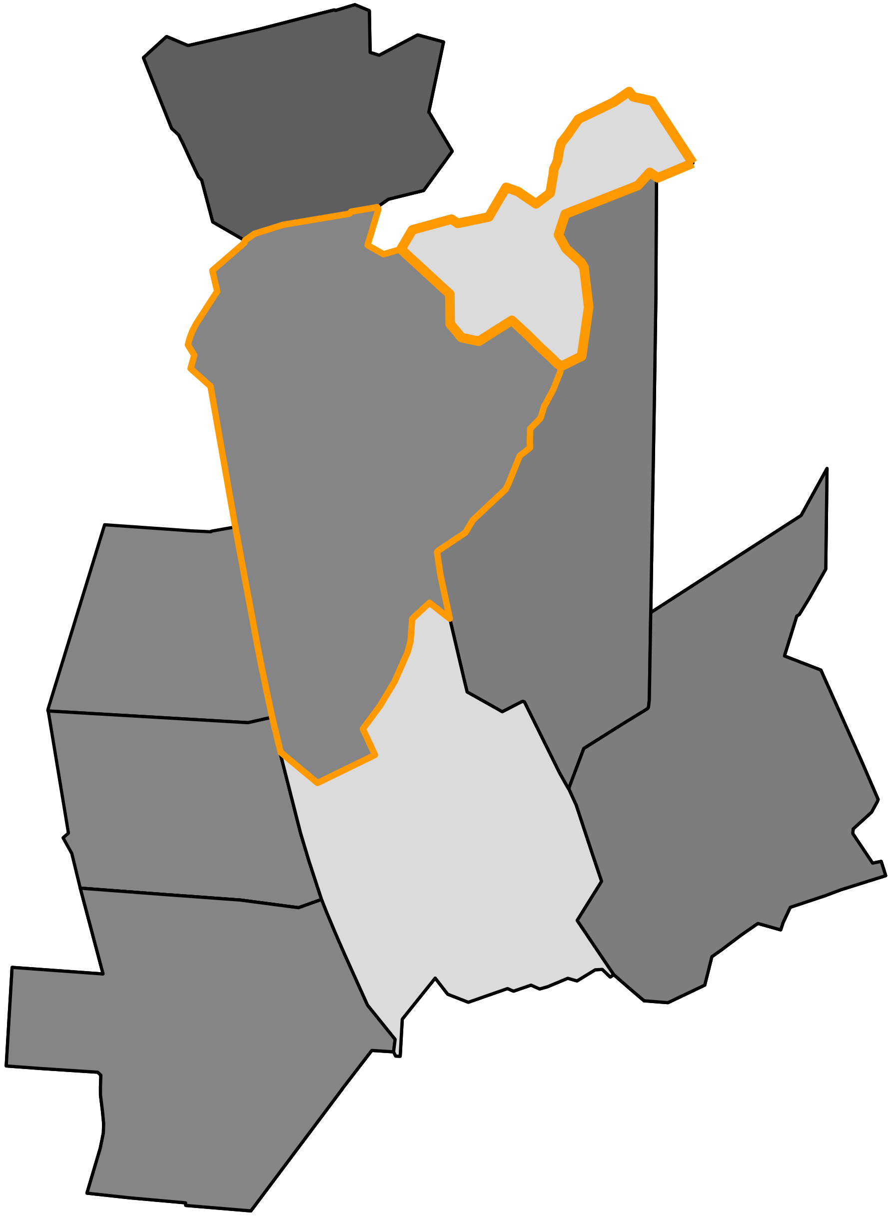


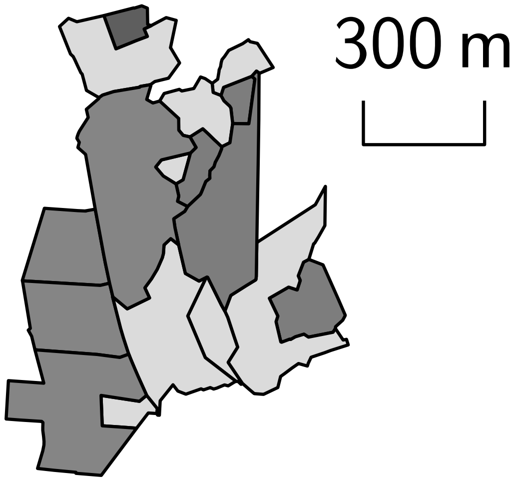
goal

An Optimal Sequence by A*

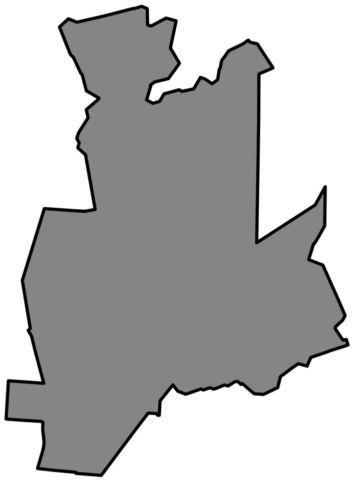


An Optimal Sequence by A*



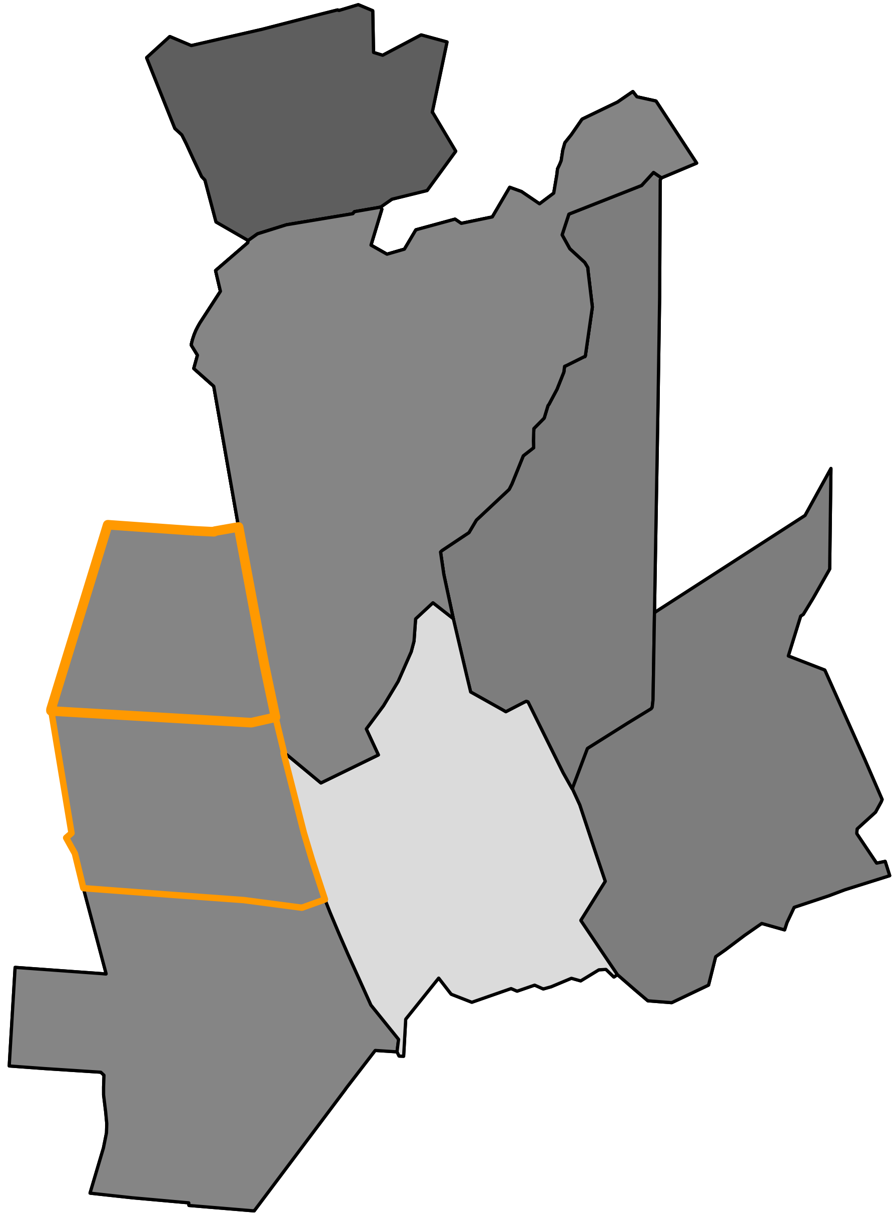


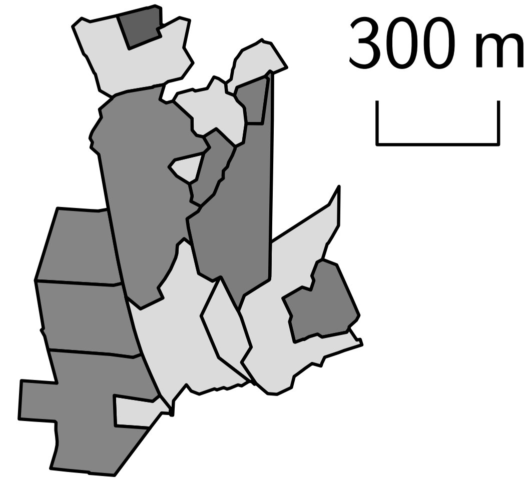
start ($n = 17$)



