Master's thesis research : results

Implementation and testing of variable scale topological data structures

Experiences with the GAP-face tree and GAP-edge forest

Martijn Meijers

TU Delft
OTB Research Institute
Section GIS Technology

Outline

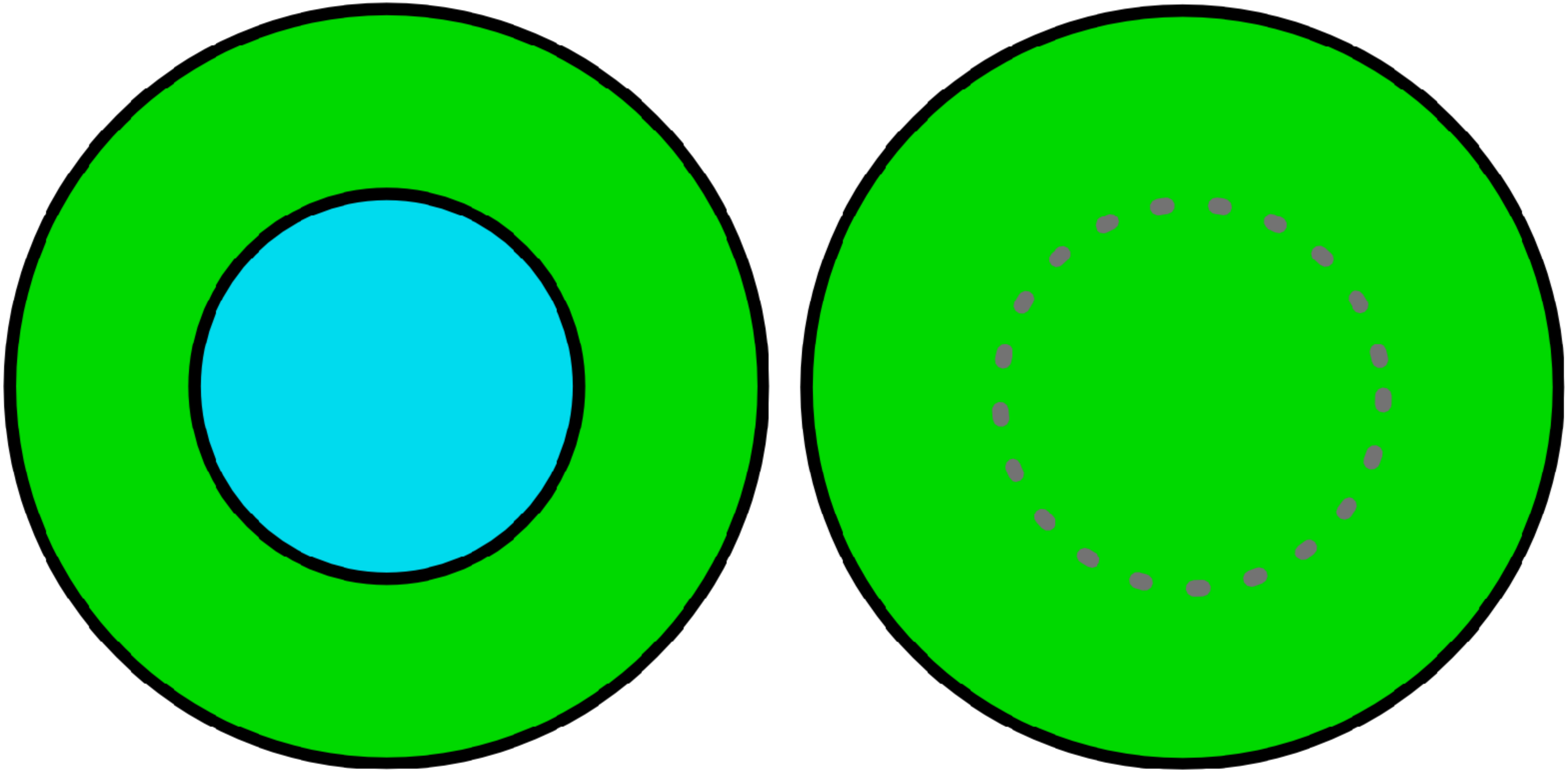
- Introduction to the problem
- Methodology: approach of research
- Relevant theories (background)
- Results
- Preliminary conclusions

Introduction

- Generalization: 'selection & simplification for appropriate detail and or purpose of maps' [ICA, 1973]
- Change in:
  - scale (paper) or level of detail (digital screen)
- Generalization by conceptual operator, implemented in GIS via algorithms [Galanda, 2003]:
  - Reclassification
  - Displacement
  - Aggregation
  - Typification
  - Exaggeration
  - Collapse
  - Elimination
  - Enlargement
  - Simplification
  - Smoothing

