Generalisation via tGAP structure

MobiMaps project (RGI-233)

Guest lecture at LBS/Geomatics

Arta Dilo

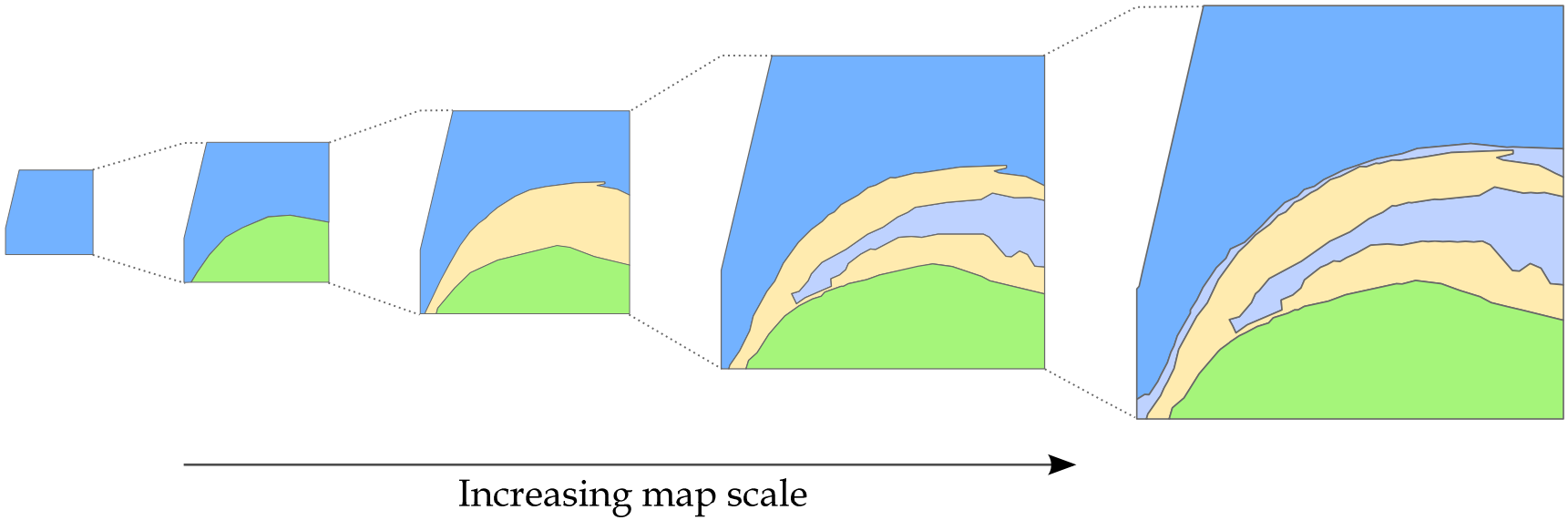
OTB Research Institute for Housing, Urban and Mobility Studies

TU Delft

Content

- Few words on generalisation
- Theory of tGAP (topological Generalised Area Partitioning)
  - How tGAP is filled (off-line)
  - How tGAP is used for (online) feature selection
- Implementation of tGAP in Oracle Spatial
  - New data types
  - Tables storing the structure
- Further research

Why generalisation?

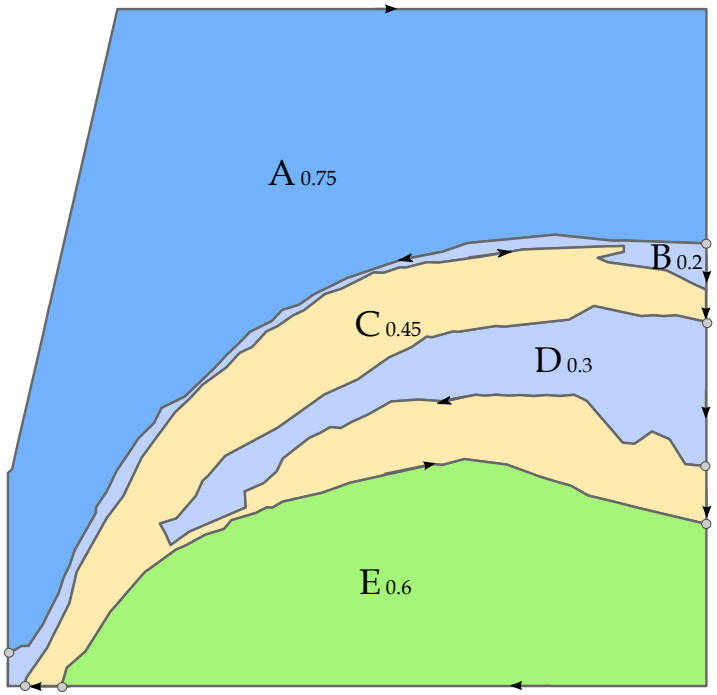


Generalisation methods & operators

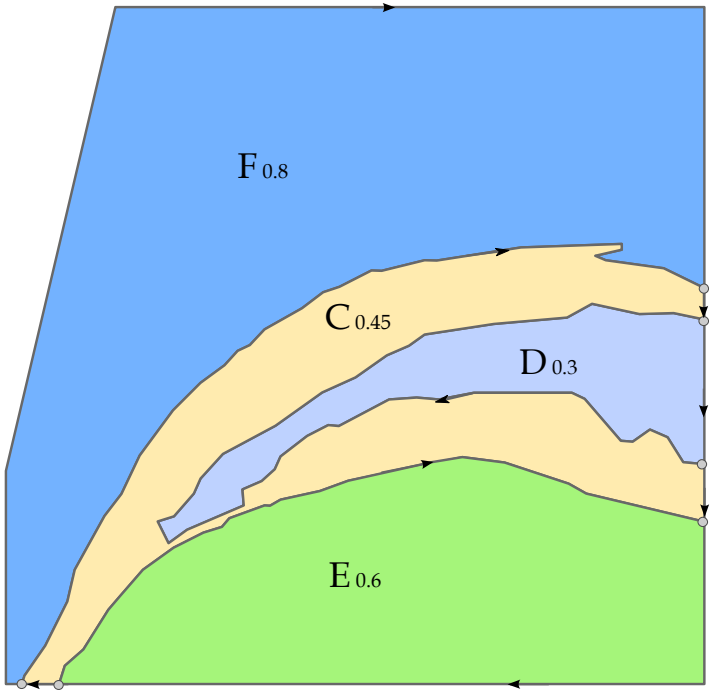
- Generalisation methods:
  - Multiple representation databases
  - On-the-fly generalisation
  - tGAP structure
    - Off-line generalisation
    - On-the-fly feature selection
- Generalisation operators:

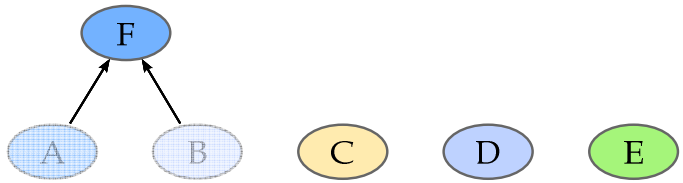
  reclassification, elimination, collapse, simplification, …

