The tGAP structure: minimizing redundancy and maximizing consistency and offering access at any LoD

Peter van Oosterom

INSPIRE Workshop on Multiple-Representation and Data Consistency, 7-8 November 2006, JRS, Ispra (VA), Italy

13 November, 2006

Section GIS Technology

TU Delft
Delft University of Technology

1. Context

- Fits in INSPIRE workshop theme ‘Technological outlook (generalisation services, adaptive zooming, mixture of multiple-representation and generalisation)’
- Two generalisation Bsik ‘Space for Geo-Information’ projects in the Netherlands: RGI-002, resp. 233. Partners: ITC, ESRI, LaserScan, ANWB, Municip of Amsterdam, Topographic Service Kadaster,…
- Two international top-ups are proposed: IGN France& Finnish Geodetic Institute, resp. Univ Hannover

Contents

1. Introduction
2. Background tGAP structure
3. First implementation
4. Client-server set-up and progressive refinement
5. Improvements
6. Conclusions

TU Delft

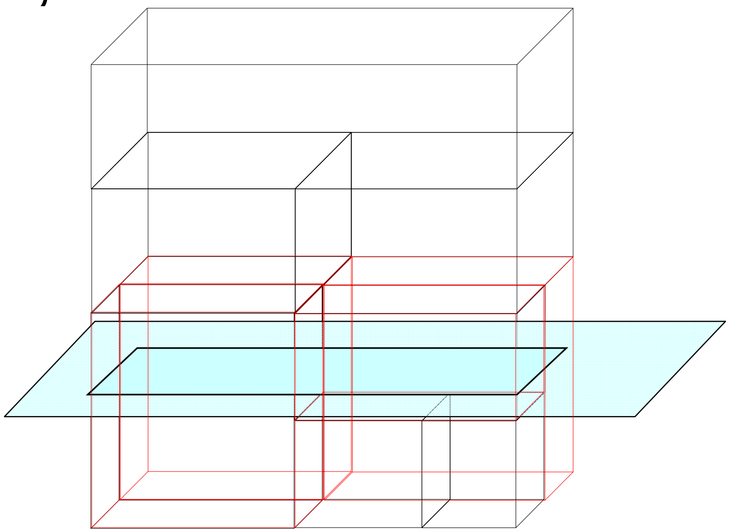
1. Introduction

- Multi-scale databases: often multiple representation
  drawbacks: redundancy, fixed levels of detail
- Scaleless data structures: single representation with additional structure to access at any level of detail

- Often also spatial organization (clustering/indexing)
- Progressive transfer: keep sending more details (compare to raster formats: data pyramids, wavelets)

1. Generalized Area Partitioning-tree (GAP-tree) history

- In normal GAP-tree (van Oosterom 1993) areas are stored as independent polygons, drawback (computed) redundancy
- Vermeij et al.'03 proposed topological GAP-tree: edges and faces (with importance range, consider as height), reduced redundancy between neighbors



- Still some redundancy left: coordinates in higher level edge also present in lower (more detailed) level edges

1. tGAP structure (GAP-face tree + GAP-edge forest)

- Also coordinate redundancy between edges at different aggregation levels is removed

- Throughout remainder of presentation examples of the tGAP-structure (creation and use) will be shown

- Creation of the tGAP-tree is shown in pairs of steps
  1. removal of least important face (merge face)
  2. removal of edges, merge of edges (BLG-tree)

TU Delft

1. Proposed solution: tGAP structure

- Variable scale: infinite amount of levels
- Base level with most detailed geometry/topology
- Create links/structure on top

- This year first tests of structure
- Based on this: further extensions and improvements suggested

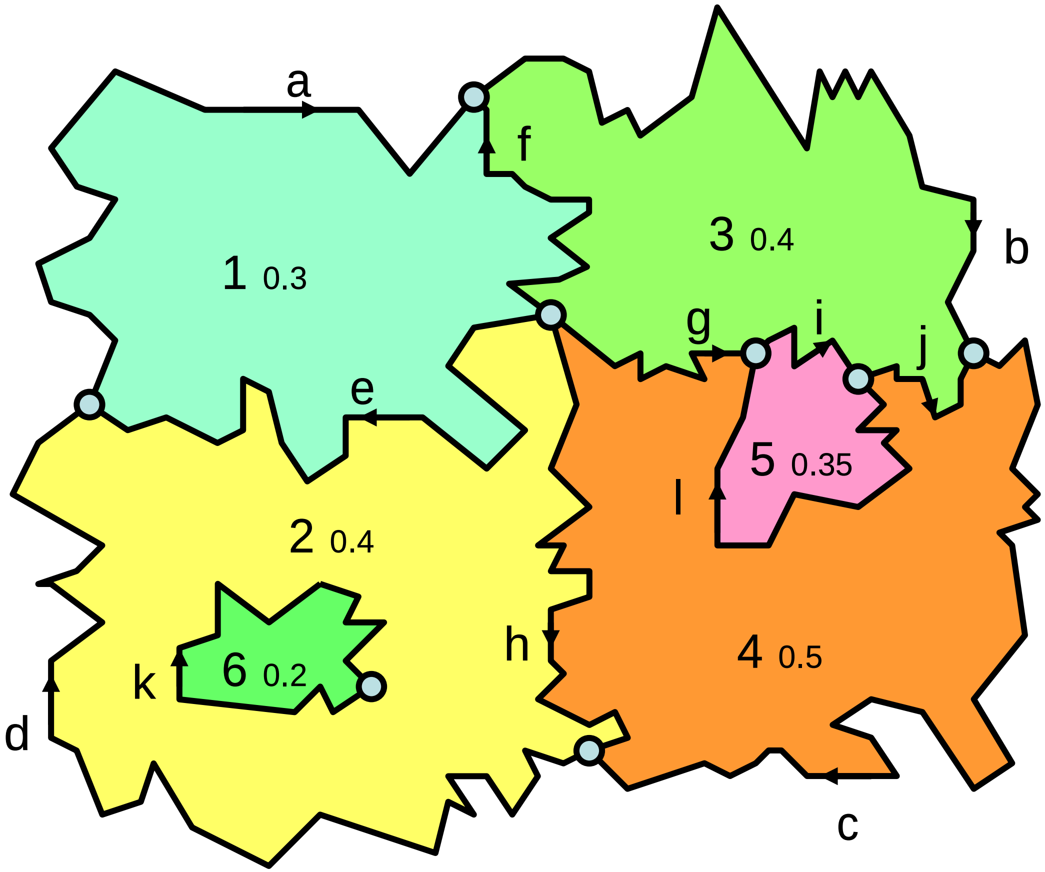
TU Delft

Contents

1. Introduction
2. Background tGAP structure
3. First implementation
4. Client-server set-up and progressive refinement
5. Improvements
6. Conclusions

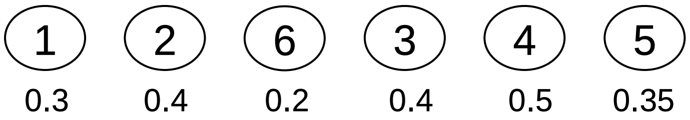
TU Delft

2. Constructing tGAP face tree

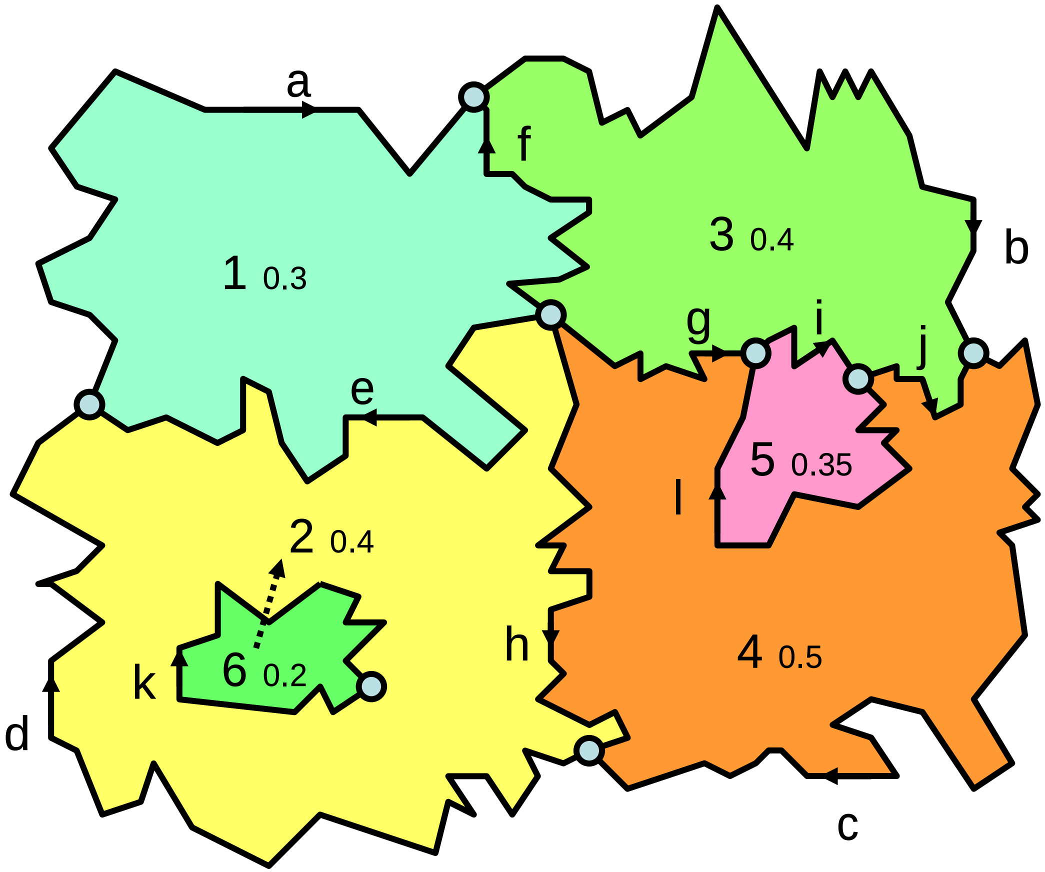


Step 0

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

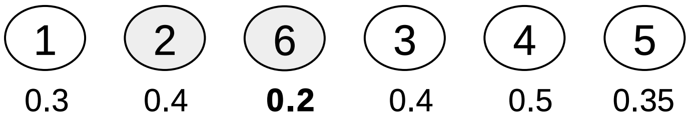


2. Constructing tGAP face tree

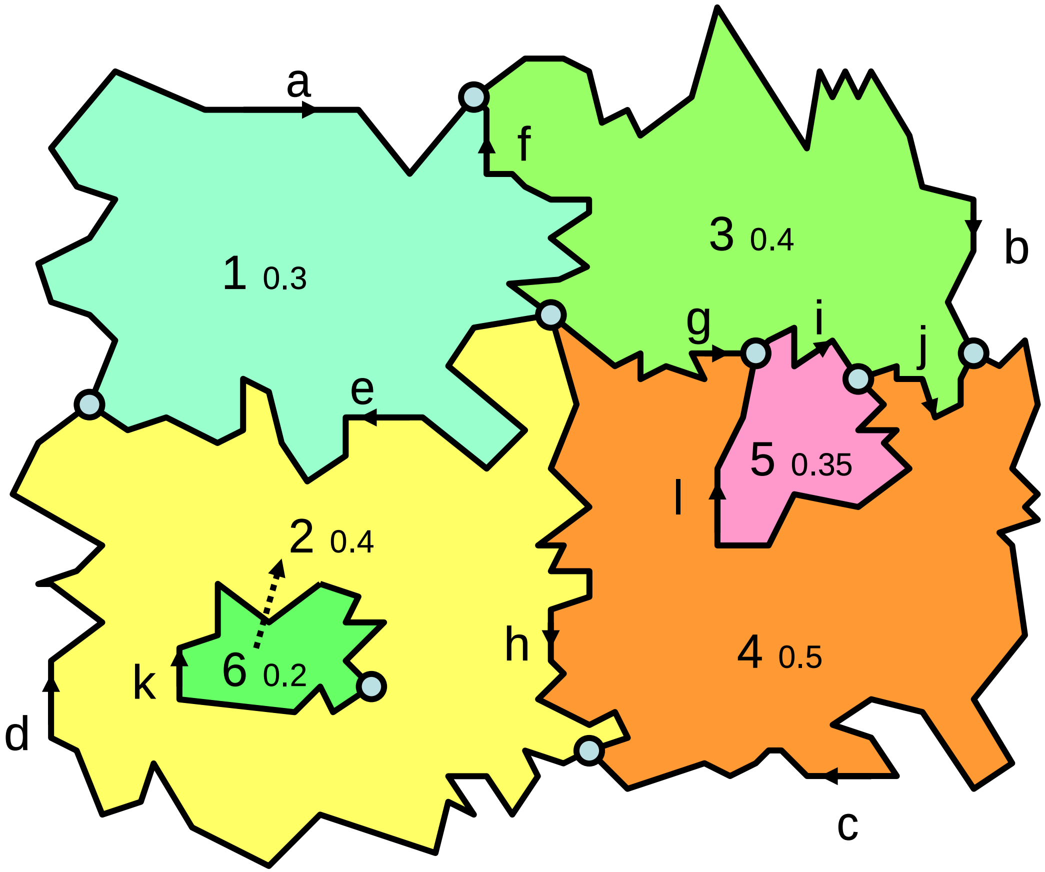


Step 1

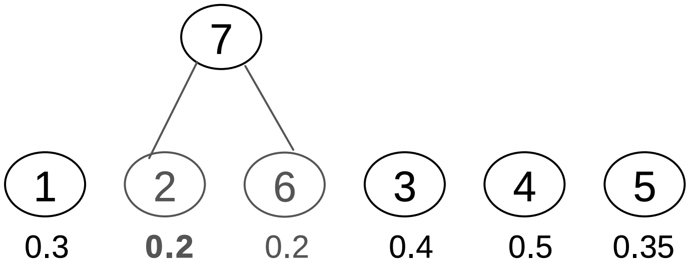
Comp(u,v) = Length(Bnd(u,v)) * Class-Similarity(u,v)