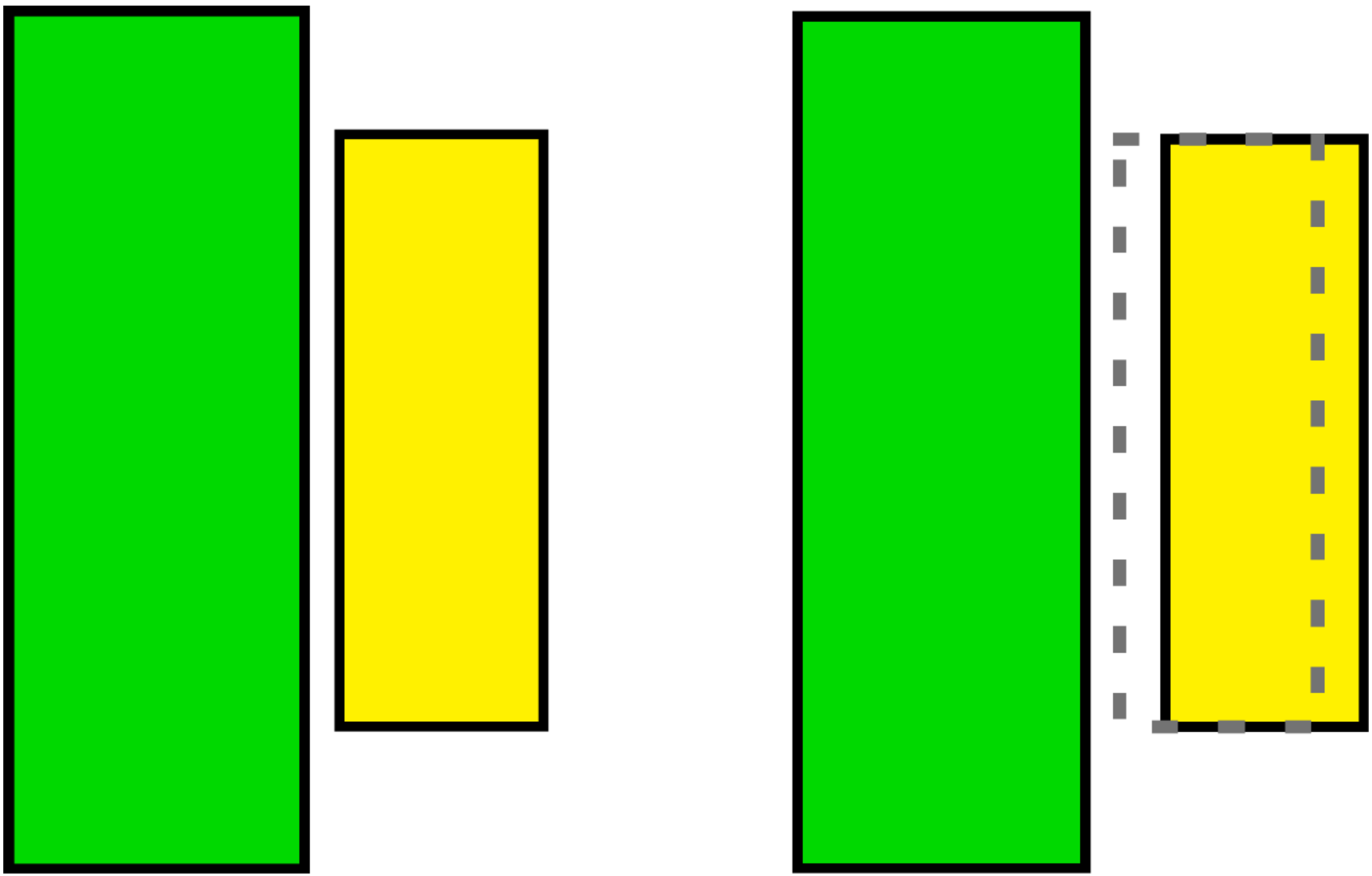
Example operator: reclassification

(taken from [Galanda, 2003])



Example operator: displacement

(taken from [Galanda, 2003])



Introduction

- Current approach (problematic, [Stoter, 2005]):
  - Problems with complexity of algorithms: *not* in real time possible to calculate wanted level of detail
  - Multiple scale storage: waterfall like process
  - Problems with managing (updating) data: redundancy
  - Not feasible in Spatial Data Infrastructure:
    - user interaction = real time
    - no pre-defined scale

Introduction

- New proposed approach [Van Oosterom, 2005 based on Van Oosterom, 1990 & Vermeij, 2003]:
  - Data structures (GAP-face tree & GAP-edge forest) store results of one-time calculation
  - No pre-defined scale: vario-scale
  - Topological data (references): no redundancy