Offline generalisation of tGAP

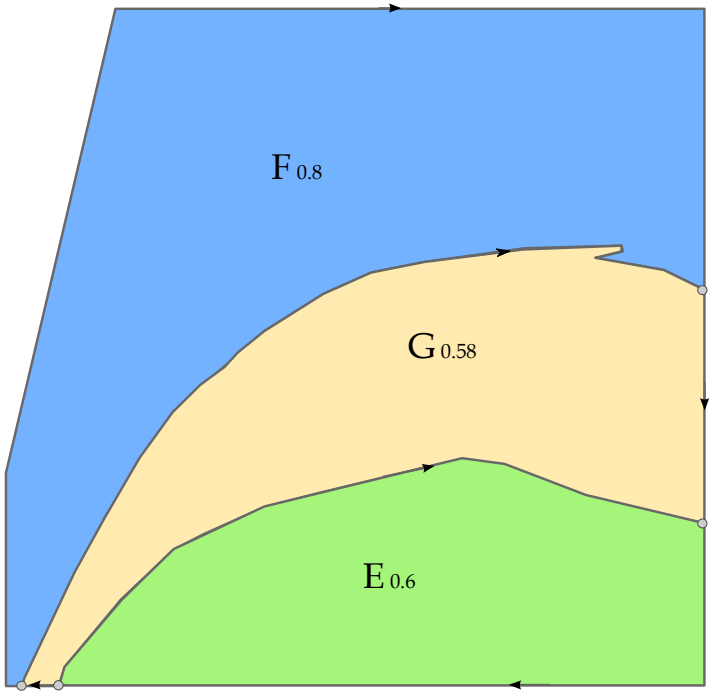


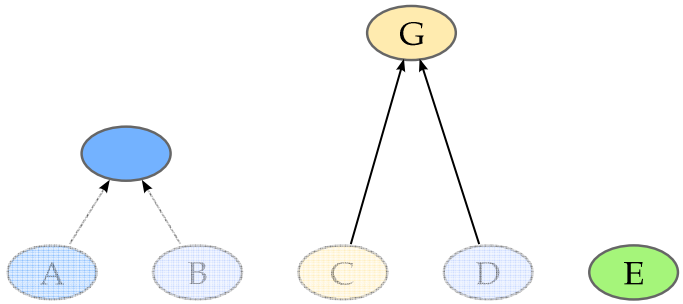
Offline generalisation of tGAP



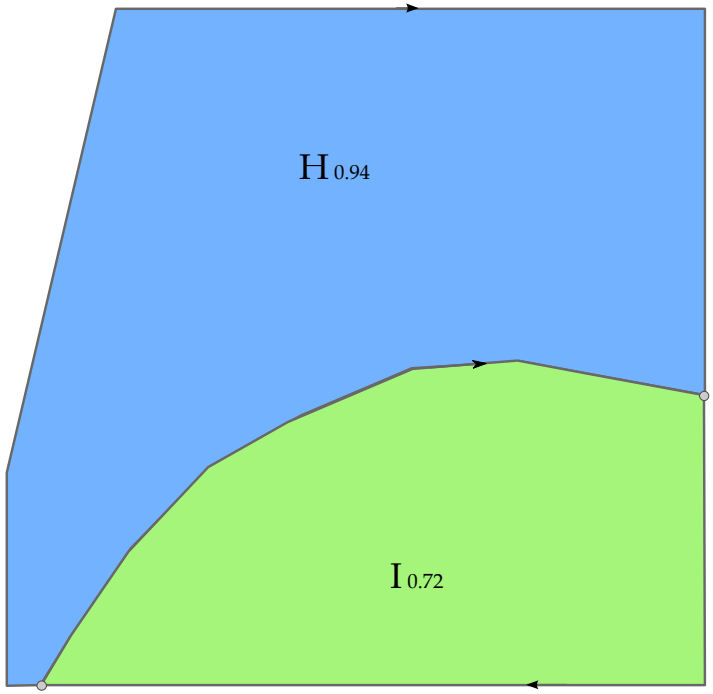


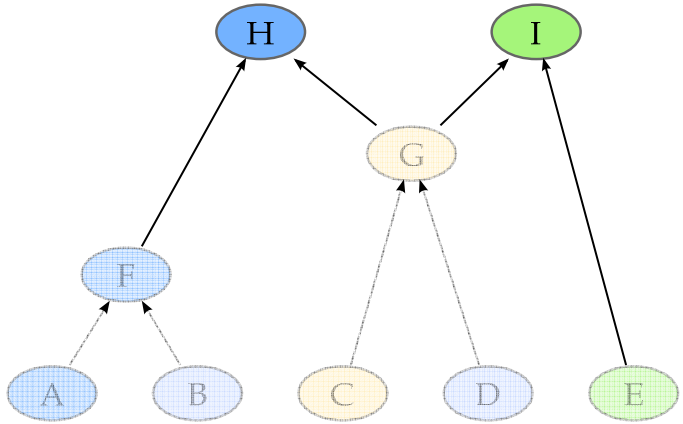
Offline generalisation of tGAP



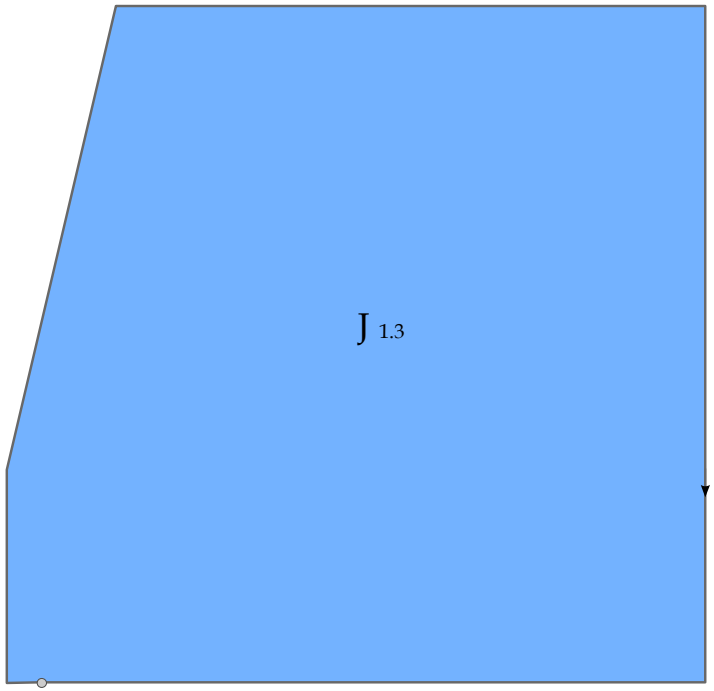


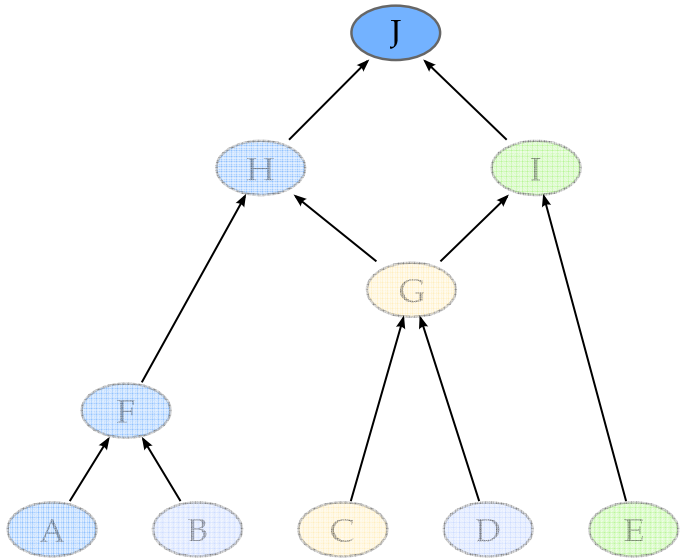
Offline generalisation of tGAP





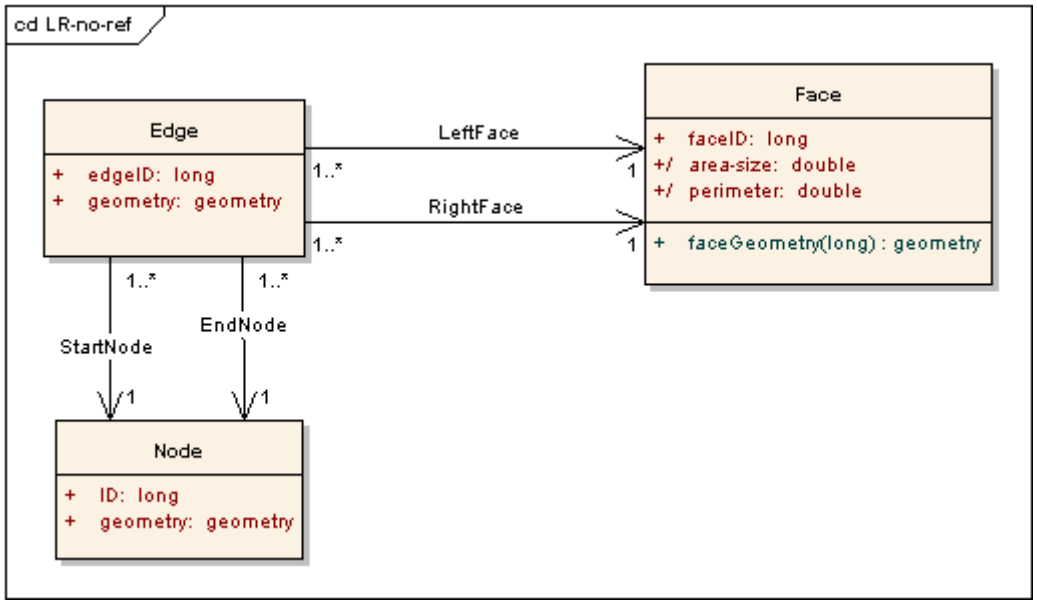
Offline generalisation of tGAP





Topological model for data storage

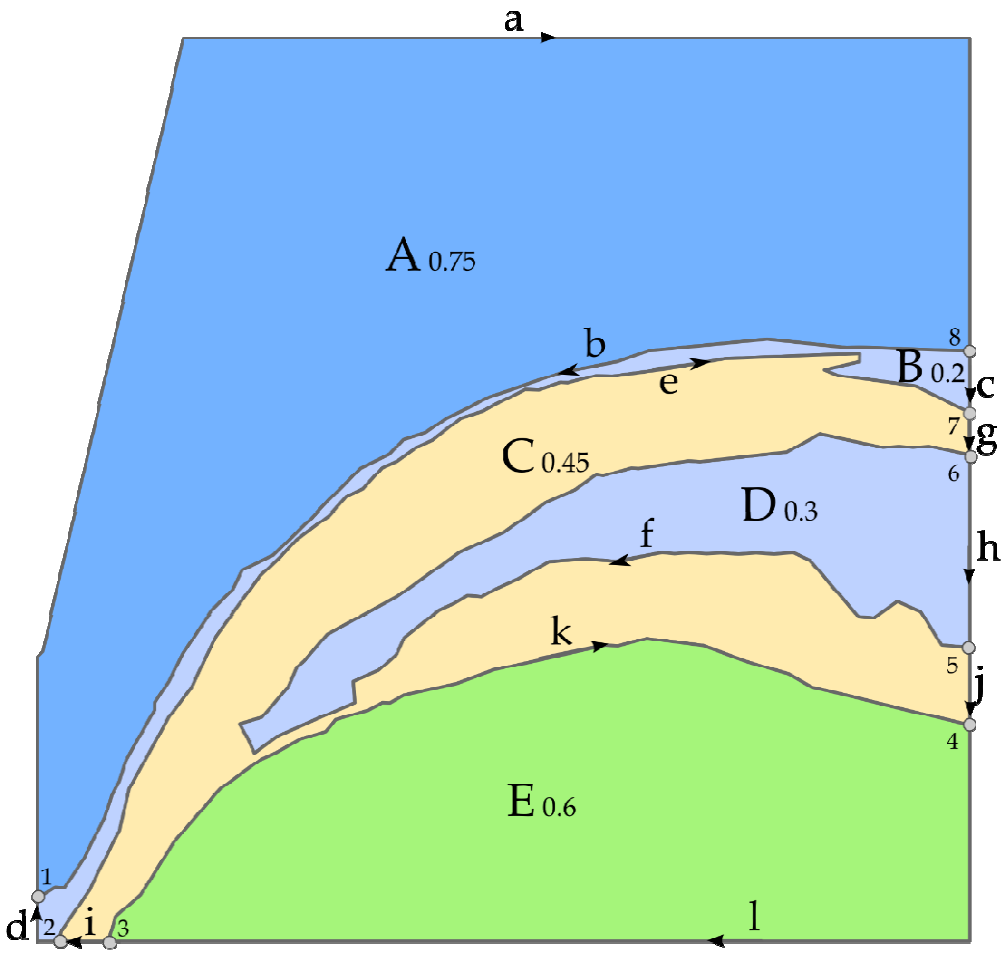
- Left-Right topology without edge references



tGAP structure composition

- tGAP structure is composed of
  - Hierarchy of faces: DAG (Directed Acyclic Graph)
  - Hierarchy of edges: edge forest
  - BLG (Binary Line Generalisation) trees for edge simplification; one BLG for each edge

Building DAG hierarchy of faces: step 0

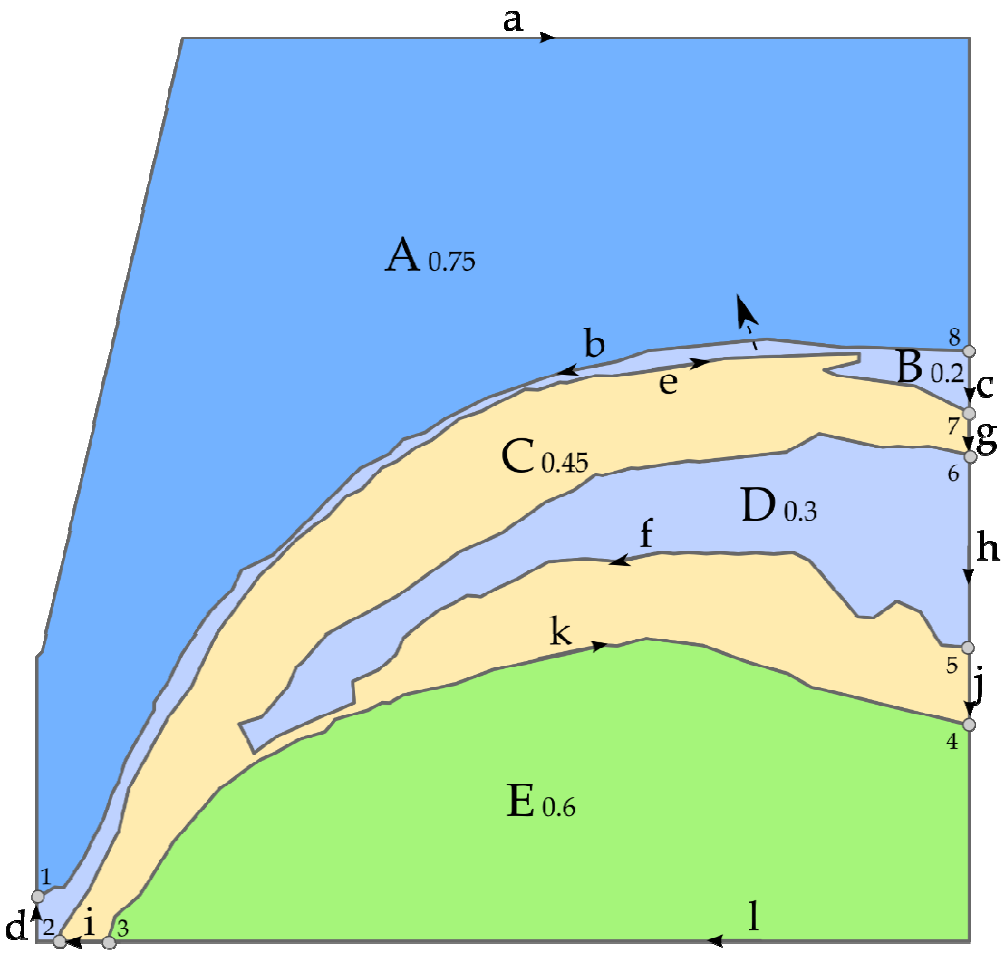


Calculate importance values:

Importance(F) = Area(F)*Class-Weight(F)

A	B	C	D	E
0.75	0.2	0.45	0.3	0.6

Building DAG hierarchy: step 1

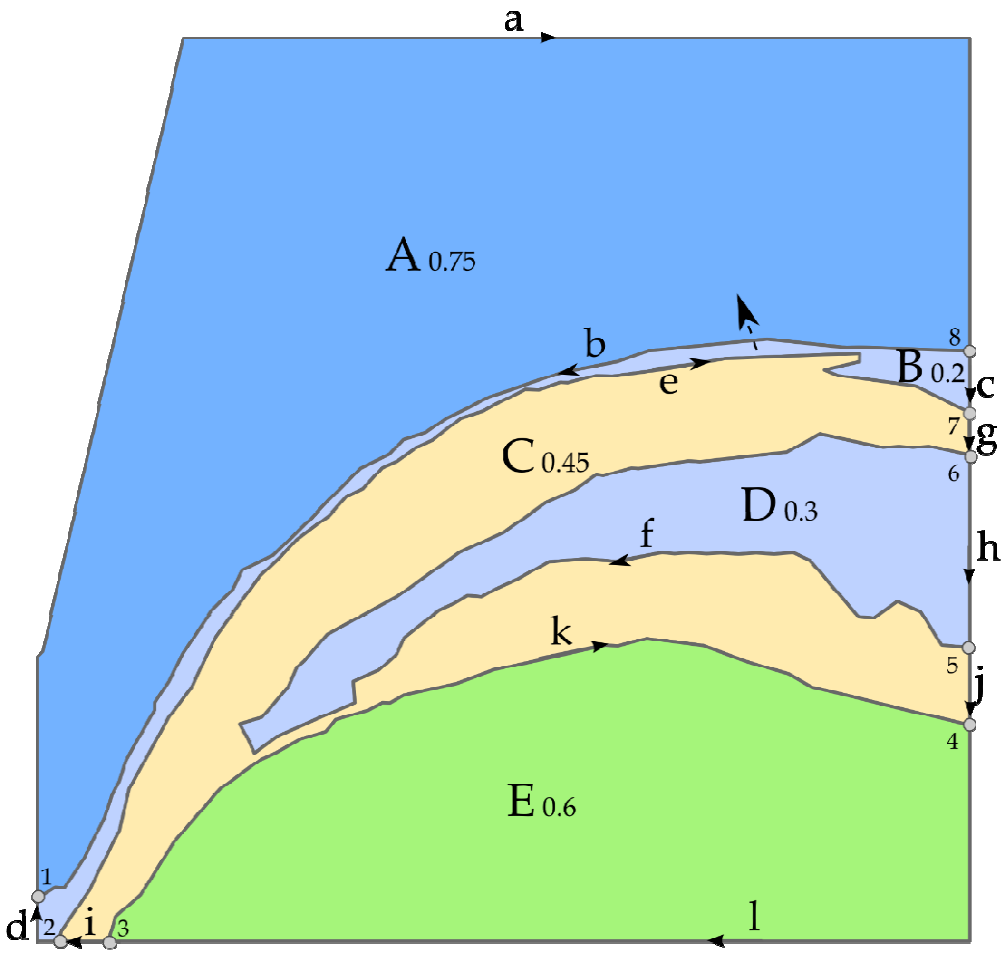


Find least important face;
Find its most compatible neighbour:

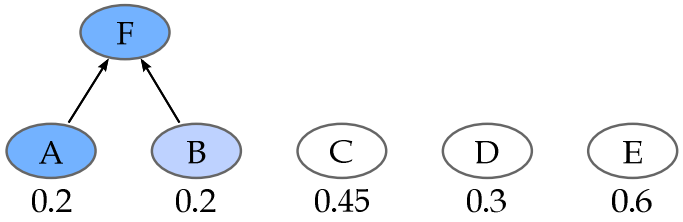
Comp(F,G) = Len(Bnd(F,G)) * Class-Similarity(F,G)