2. Constructing tGAP face tree

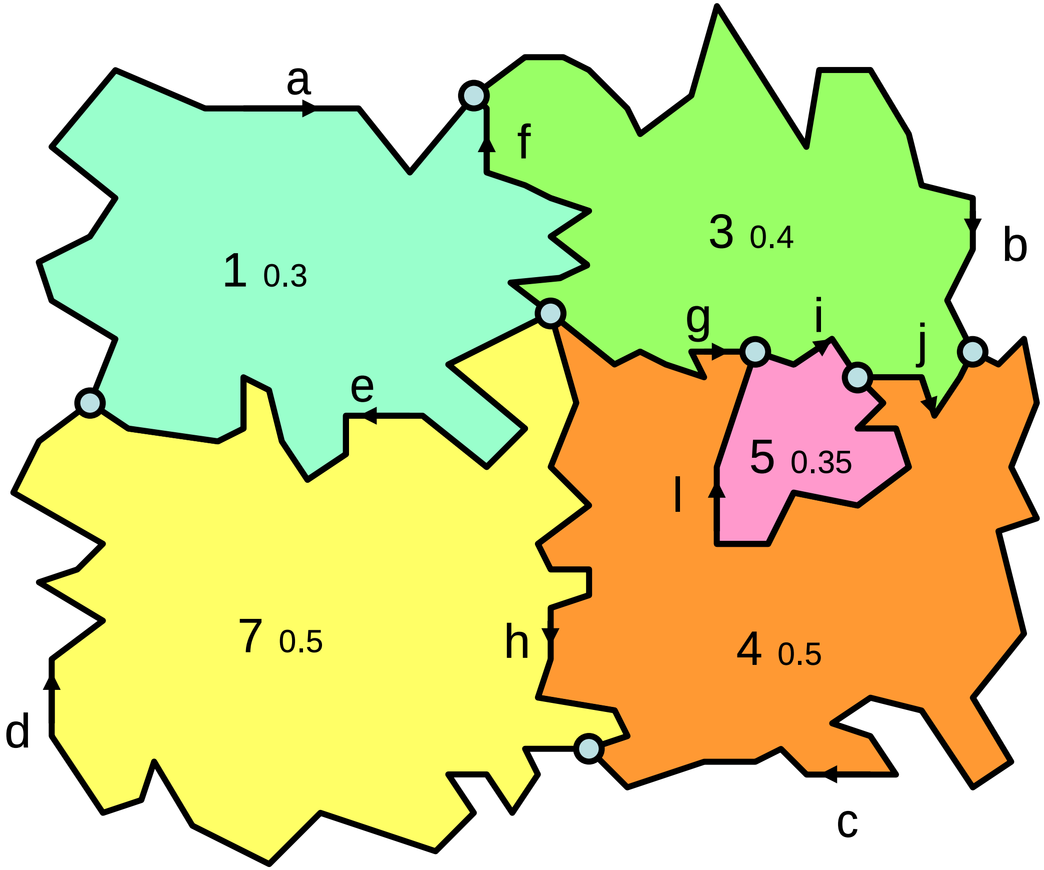


Step 1



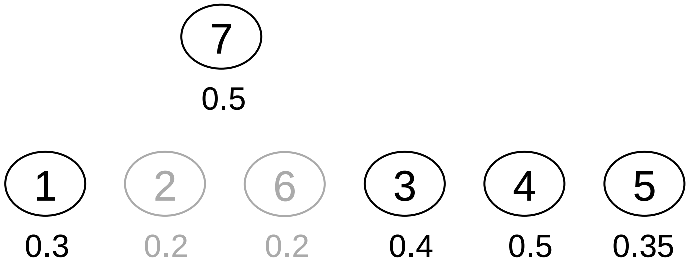
TU Delft

2. Constructing tGAP face tree

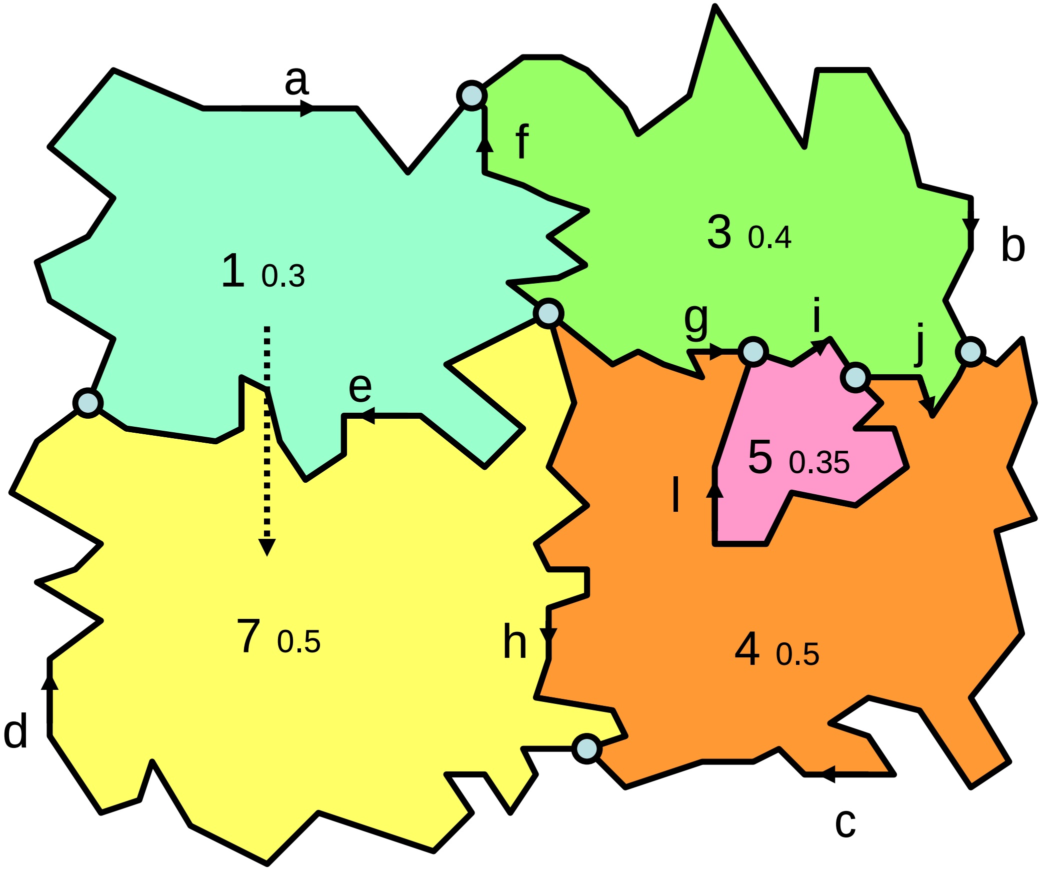


Step 1

Importance(7) = 0.5

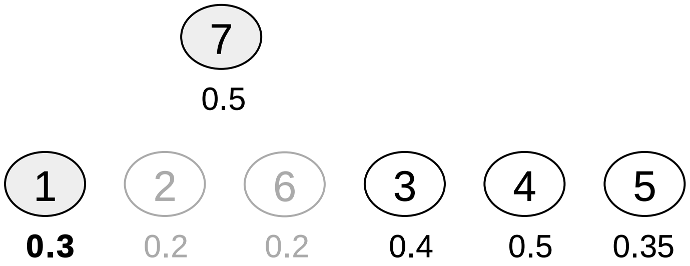


2. Constructing tGAP face tree



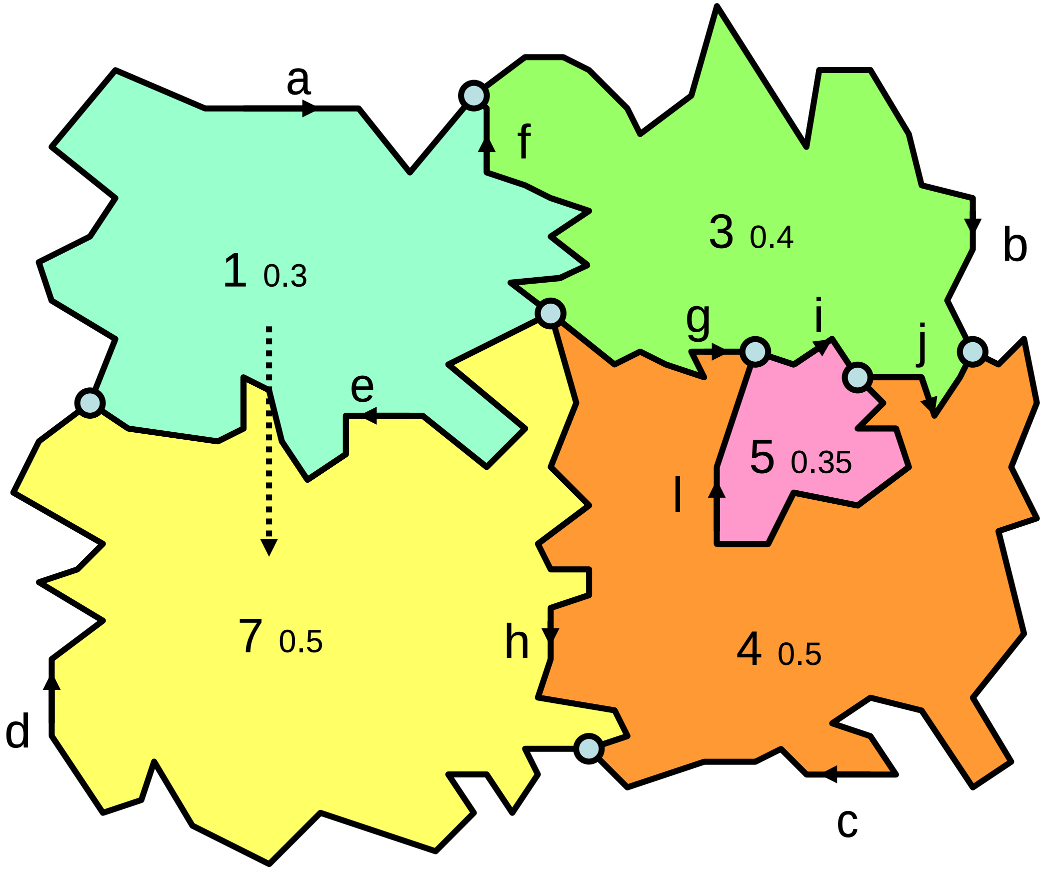
Step 2

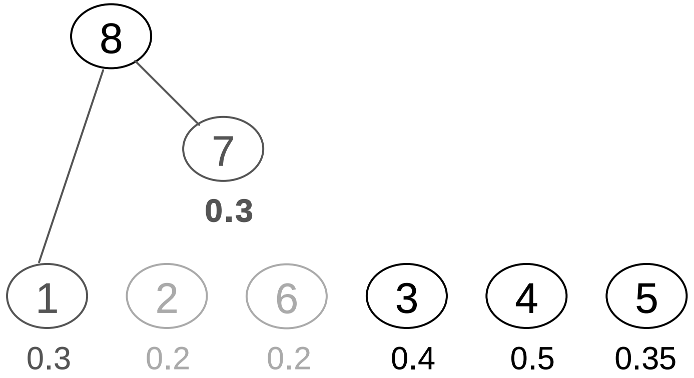
$\text{Max}_{v} \{\text{Comp}(1, v)\} = \text{Comp}(1, \mathbf{7})$