Introduction

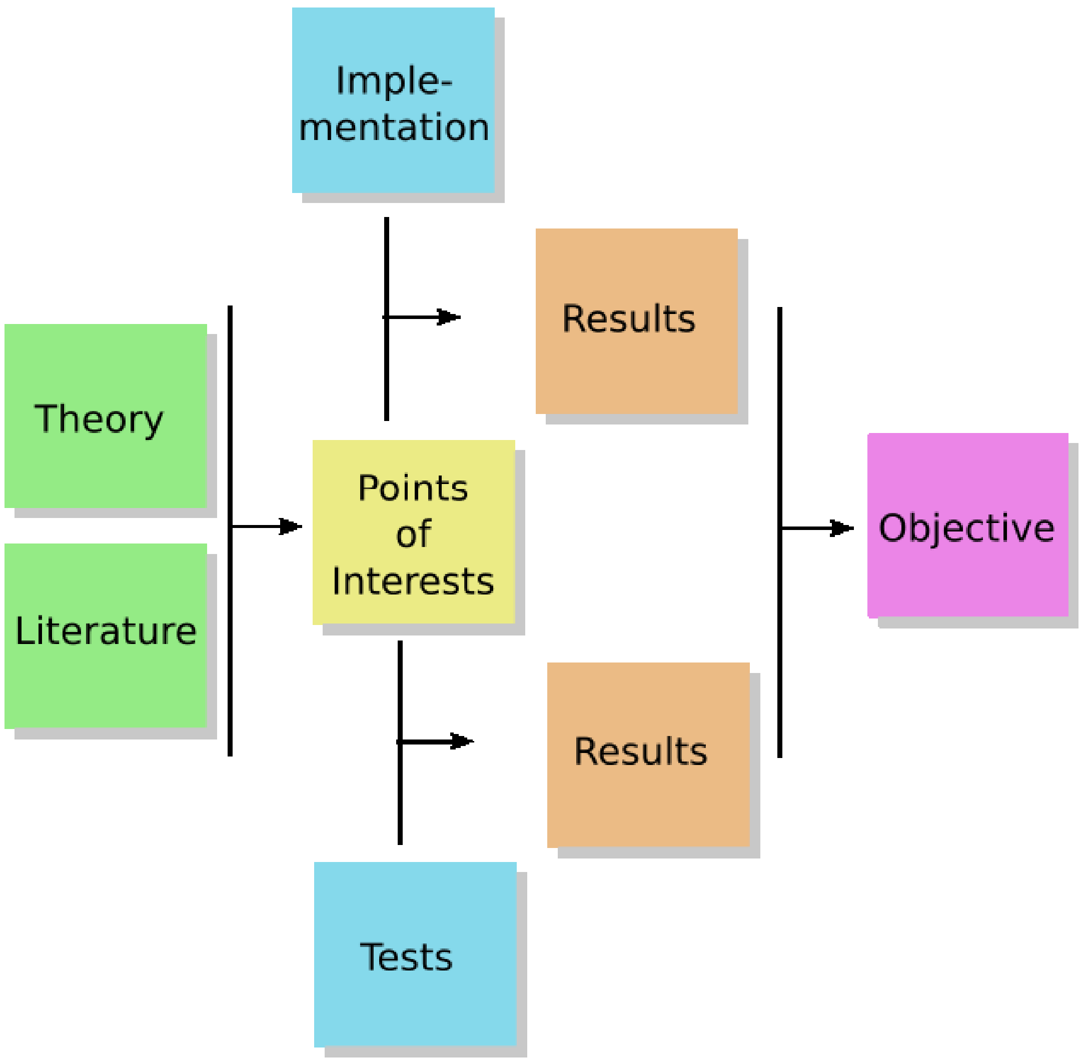
- Problem:
  - Only theory about new data structures
- Research objective:
  - Verification by:
    - literature study
    - Implementation experiment / test bed

Not: focus on generalization algorithms

But: focus on data structure:

- storing result of algorithms
- querying and visualization is possible on several levels of details (vario-scale)