A	B	C	D	E
0.75	**0.2**	0.45	0.3	0.6

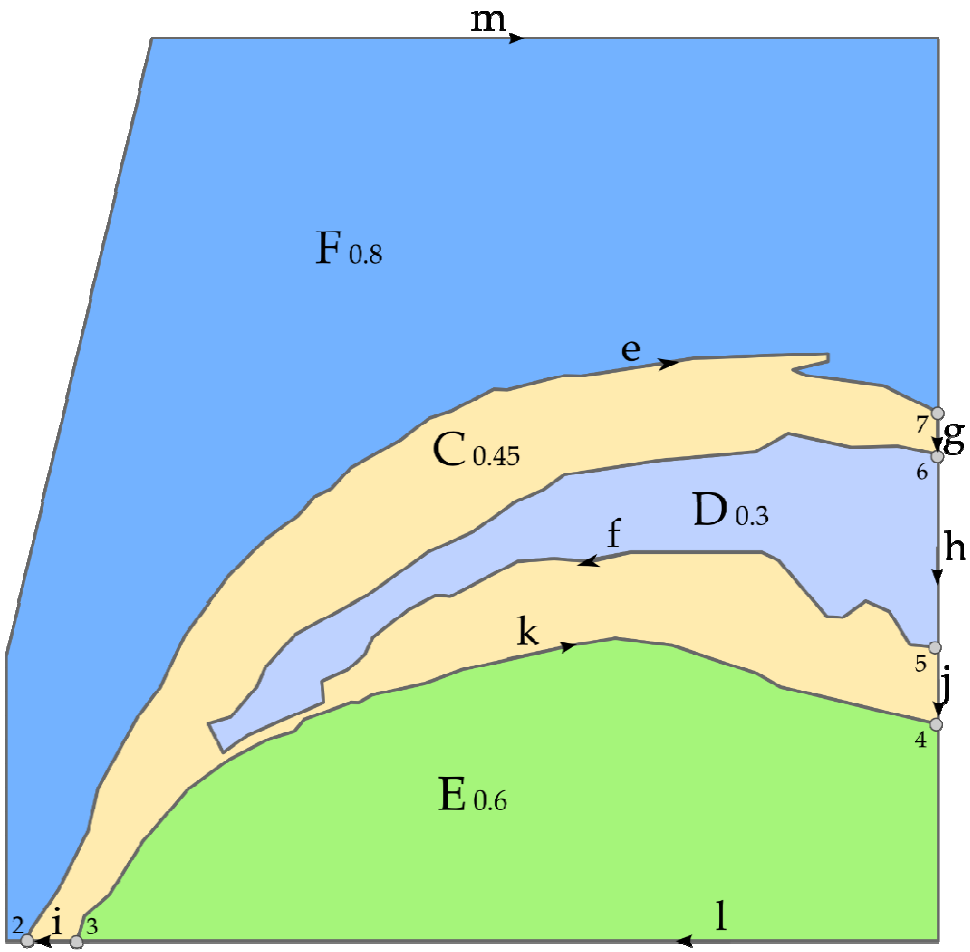
Building DAG hierarchy: step 1



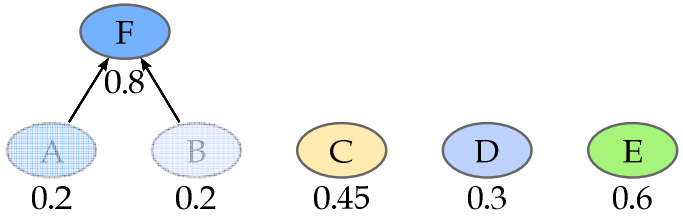
Merge faces



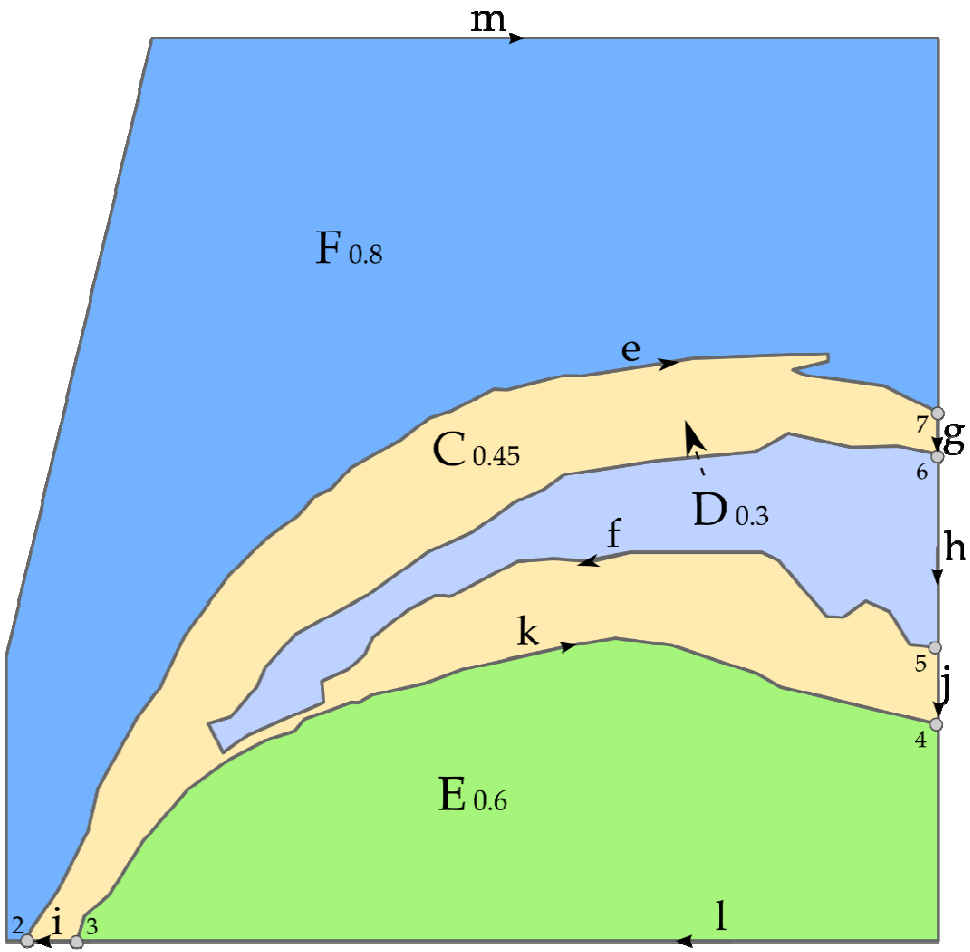
Building DAG hierarchy: step 1



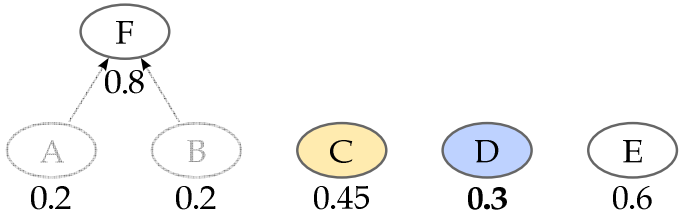
Calculate importance of the new face



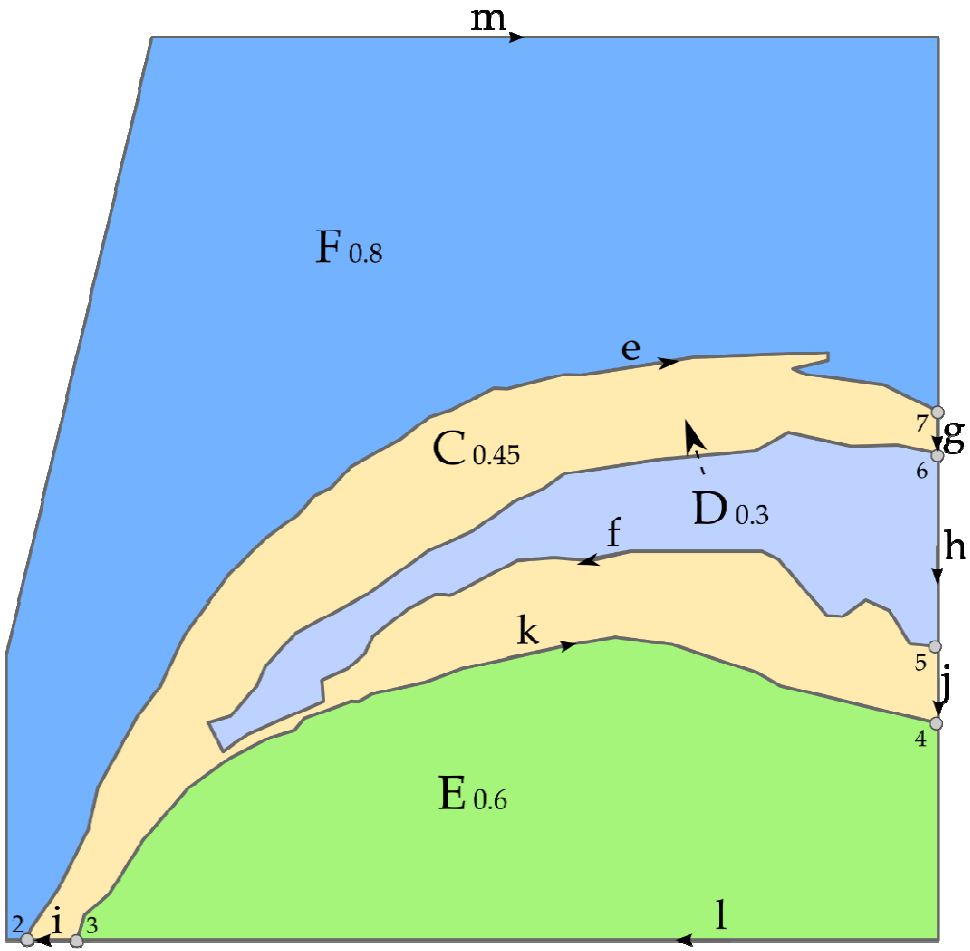
Building DAG hierarchy: step 2



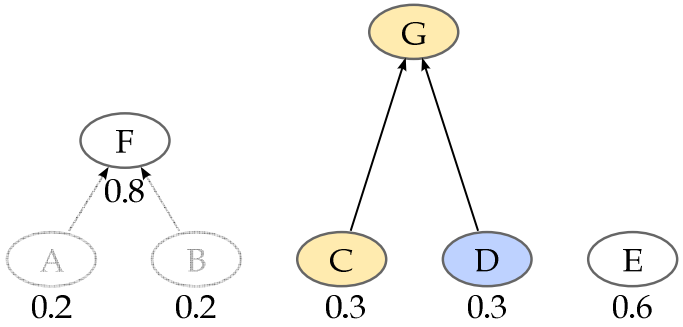
Find least important face;
Find its most compatible neighbour