2. Constructing tGAP face tree

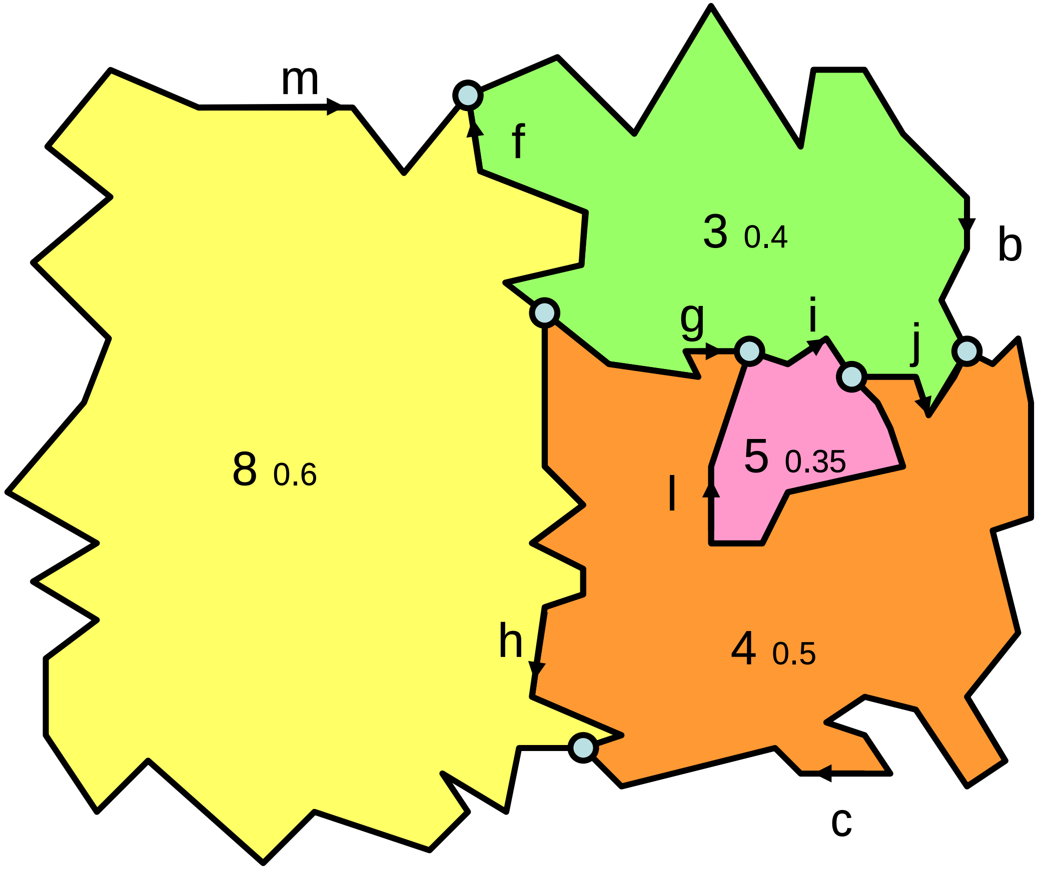
Step 2





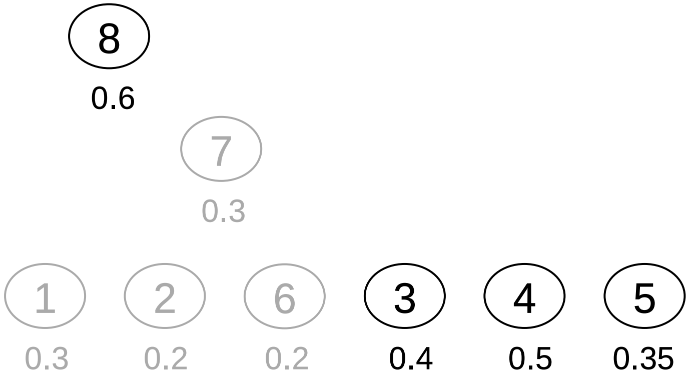
TU Delft

2. Constructing tGAP face tree

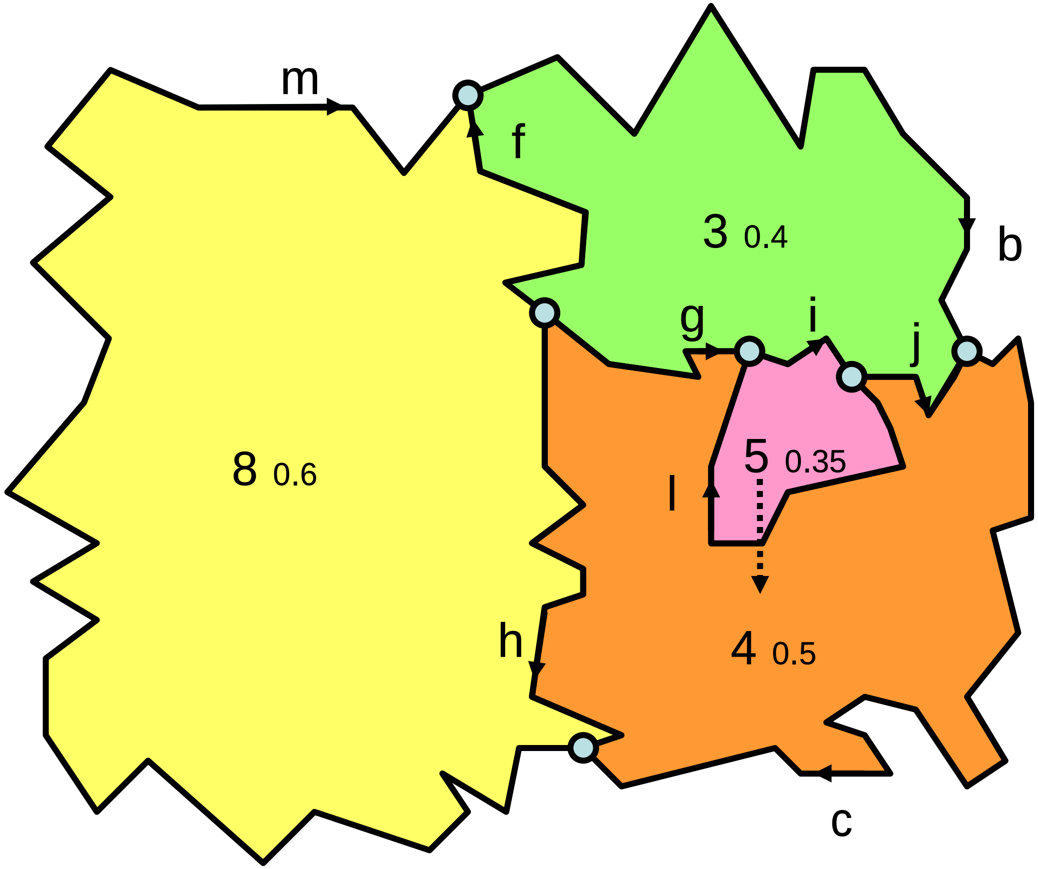


Step 2

Importance(8) = 0.6

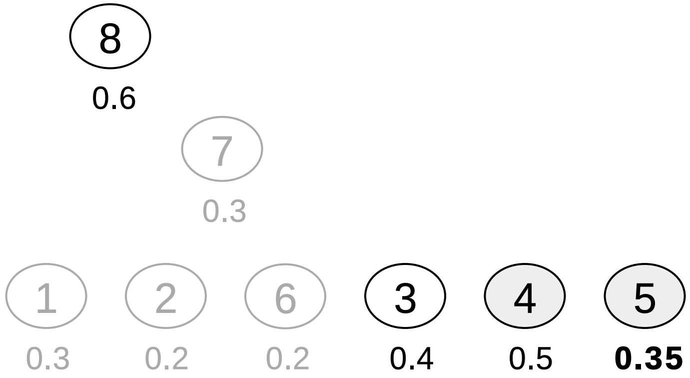


2. Constructing tGAP face tree



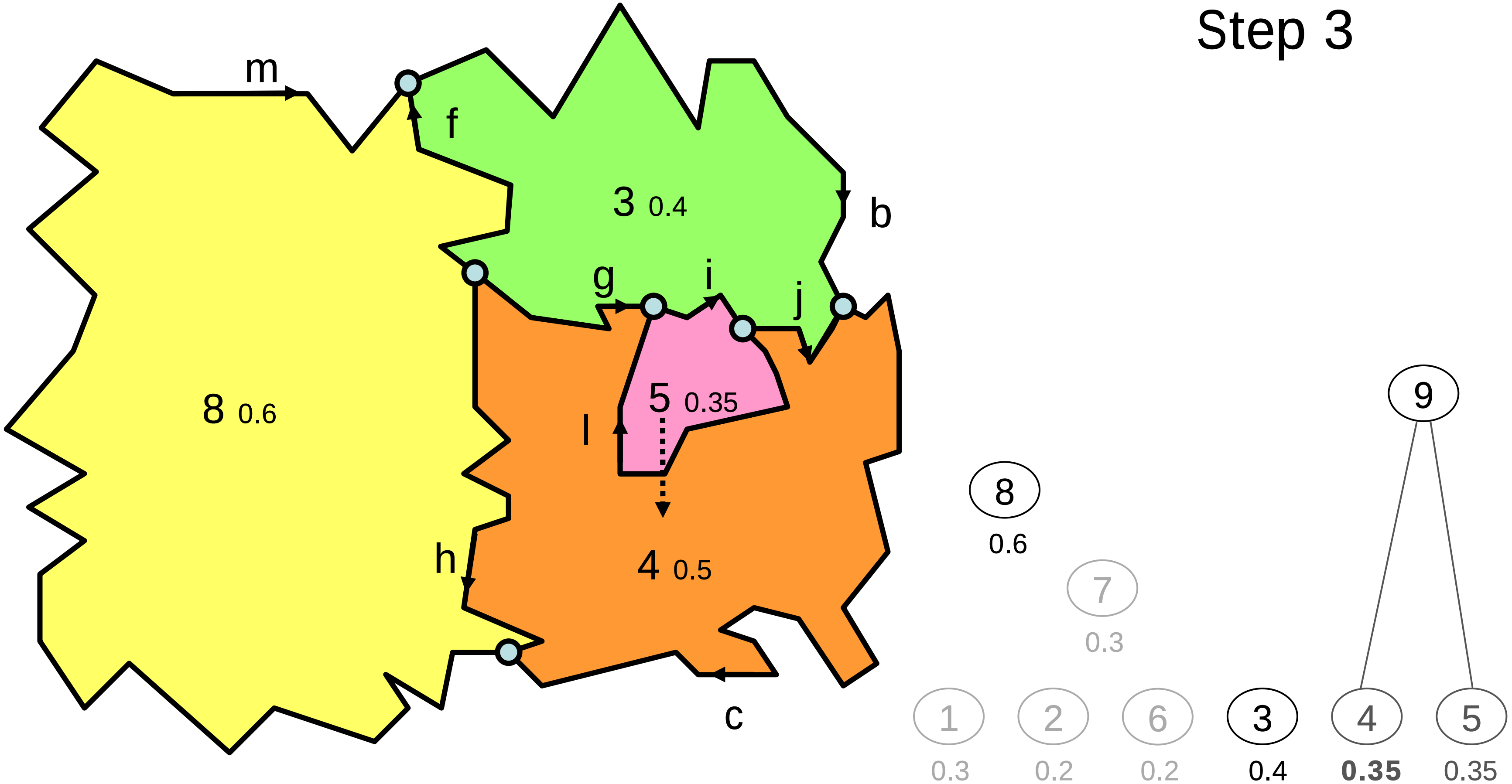
Step 3

$\text{Max}_v \{\text{Comp}(5, v)\} = \text{Comp}(5, \mathbf{4})$

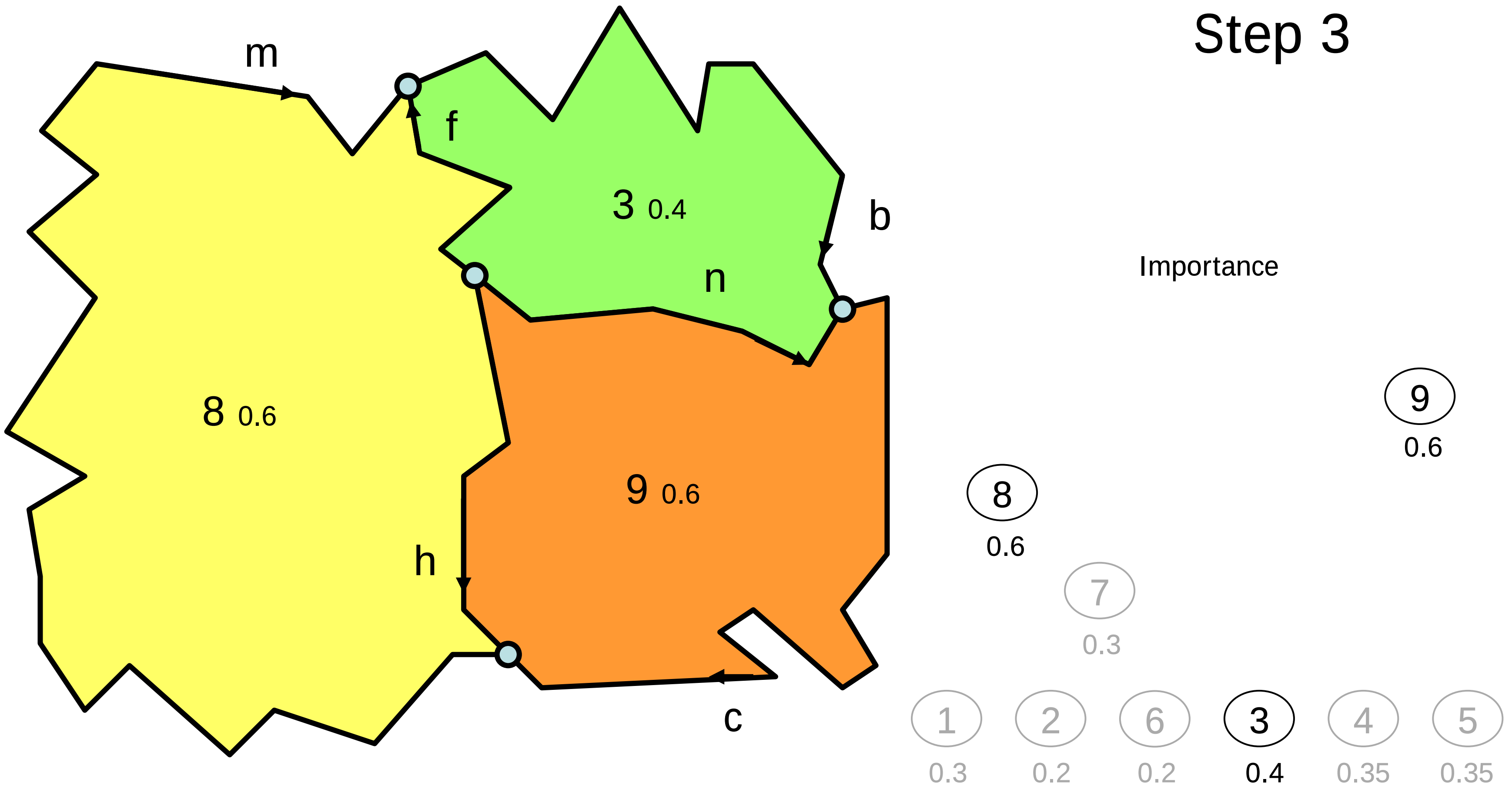


2. Constructing tGAP face tree

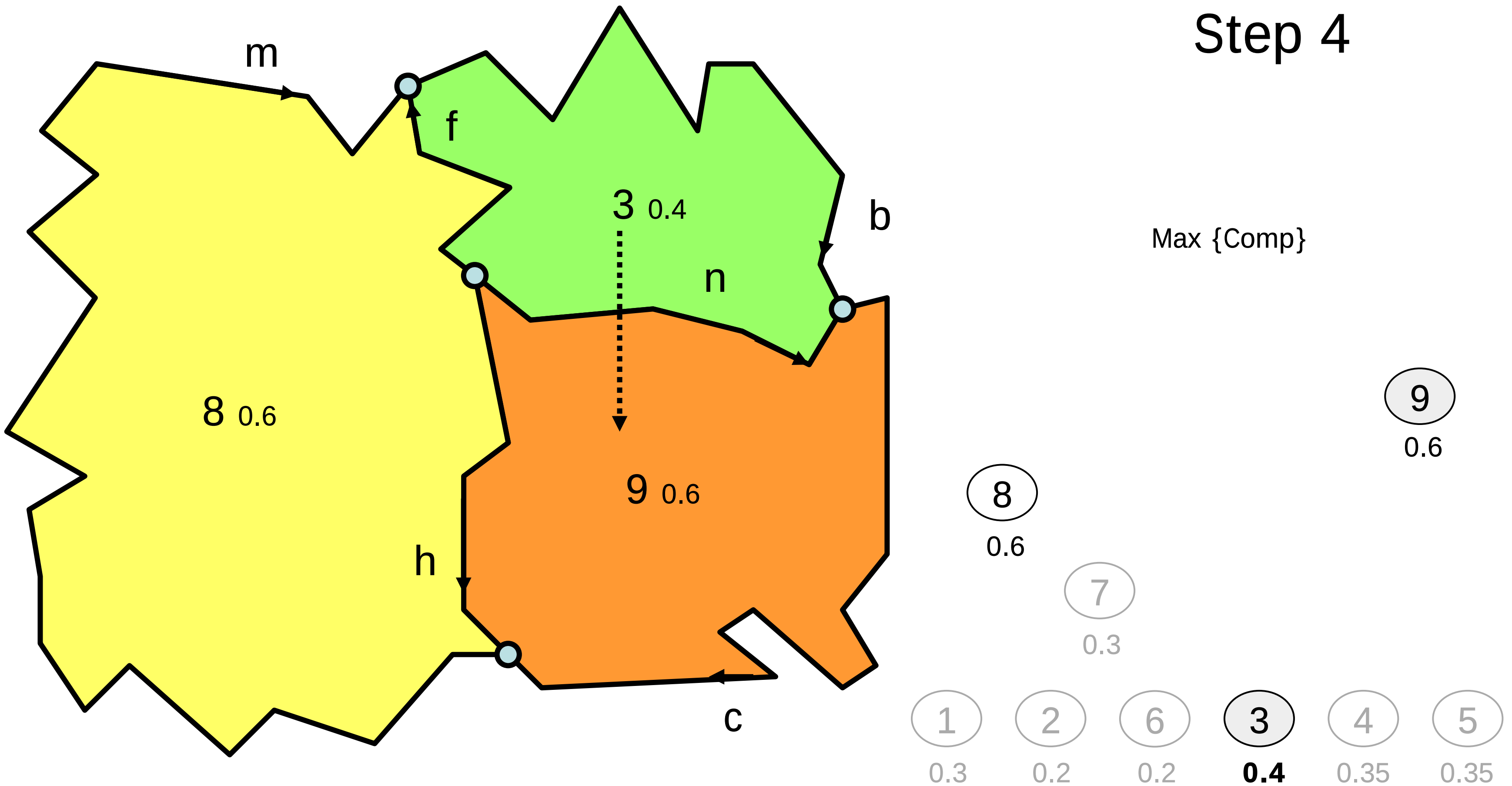
Step 3



2. Constructing tGAP face tree

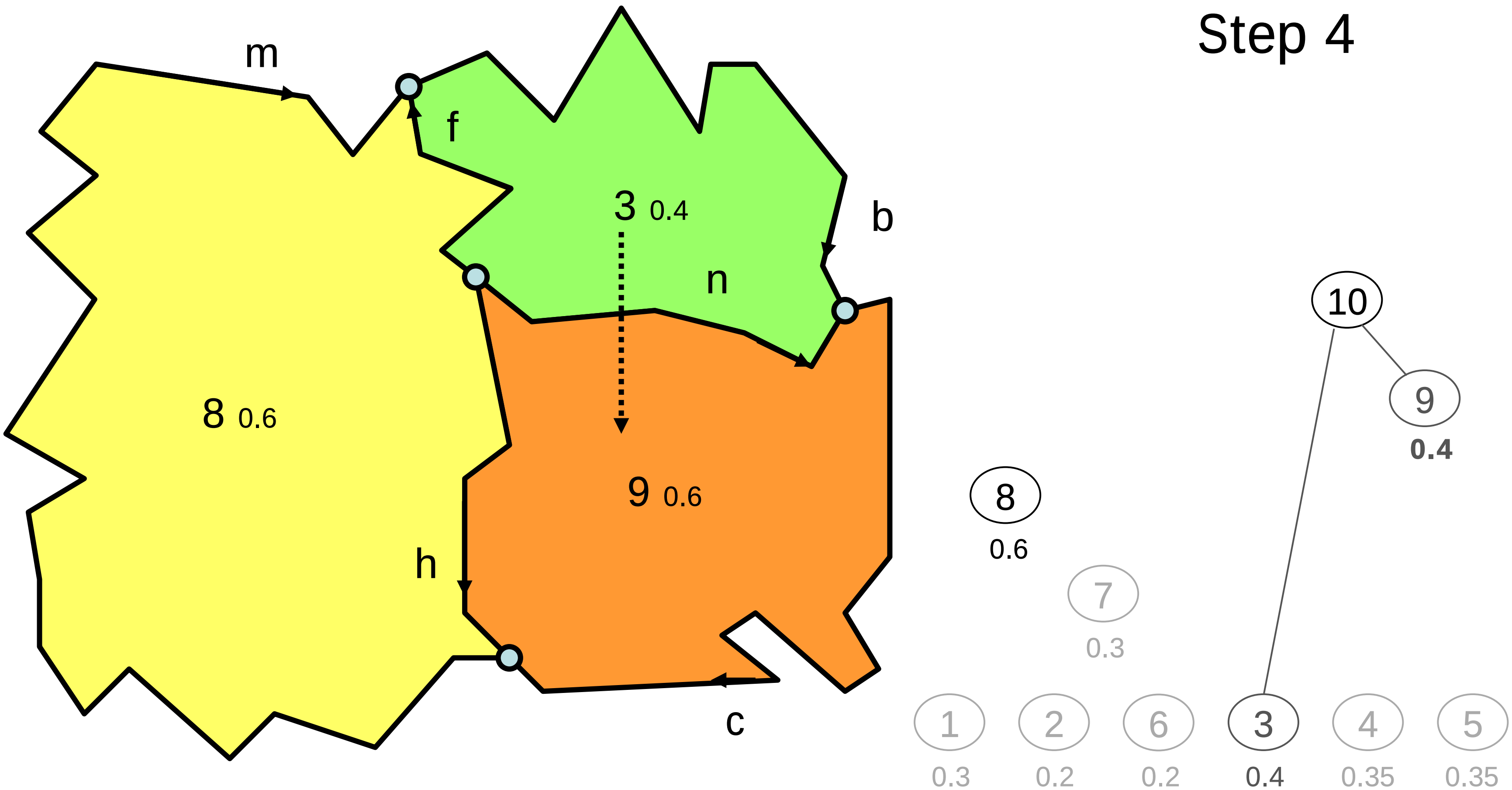


2. Constructing tGAP face tree



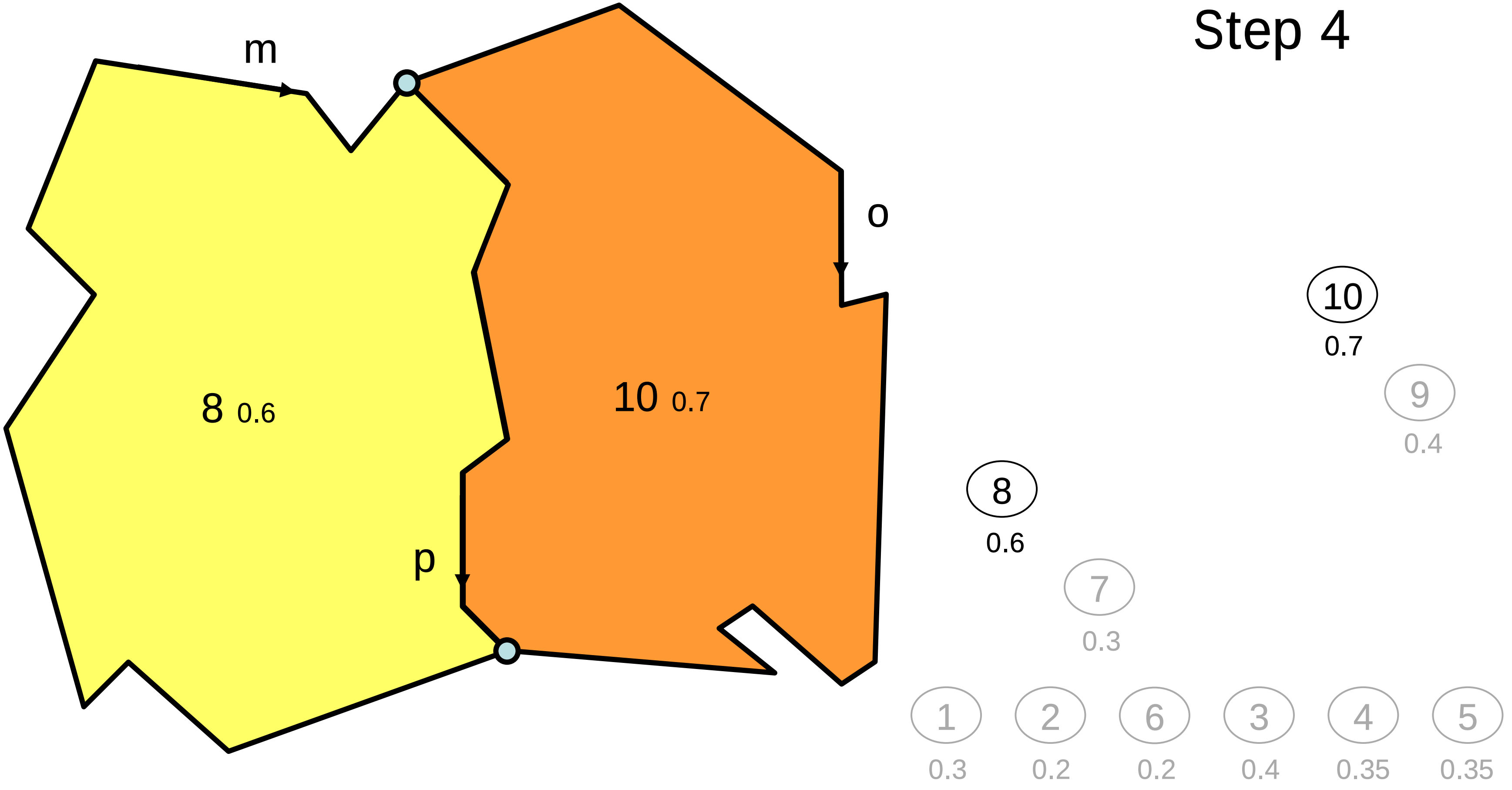
TU Delft

2. Constructing tGAP face tree