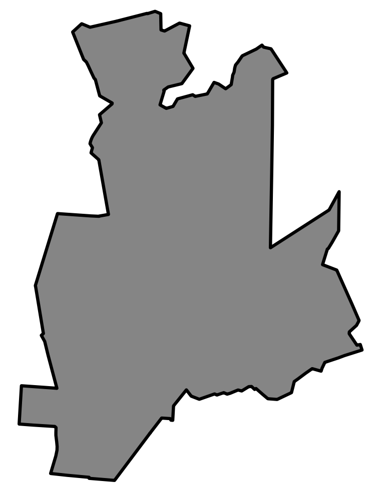
goal

An Optimal Sequence by A*



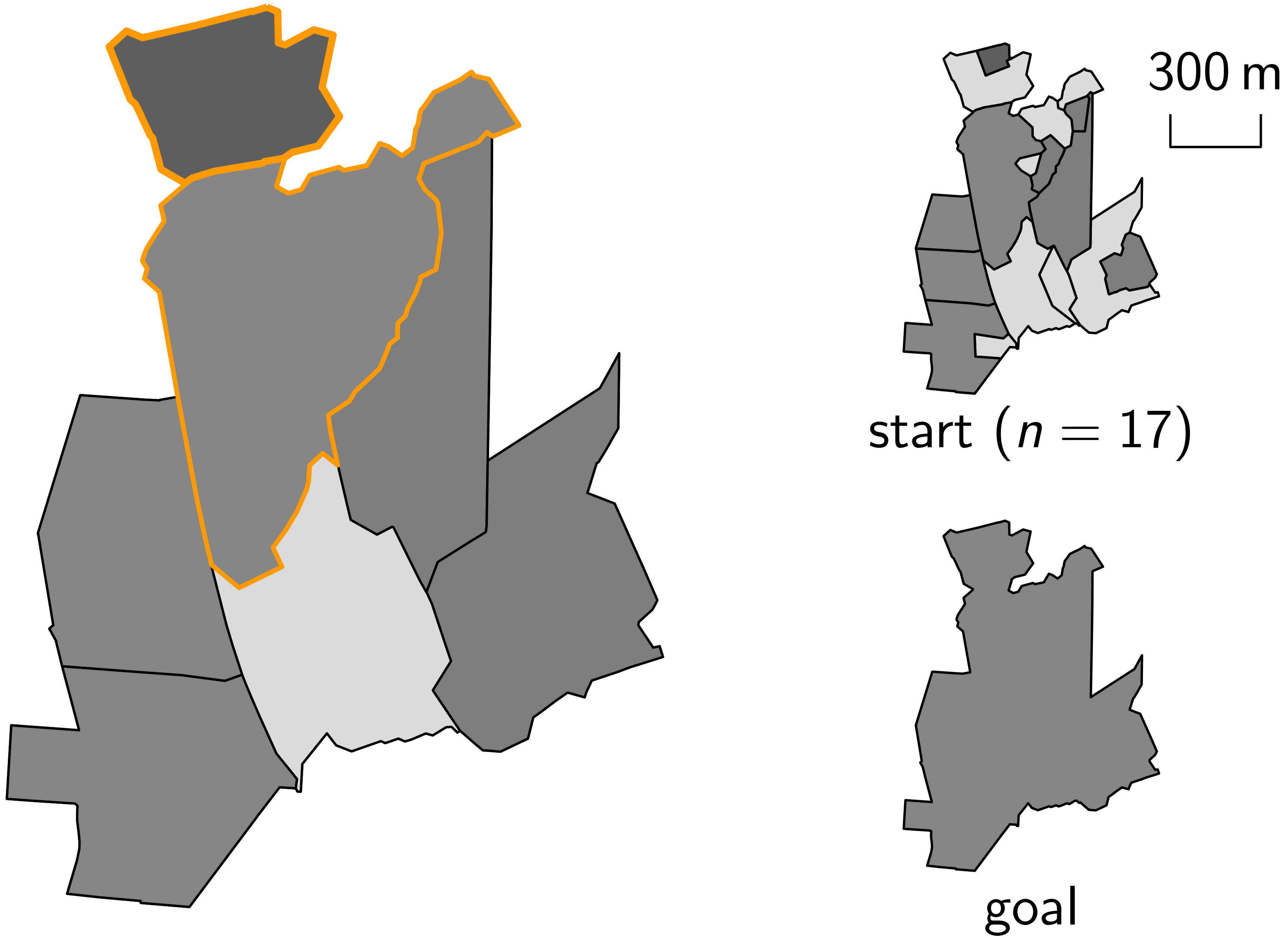


start ($n = 17$)

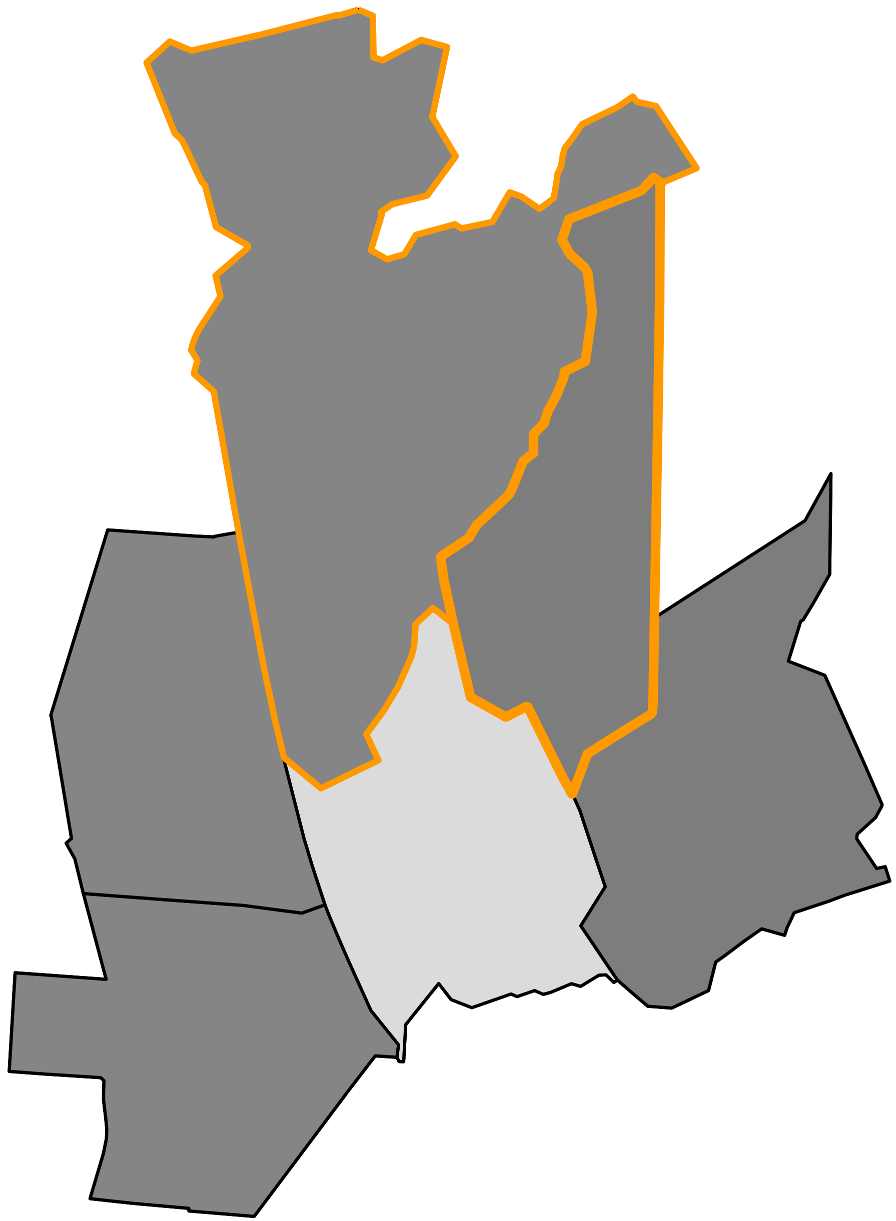


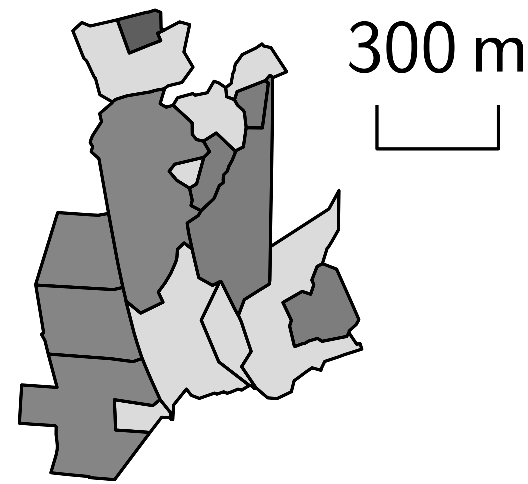
goal

An Optimal Sequence by A*

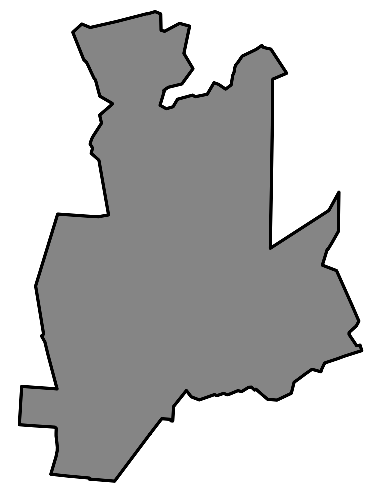


An Optimal Sequence by A*



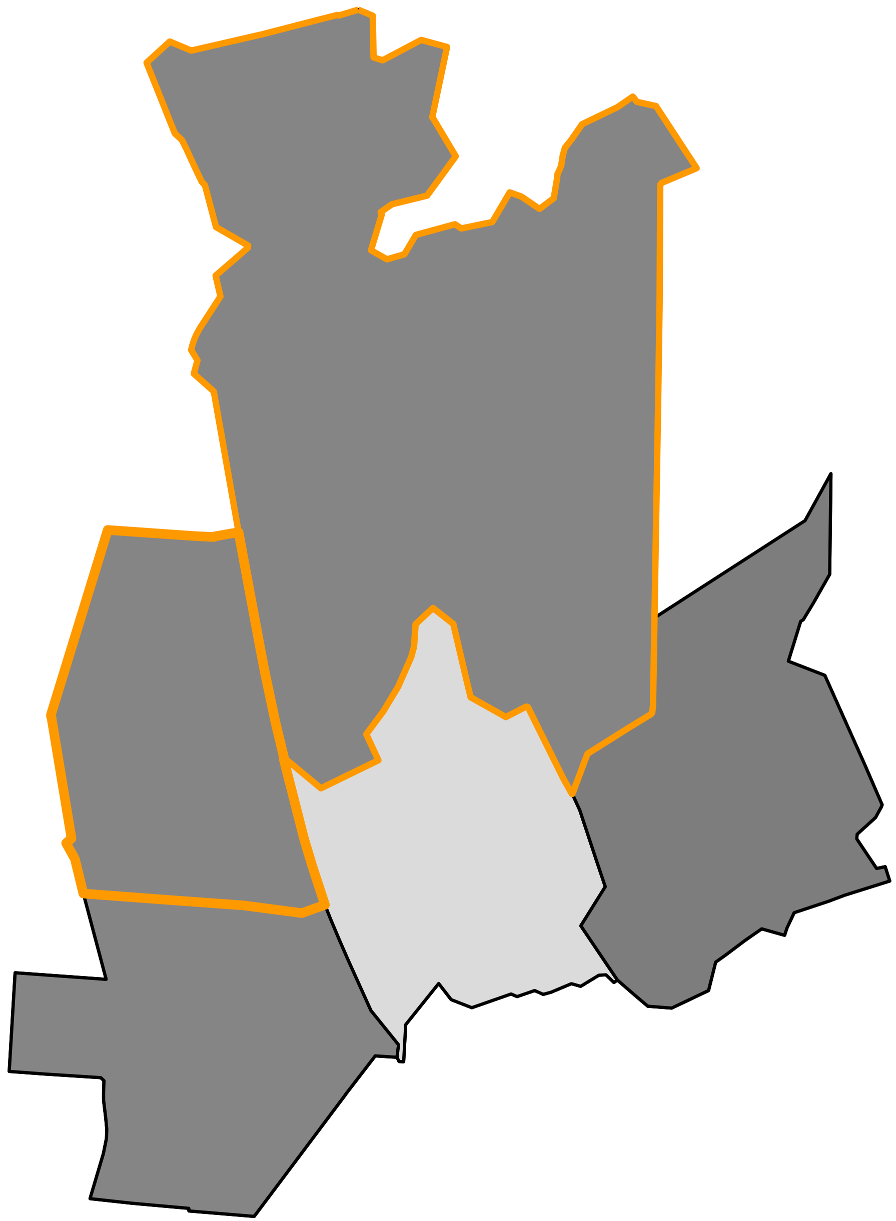


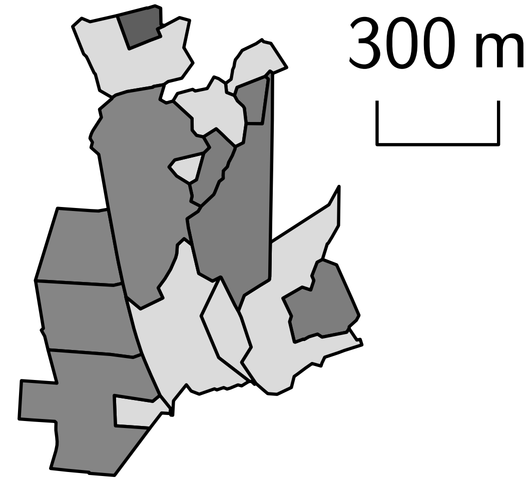
start ($n = 17$)