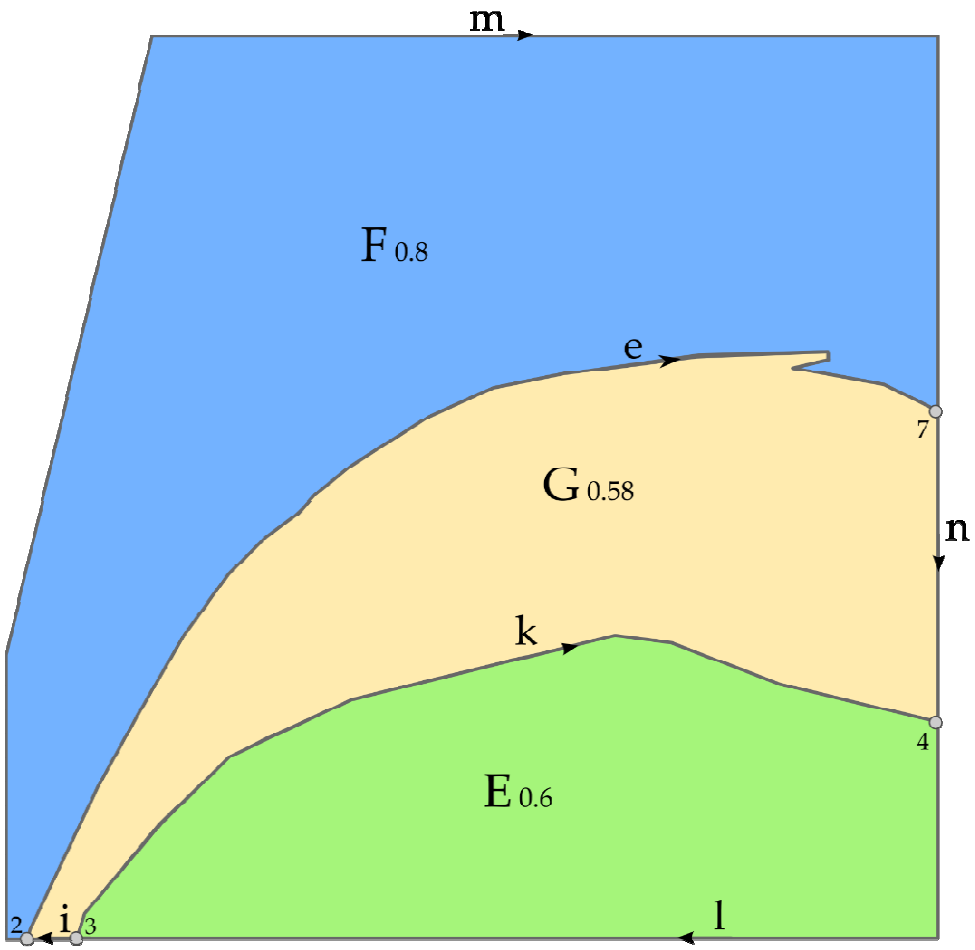
Building DAG hierarchy: step 2



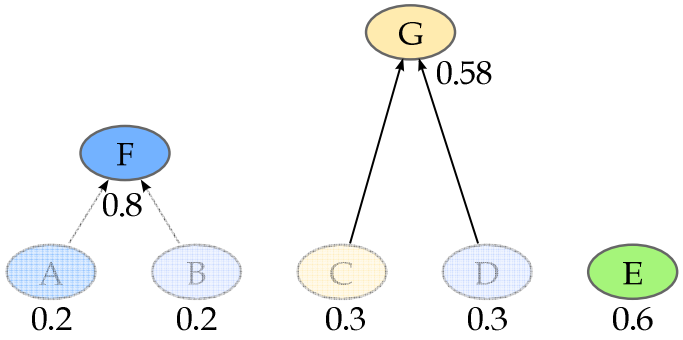
Merge faces



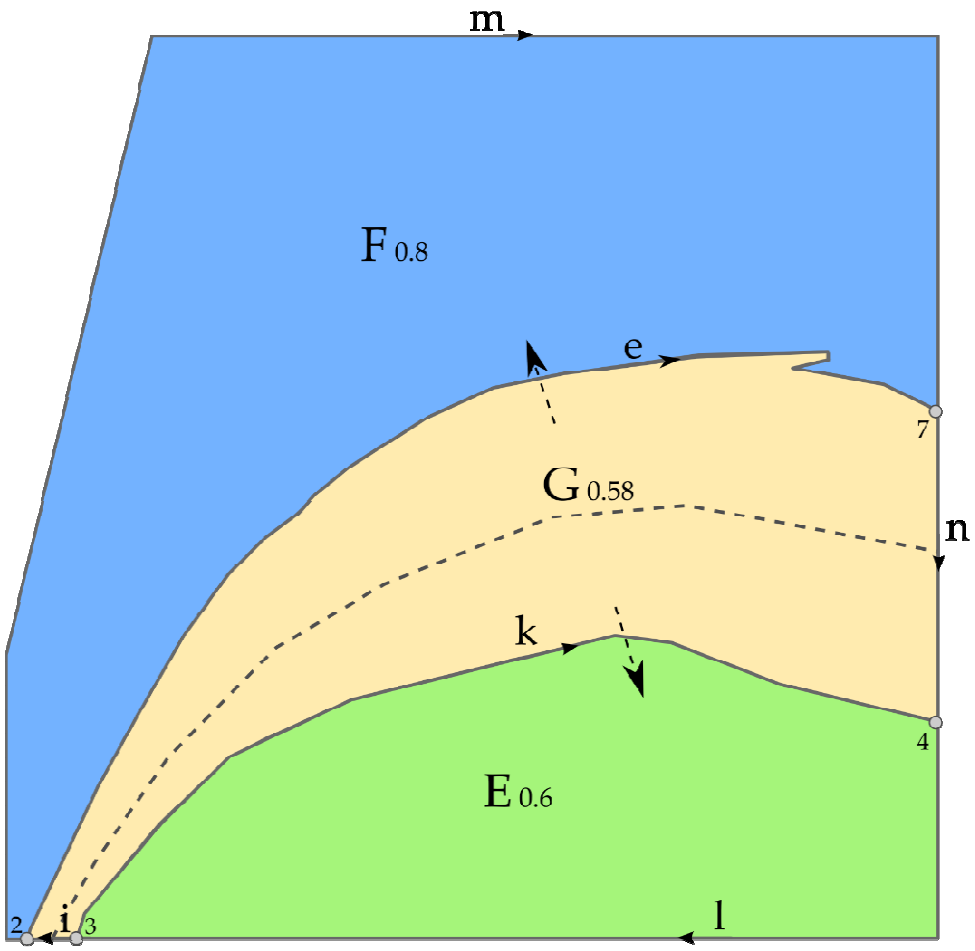
Building DAG hierarchy: step 2



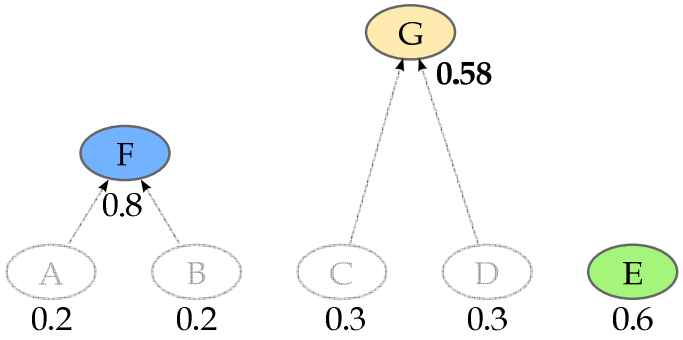
Calculate importance of the new face



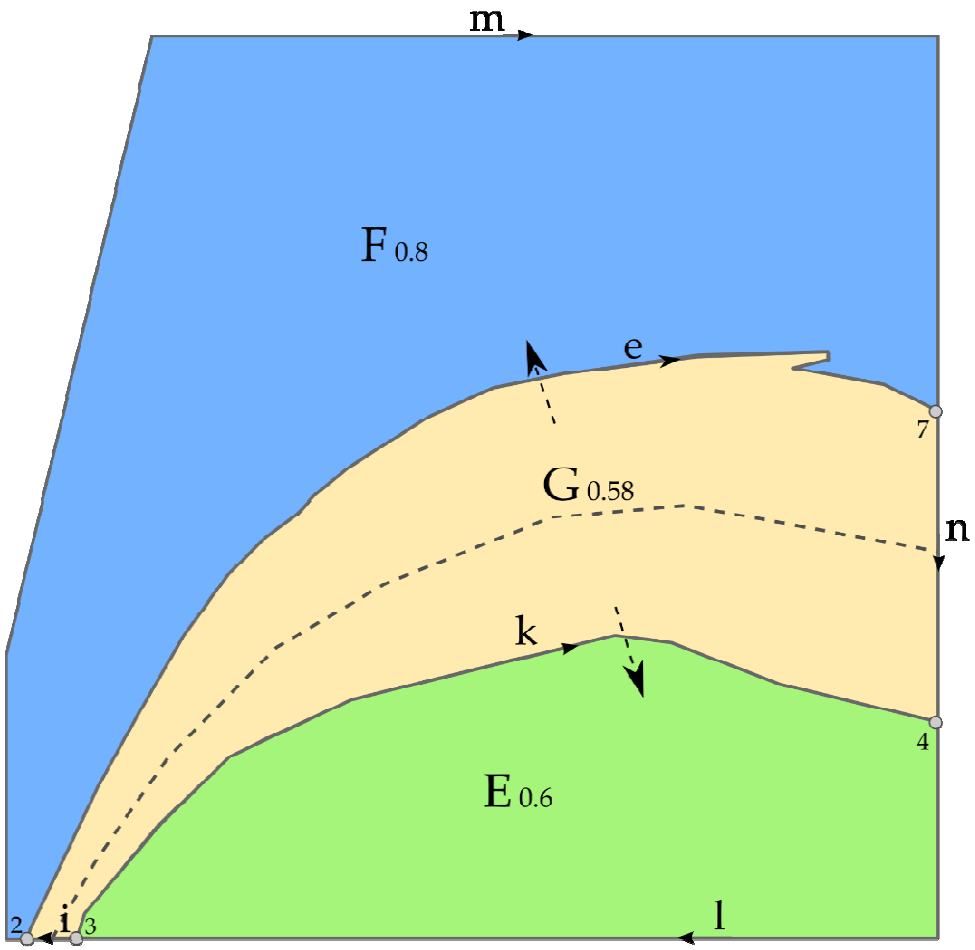
Building DAG hierarchy: step 3



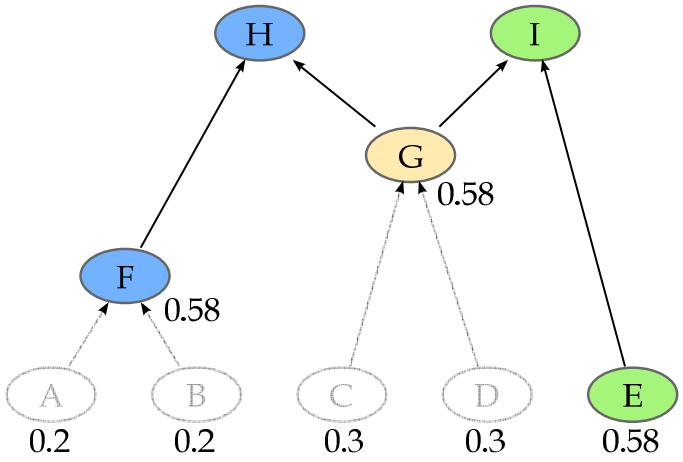
Find least important face;
Find most compatible neighbours



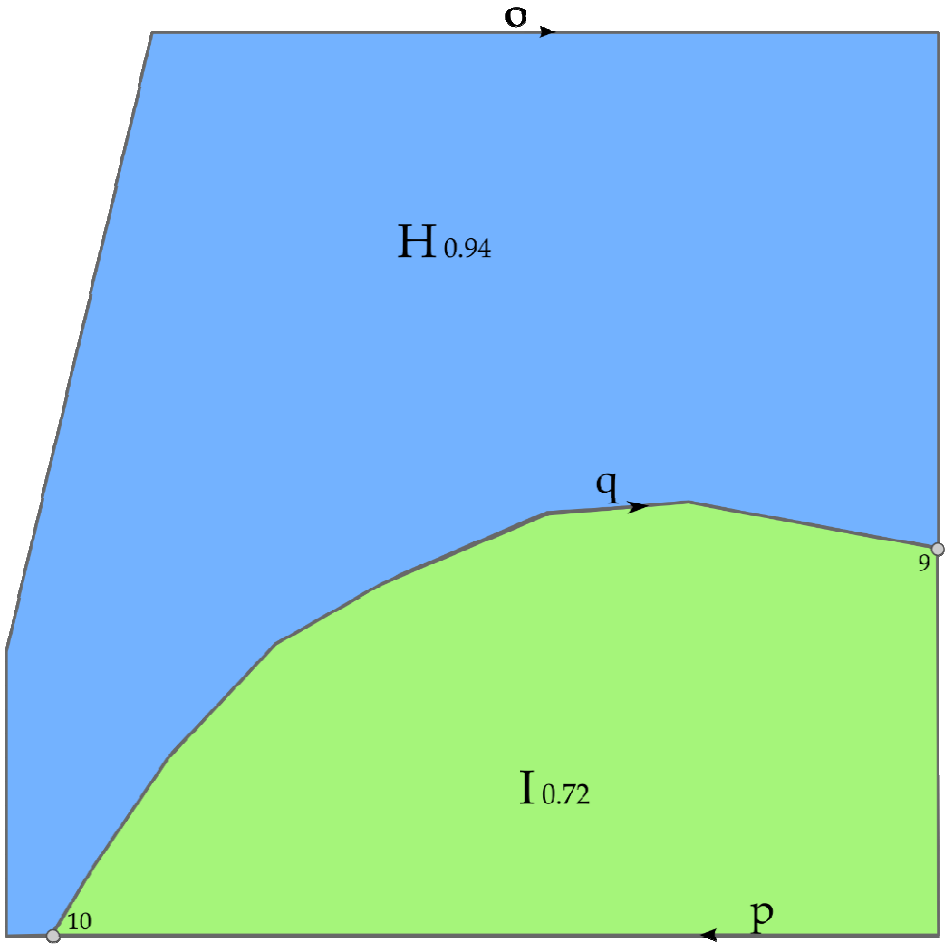
Building DAG hierarchy: step 3



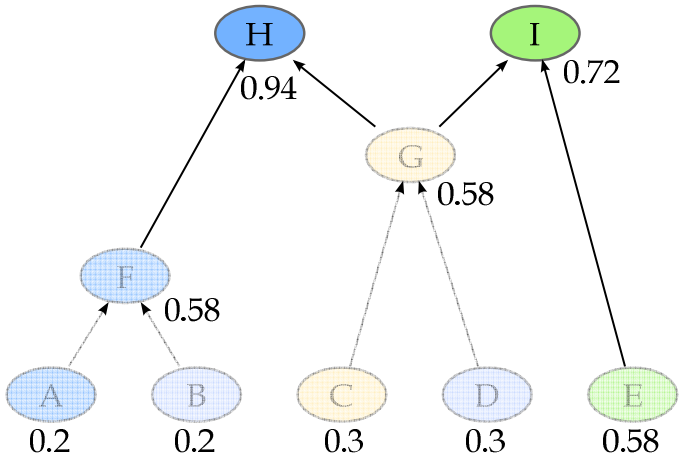
Split to an axis;
Merge parts to neighbour faces



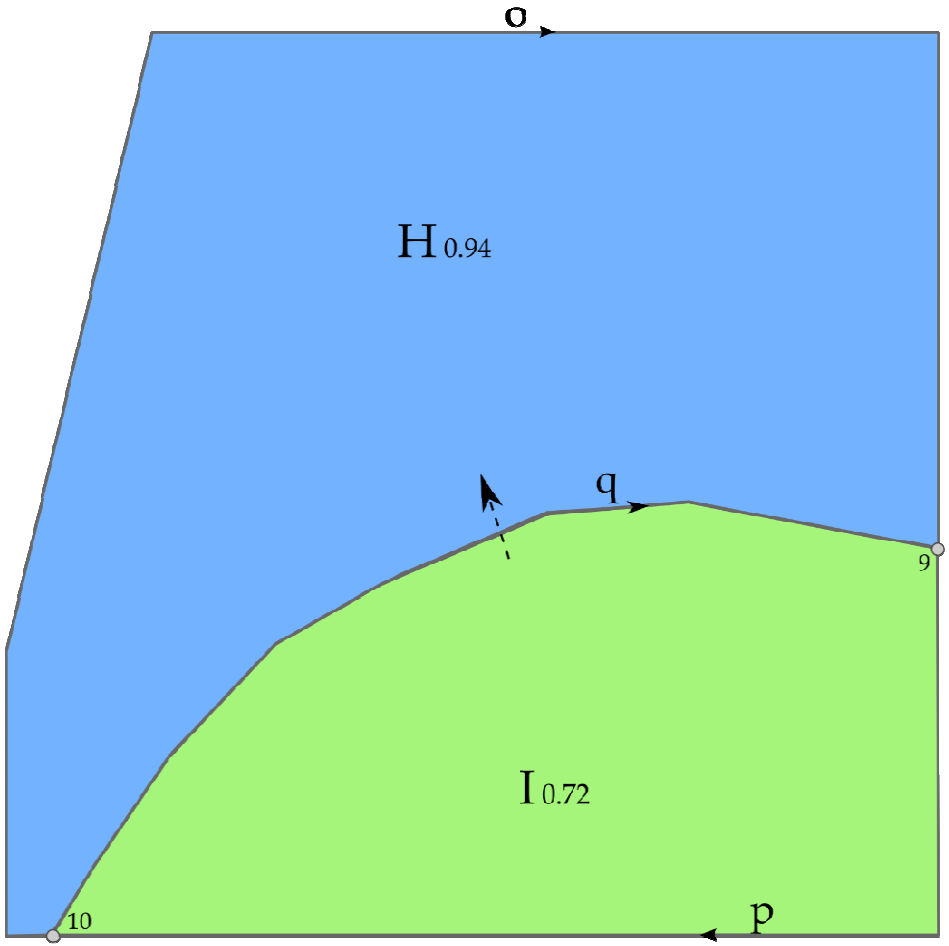
Building DAG hierarchy: step 3



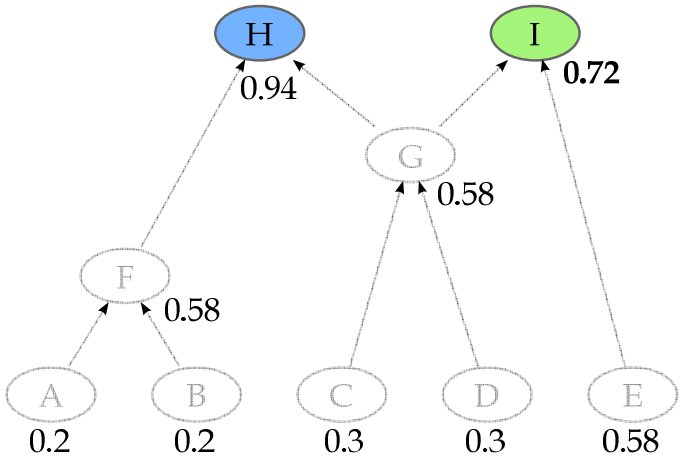
Calculate importance of the new face**s**



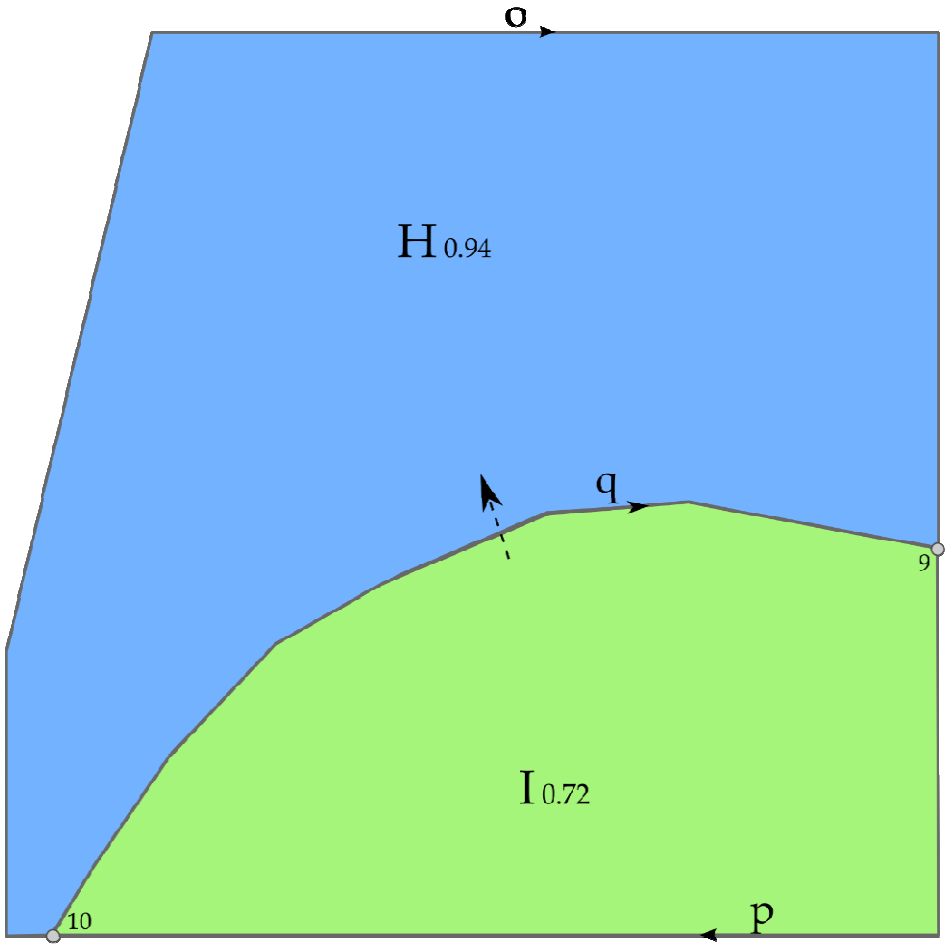
Building DAG hierarchy: step 4



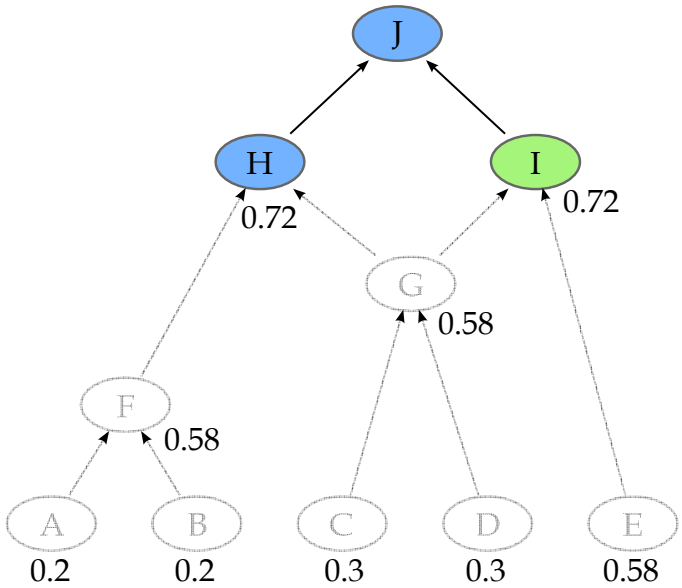
Find least important face;
Find most compatible neighbour