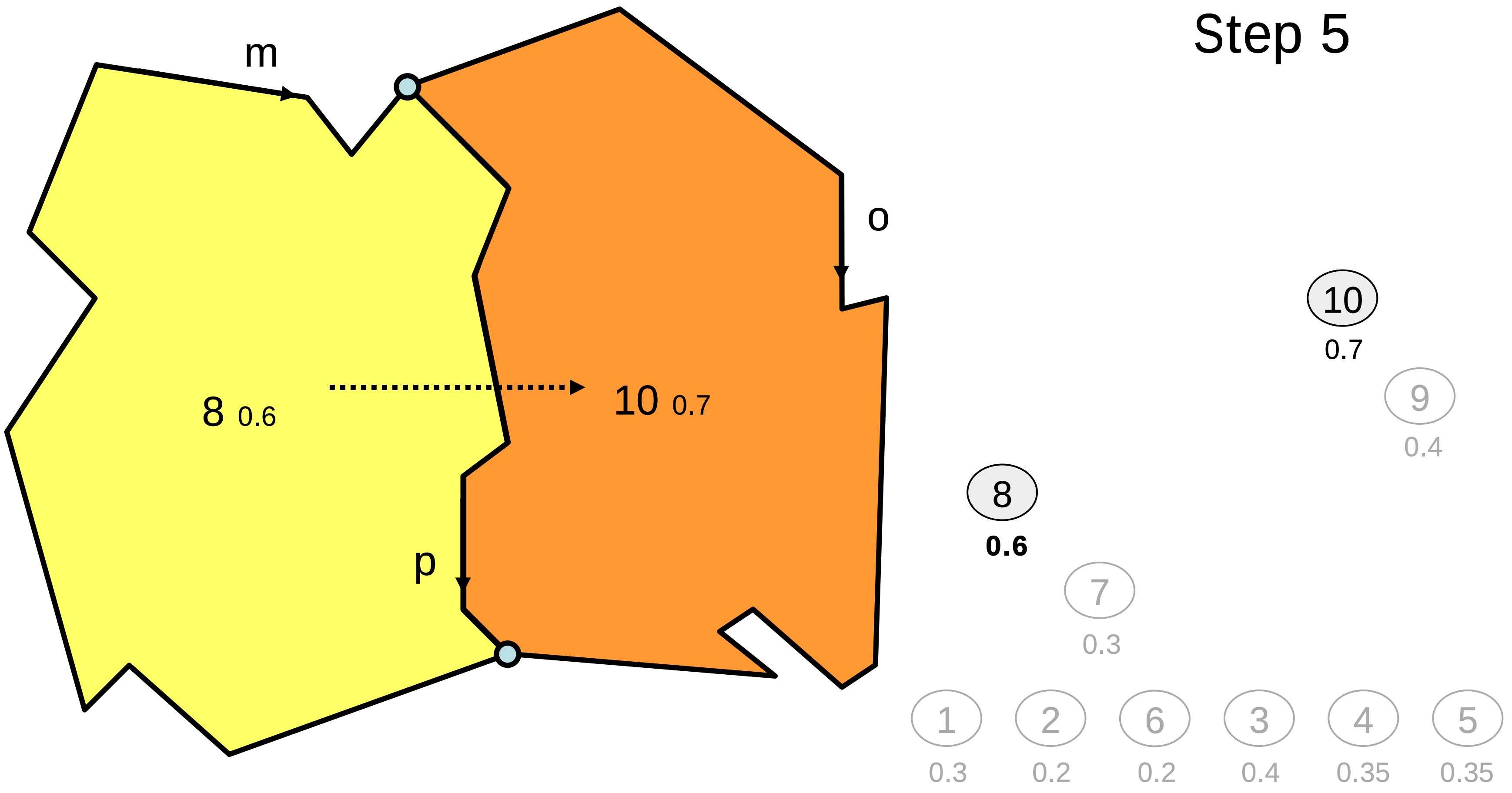
TU Delft

2. Constructing tGAP face tree



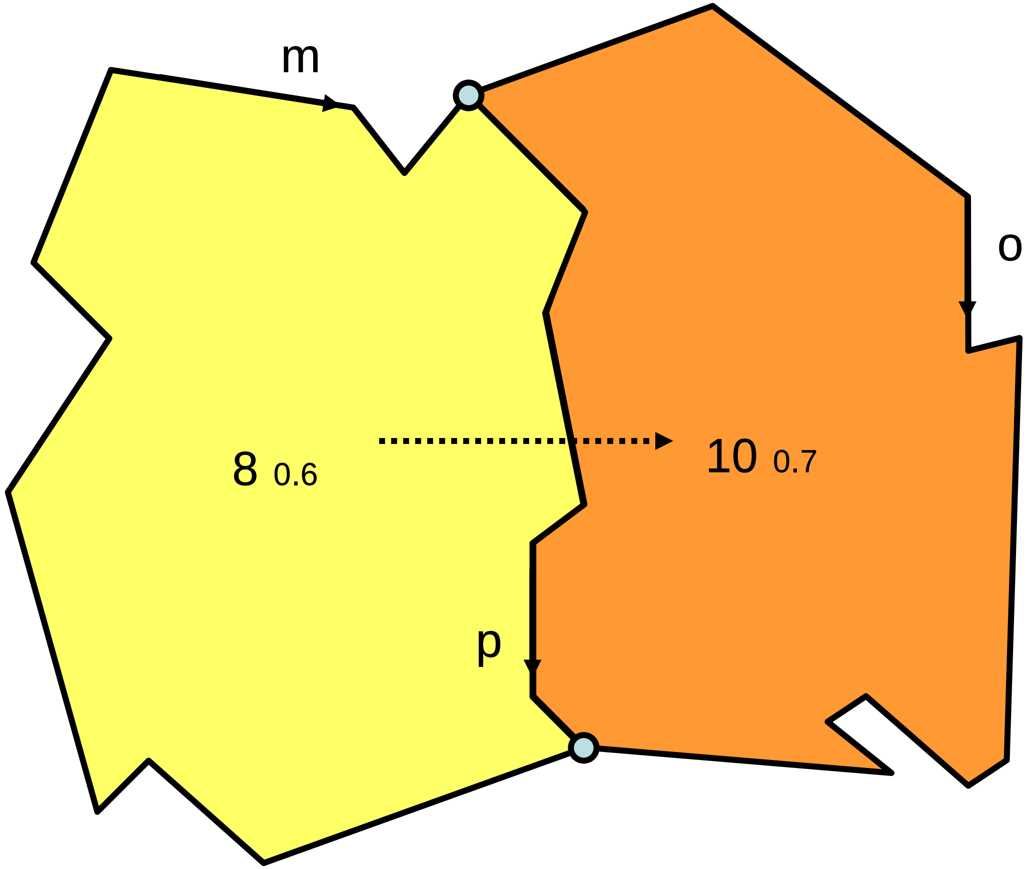
TU Delft

2. Constructing tGAP face tree

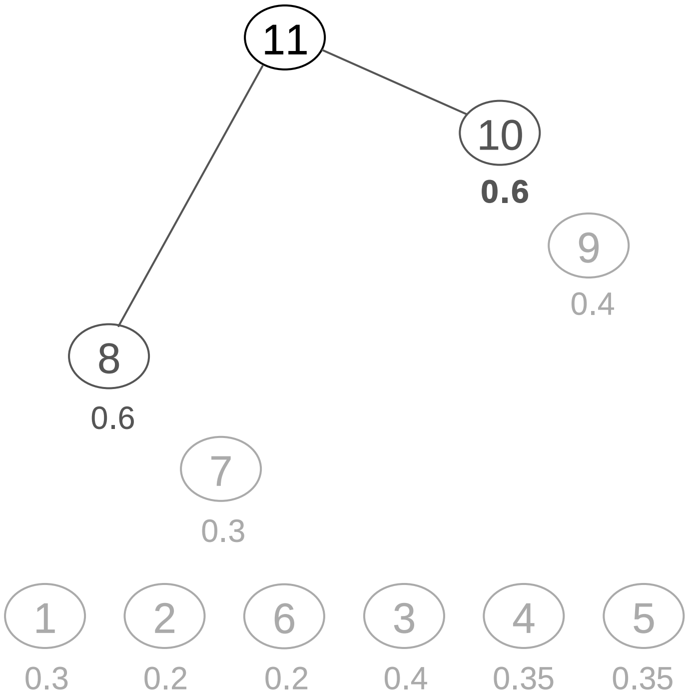


TU Delft

2. Constructing tGAP face tree



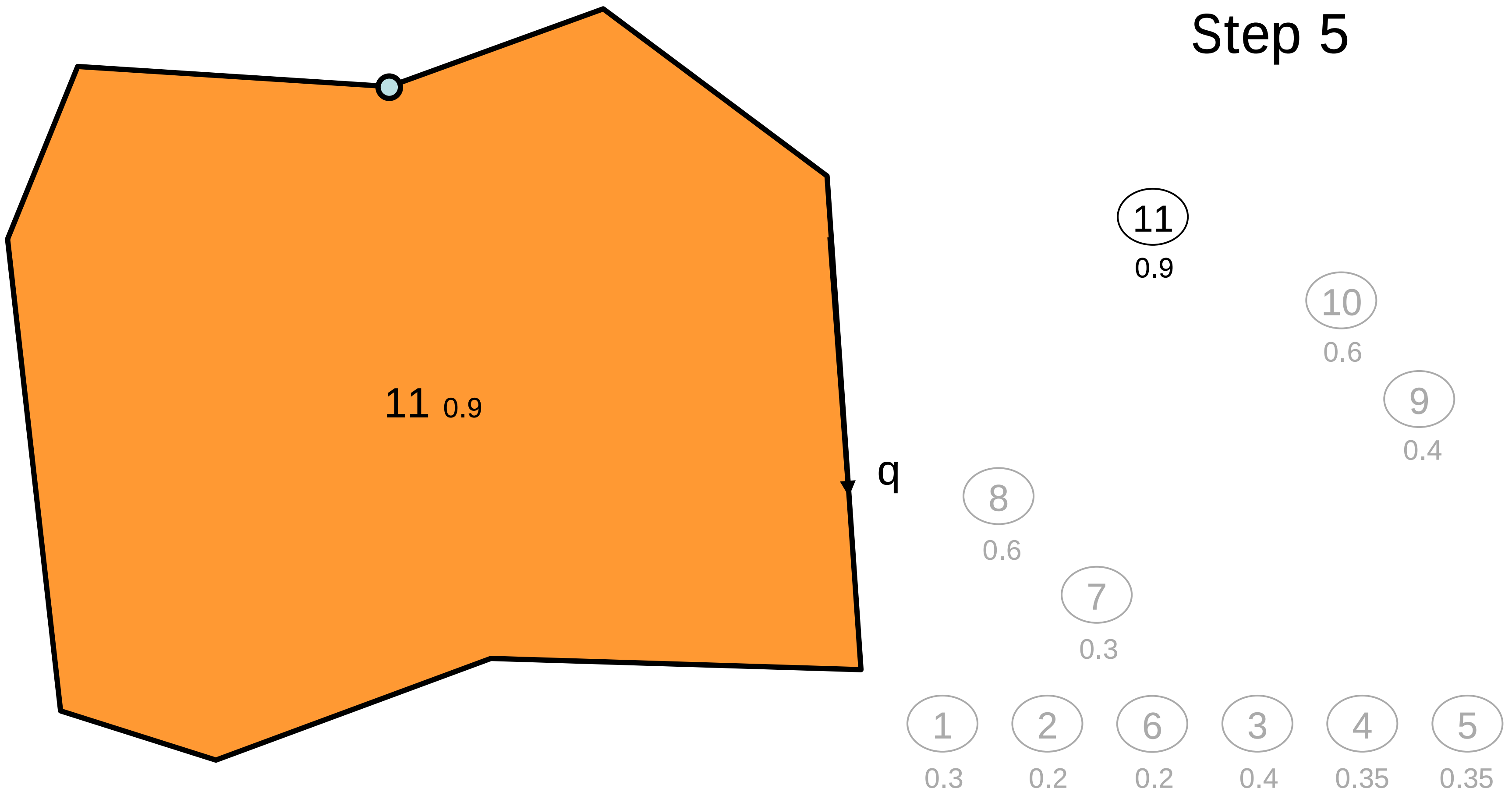
Step 5



TUDelft

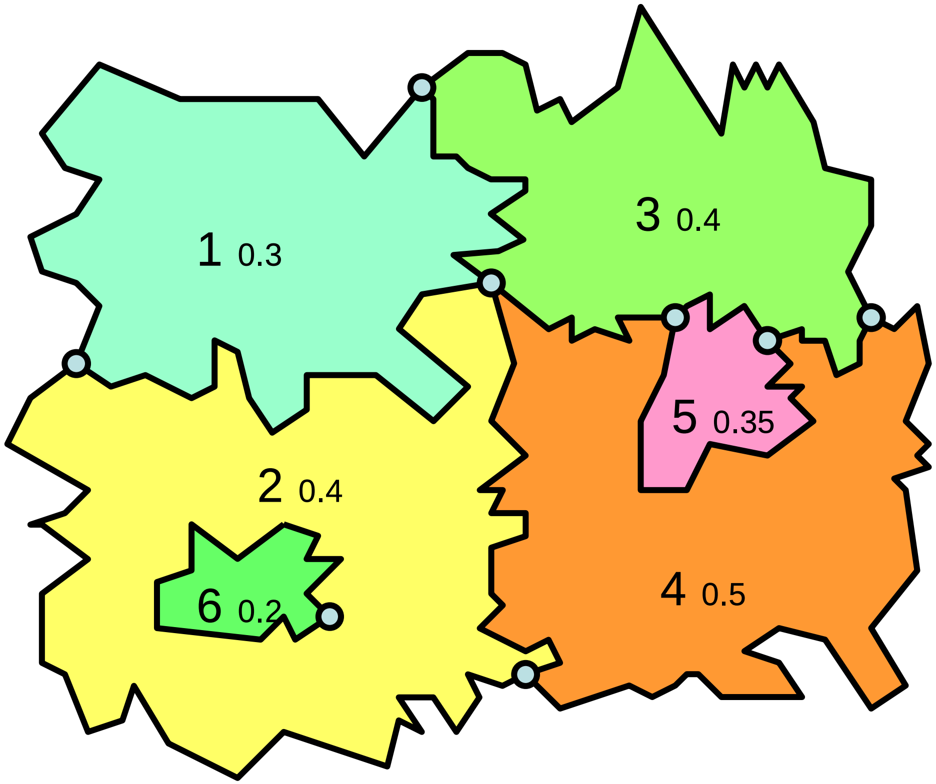
2. Constructing tGAP face tree

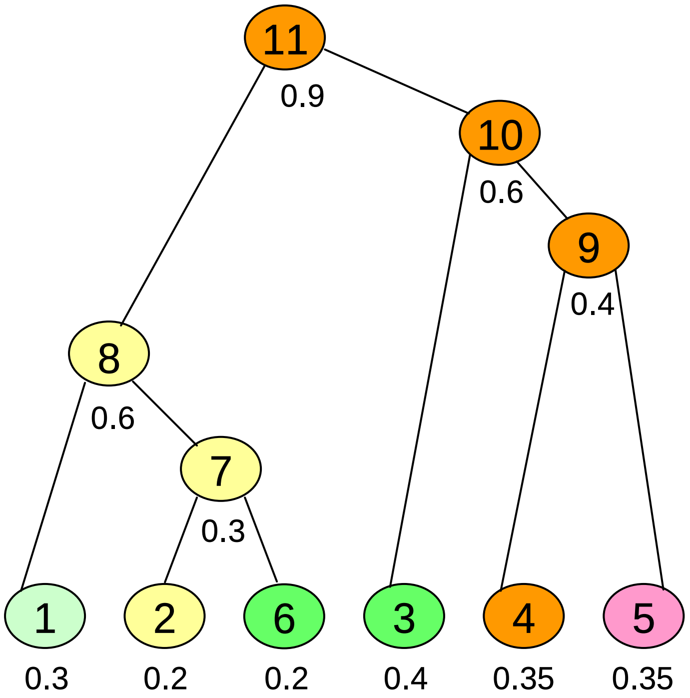
Step 5



TU Delft

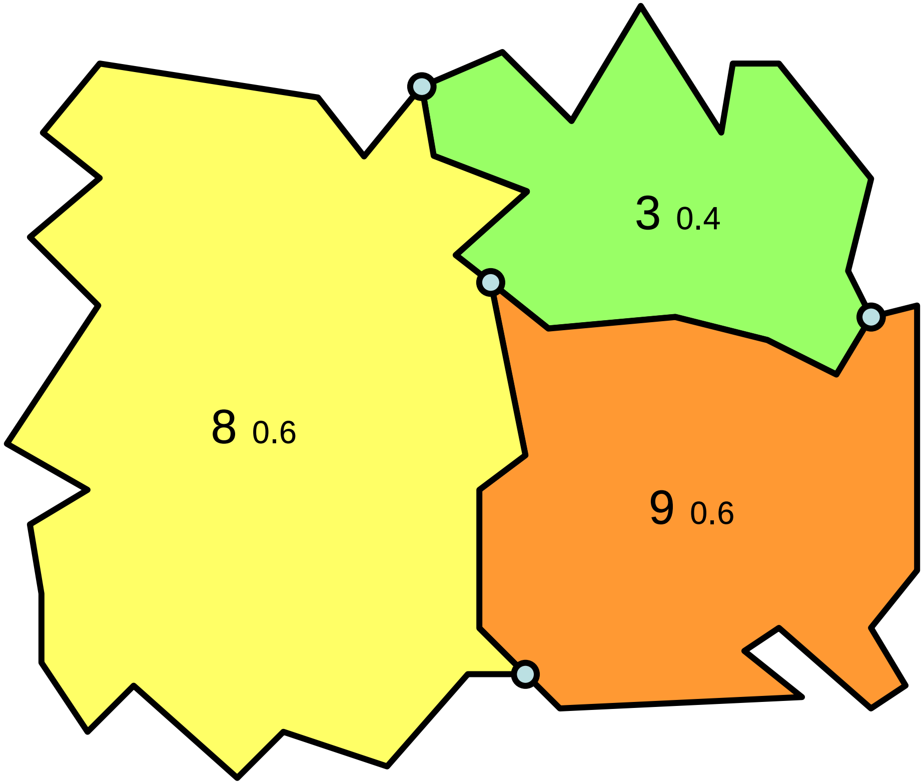
2. tGAP face tree



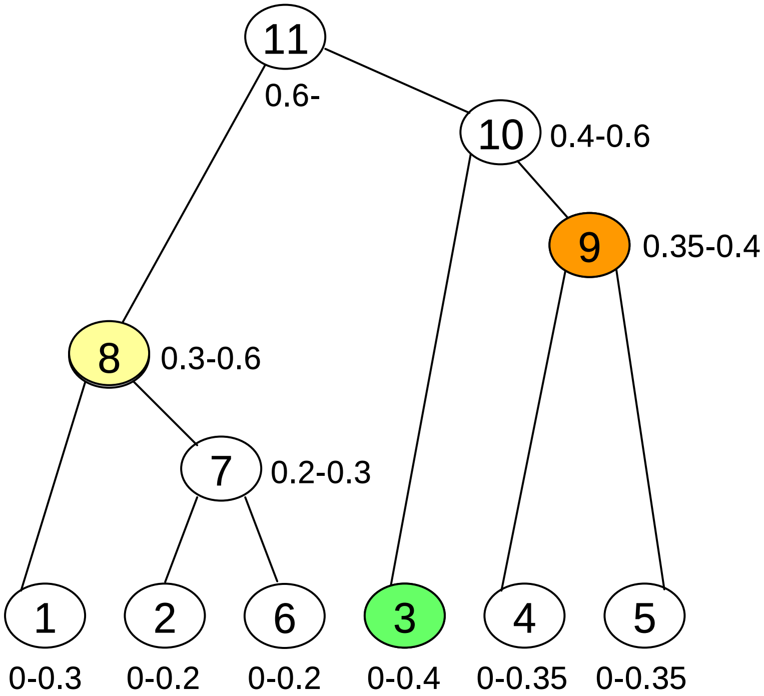


TU Delft

2. Using tGAP face tree

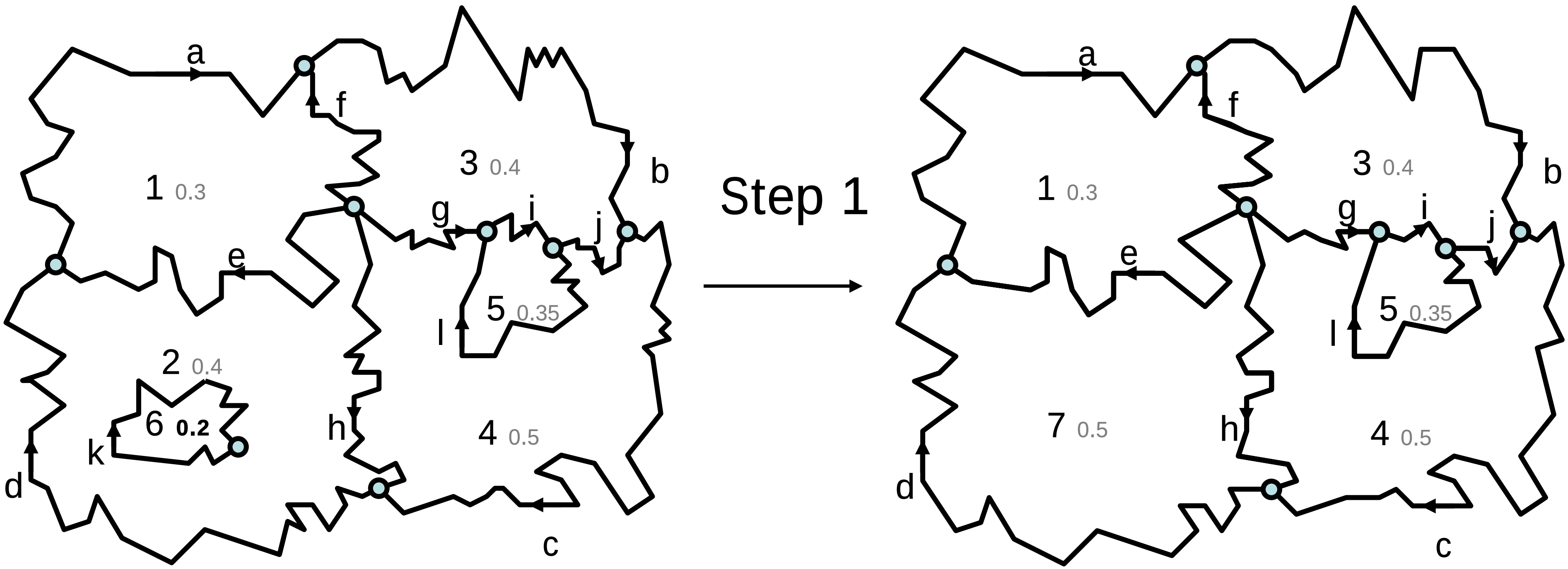


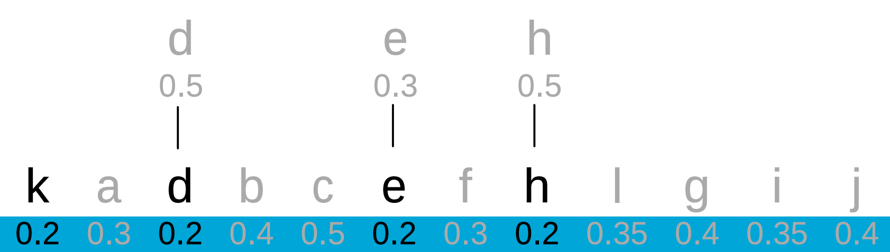
Selection of faces overlap search region & Importance = 0.38