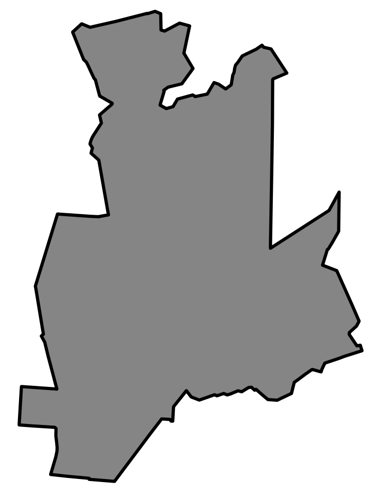
goal

An Optimal Sequence by A*



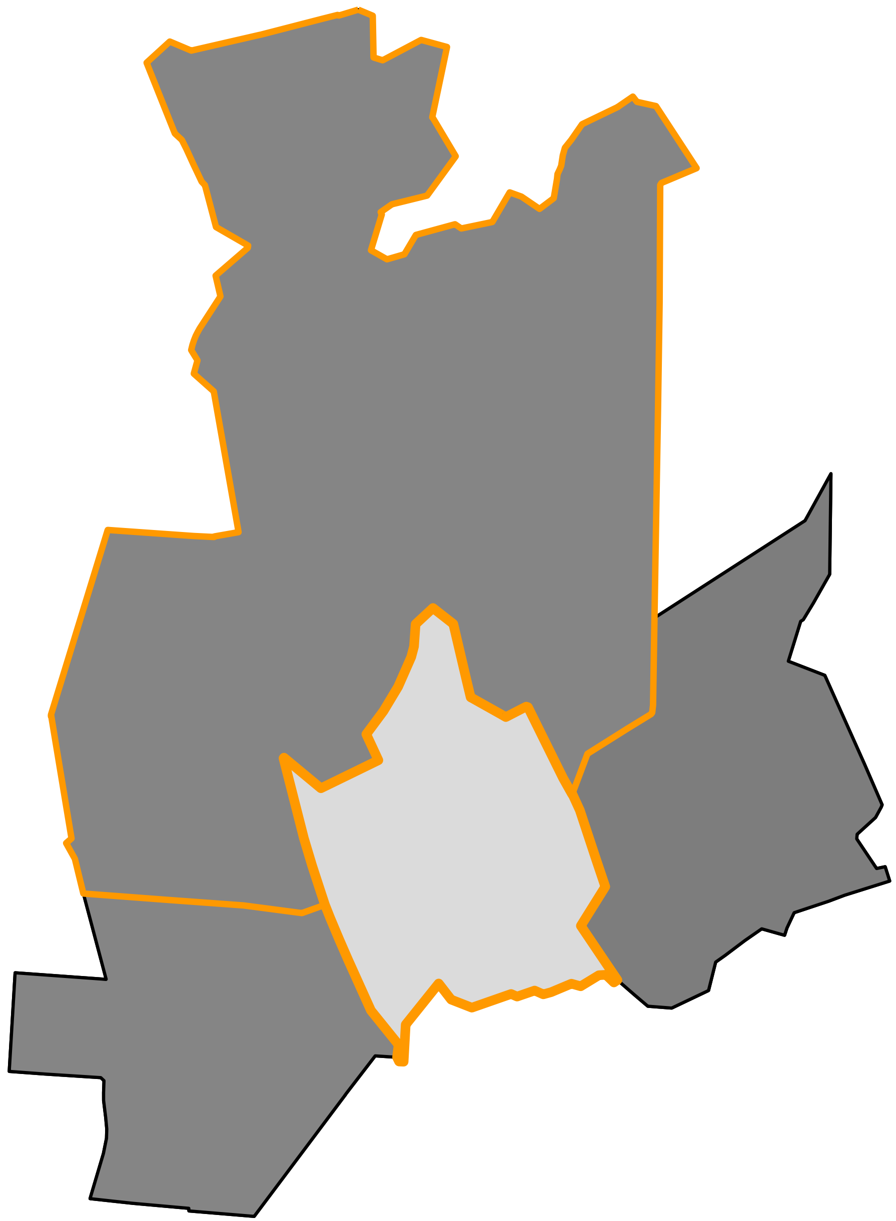


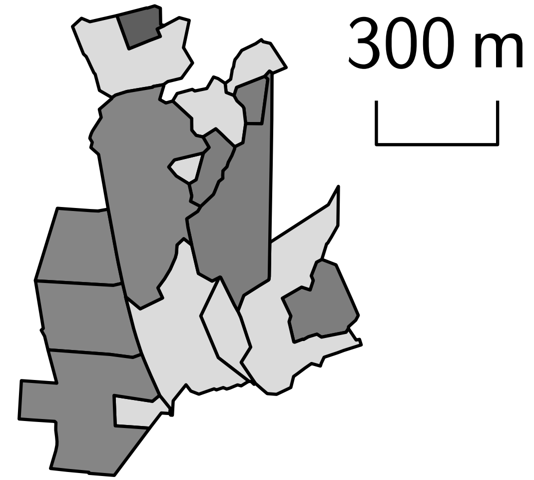
start ($n = 17$)



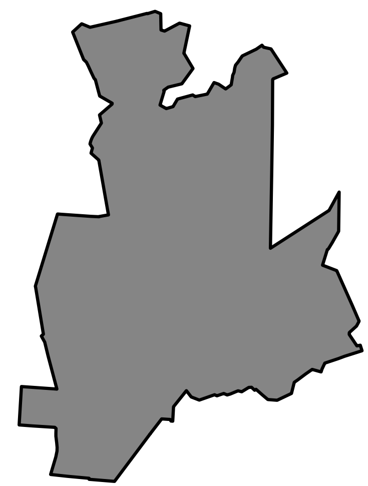
goal

An Optimal Sequence by A*



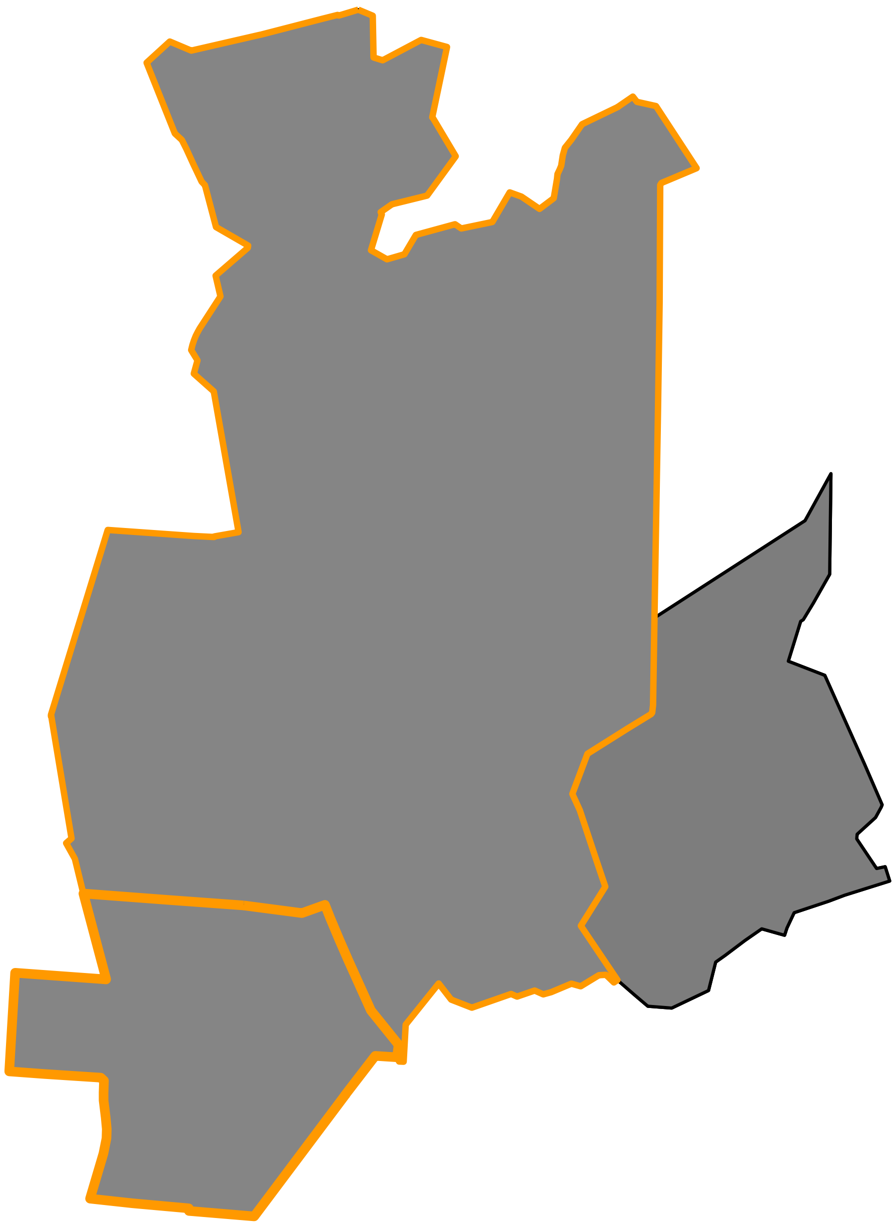


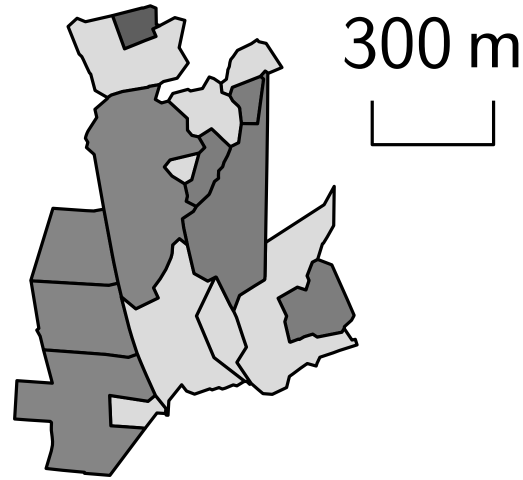
start ($n = 17$)