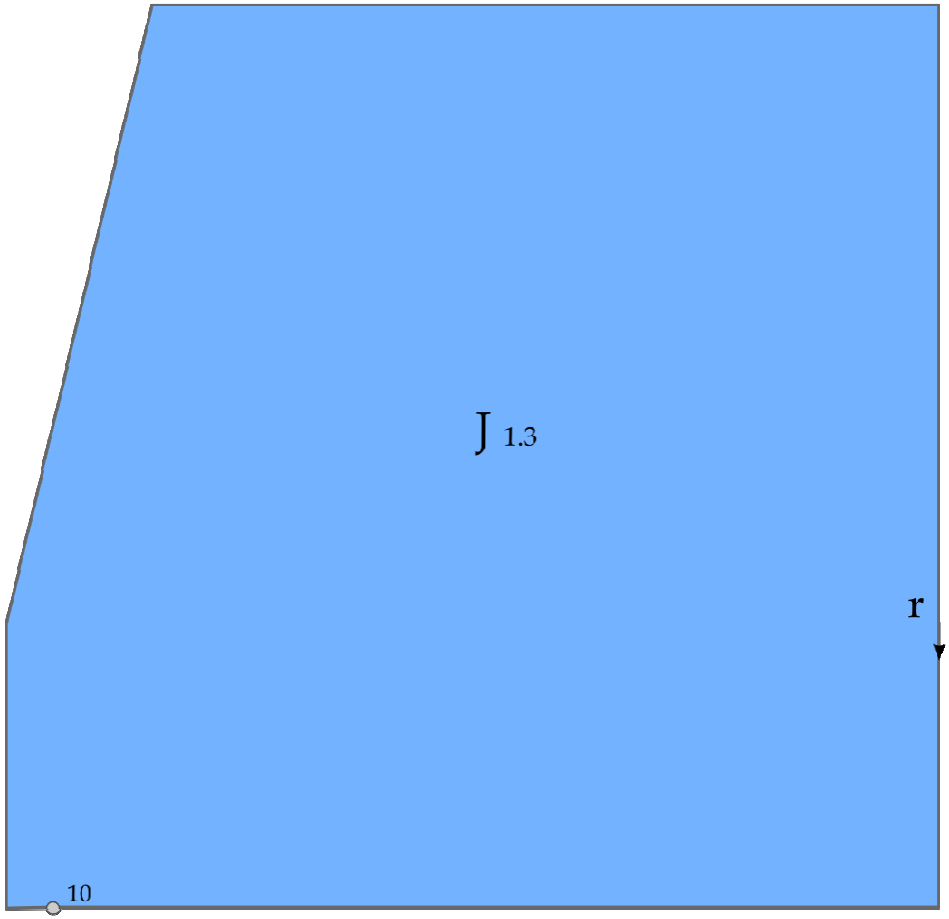
Building DAG hierarchy: step 4



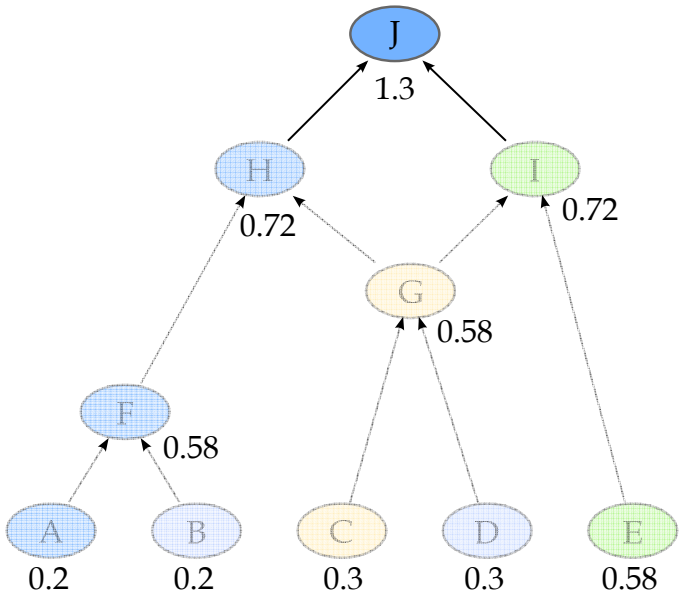
Merge faces



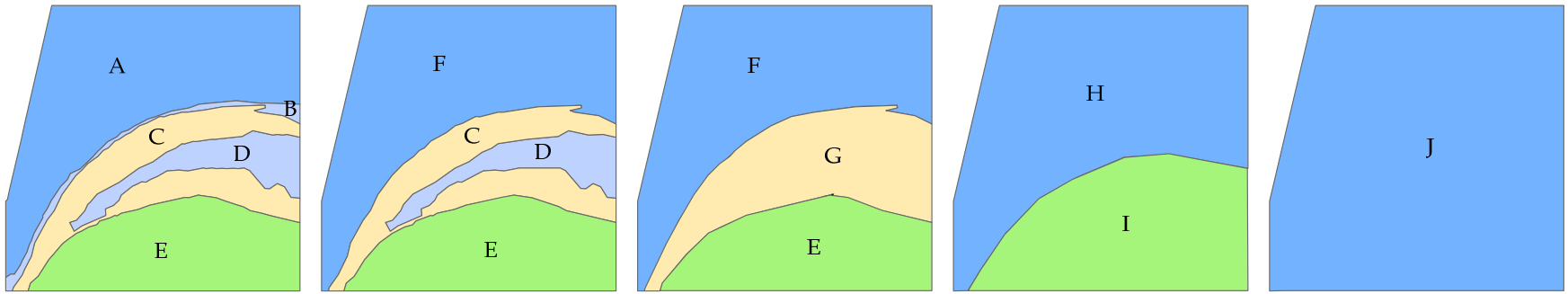
Building DAG hierarchy: step 4

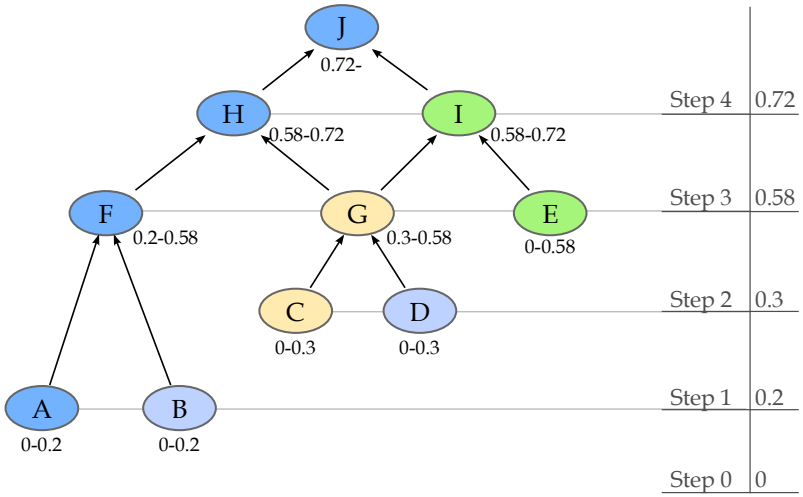


Calculate importance of the new face

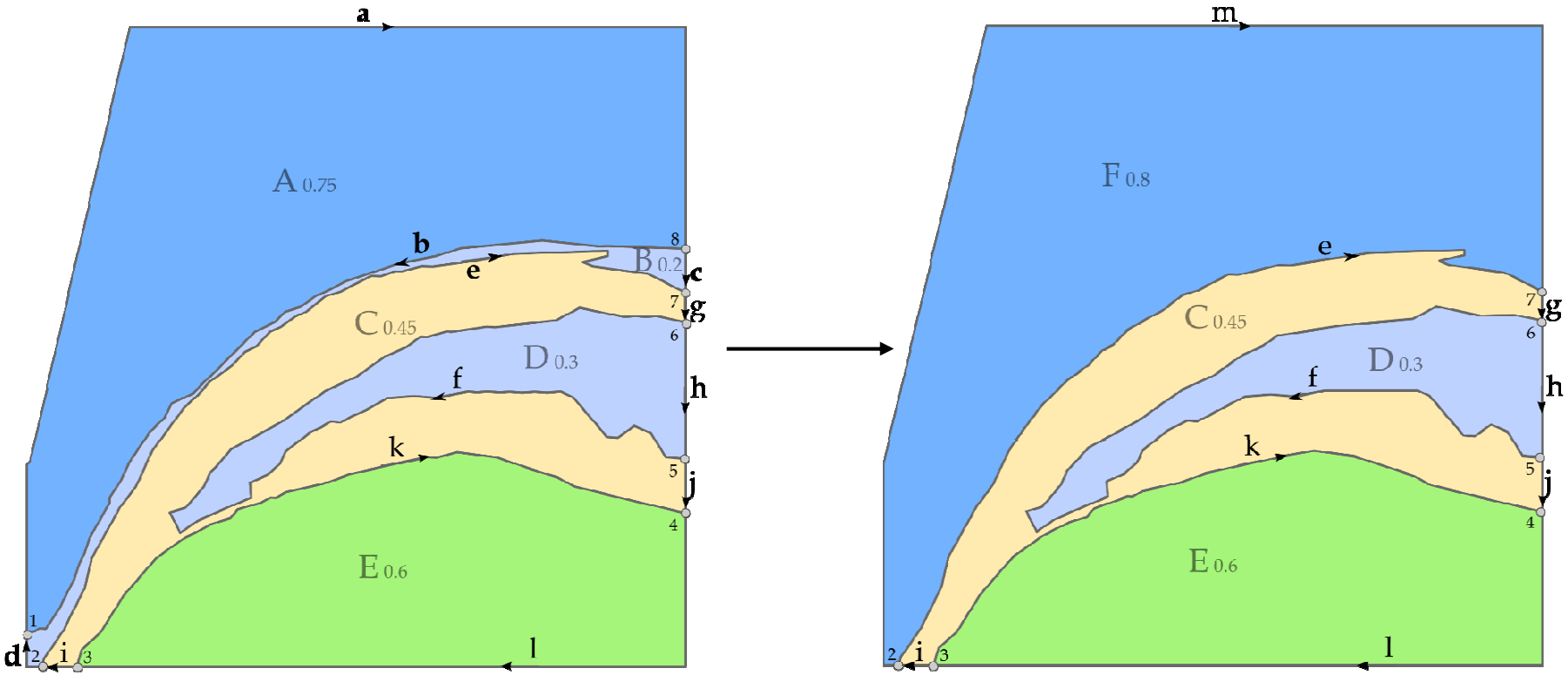


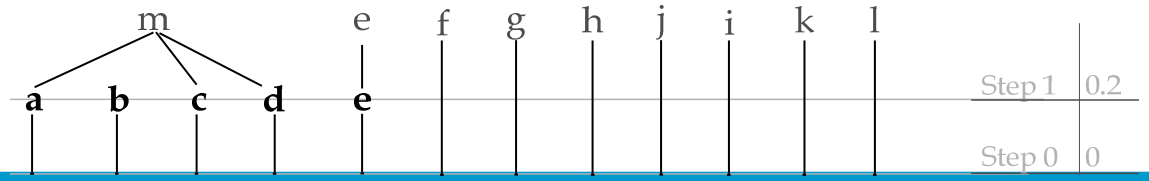
DAG hierarchy of faces



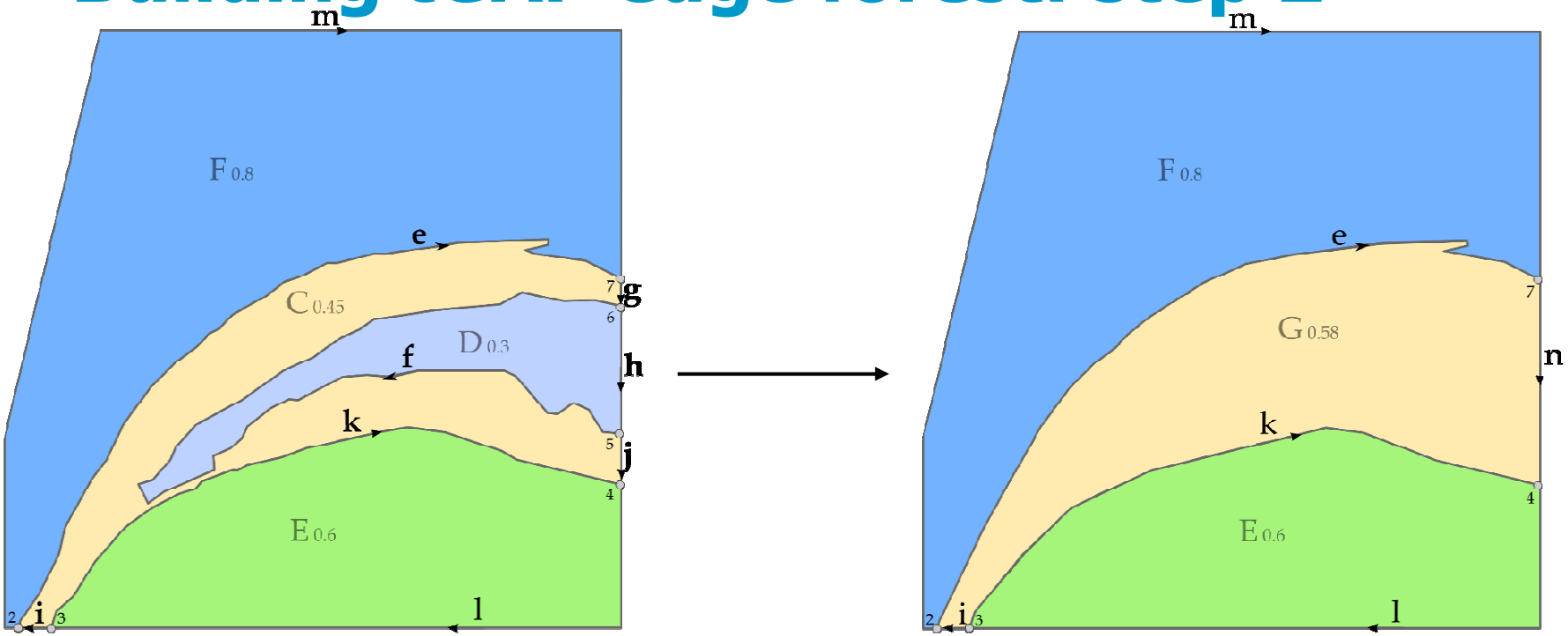


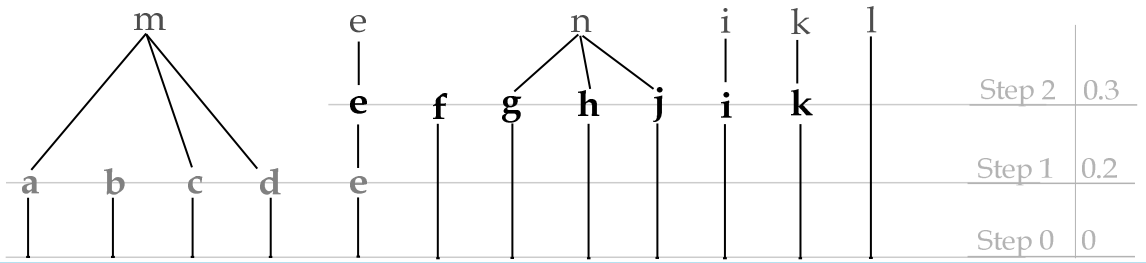
Building tGAP edge forest: step 1