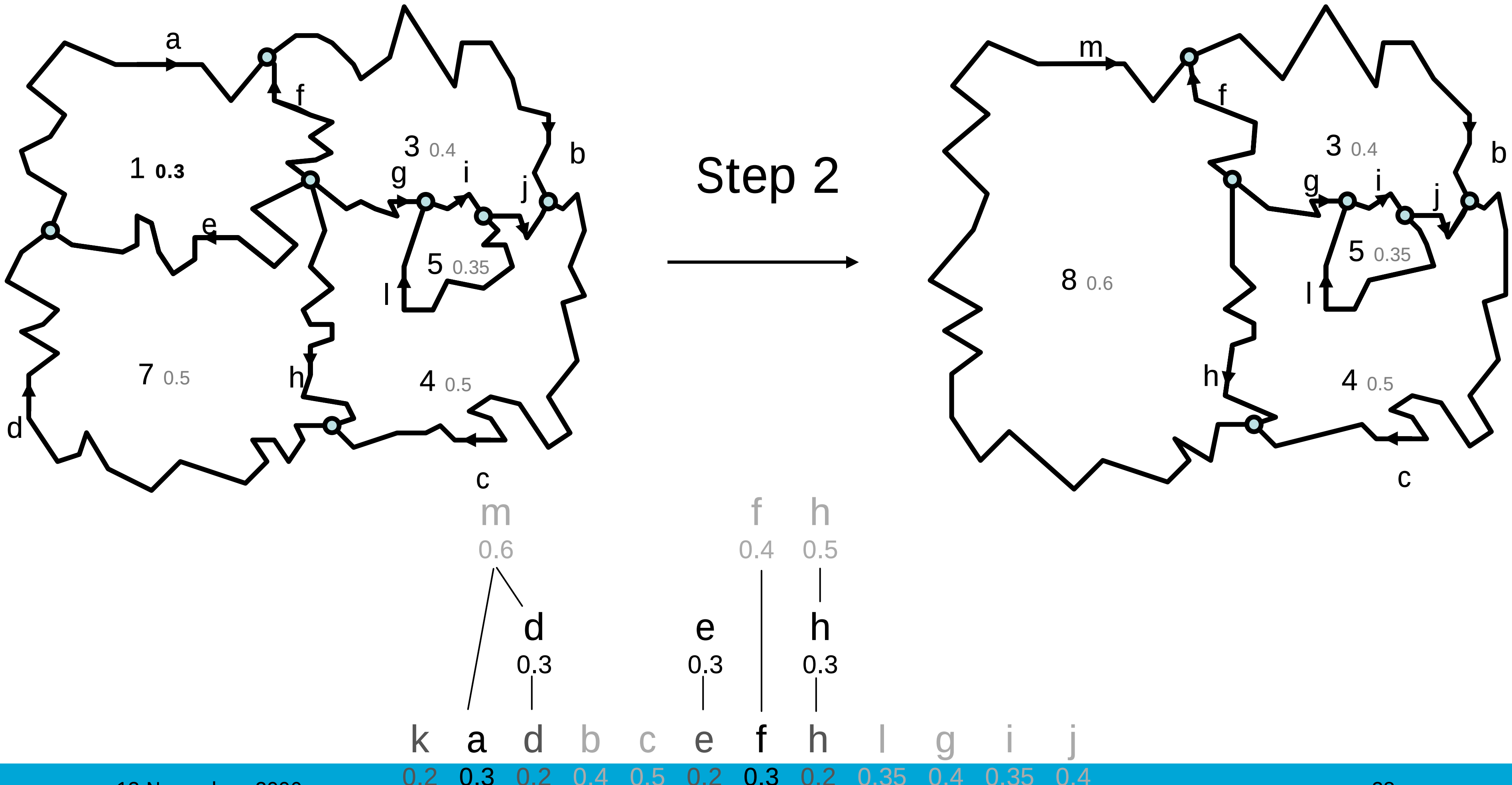
TU Delft

2. Constructing tGAP edge forest

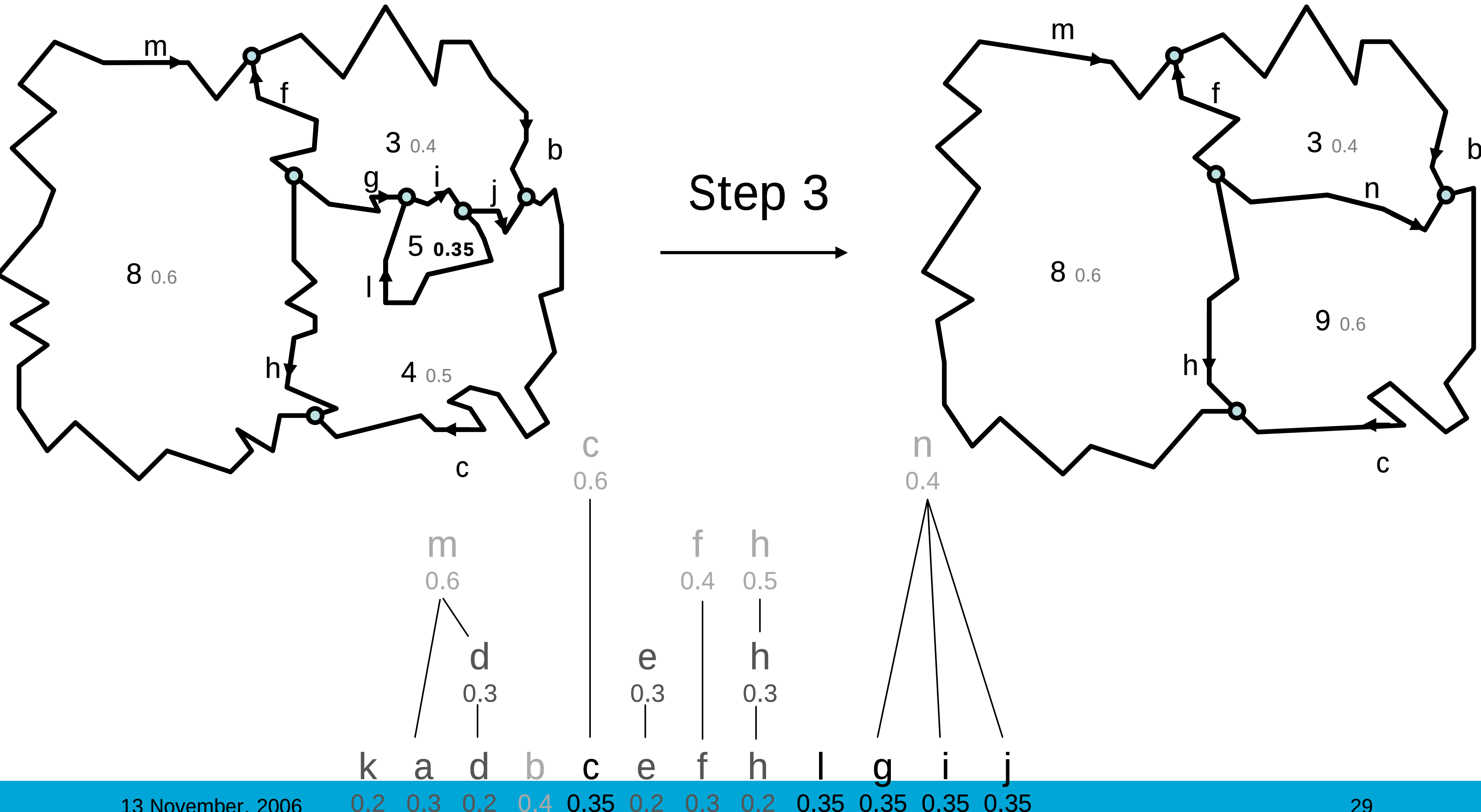




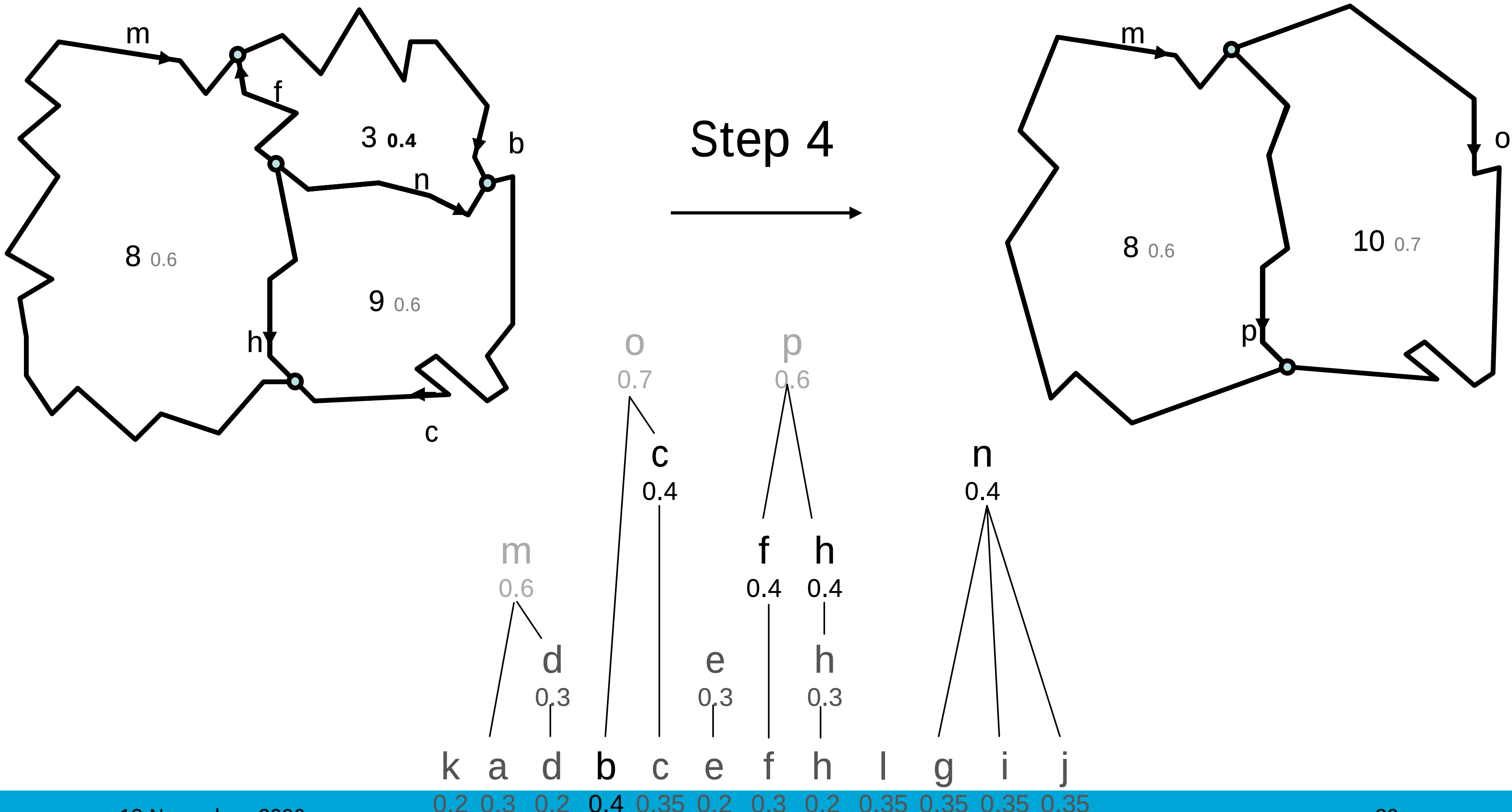
2. Constructing tGAP edge forest



2. Constructing tGAP edge forest

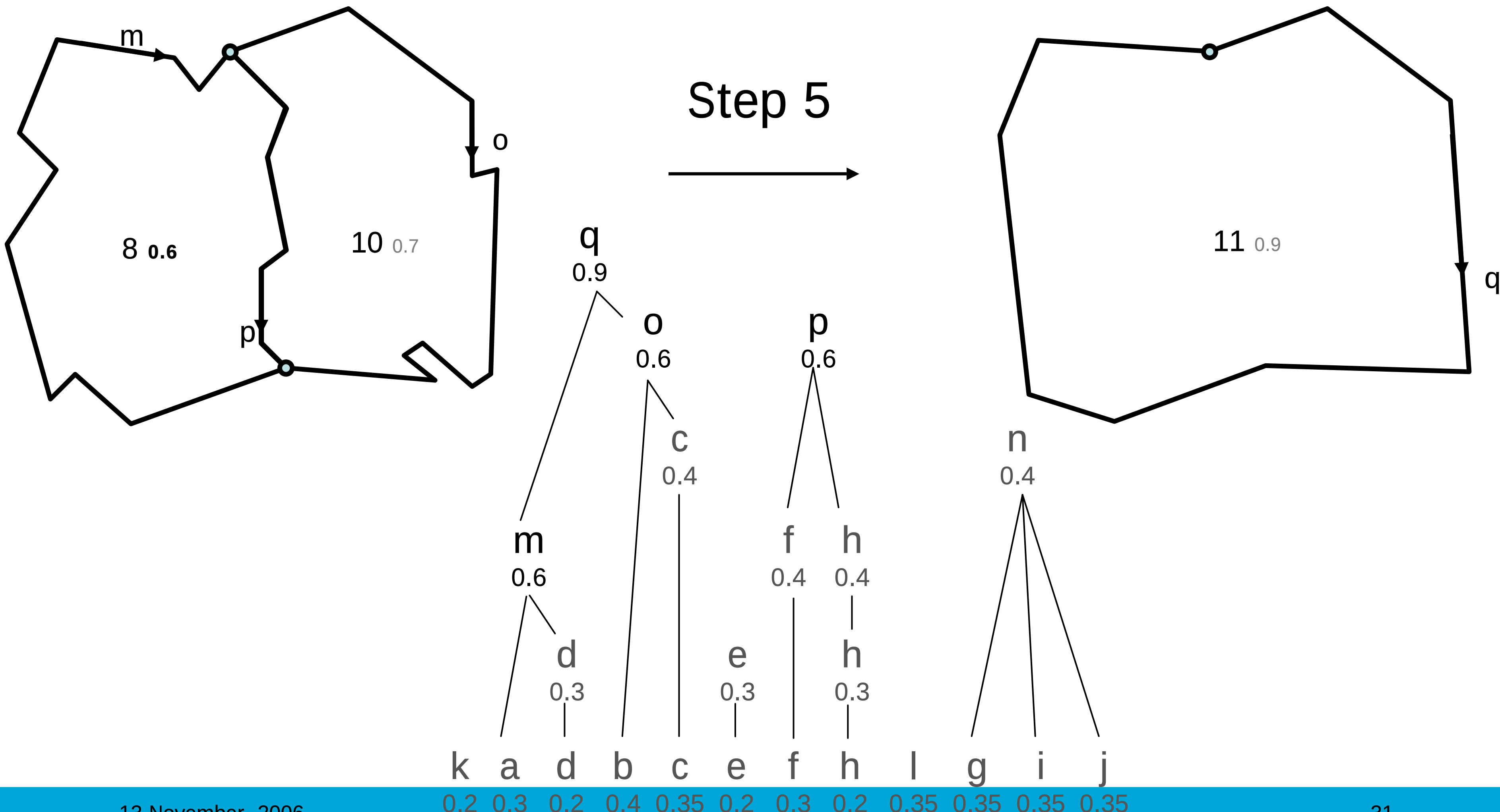


2. Constructing tGAP edge forest

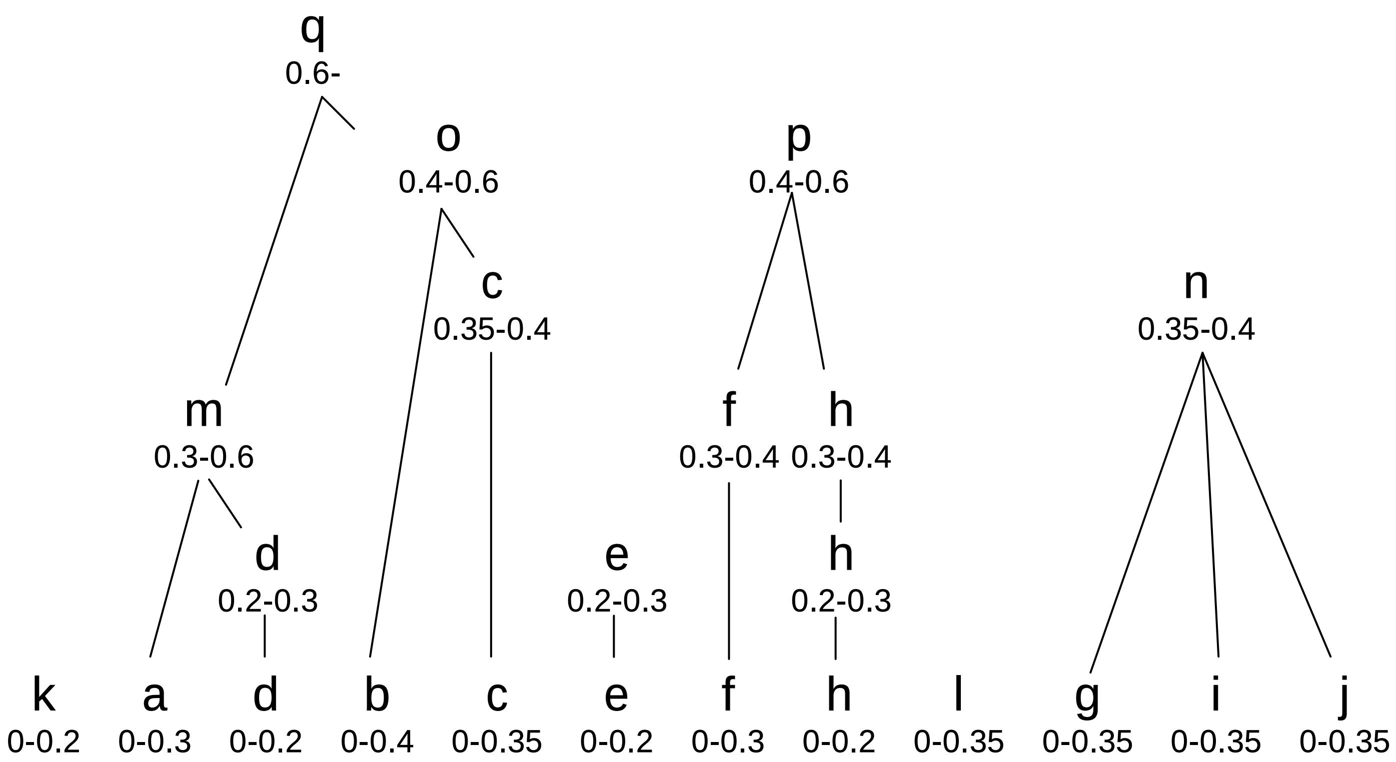


TU Delft

2. Constructing tGAP edge forest

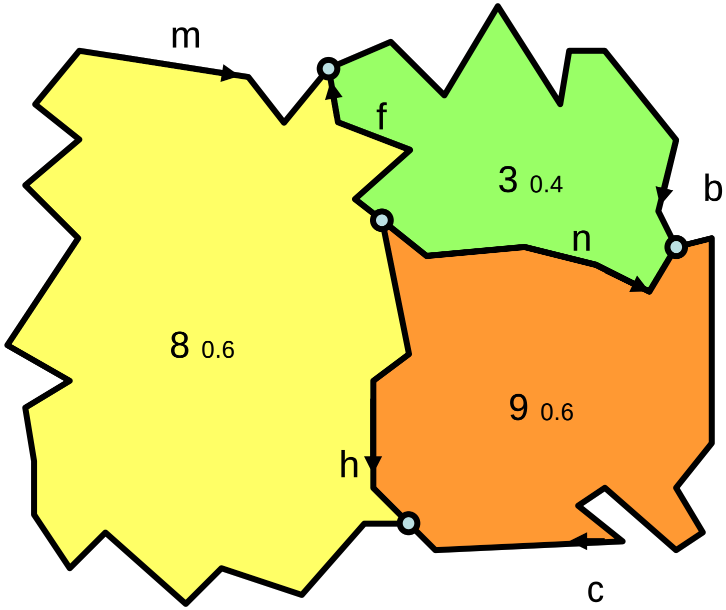


2. tGAP edge forest

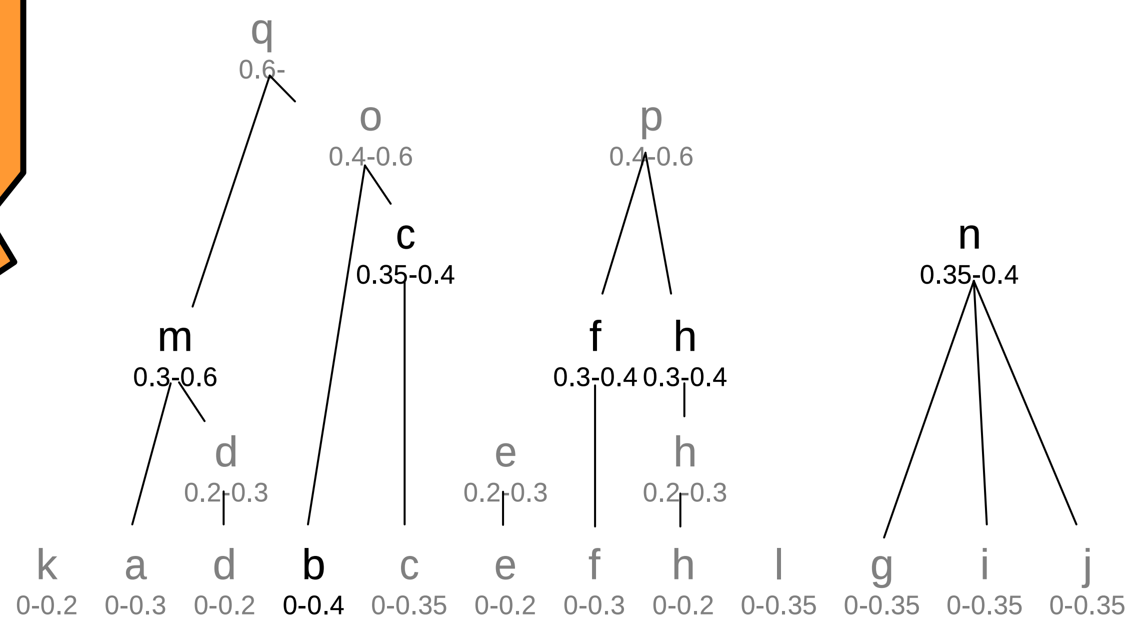


TU Delft

2. Using tGAP edge forest

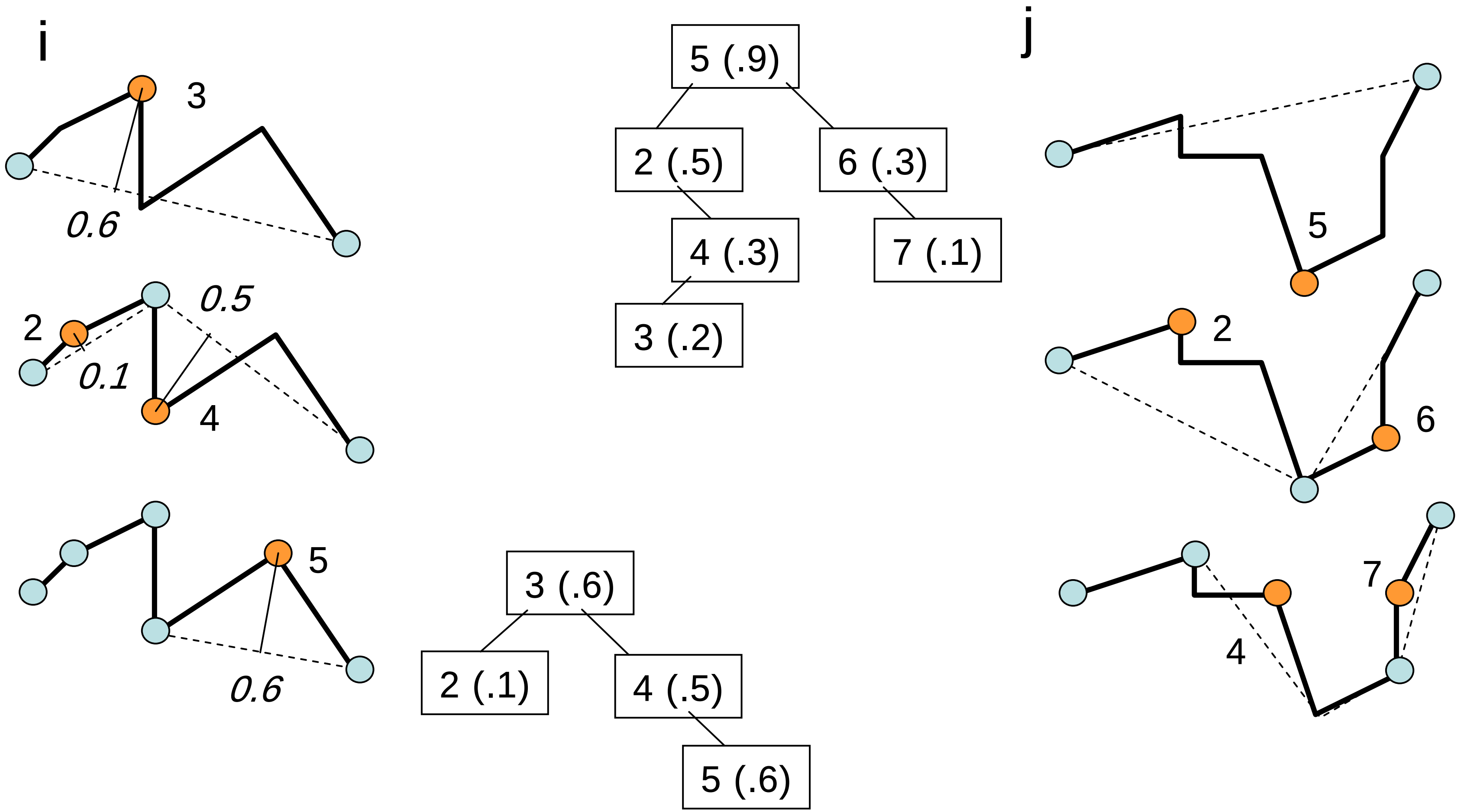


Selection of edges overlap search region & Importance = 0.38



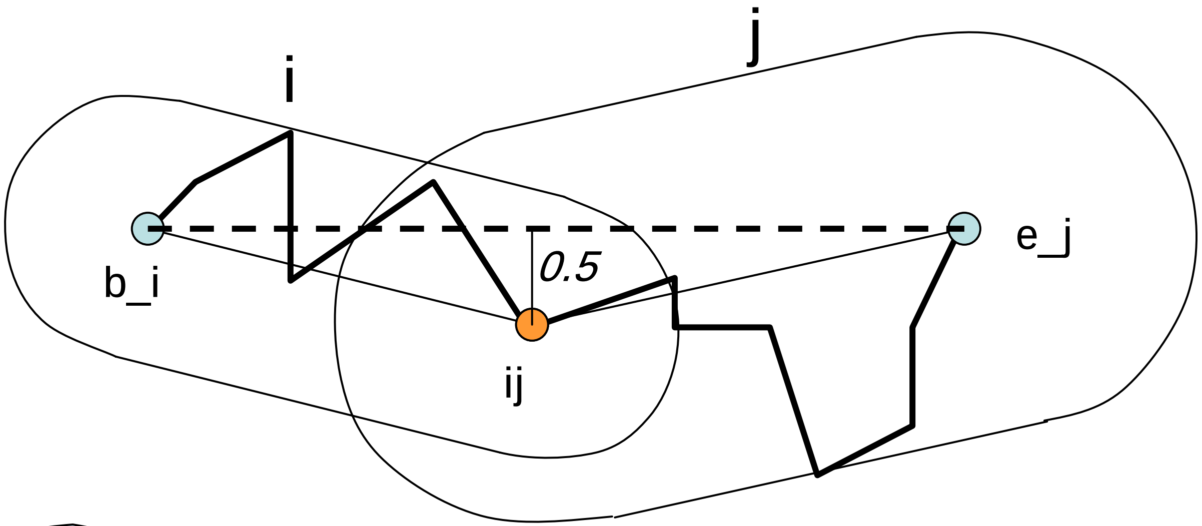
TUDelft