goal

An Optimal Sequence by A*

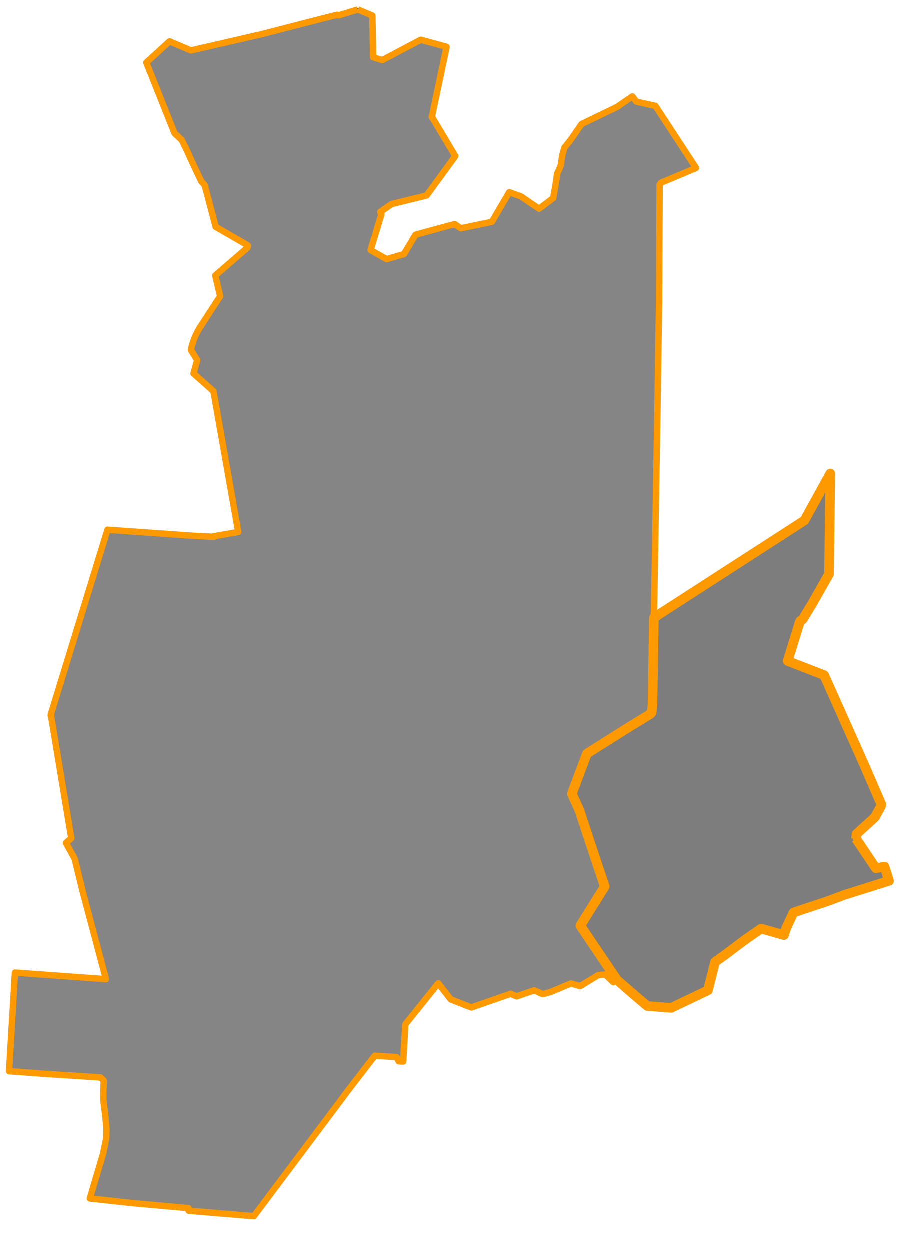


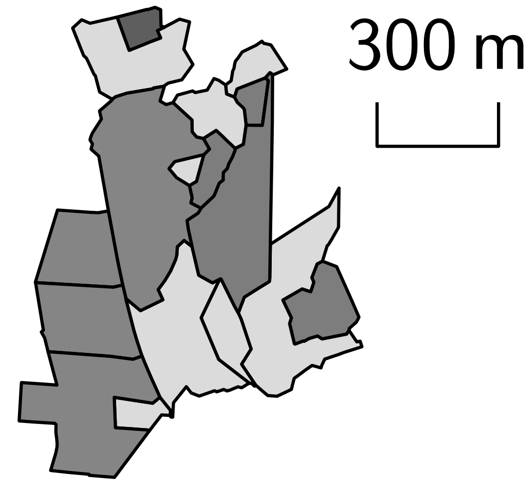


start ($n = 17$)

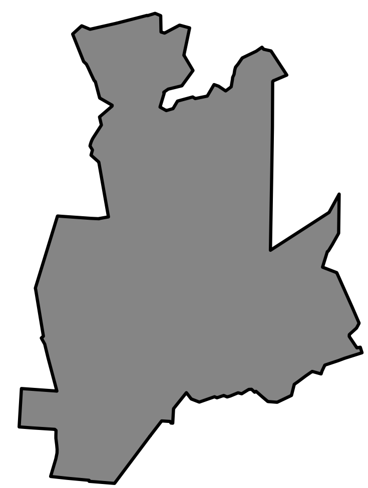
goal

An Optimal Sequence by A*



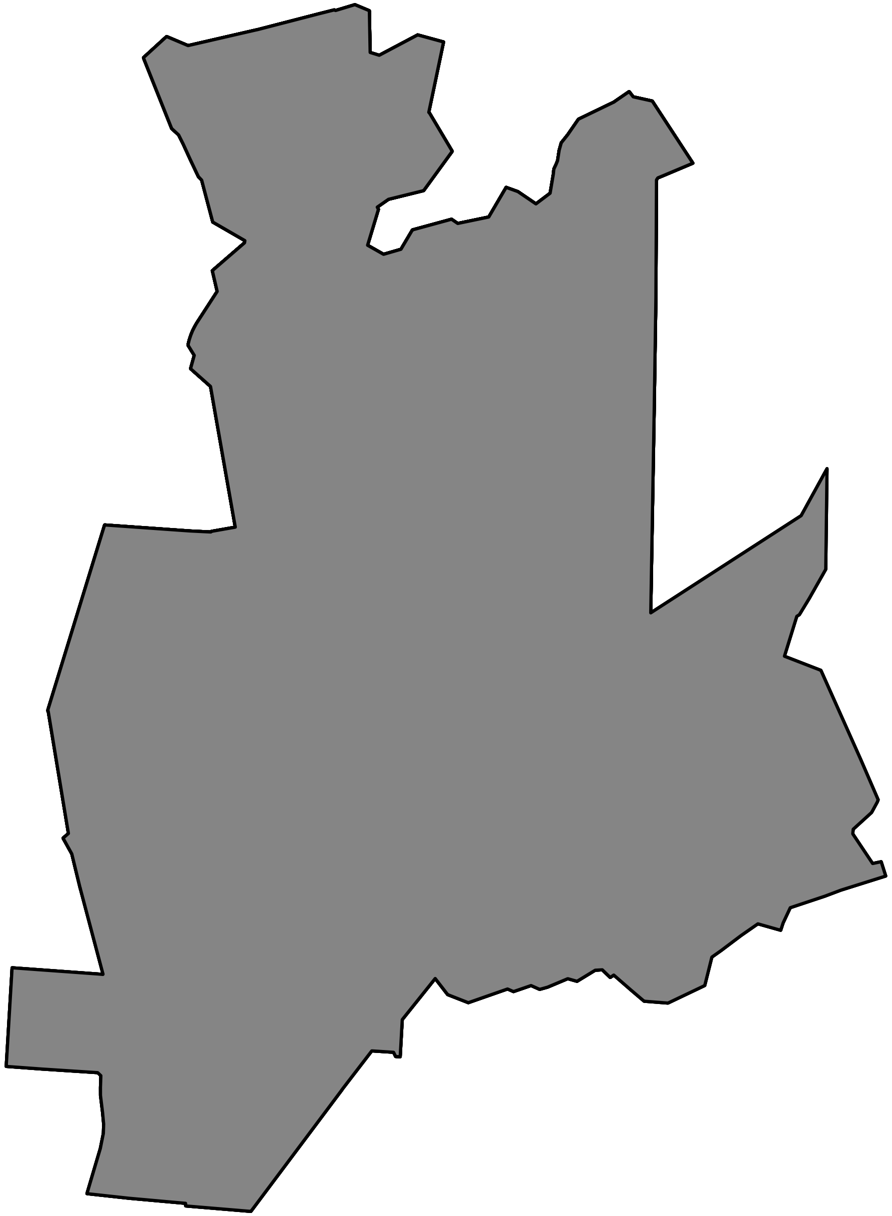


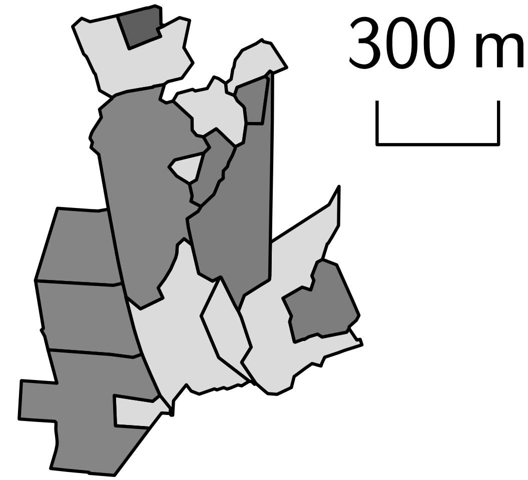
start ($n = 17$)



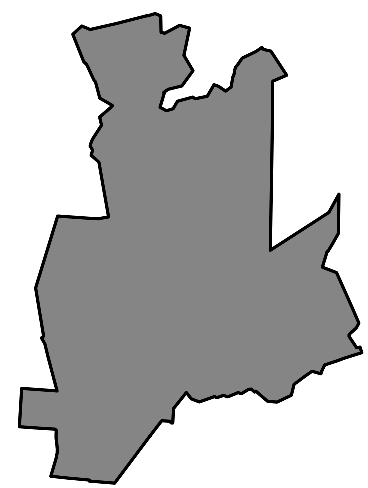
goal

An Optimal Sequence by A*

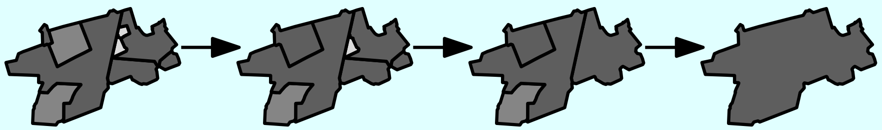
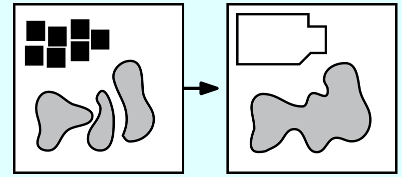
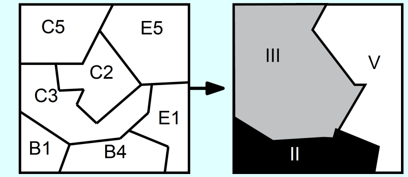
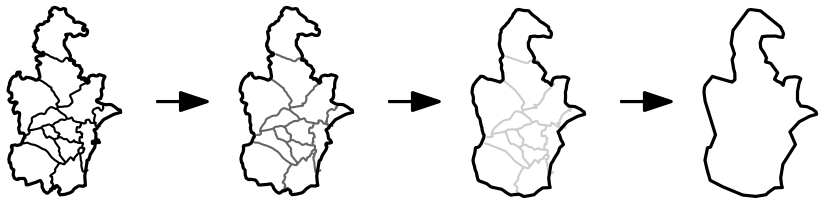
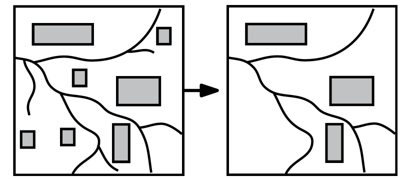
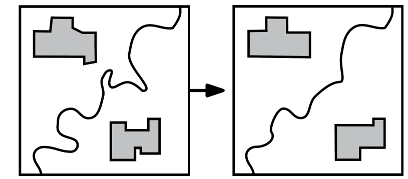
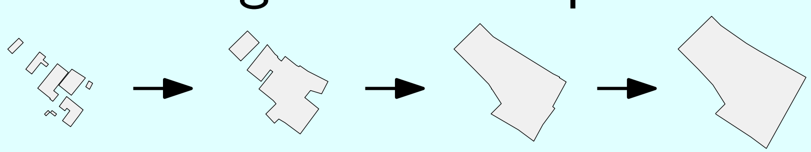
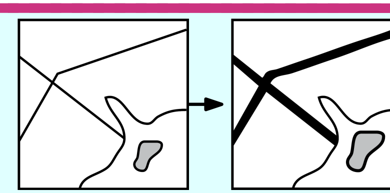
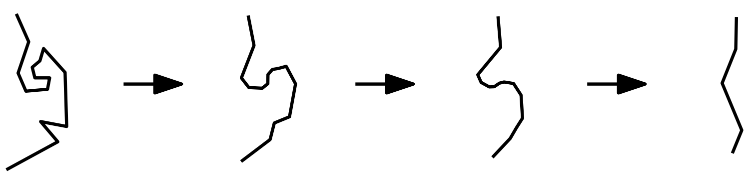
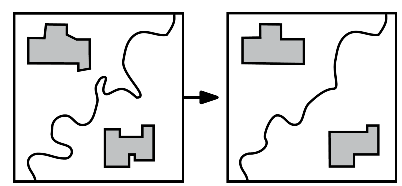
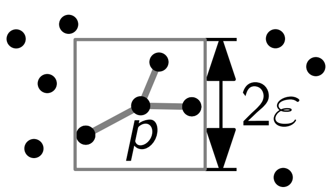




start ($n = 17$)