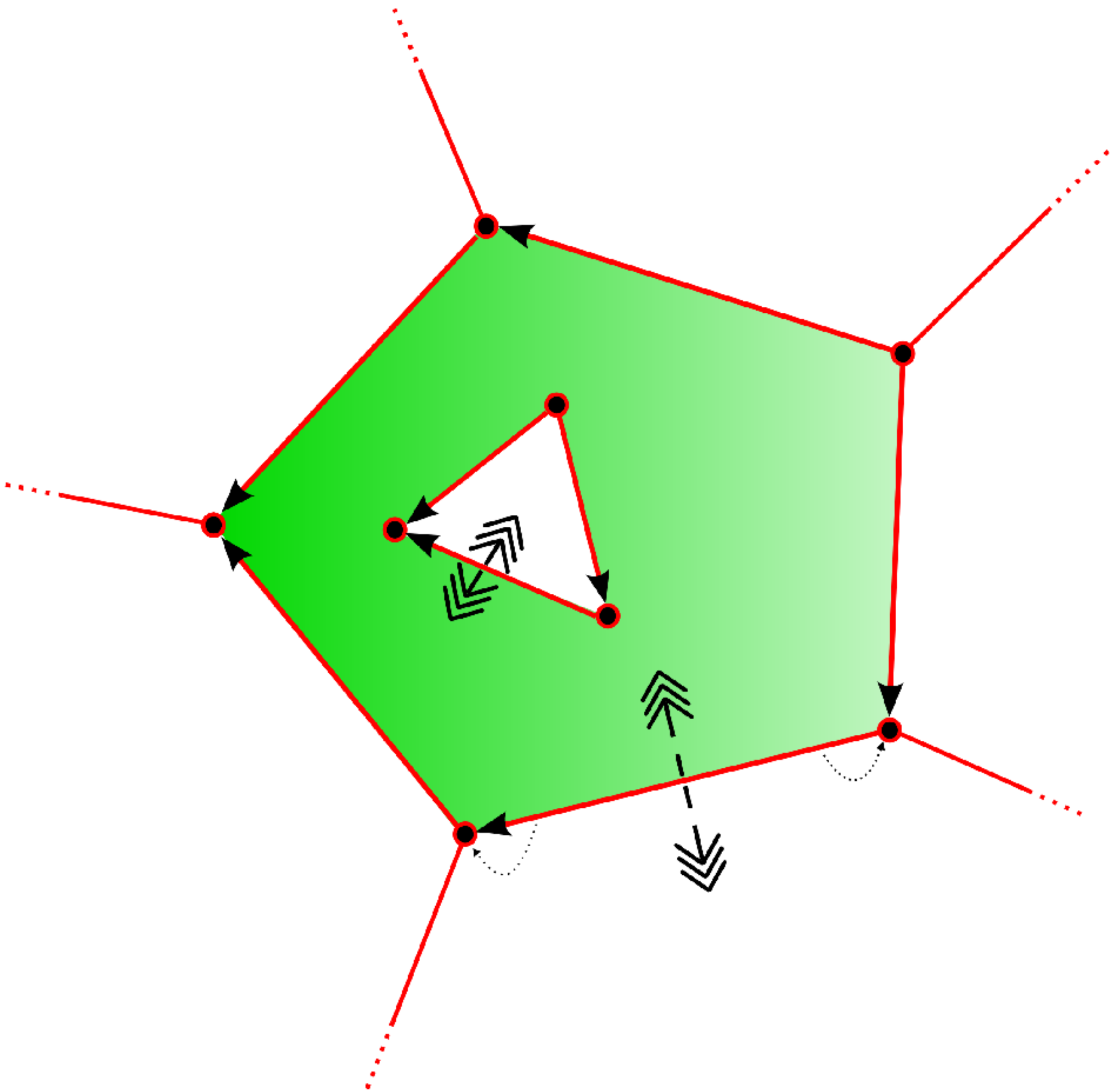
Methodology



- Research framework
  - Literature study
  - Experiment
    - Implementation: test bed
    - Tests
  - Results
  - Objective
    - Test performance (broadest sense) of data structures for generalization