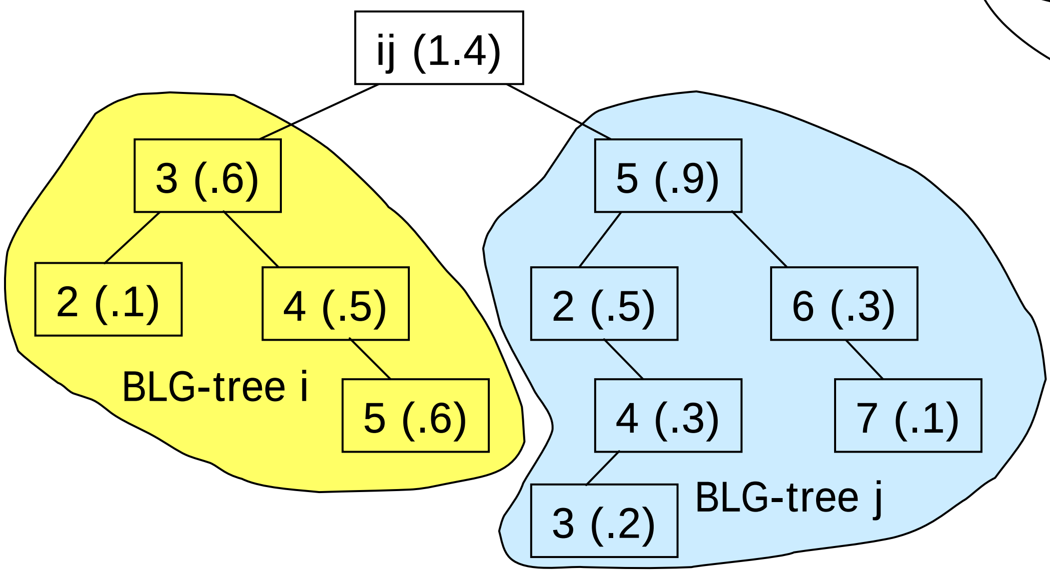
2. BLG edge trees



TU Delft

2. Joined BLG trees





err_ij=dist(point(ij), line(b_i,e_j))
+ max(err_i, err_j)=

0.5+0.9=1.4

Contents

1. Introduction
2. Background tGAP structure
3. First implementation
4. Client-server set-up and progressive refinement
5. Improvements
6. Conclusions

TU Delft

3. First implementation

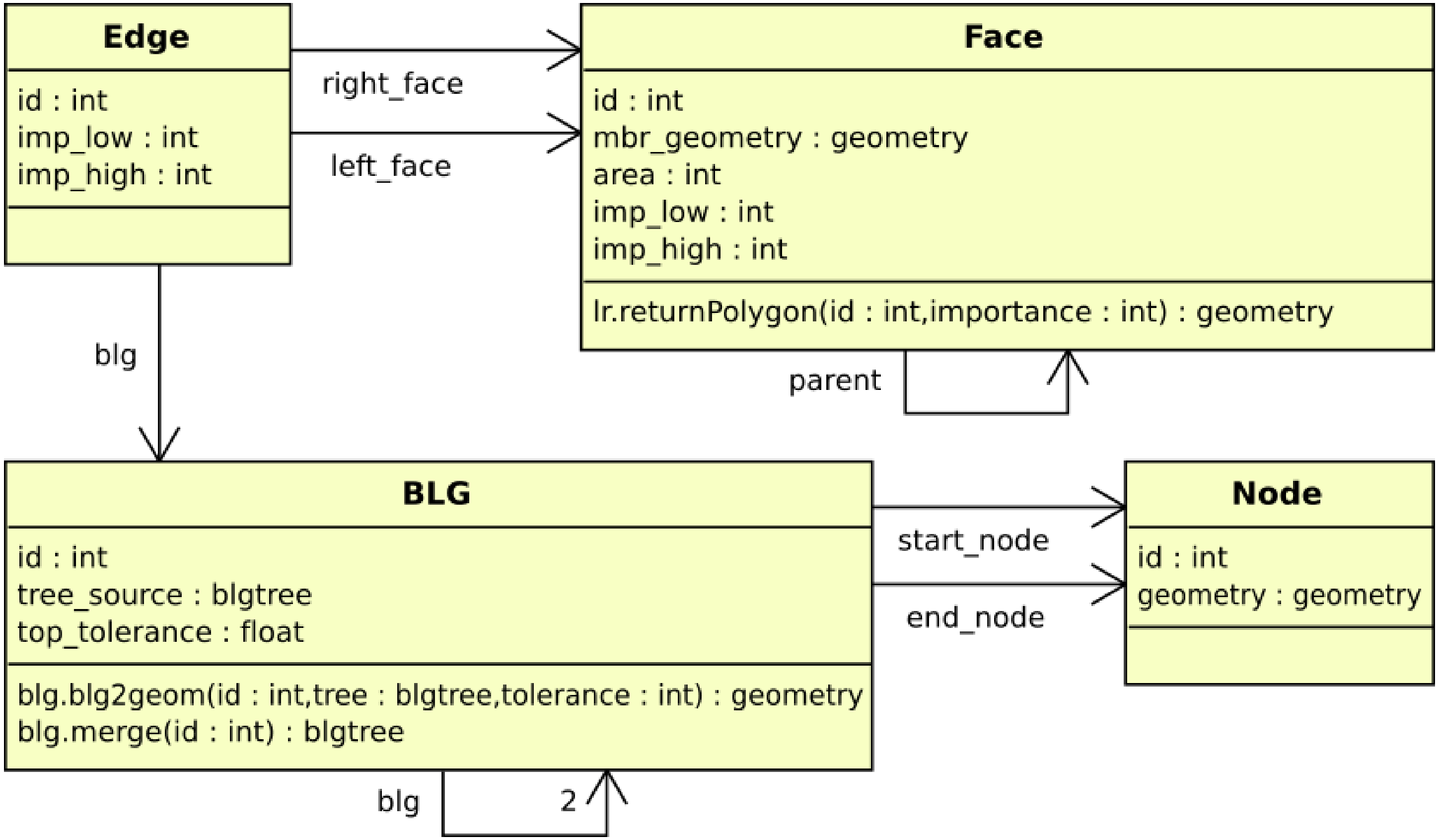
- Object-relational model
- Spatial data types available (incl. BLG-tree polyline)
- Tables for tgap_face, tgap_edge, tgap_blg, tagp_node
- Heavy use of views (and functions) to avoid redundant storage, but to provide 'easy access'
- Functional index (3D R-tree: 2D box+imp range)
- Oracle spatial