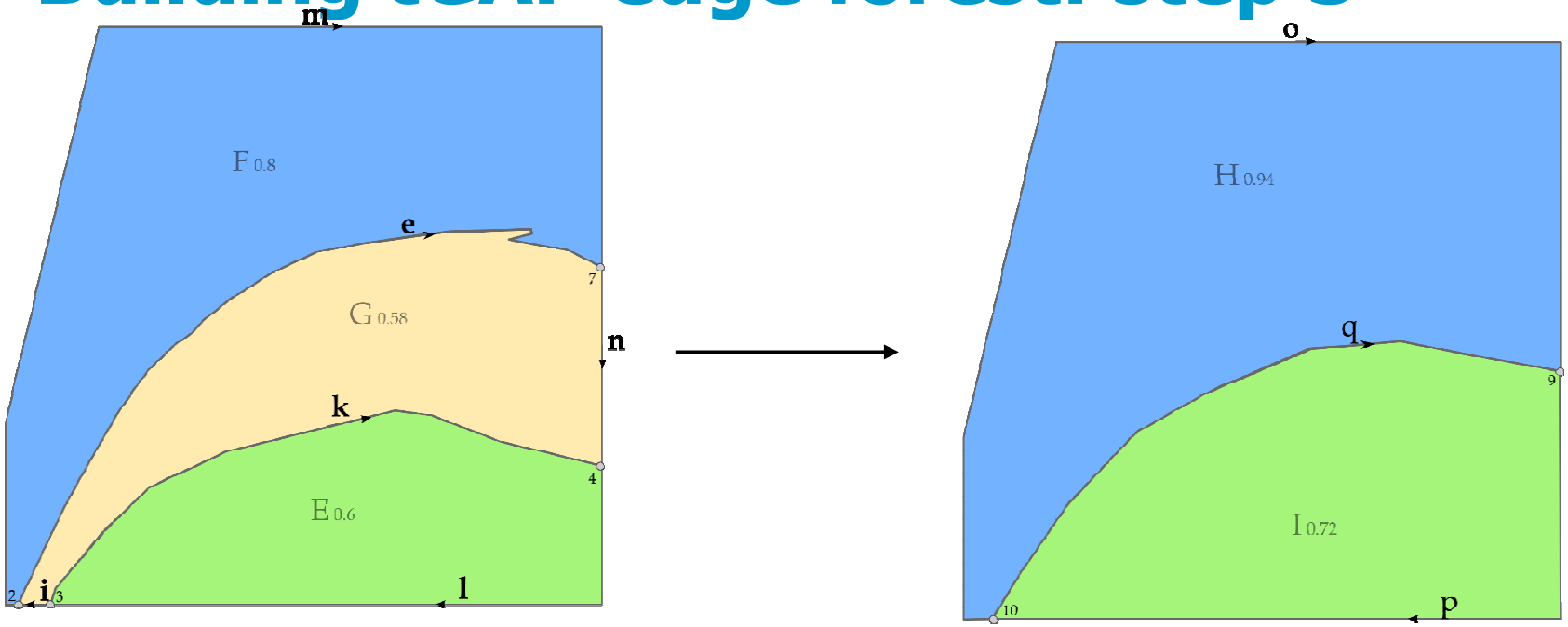


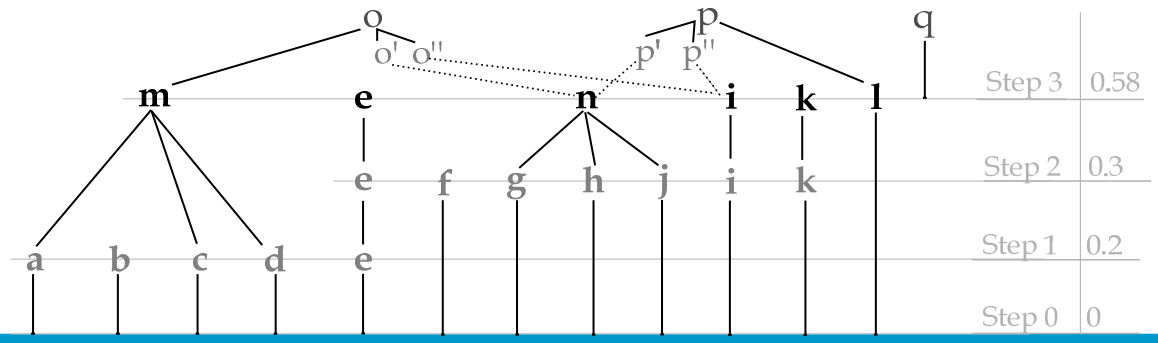
Building tGAP edge forest: step 2



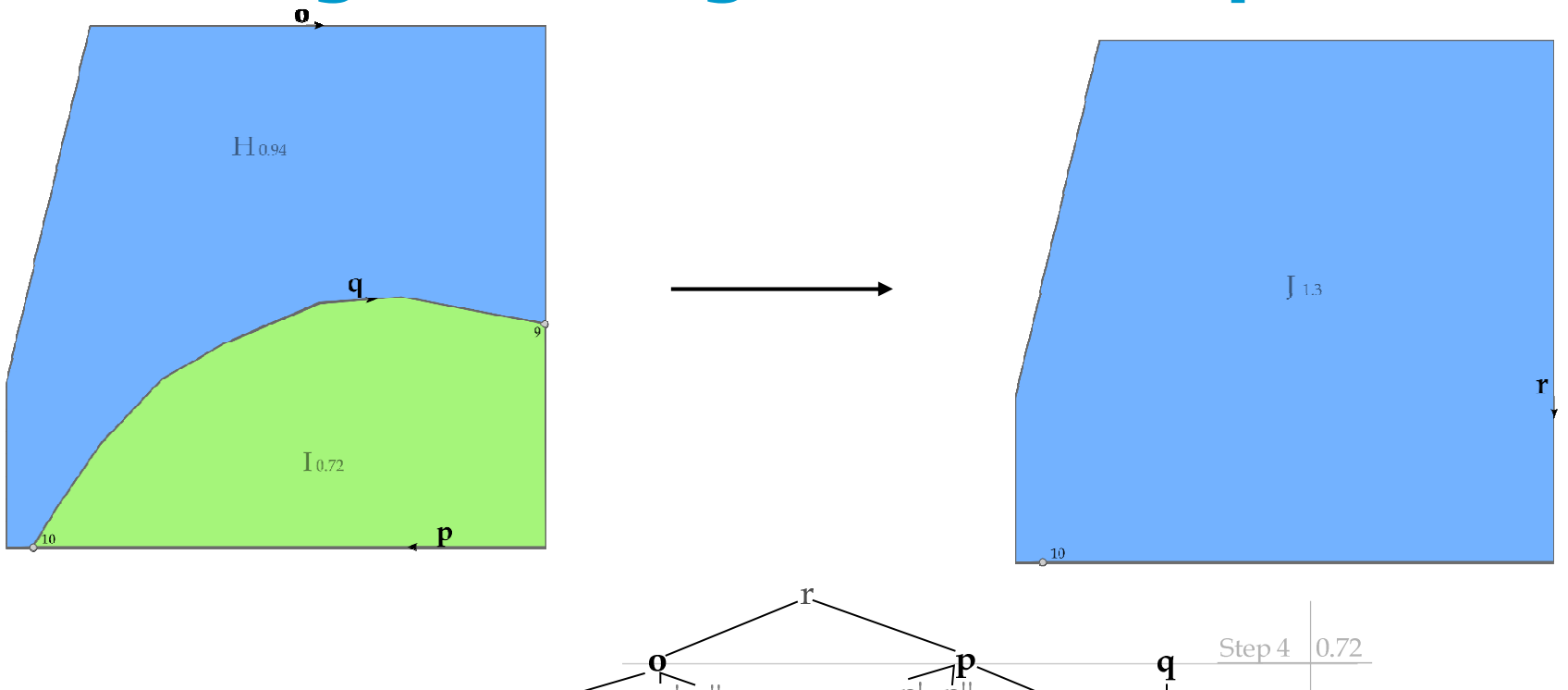


Building tGAP edge forest: step 3

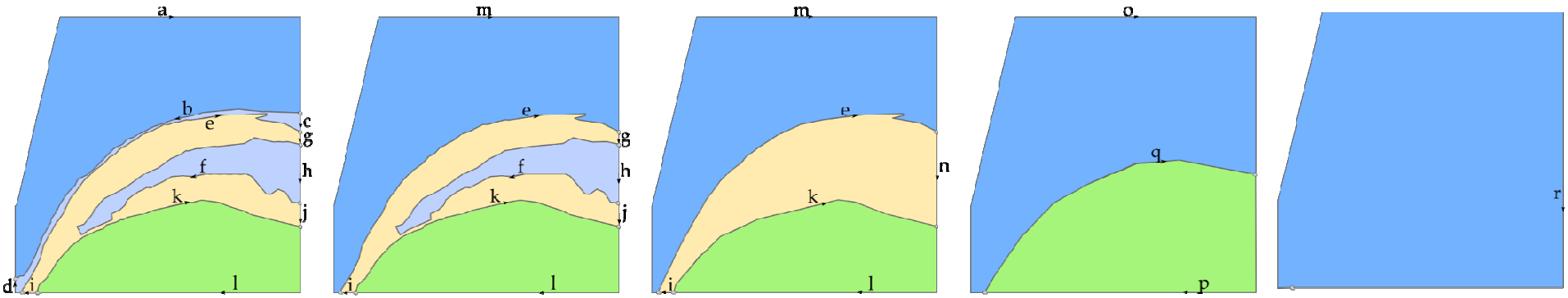


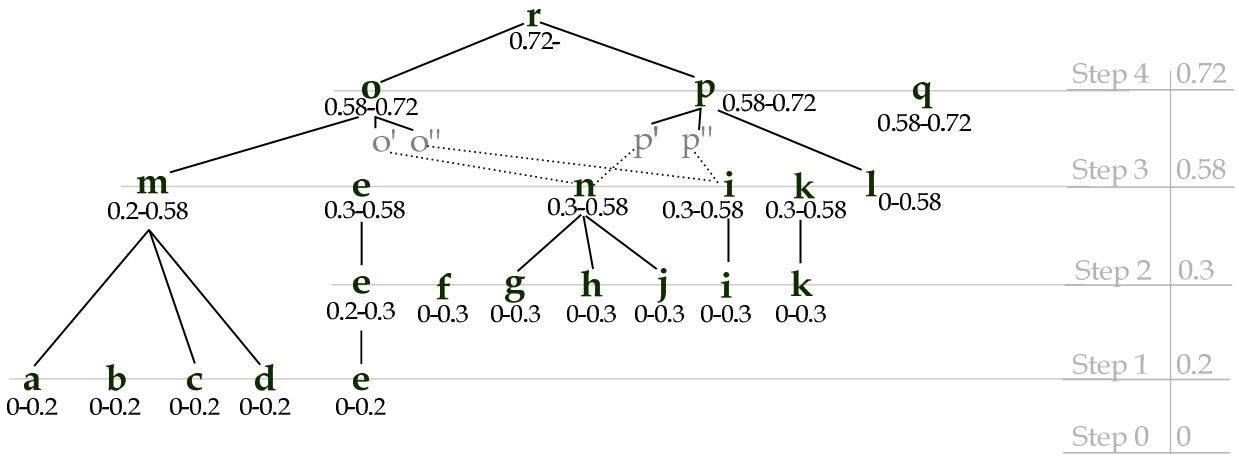


Building tGAP edge forest: step 4



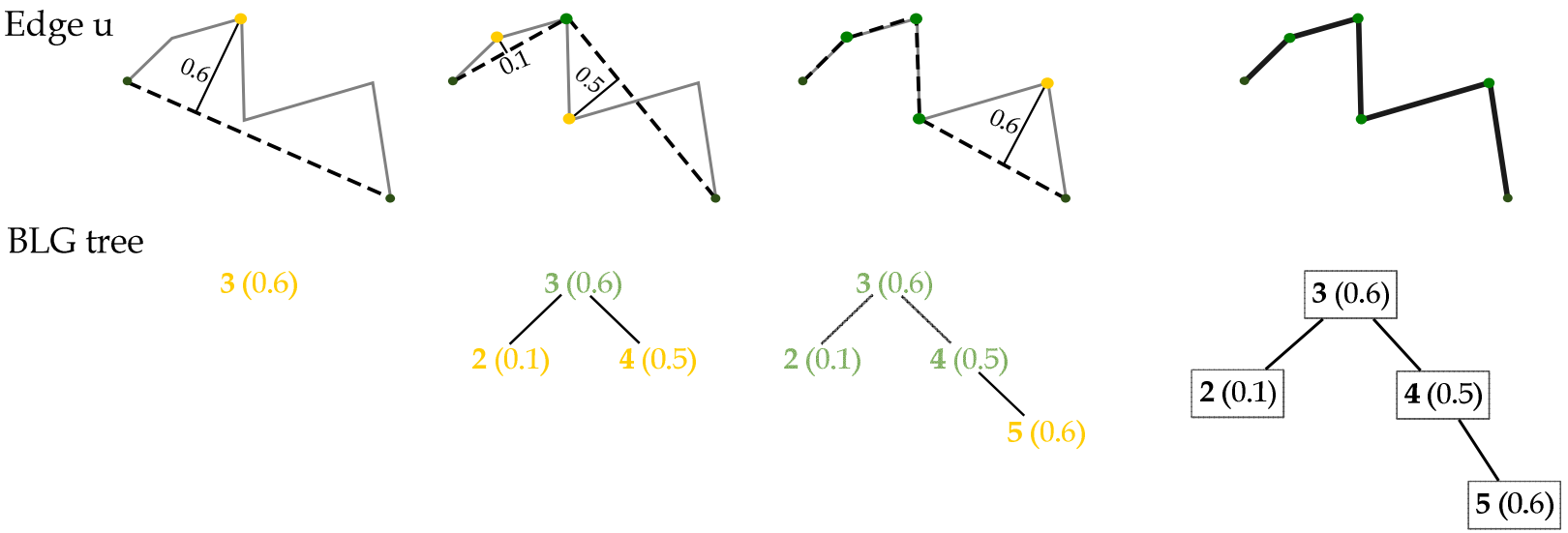
Complete tGAP edge forest





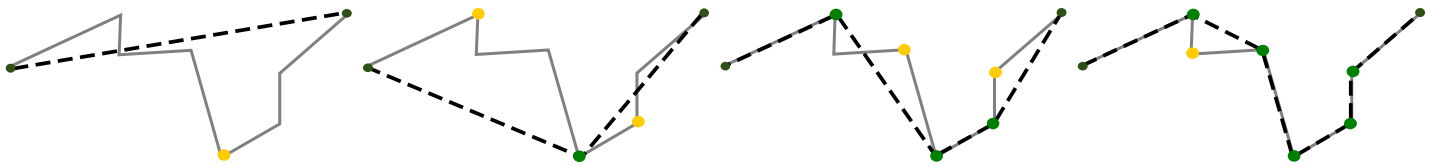
BLG edge trees

- Douglas-Peuker algorithm for edge simplification
- Results stored in a BLG tree