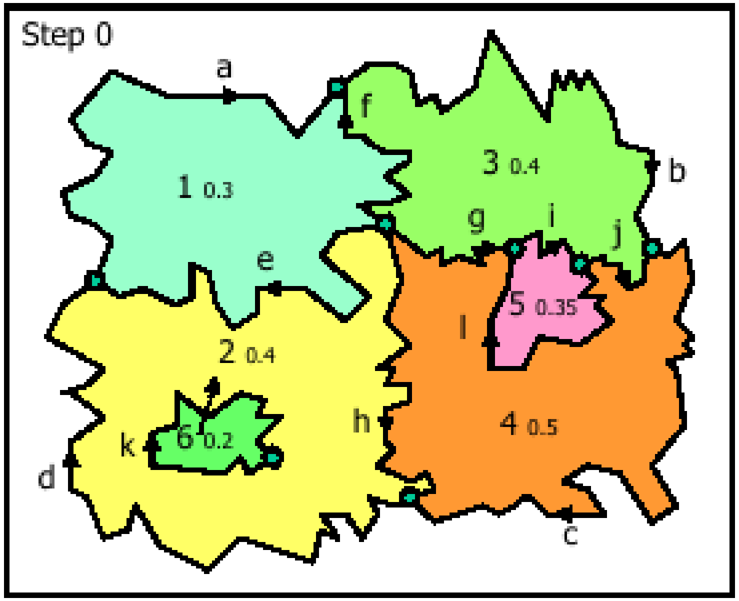
Background

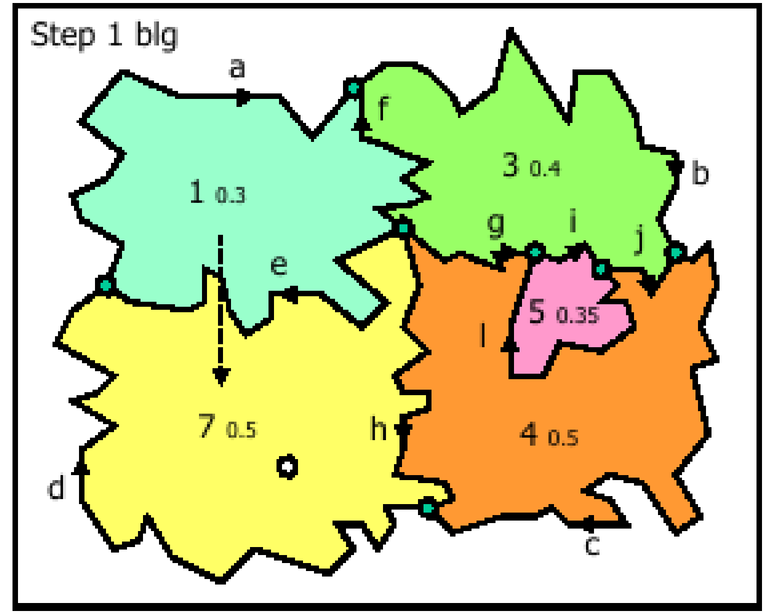
- Topology versus geometry
- Topological face:
  - {edges}
  - {nodes}
  - {references}

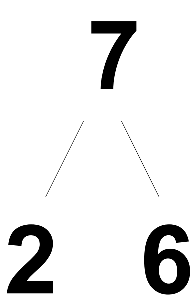


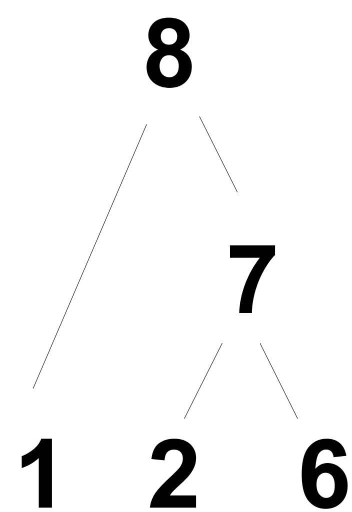
Background

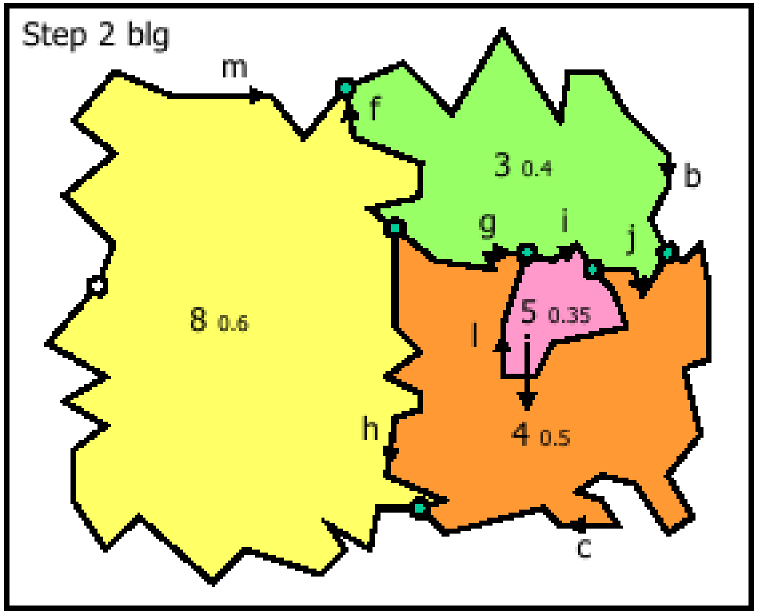
- GAP-face tree
  - Generalized Area Partition
  - Tree structure
  - Based on GAP-tree
- Store face information:
  - Which faces at which scale:
    - Selection on importance instead of scale
  - How faces are merged
- Example
  - Pictures taken from [Van Oosterom, 2005]