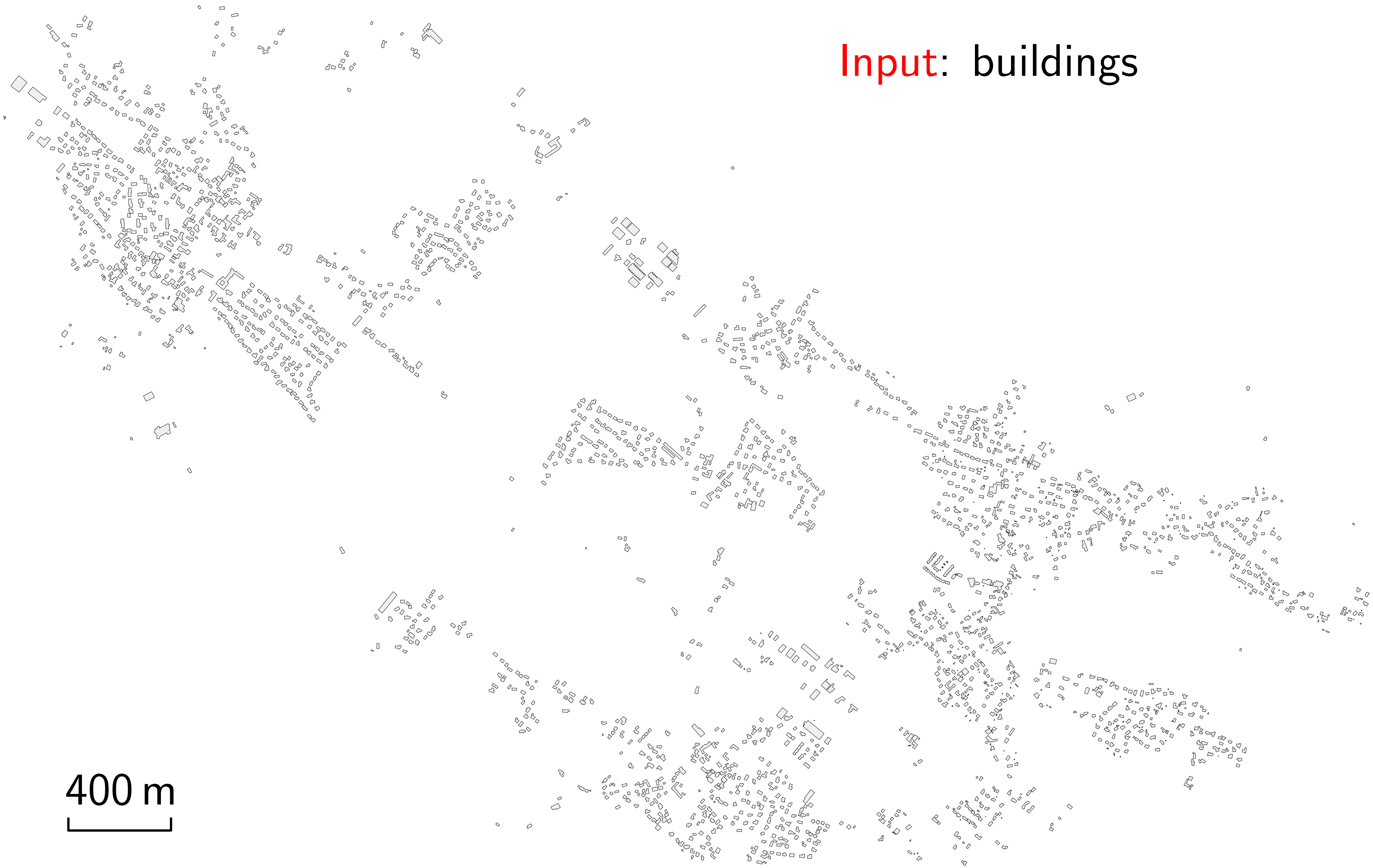


goal

Contents of Thesis	Optim.	Related Generalization
Optimal sequence for aggregation 	$A^\star$ ILP	 Aggregation  Classification
Administrative boundaires 	DP	 Elimination  Simplification
Buildings to built-up areas 	MST	 Exaggeration; Aggregation, Simplification, Elimination
Morphing polylines 	LSA DP	 Simplification
Choosing right data structures  SortedDictionary, SortedSet, ...		

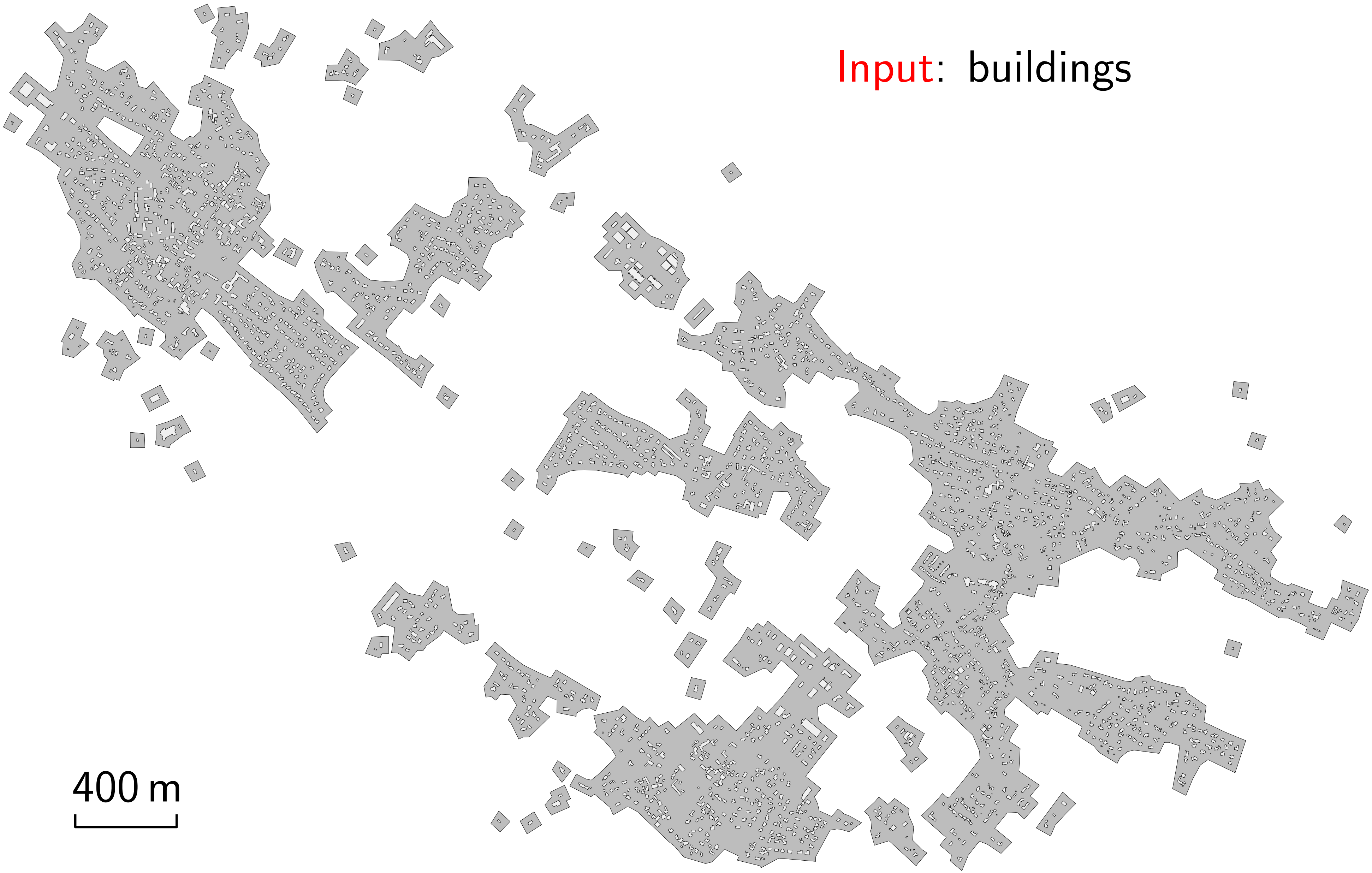
Generalizing Buildings to Built-up Areas

Input: buildings



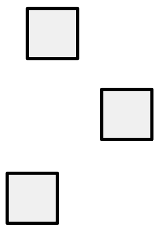
Generalizing Buildings to Built-up Areas

Input: buildings



Aggregate and Grow

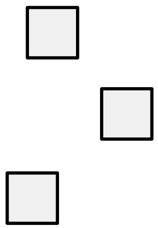
- Aggregate buildings that are too close, when zooming out



original buildings

Aggregate and Grow

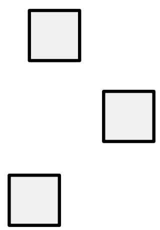
- Aggregate buildings that are too close, when zooming out
- Bridges and buildings constitute
  a minimum spanning tree (MST)



original buildings

Aggregate and Grow

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



original buildings

$t = 0$

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)