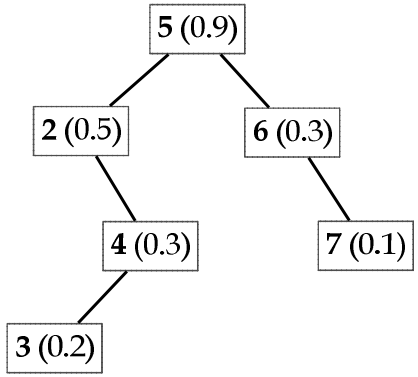


BLG edge trees

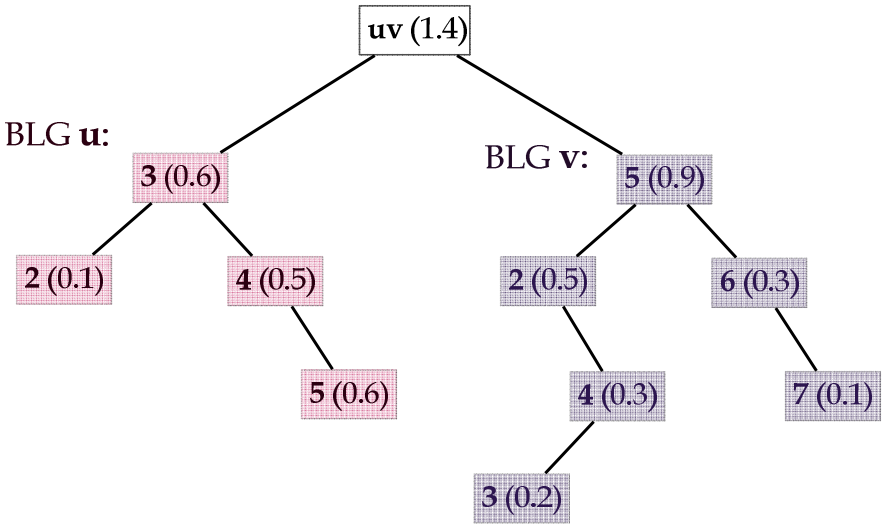
•BLG tree of another edge, **v**:

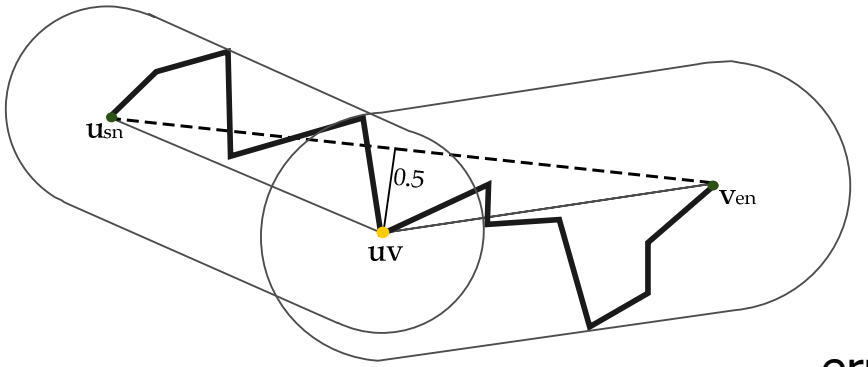


v: BLG tree



Joined BLG trees



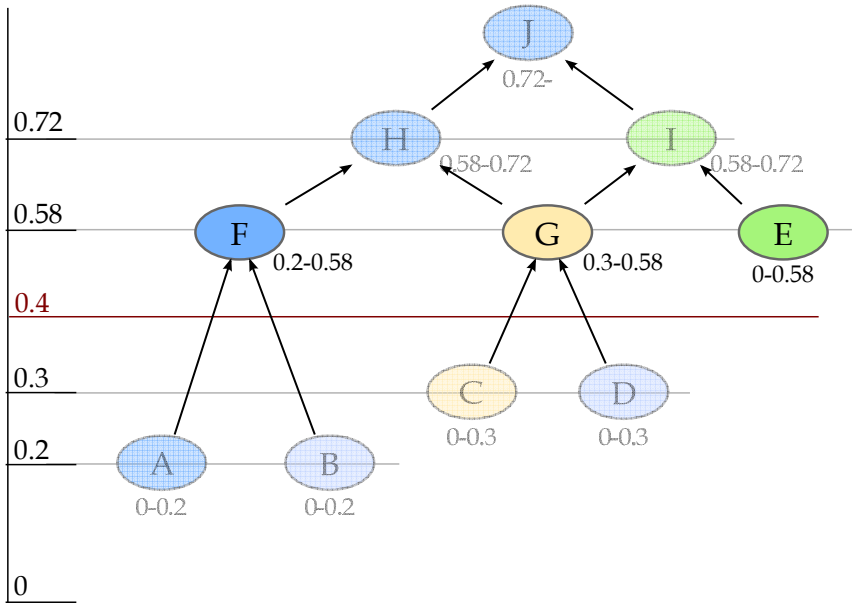


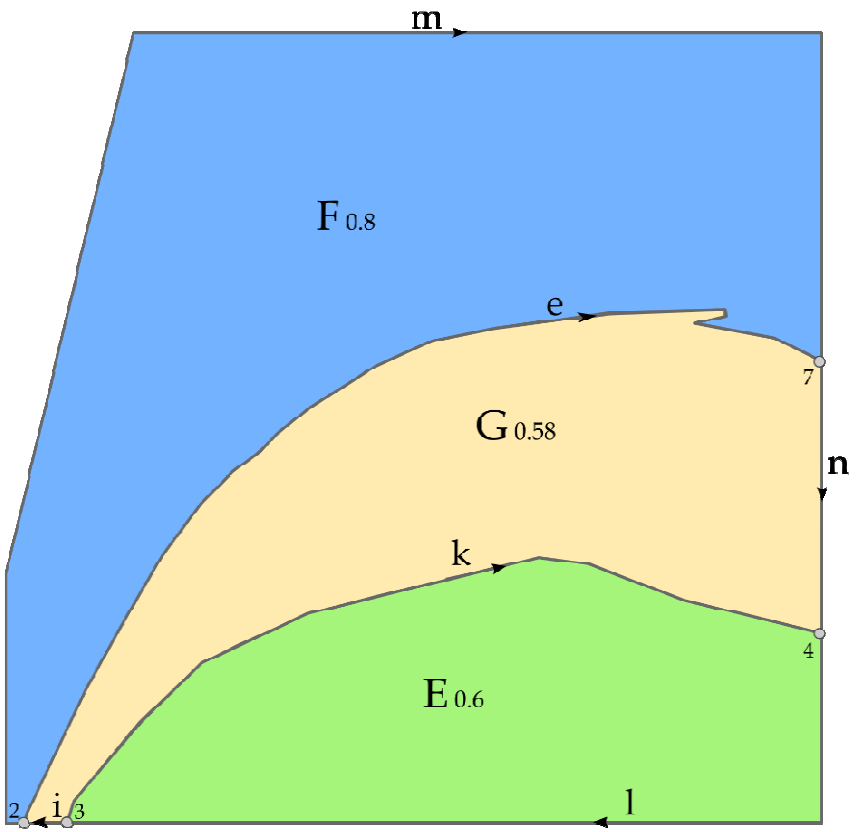
err_uv = dist(point(uv), line(u_{sn},v_{en})) +
max{err_u, err_v}=