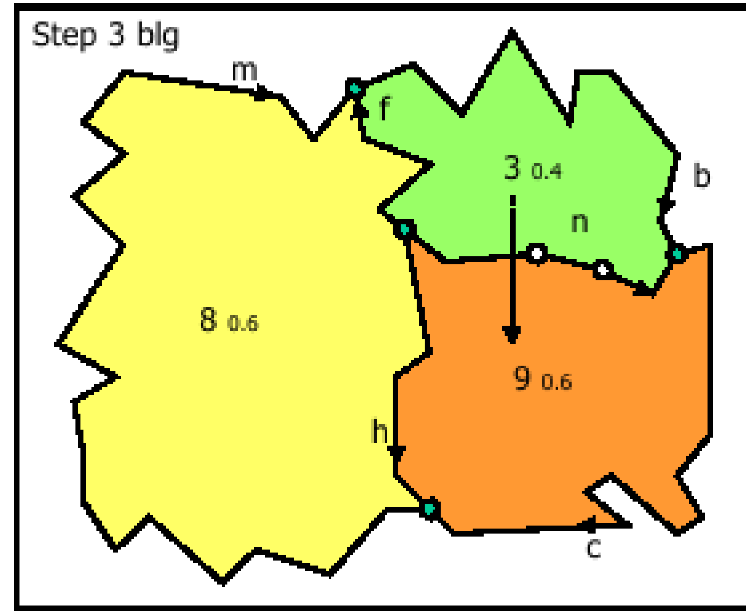

Step 0
a
f
1 0.3
3 0.4
b
g
i
j
e
5 0.35
l
2 0.4
h
4 0.5
k
6 0.2
d
c

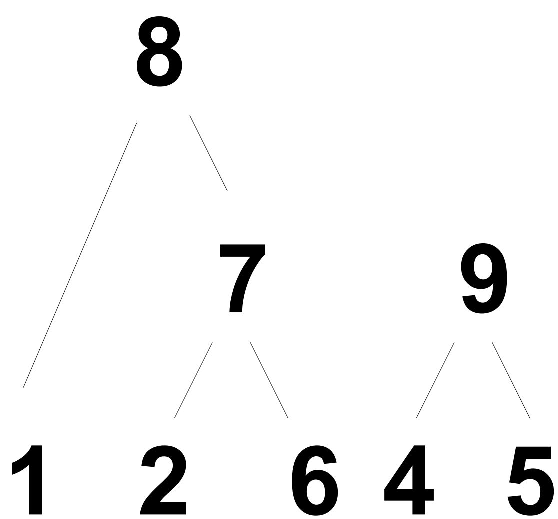

Step 1 blg
a
f
1 0.3
3 0.4
b
g
i
j
e
5 0.35
l
7 0.5
h
4 0.5
d
c

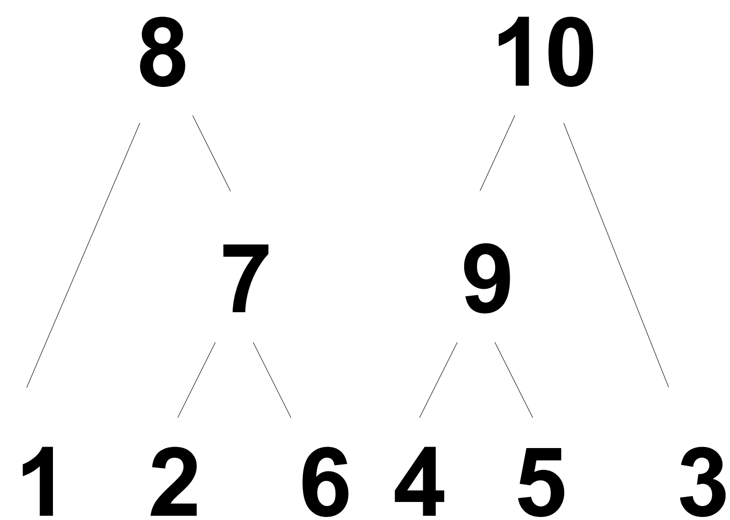