3: tGAP structure: combination of structures

- Uses topology
- Stores results of Generalization
- Suitable for Area Partitioning

GAP face tree	allow face selection
GAP edge forest	allow line selection
BLG tree	allow line simplification
3D R-tree	allow fast selection

3. UML class diagram tGAP structure



3. tGAP storage requirements

- Several test datasets (small/medium/large A'dam): cadastral and topographic data (1:1.000-1:10.000)
- Plain (base scale) polygon storage 82 Mb
- Lean topology (base scale storage) 107 Mb (fact 1.3, note that Oracle spatial topology requires fact 3.0)
- Current tGAP (vario scale storage) 491 Mb (fact 6.0)

	#face/ Mb	*#edge/ Mb*	*#blg/ Mb*	*#node/ Mb*	*Total Mb*
Basic topology	170.368/ 2	418.530/ 94	-/ 0	281.216/ 11	107
tGAP structure	340.735/ 56	7.113.680/ 291	658.219/ 133	281.216/ 11	491

3. tGAP storage improvements

- tgap_face: less attributes; area, mbr, perhaps parent..
- tgap_edge ‘explodes’: 17 times more than base edges, many versions of same edge (at different imp levels). However only few attributes change left, right, imp → all versions of edge in same record+varray’s for variable attributes
- Actually: the fact that the faces form a tree and the edges a forest is never used. Only the fact that the scale (imp) ranges of the different representation are forming a scale partition (no overlap, no gaps)
- Expected size mean and lean tGAP: factor 3 smaller

TU Delft

Contents

TUDelft

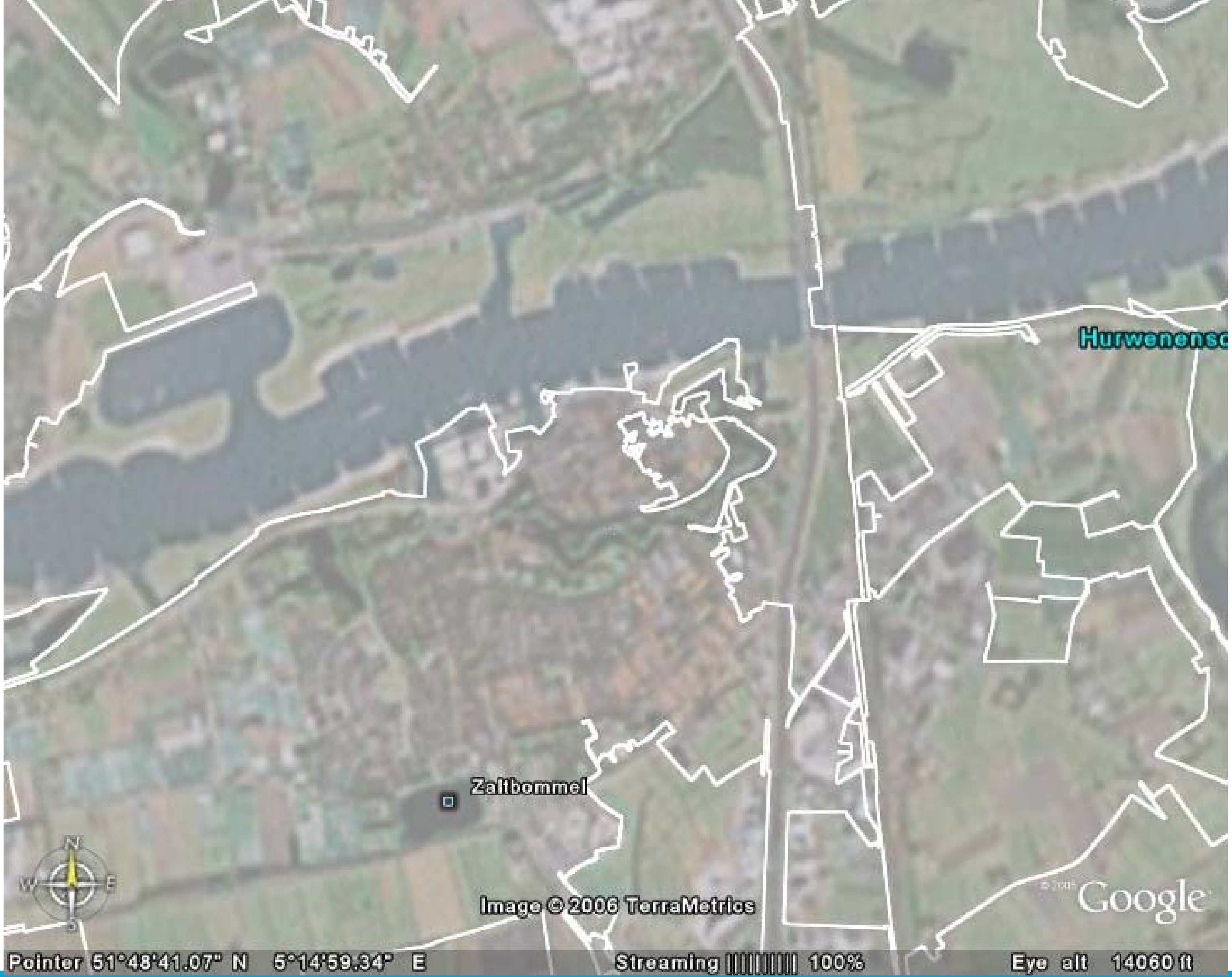
4. tGAP initial visualization: polygons at arbitrary scale in Google Earth

1. DBMS Server: Oracle spatial with tGAP as discussed → Polygons generated for arbitrary importance and tolerance (BLG-tree)
2. Middleware (Apache web server + Python, GDAL): coord transf, KML → Polygons transformed
3. Frontend: Google Earth → Polygons visualized

- Communication:
  - 2←→3: HTTP get/KML and
  - 1←→2: OCI (query, result set)

TU Delft


N
W
E
S
Image © 2006 GeoContent
Image © 2006 TerraMetrics
© 2005 Google
Pointer 51°48'40.85" N 5°15'16.88" E
Streaming |||||||||| 100%
Eye alt 54.37 mi


Hurwenensc
Zaltbommel
Image © 2006 TerraMetrics
Google
Pointer 51°48'41.07" N 5°14'59.34" E
Streaming |||||||||| 100%
Eye alt 14060 ft

TU Delft

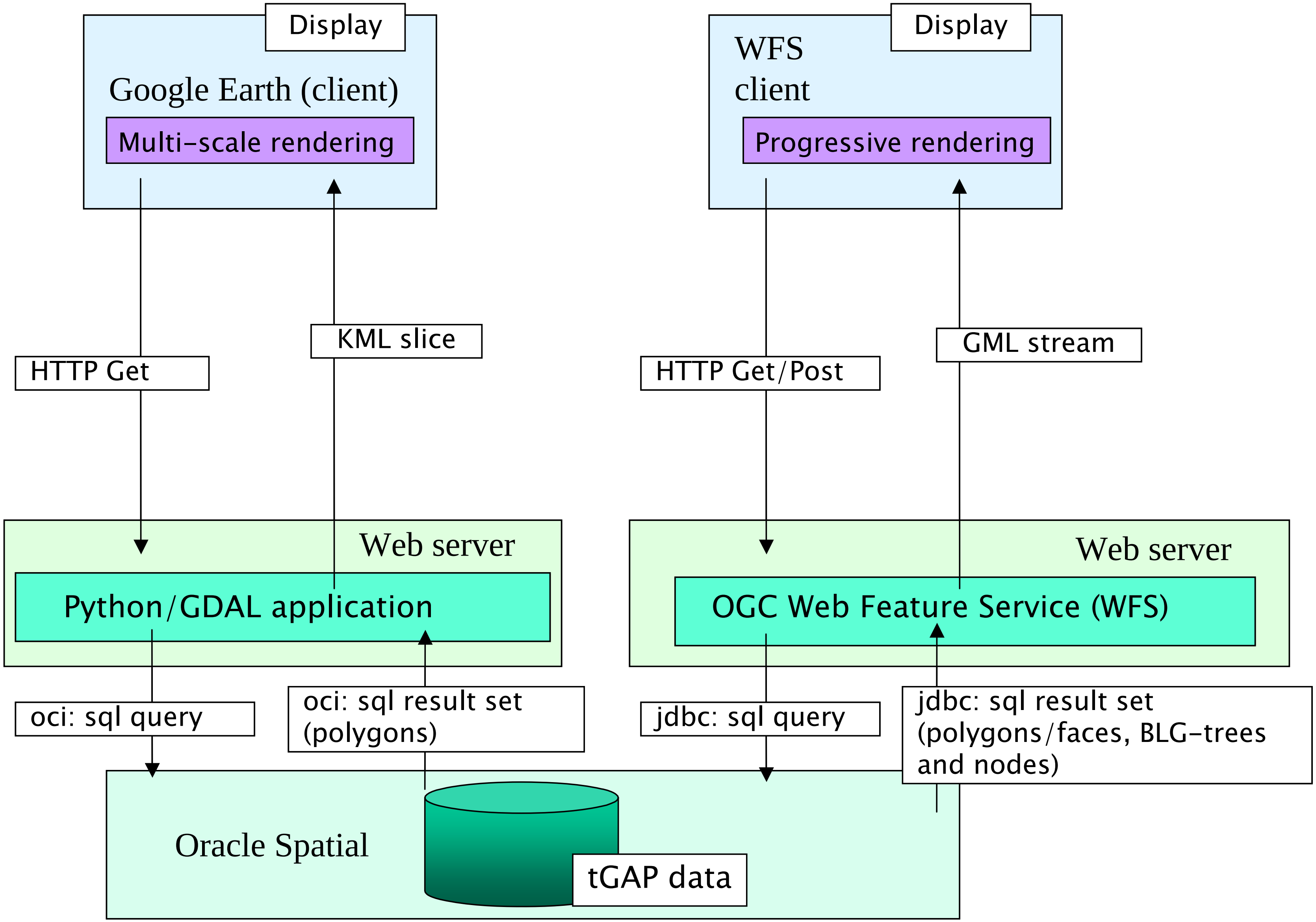

N
W
E
S
Image © 2006 TerraMetrics
© 2005 Google
Pointer 51°48'49.00" N 5°15'06.88" E
Streaming |||||||||| 100%
Eye alt 5483 ft


Image © 2006 TerraMetrics
© 2005 Google
Pointer 51°48'40.95" N 5°15'17.50" E
Streaming |||||||||| 100%
Eye alt 1478 ft

4. Polygons or structure?

- Current implementation has focus on server
- Client gets only polygons:
  1. No topology structure
  2. No progressive refinement
- Polygons are requested for every wanted scale (importance)
- Improvements for progressive transfer:
  1. Send importance range polygons (sorted) → smart client
  2. Send tGAP structure → needs smarter client (tGAP aware)

TU Delft


Display
Google Earth (client)
Multi-scale rendering
Display
WFS
client
Progressive rendering
HTTP Get
KML slice
HTTP Get/Post
GML stream
Web server
Python/GDAL application
Web server
OGC Web Feature Service (WFS)
oci: sql query
oci: sql result set
(polygons)
jdbc: sql query
jdbc: sql result set
(polygons/faces, BLG-trees
and nodes)
Oracle Spatial
tGAP data

TU Delft

4. Streaming of importance range (and first compared with a cut)

- A cut (or slice) of single importance

```
select face_id as id, '101' as impLevel,
  RETURN_POLYGON(face_id, 101) as geom
from tgap_face
where imp_low <= 101 and 101 < imp_high;
```

- A ordered range of importance values

```
select face_id as id, imp_high-1 as impLevel,
  imp_low, imp_high,
  RETURN_POLYGON(face_id,imp_high-1) as geom
from tgap_face
where imp_high > 90
order by imp_high desc;
```


TU Delft

4. Smart client for polygon range

- Alternatives:
  - Render step by step: start with most coarse polygon, then replace it by its two children. Repeat this step when receiving more detailed polygons
  - Collect polygons for a while and render at a number of larger steps (and morph between steps 'smooth zooming')
- The cached range (imp) of polygons can also be used at client side for (smart) zooming
- Note no topology used and also no line simplification

TU Delft

4. Smarter client: Progressive refinement based on tGAP structure

- Server starts sending most important nodes in GAP face-tree/edge-forest (in selected search rectangle)
- Client builds partial copy of GAP/BLG-structure
  → can be used to display coarse impression
  → every (x) seconds this structure is redisplayed
- Server keeps on sending more data and GAP/BLG-structure at client is growing (with more details)
- Possible stop criteria:
  1. 1000 objects (meaningful info density on screen)
  2. Required imp level is reached (with tolerance value)
  3. User interrupts the client

TU Delft

4. Extension to OGC/ISO WFS

- It is possible to specify imp range in Filter part of GetFeature request and using ogc:SortBy
- Not ideal because it is not clear that this is about scale, streaming, progressive transfer/refinement
- Deeper integration in WFS (called WFS-R):
  1. GetCapabilities should indicate if server supports progressive refinement
  2. Reporting of the min and max imp of a theme
  3. New request type GetFeatureByImportance

4. Example WFS-R request

```
<wfs:GetFeatureByImportance service="WFS"
 version="1.0.0" outputFormat="GML2" ...>
 <wfs:Query typeName='tgap_face' minImp='5' maxImp='8'>
  <ogc:Filter>
   <ogc:BBOX>
    <ogc:PropertyName>geom</ogc:PropertyName>
    <gml:Box srsName="…epsg.xml#28992">
     <gml:coordinates>
      136931,416574 139382,418904
     </gml:coordinates>
    </gml:Box>
   </ogc:BBOX>
  </ogc:Filter>
  <ogc:SortBy>gdmc:imp_high D</ogc:SortBy>
 </wfs:Query>
</wfs:GetFeatureByImportance>
```


TU Delft

Contents

1. Introduction
2. Background tGAP structure
3. First implementation
4. Client-server set-up and progressive refinement
5. Improvements
   - dynamic updates
   - non-area objects
   - other generalization operations
6. Conclusions

TU Delft

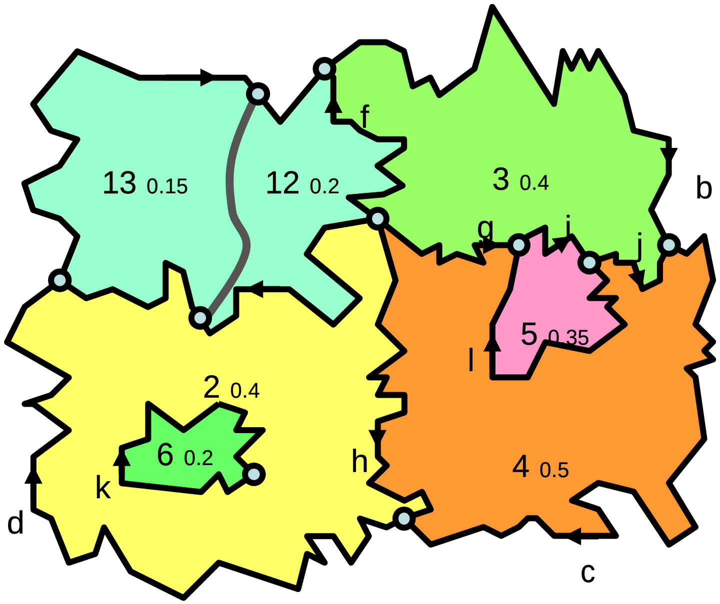
5. Improvements on theoretical aspects/dynamic updates