0.5+max{0.6,0.9} = 0.5 + 0.9 = 1.4

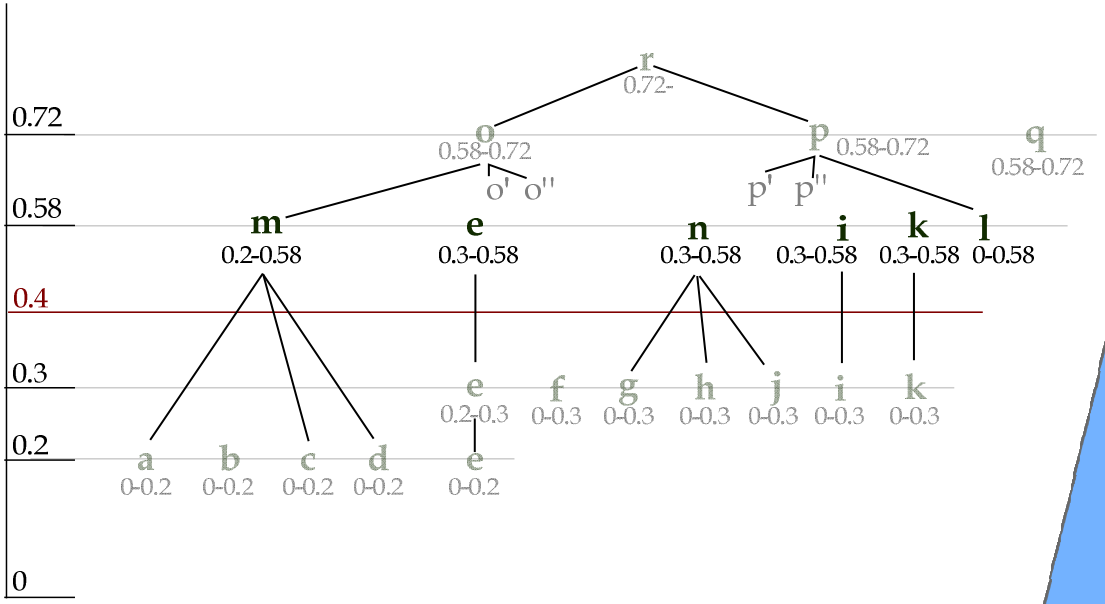
Using DAG hierarchy for selection

Importance = 0.4

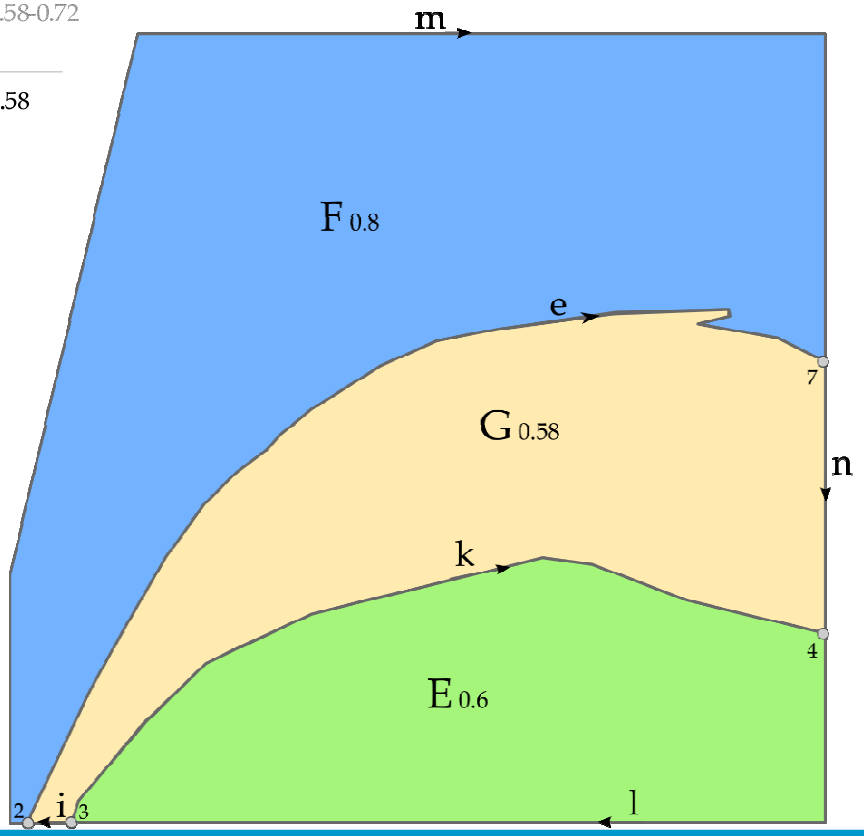




Using edge forest for selection

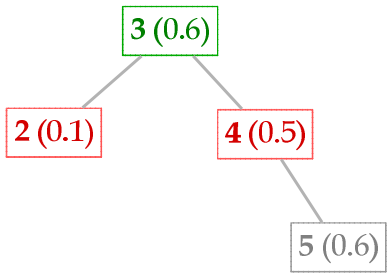


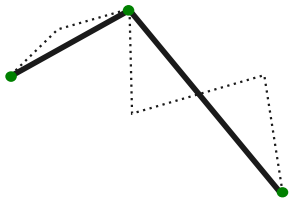
Importance = 0.4



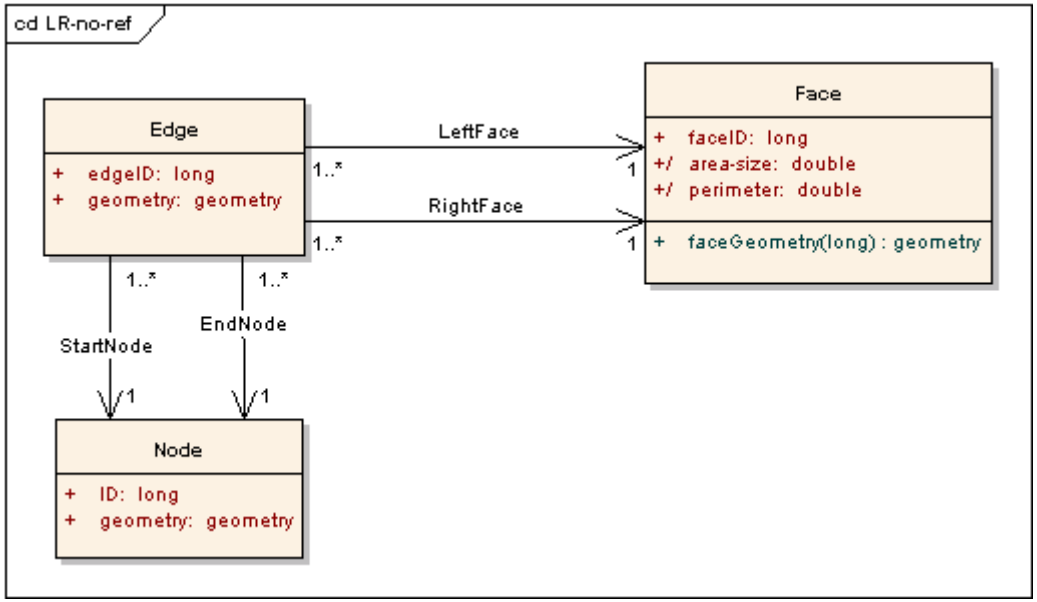
Using BLG trees for edge simplification

•Selecting vertices of BLG tree **u** for tolerance = 0.52

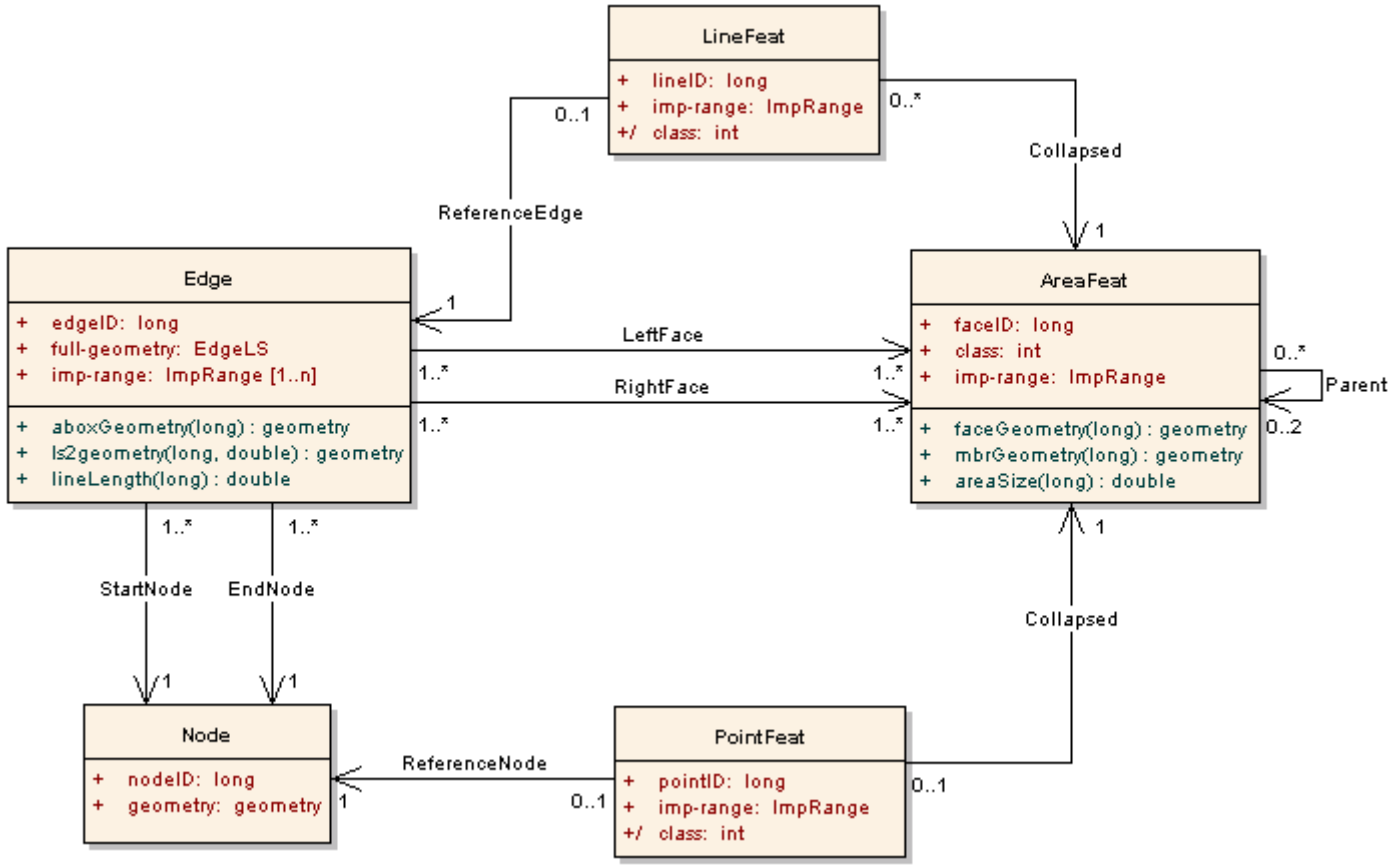




Implementation
Left-Right topology without edge references



Implementation: tGAP classes



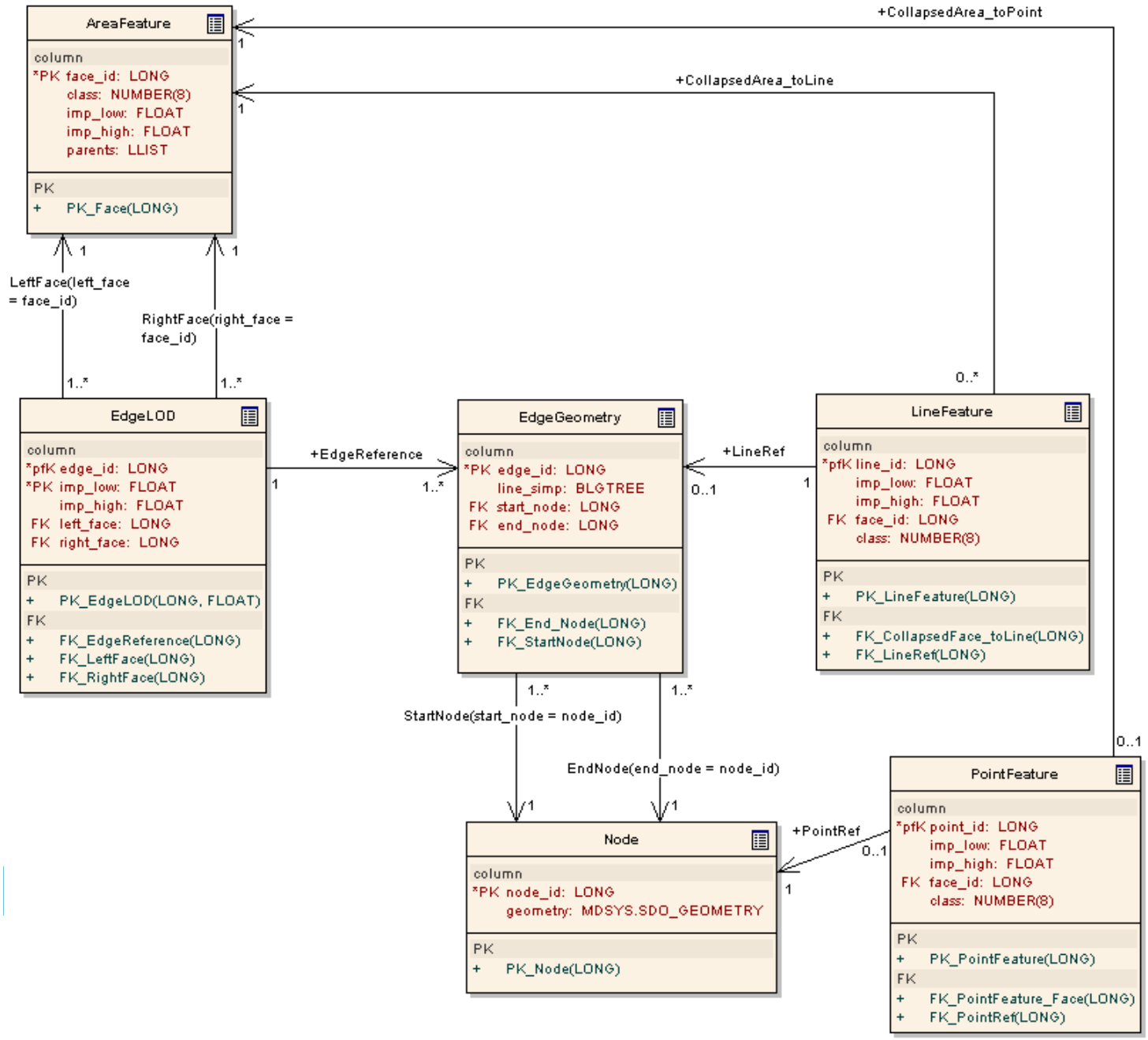
Implementation: new data types

- Data type for (variable detail) edge geometry

```
struct BLGTREE {
    double x_coord;
    double y_coord;
    float tolerance;
    struct BLGTREE *left_node;
    struct BLGTREE *right_node;
}
```

For joined BLG trees left & right_node refer to left & right BLG trees

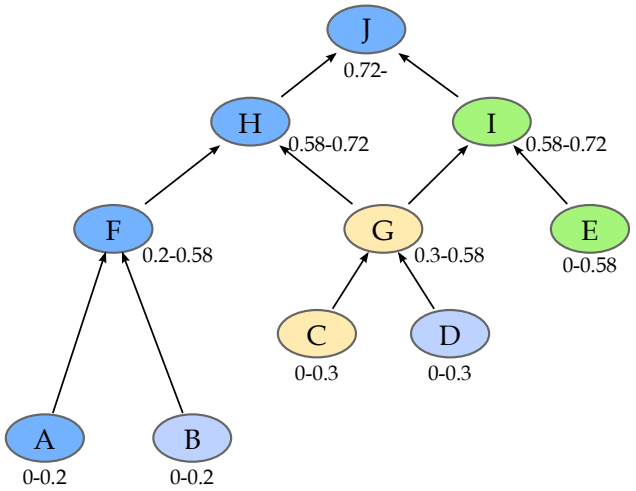
Implementation: tGAP tables



TU Delft

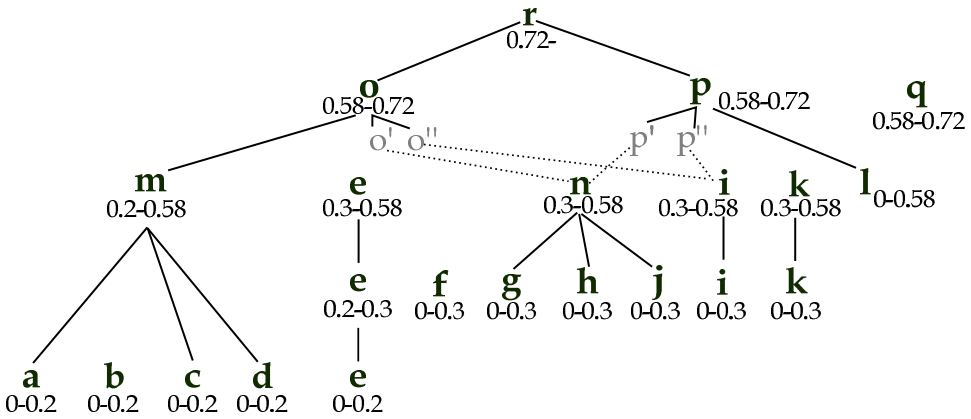
Implementation: AreaFeature table

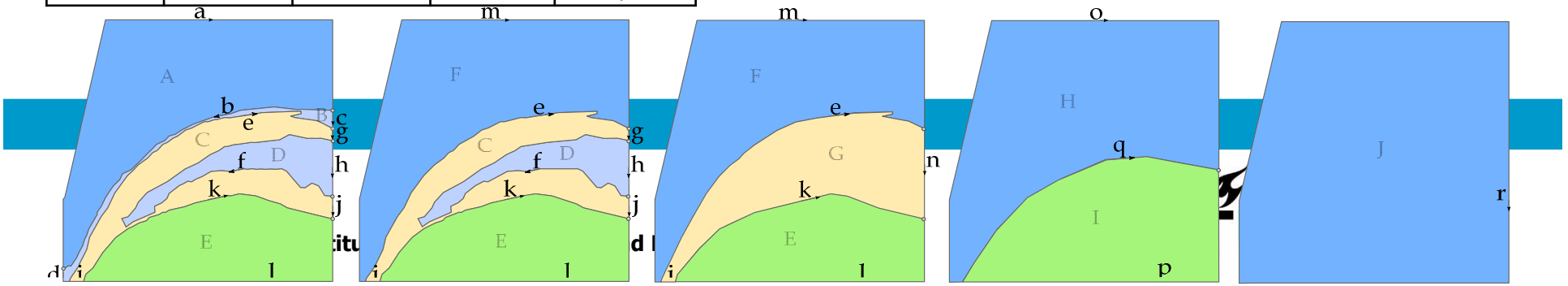
AreaFeature				
face-id	class	imp-low	imp-high	parents
A	1	0	0.2	F
B	3	0	0.2	F
C	2	0	0.3	G
D	3	0	0.3	G
E	4	0	0.58	I
F	1	0.2	0.58	H
G	2	0.3	0.58	H, I
H	1	0.58	0.72	J
I	4	0.58	0.72	J
J	1	0.72		



EdgeLOD table

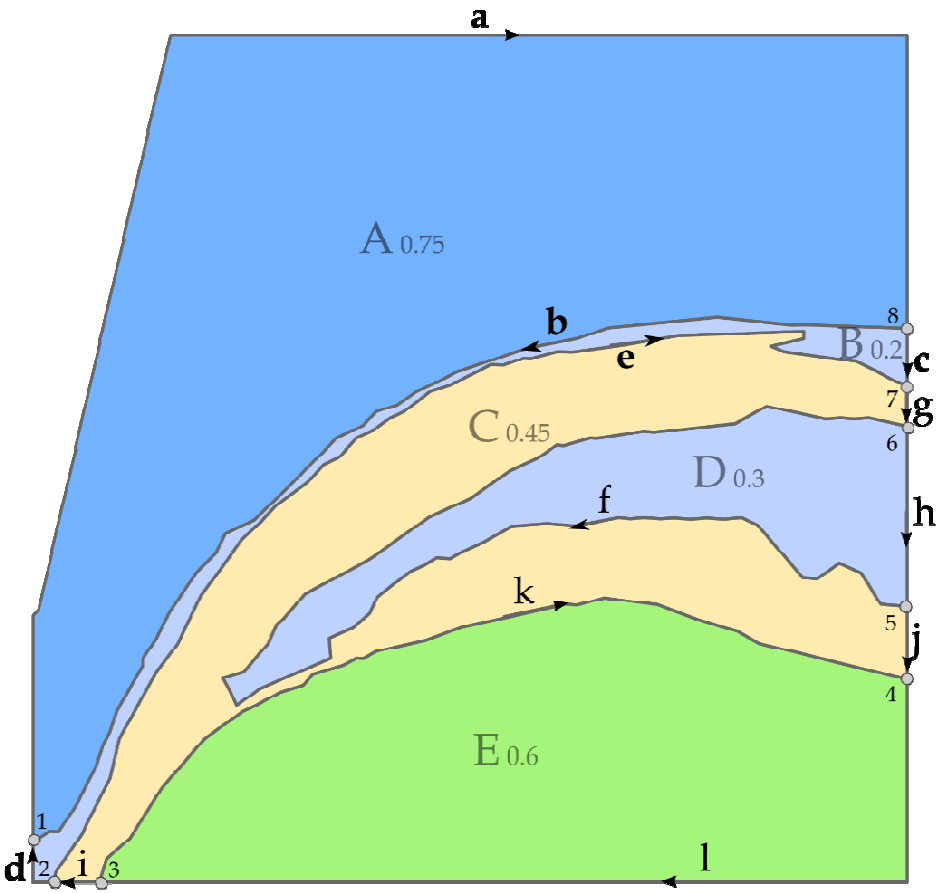
EdgeLOD				
edge-id	imp-low	imp-high	left-face	right-face
a	0	0.2	0	A
b	0	0.2	B	A
c	0	0.2	0	B
d	0	0.2	0	B
e	0	0.2	B	C
f	0	0.3	C	D
g	0	0.3	0	C
h	0	0.3	0	D
i	0	0.3	0	C
j	0	0.3	0	C
k	0	0.3	C	E
e	0.2	0.3	F	C
l	0	0.58	0	E
m	0.2	0.58	0	F
e	0.3	0.58	F	G
i	0.3	0.58	0	G
k	0.3	0.58	G	E
n	0.3	0.58	0	G
o	0.58	0.72	0	H
p	0.58	0.72	0	I
q	0.58	0.72	H	I
r	0.72		0	J





Implementation: EdgeGeometry table

EdgeGeometry			
edge-id	line-simp	start-node	end-node
a	<blgtree>	1	8
b	<blgtree>	8	1
c	<blgtree>	8	7
d	<blgtree>	2	1
e	<blgtree>	2	7
f	<blgtree>	5	6
g	<blgtree>	7	6
h	<blgtree>	6	5
i	<blgtree>	3	2
j	<blgtree>	5	4
k	<blgtree>	3	4
l	<blgtree>	4	3
m	<blgtree>	2	7
n	<blgtree>	7	4
o	<blgtree>	10	9
p	<blgtree>	9	10
q	<blgtree>	10	9
r	<blgtree>	10	10



More tGAP issues & further research

- Algorithm performing the off-line generalisation
- Mobile (or web) application
  - Progressive transfer
  - Visualisation on client
- Include other operators in generalisation process
- Updating tGAP structure
- …

References

1. Peter van Oosterom. Variable-scale topological data structures suitable for progressive data transfer: the gap-face tree and gap-edge forest. *Cartography and geographic information science*, 32:331–346, 2005.
2. Peter van Oosterom. *Reactive Data Structures for Geographic Information Systems*. PhD thesis, Department of Computer Science, Leiden University, December 1990.
3. Martijn Meijers. Implementation and testing of variable scale topological data structures. Master's thesis, TU Delft, June 2006.
4. Tinghua Ai and Peter van Oosterom. Gap-tree extensions based on skeletons. In Dianne Richardson and Peter van Oosterom, editors, *10th International Symposium on Spatial Data Handling*, pages 501–513, 2002.
5. Martin Galanda. *Automated Polygon Generalization in a Multi Agent System*. PhD thesis, University of Zürich, 2003.