7
2
6

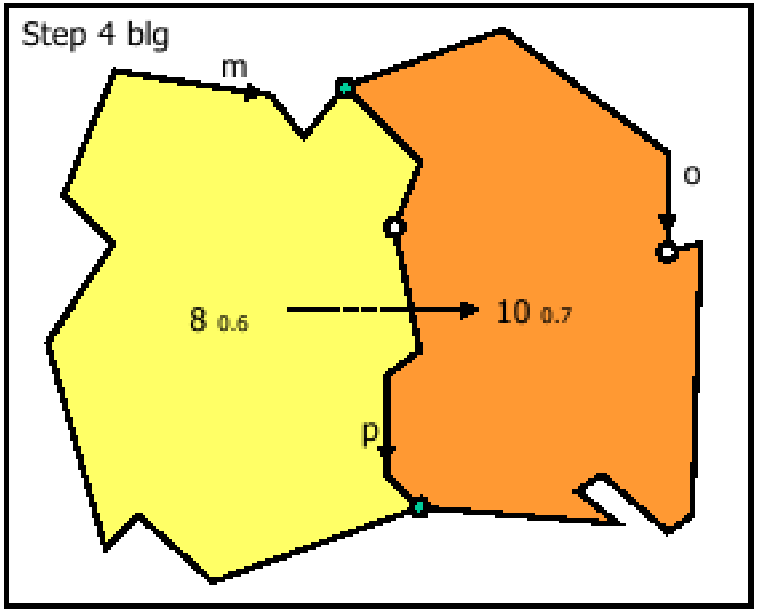

8
7
1
2
6

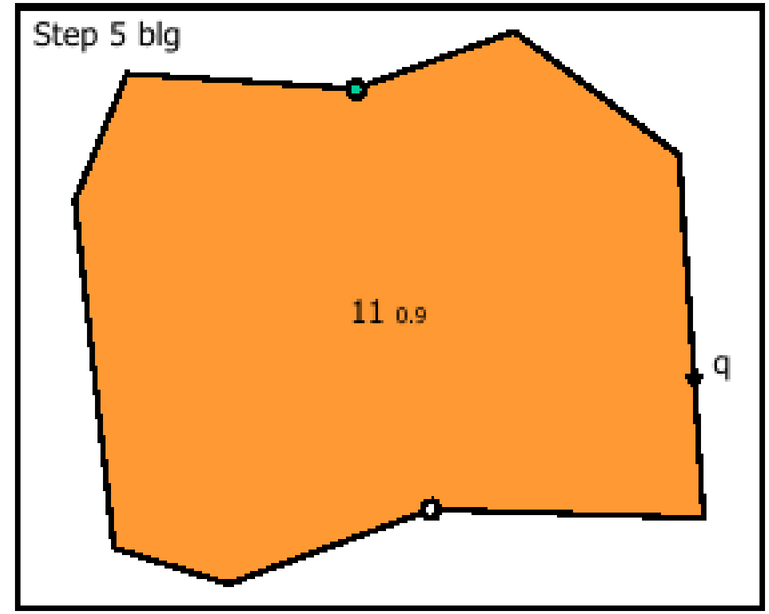

Step 2 blg
m
f
3 0.4
b
g
i
j
8 0.6
5 0.35
l
4 0.5
h
c

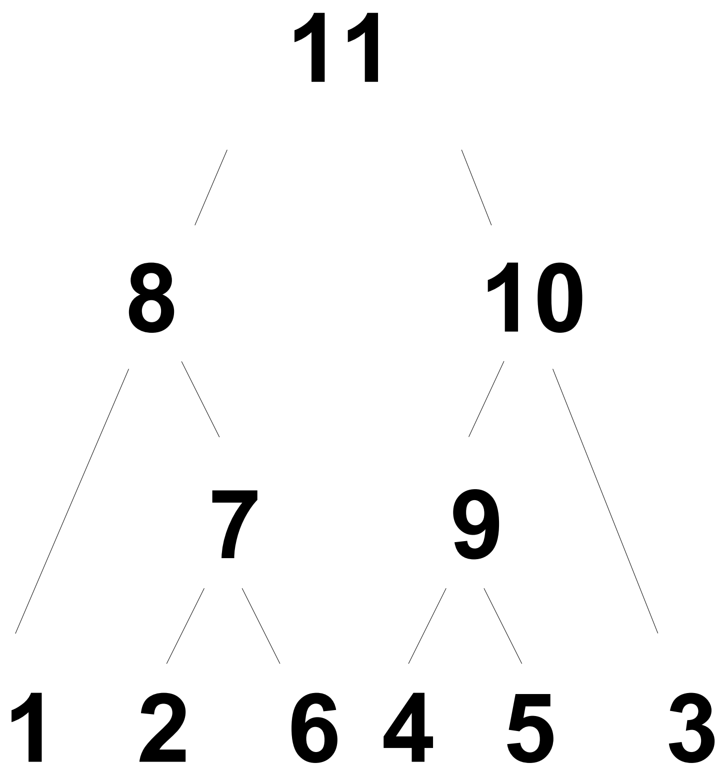

Step 3 blg
m
f
3 0.4
b
n
8 0.6
9 0.6
h
c