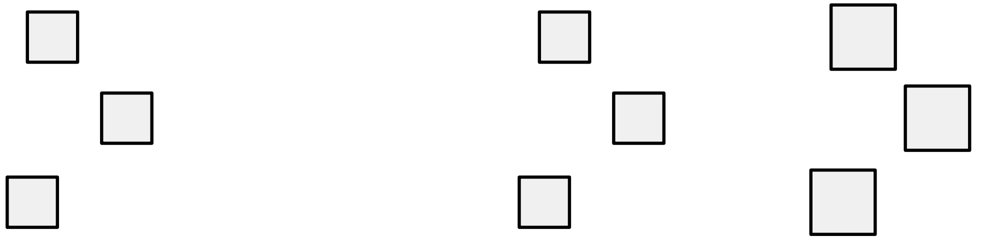
original buildings $t = 0$

add bridge $t = 0.4$

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

$t = 0$

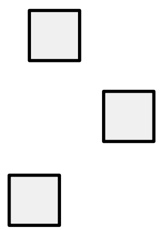
add bridge grow

$t = 0.4$

zoom out:
t increases

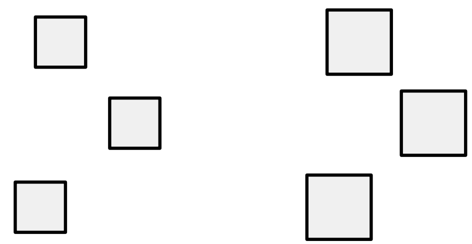
Aggregate and Grow

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



original buildings

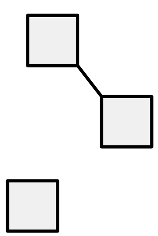
$t = 0$



add bridge   grow

$t = 0.4$

zoom out:
t increases

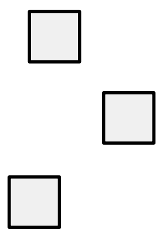


add bridge

$t = 0.6$

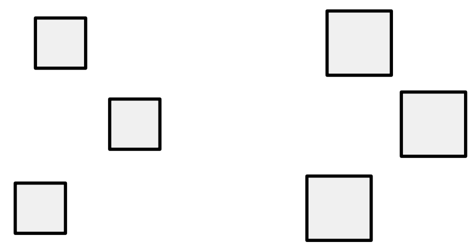
Aggregate and Grow

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



original buildings

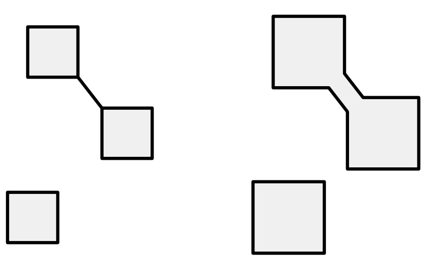
$t = 0$



add bridge grow

$t = 0.4$

zoom out:
t increases

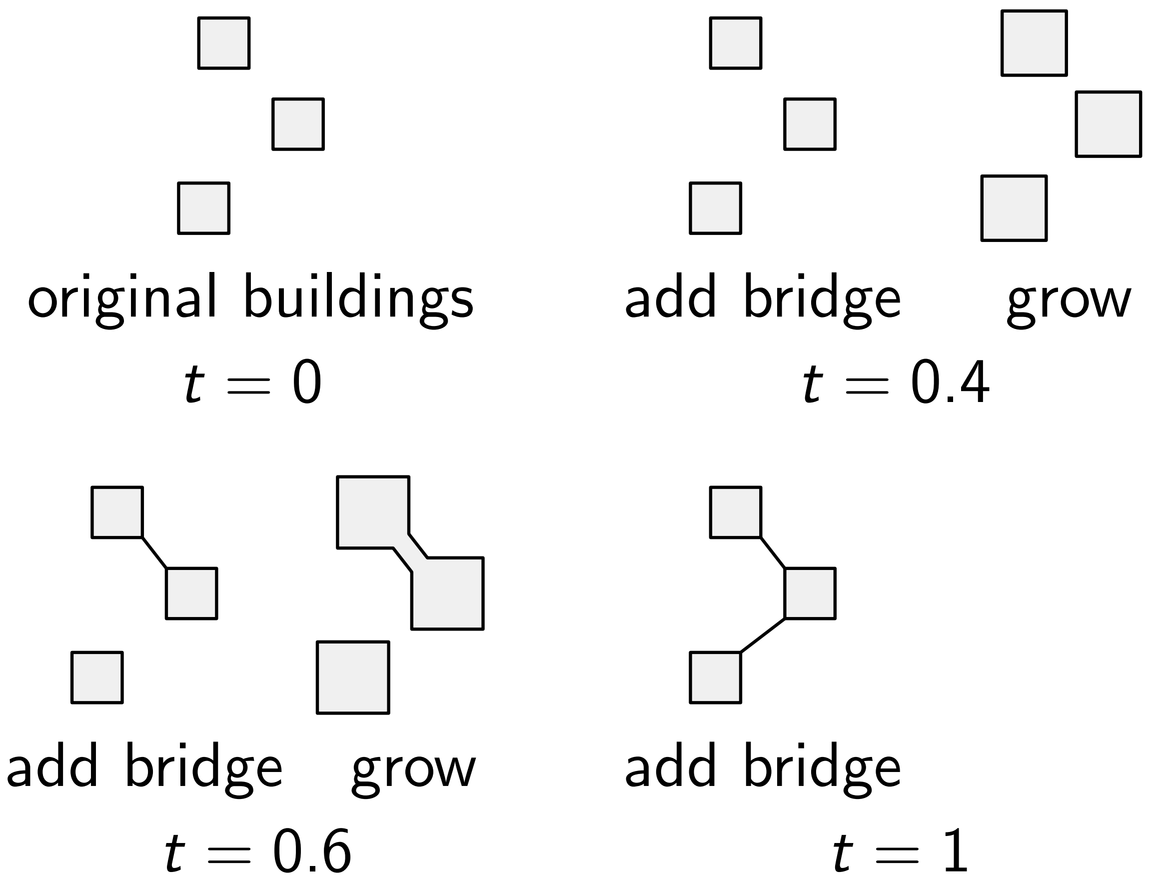


add bridge grow

$t = 0.6$

Aggregate and Grow

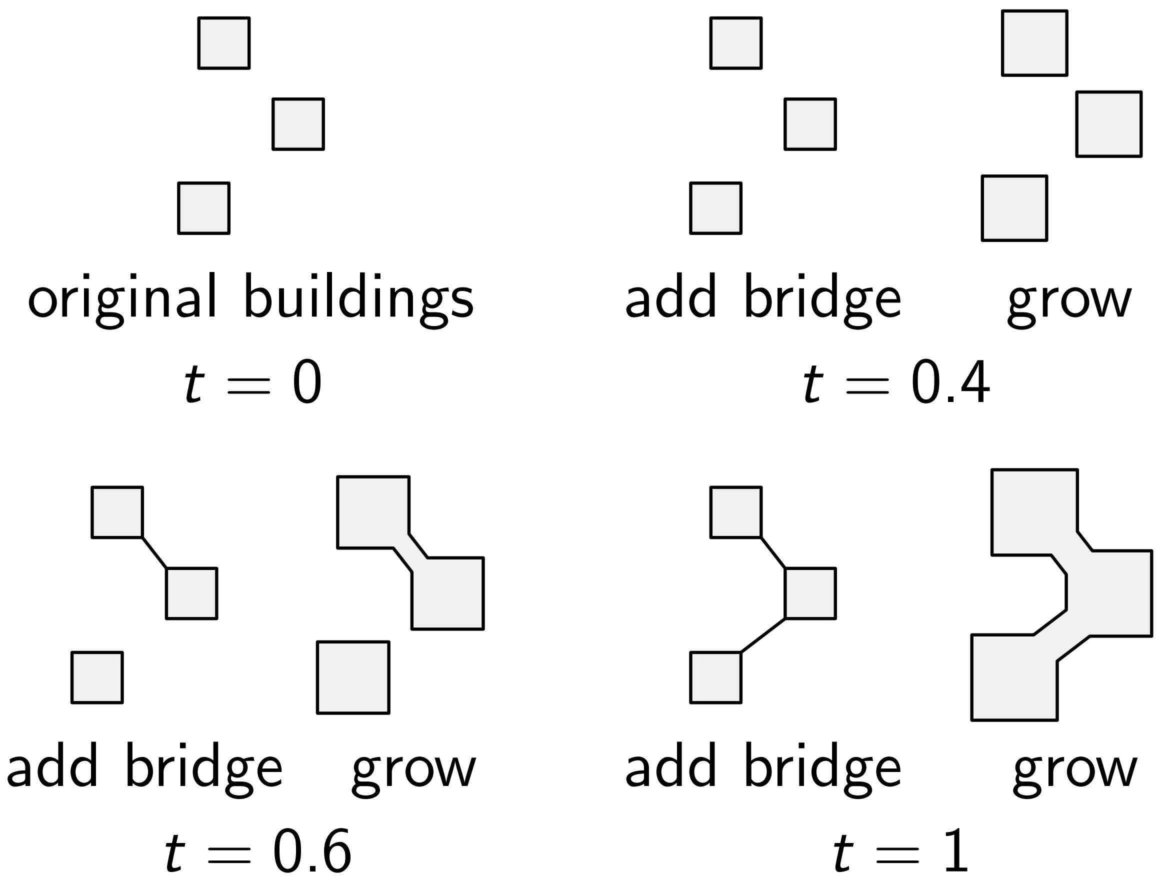
- Aggregate buildings that are too close, when zooming out
- Bridges and buildings constitute
  a minimum spanning tree (MST)