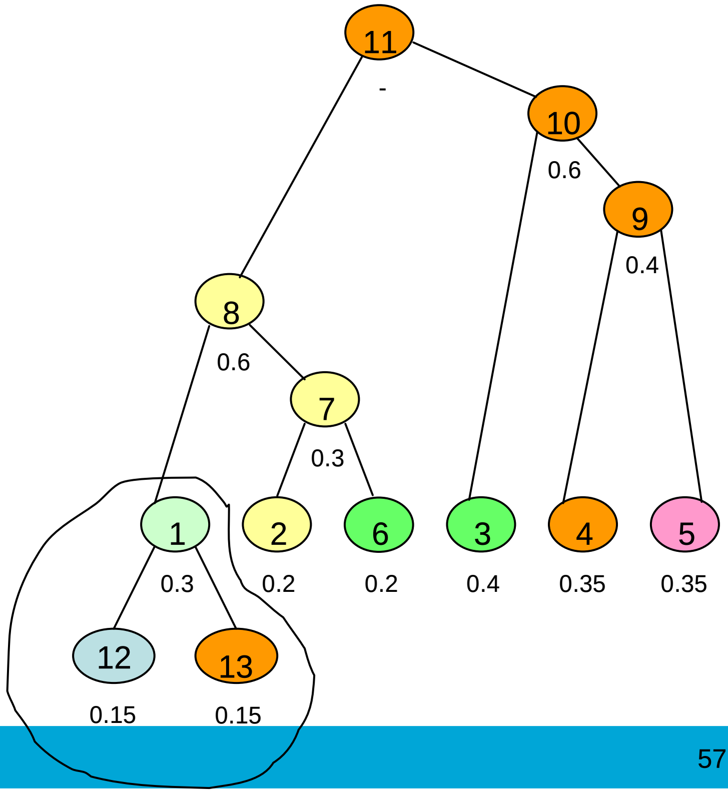
- More formal description (based on axioms, properties)
- Change the principle of creating the tGAP from a global minimum to some local minima
- Data editing (at most detailed level), local propagation to higher levels, dynamic structures
- Update the source data without rebuilding the tGAP structure: keep updates as local as possible (propagate up the ‘tree’/ scale-ranges above)

5. Updating tGAP (1)

Local update, control the propagation effect

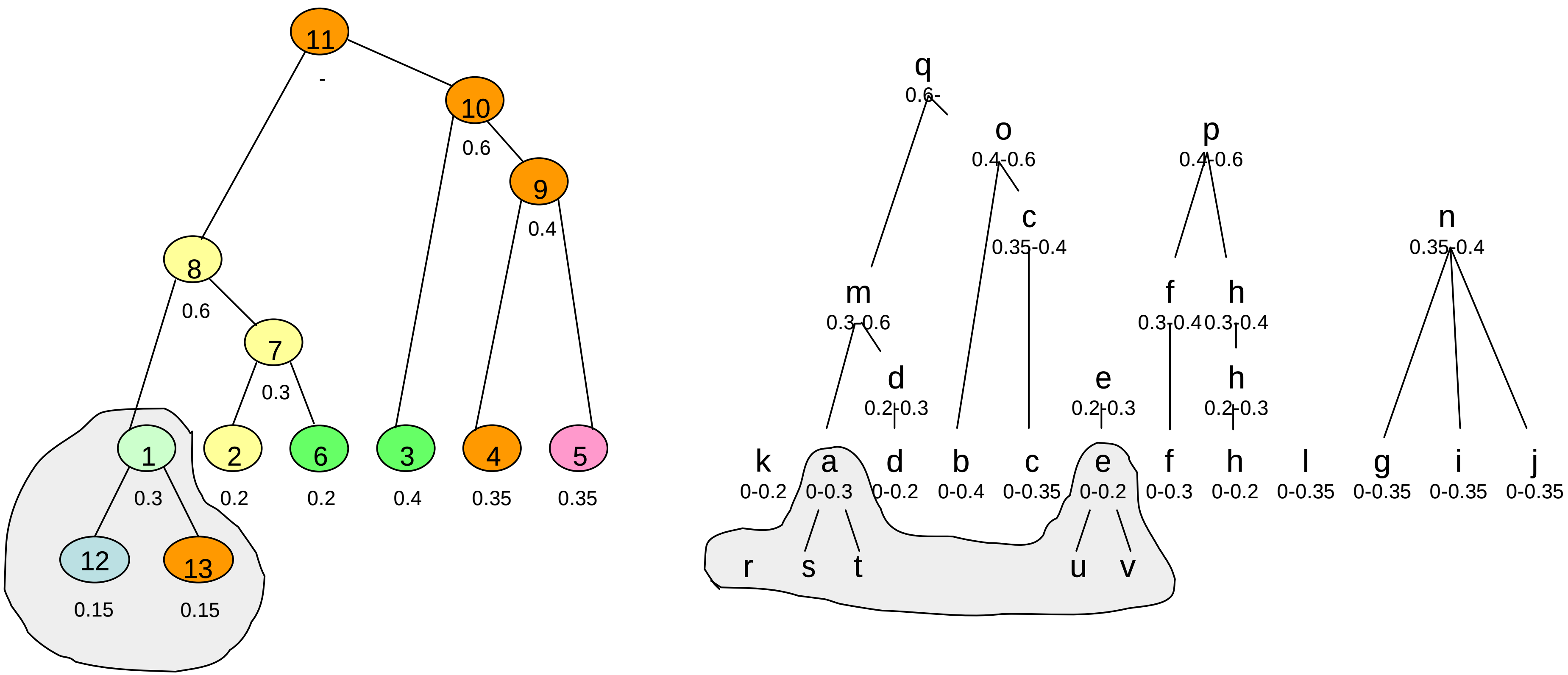
- Types of update: split, merge, boundary change,
- Effect: face tree (branch), edge forest (part), BLG trees





5. Updating tGAP (2)

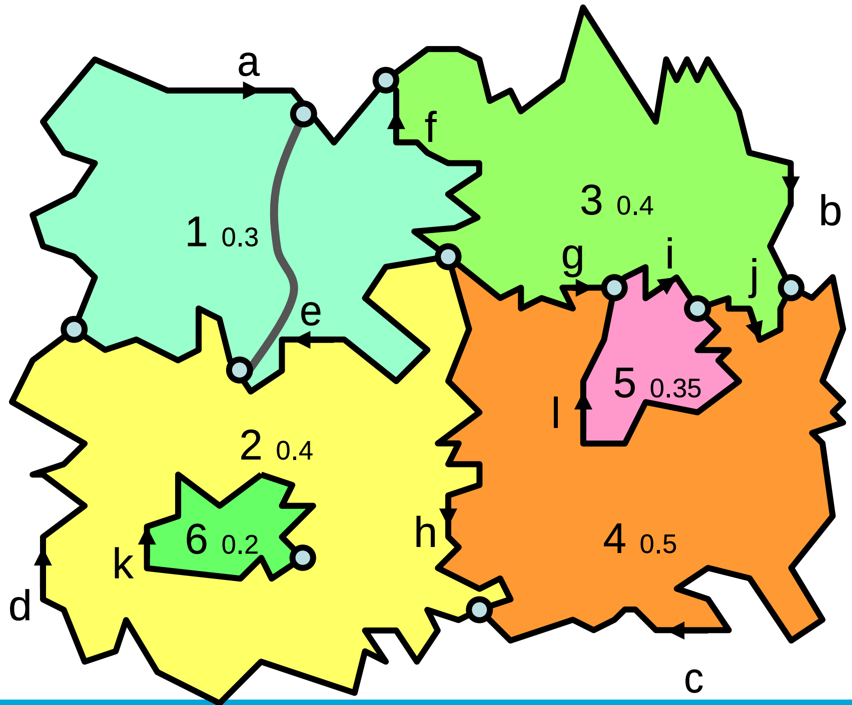
Branch of face tree determines the part of edge forest



Updating tGAP (3)

Effect on BLG trees is local: each edge has a BLG-tree

- New edges & respective BLG-trees
- Old edges and their BLG: delete or use for joined edges

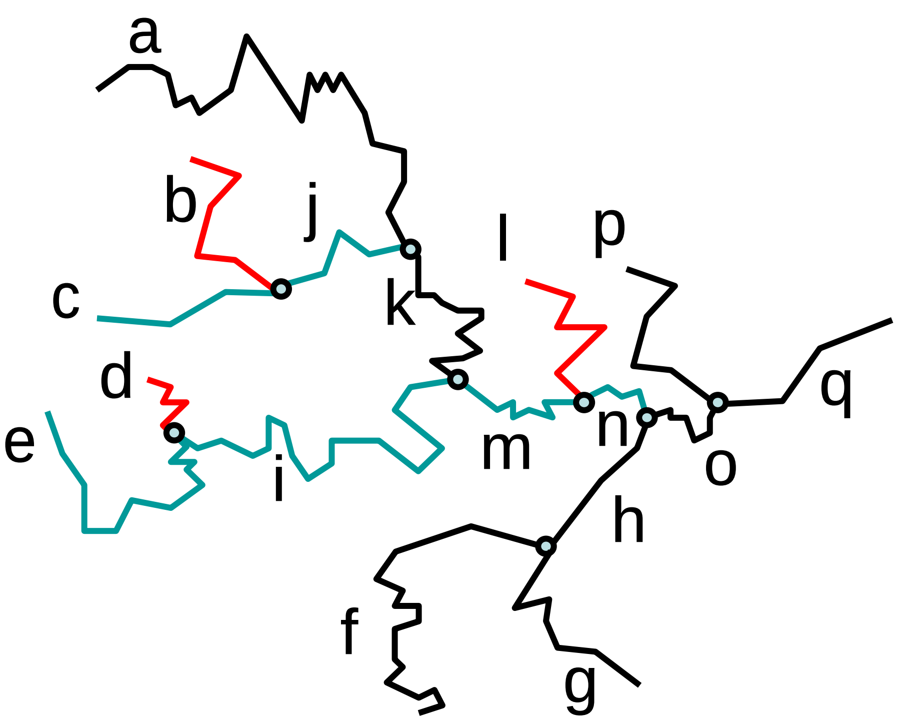


5. Improvements: support of non-area objects (1)

- Support for non-area objects (Reactive-tree for index) fits in tGAP structure:
  1. Points: own table with importance range
  2. Lines: same but now with reference to BLG-repr.
  3. Also combine 2 less important lines in 1 (e.g. after removal of least important branch)
- This enables: the change from area to line (or point) representation at certain moment. Similar to normal GAP-face tree when face is removed, but now at same time it is introduced in node or edge table (with link)

TU Delft

5. Improvements: support of non-area objects, linear network



All edges represented by BLG-trees

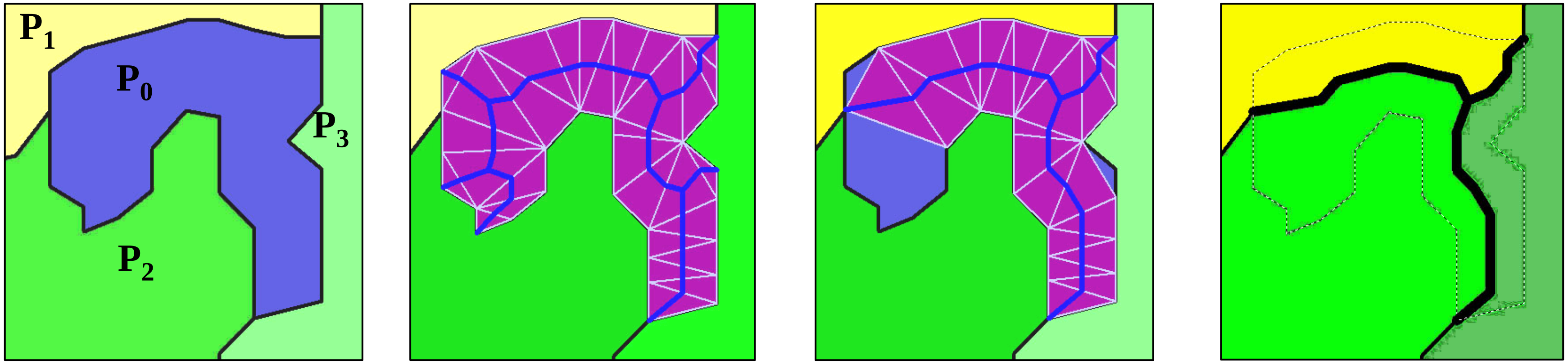
Removal of unimportant edge d will lead to merging BLG-trees of e and i

Remove l → merge m+n
Remove b → merge c+j
...
This fits in tGAP structure!

TU Delft

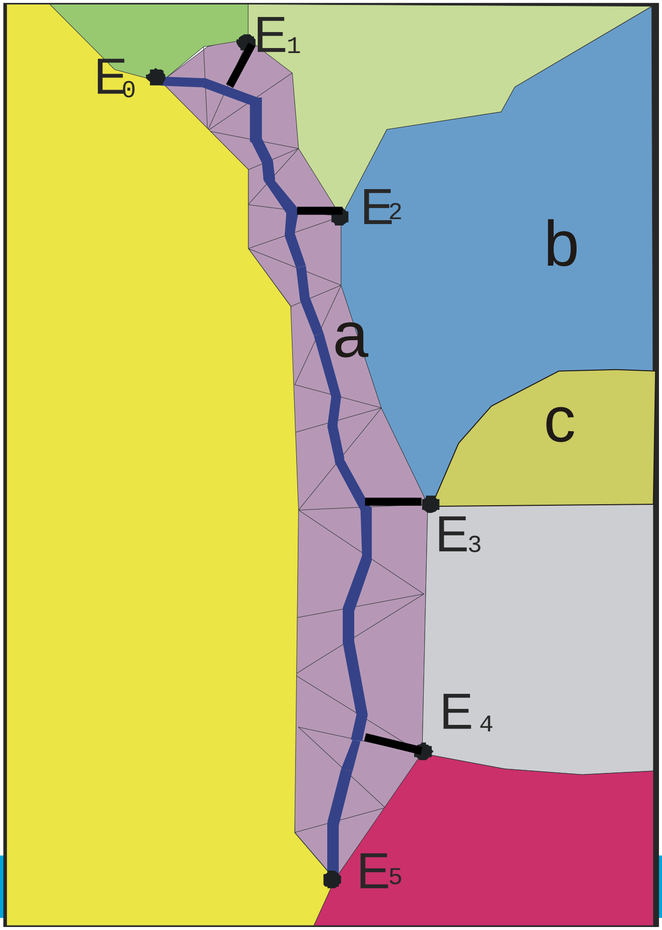
5. Improvements: support of other generalization operations (1)

- Consider collapsing of areas in lines (or points) $\Rightarrow$
  - Option 1: include ‘additional collapse/split’ lines at lowest level (and a feature may the be a collection of faces)
  - Option 2: face tree $\rightarrow$ directed acyclic graph (DAG)
    But similar to tree: does not have to be store explicitly (but scale-ranges should fit)

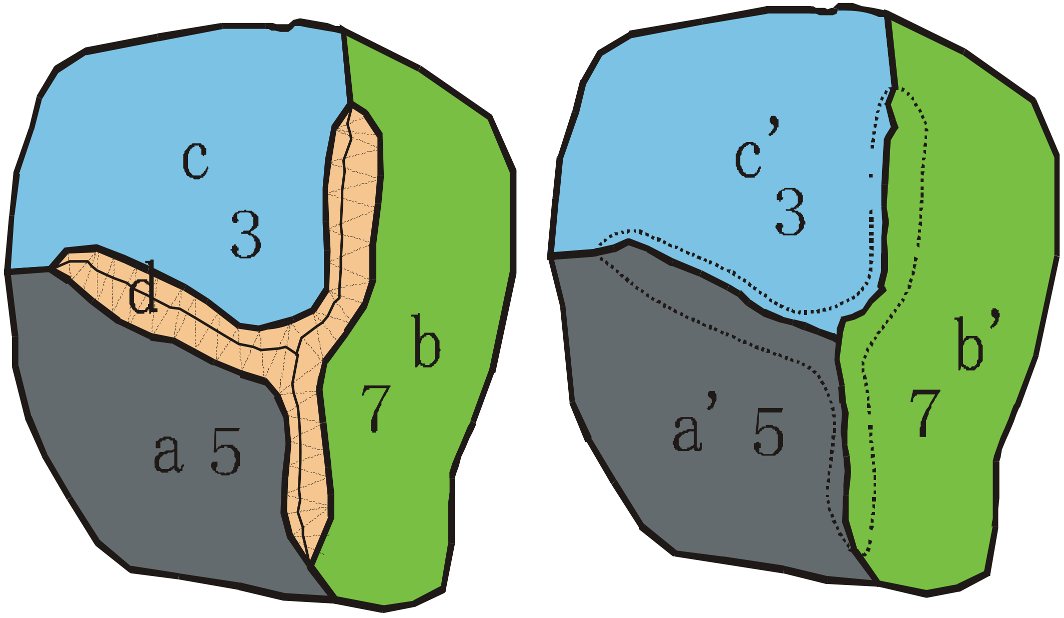


TUDelft

Many neighbors



Weighted skeleton



Both option 1 and 2 fit in tGAP structure

5. Improvements: support of other generalization operations (2)

- Displacement: fits, make sure that end-scale range of first representation, start scale range of second one
- Typification: fits, end scale range of a few
- Heterogeneous aggregation in new class: fits, end scale range of components, scale scale range of aggregate (reuse outer geometry)
- Enlargement: fits (kind of counterpart of collapse)

- Notes: 1. Rules/algorithms to take these decisions needed, 2. Vario-scale structure starts to resemble a MRDB (with minimal redundancy and non-fixed scales)

Contents

1. Introduction
2. Background tGAP structure
3. First implementation
4. Client-server set-up and progressive refinement
5. Improvements
6. Conclusions

TU Delft

6. Conclusions, main results

- First time ever non-redundant geometry scaleless data structure has been implemented (based on topology)
- tGAP is well suited for web environment (progressive)
- The class importance values and classes compatibility matrix are crucial for quality of the structure (same is true for other rules/algorithms to fill the structure)
- Semantic aspect (also attributes) needs further attention; e.g. what to do with attributes after merging: sum, min, avg, ... (depends on meaning attribute)
- Independent themes → multiple tGAP structures
- Views can be used for ‘dumb’ clients (non-tGAP-aware)

6. More improvements/future work

- Generalization is application (task) dependent
  → more than 1 tGAP structure on same base topol (compare to multiple indices on same table)
- 'Bug': different edges of narrow features may cross when generalizing → avoid this during creation by tests (and state corresponding correct imp/tol value)
- Benchmarks have to be performed with alternatives (multiple-representation approaches and redundant scaleless approaches)
- Two important test client environments:
  1. Desktop GIS
  2. Distributed Web GIS

TU Delft

6. Some observations on dimensions

- Tree structure not really needed in implementation
- Important: tGAP structure translates 2D space and 1D scale in an integrated 3D topological representation: no overlaps and no gaps (in space and scale)

- (Starting with 3D space and adding scale results in 4D)
- Starting with 3D space and time (history) and adding scale results in 5D topological structure (again no gaps/overlaps in space, time or scale), well defined neighbors in space, time and scale directions

TU Delft