zoom out:
t increases

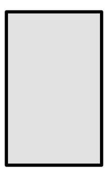
Aggregate and Grow

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



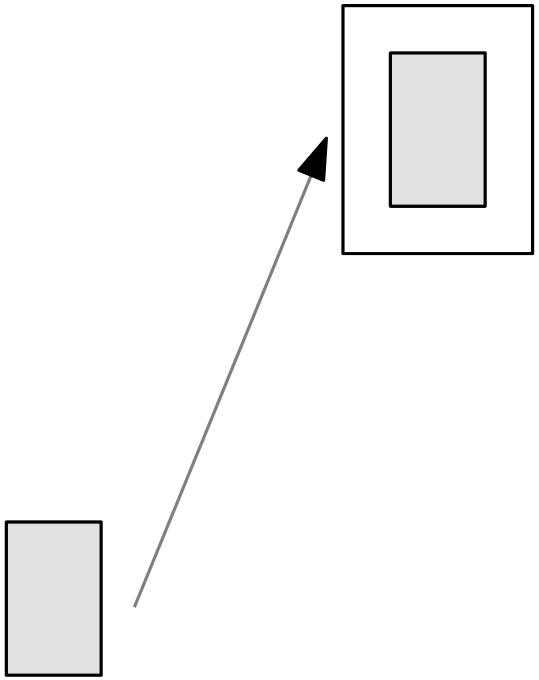
zoom out:
t increases

Three Join Types of Buffering



building

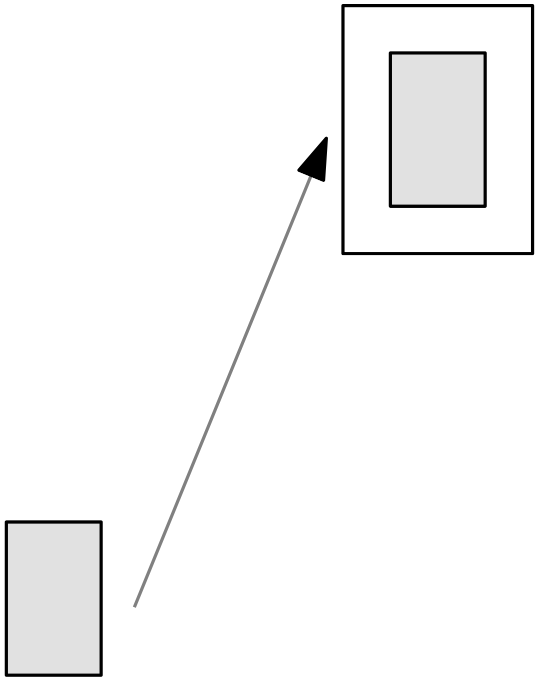
Three Join Types of Buffering



miter:
keep right angles

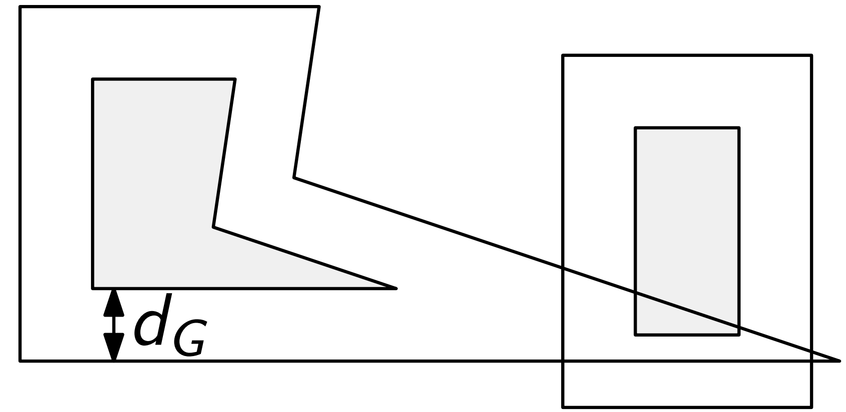
building

Three Join Types of Buffering

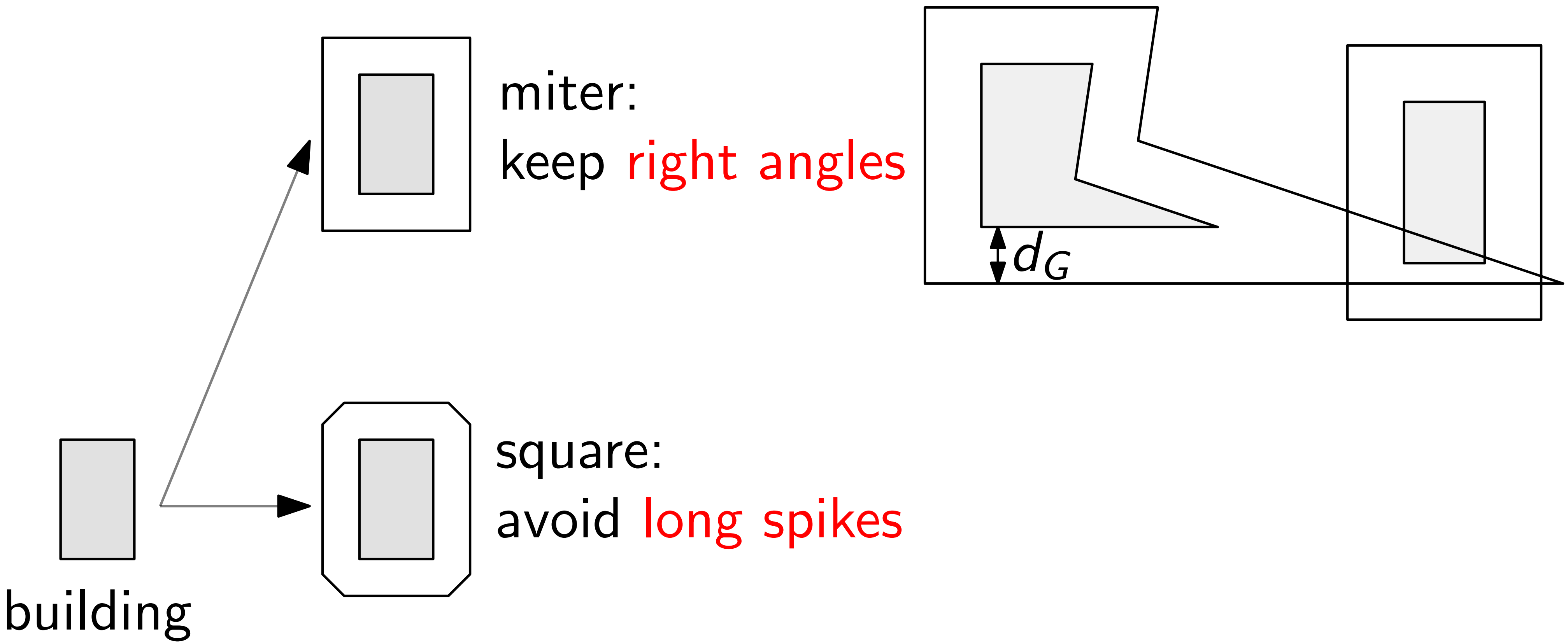


building

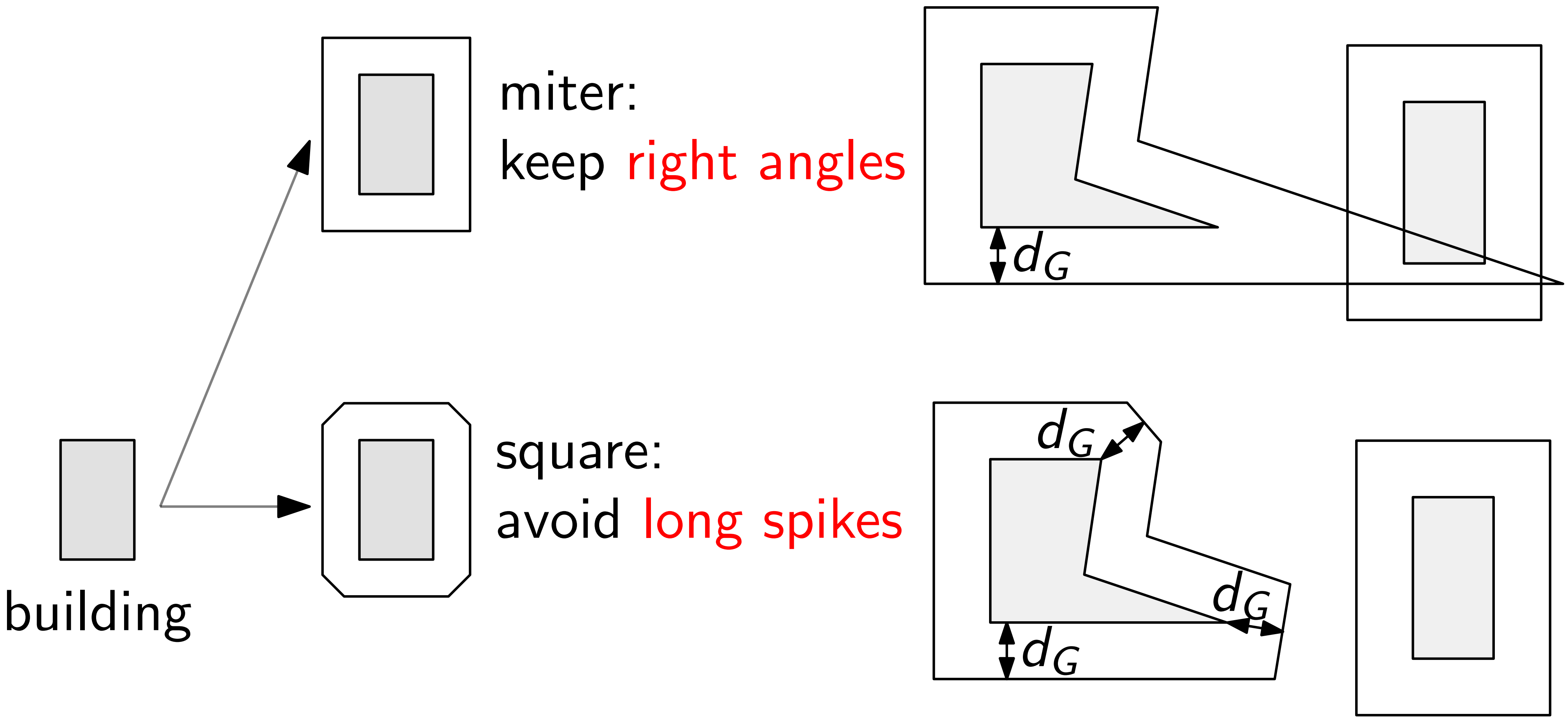
miter:
keep right angles



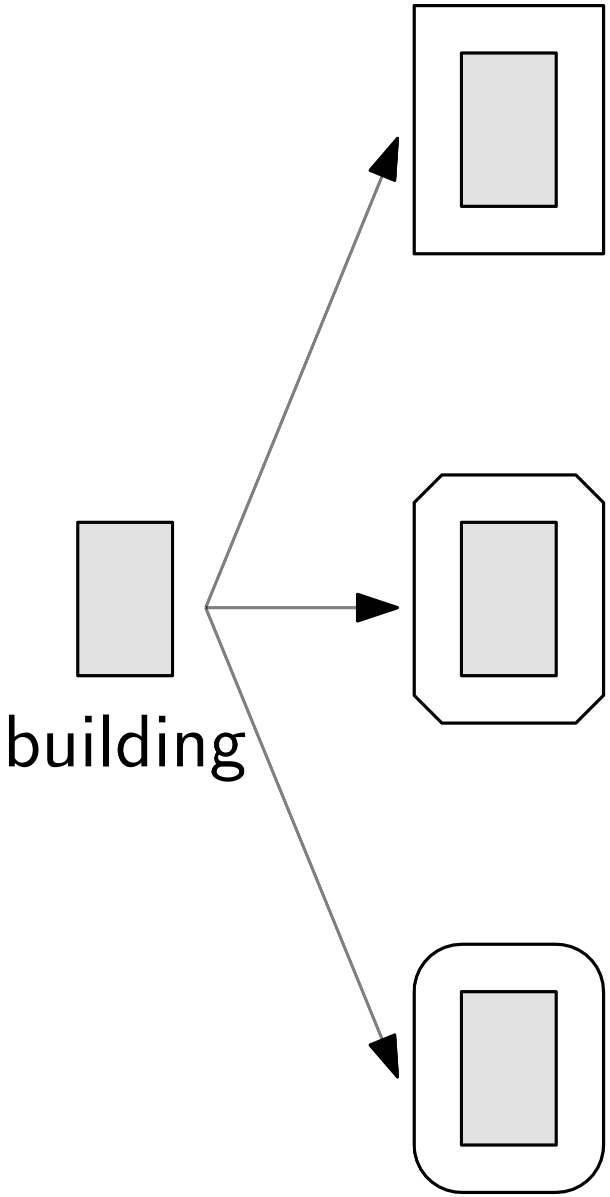
Three Join Types of Buffering



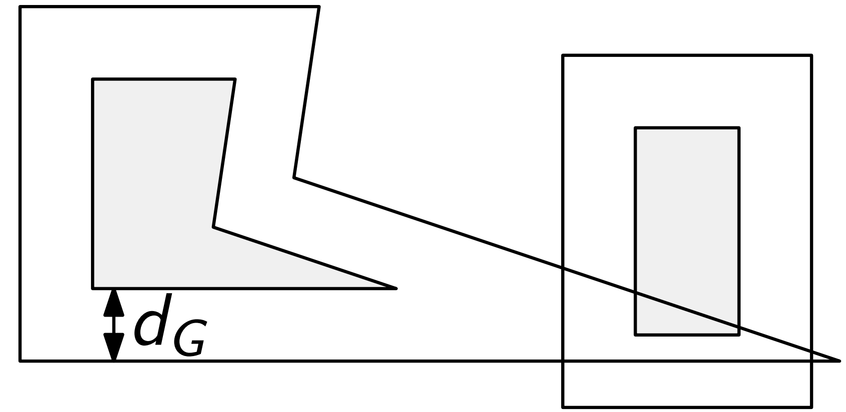
Three Join Types of Buffering



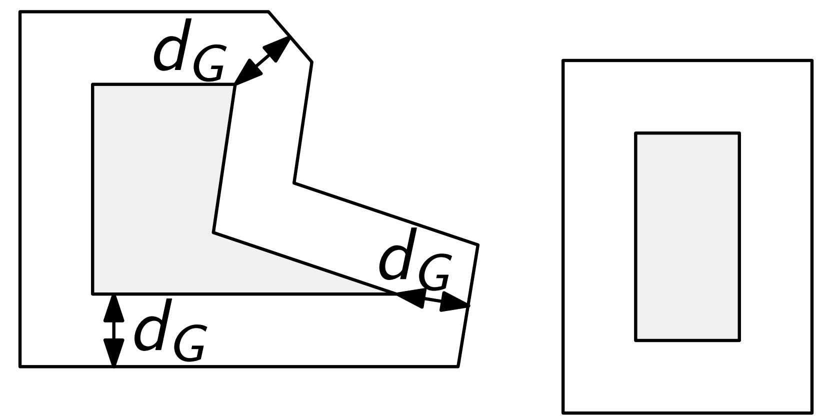
Three Join Types of Buffering



miter:
keep right angles

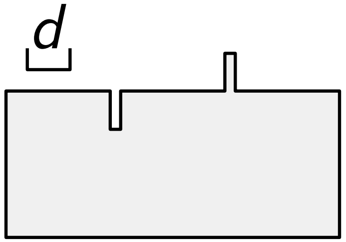


square:
avoid long spikes



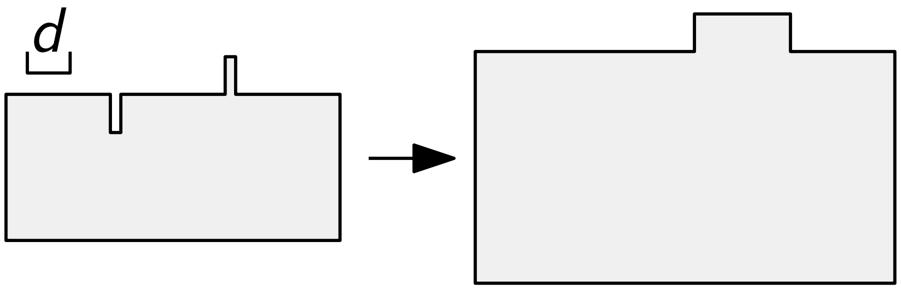
round:
detect if two buildings are too close

Simpifying Based on Dilation and Erosion



polygon

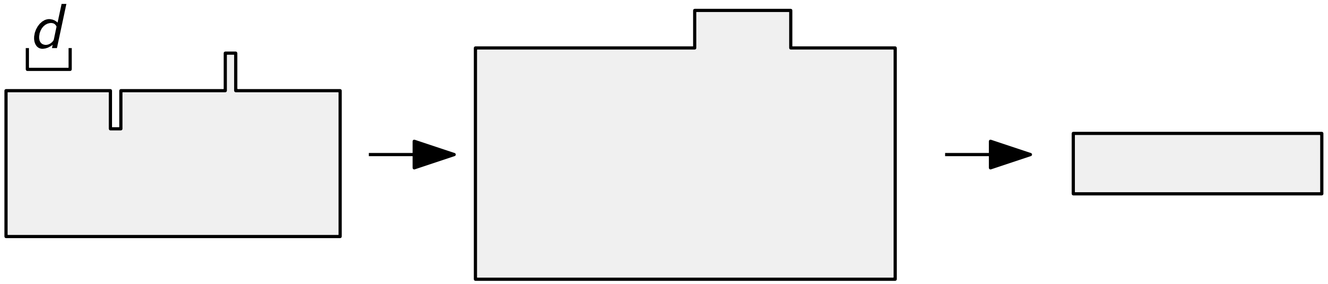
Simpifying Based on Dilation and Erosion



polygon

dilate with d:
remove dents

Simpifying Based on Dilation and Erosion

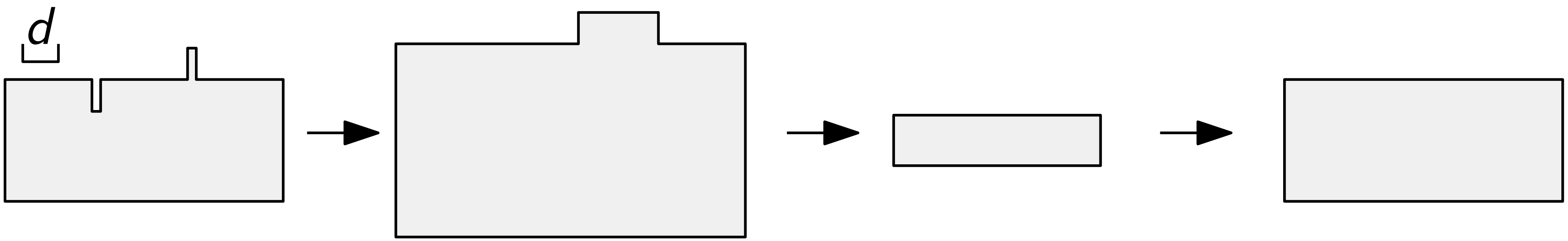


polygon

dilate with d:
remove dents

erode with $2d$:
remove bumps

Simpifying Based on Dilation and Erosion



polygon

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

Animation

Animation

Animation

Animation

Animation

Animation

Animation

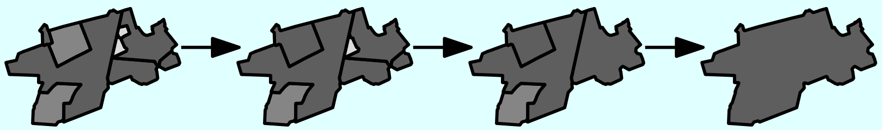
Animation

Animation

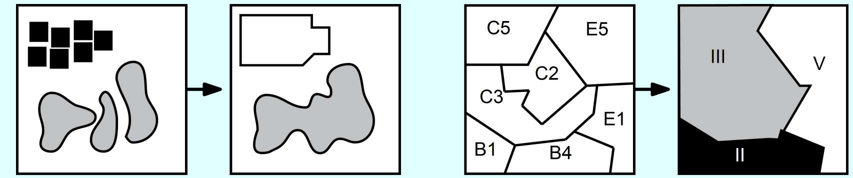
Animation

Contents of Thesis	Optim.	Related Generalization

Optimal sequence for aggregation

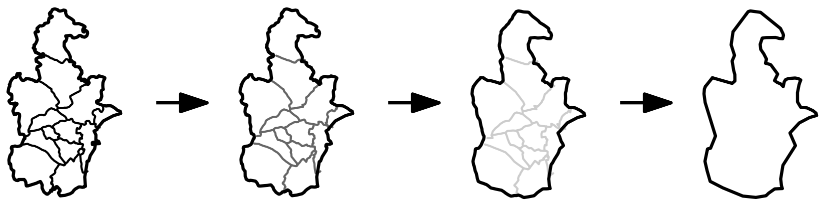


A* ILP

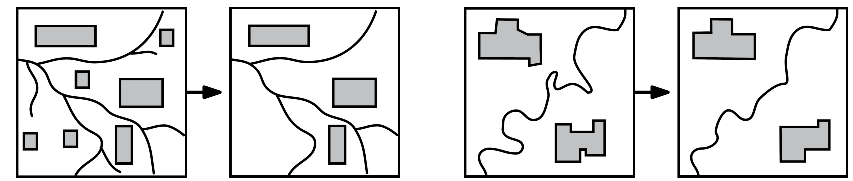


Aggregation Classification

Administrative boundaires

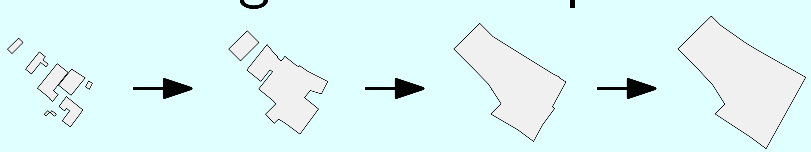


DP

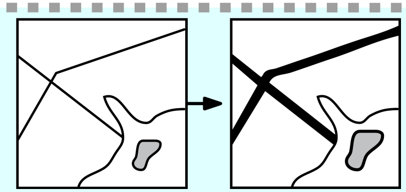


Elimination Simplification

Buildings to built-up areas



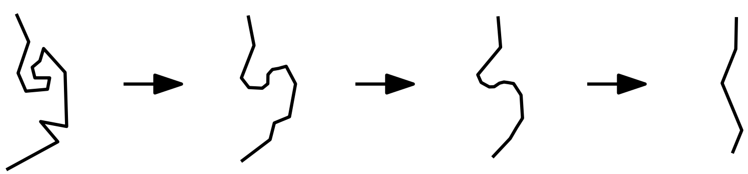
MST



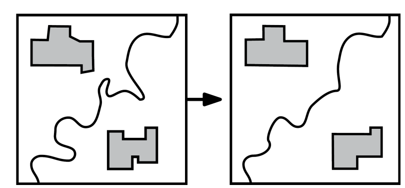
Exaggeration

Aggregation,
Simplification,
Elimination

Morphing polylines

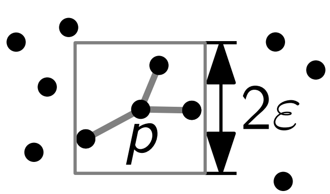


LSA
DP



Simplification

Choosing right data structures



SortedDictionary,
SortedSet, . . .

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

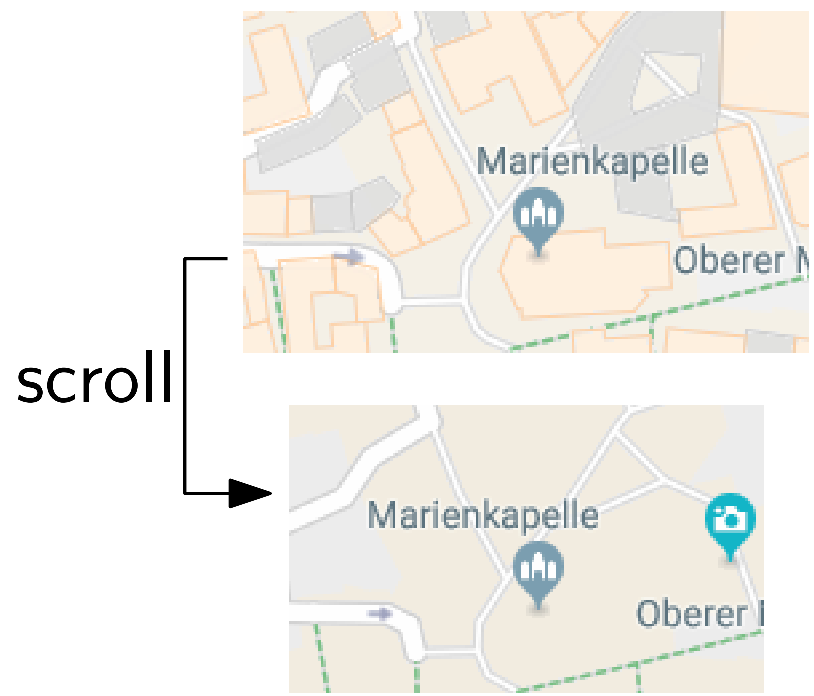
- Improve our methods
- Usability testing

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, ...)



Conclusion

Thank you!

- 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, ...)