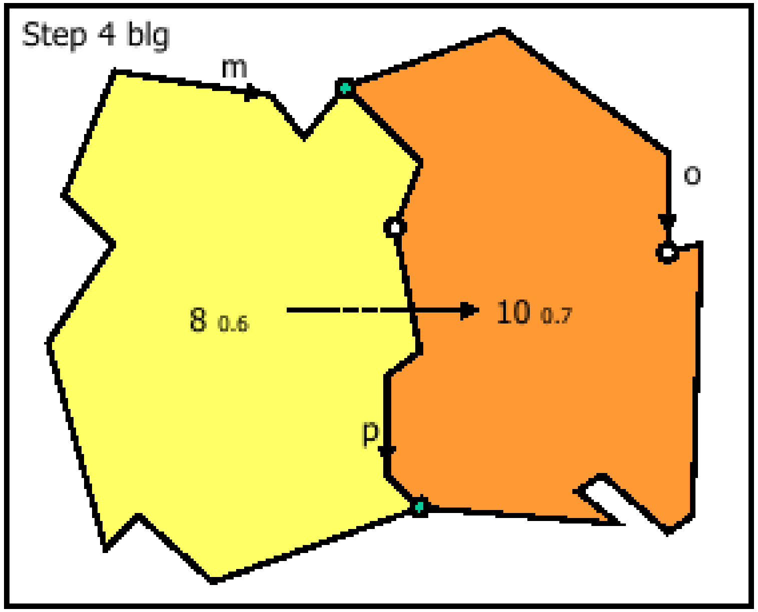

8
7
9
1
2
6
4
5


8
10
7
9
1
2
6
4
5
3

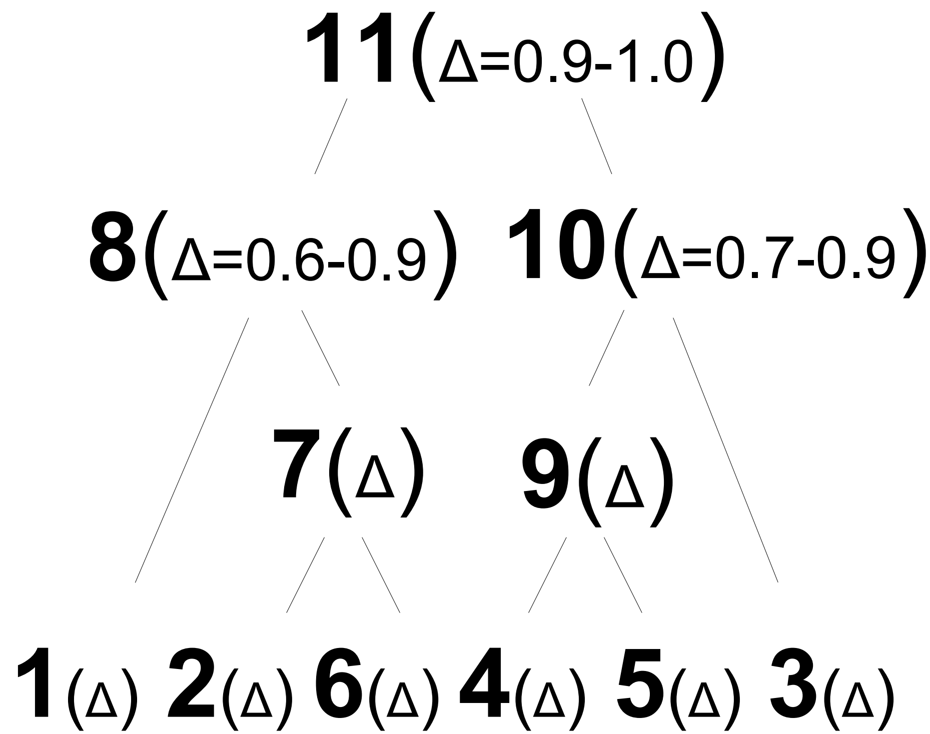

Step 4 blg
m
o
p
8 0.6
10 0.7

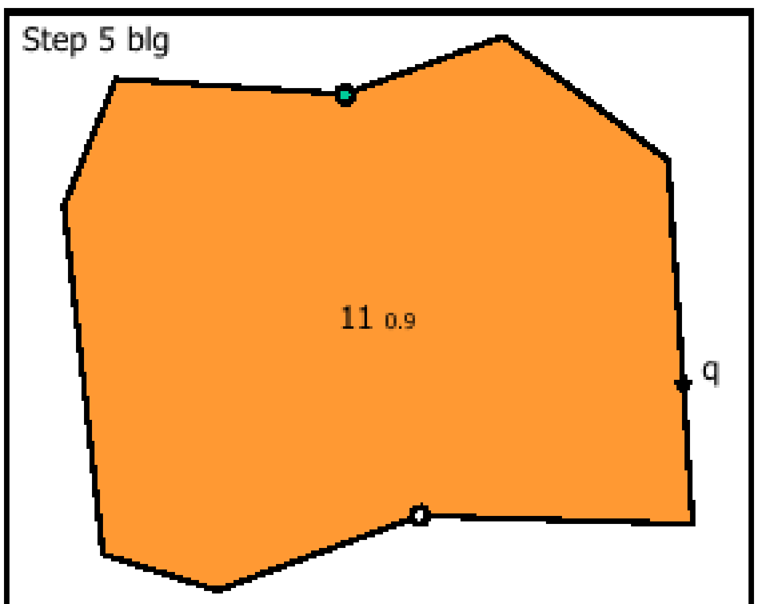

Step 5 blg
q
11 0.9


11
8
10
7
9
1
2
6
4
5
3



$$\Delta = [imp\_low, imp\_high)$$

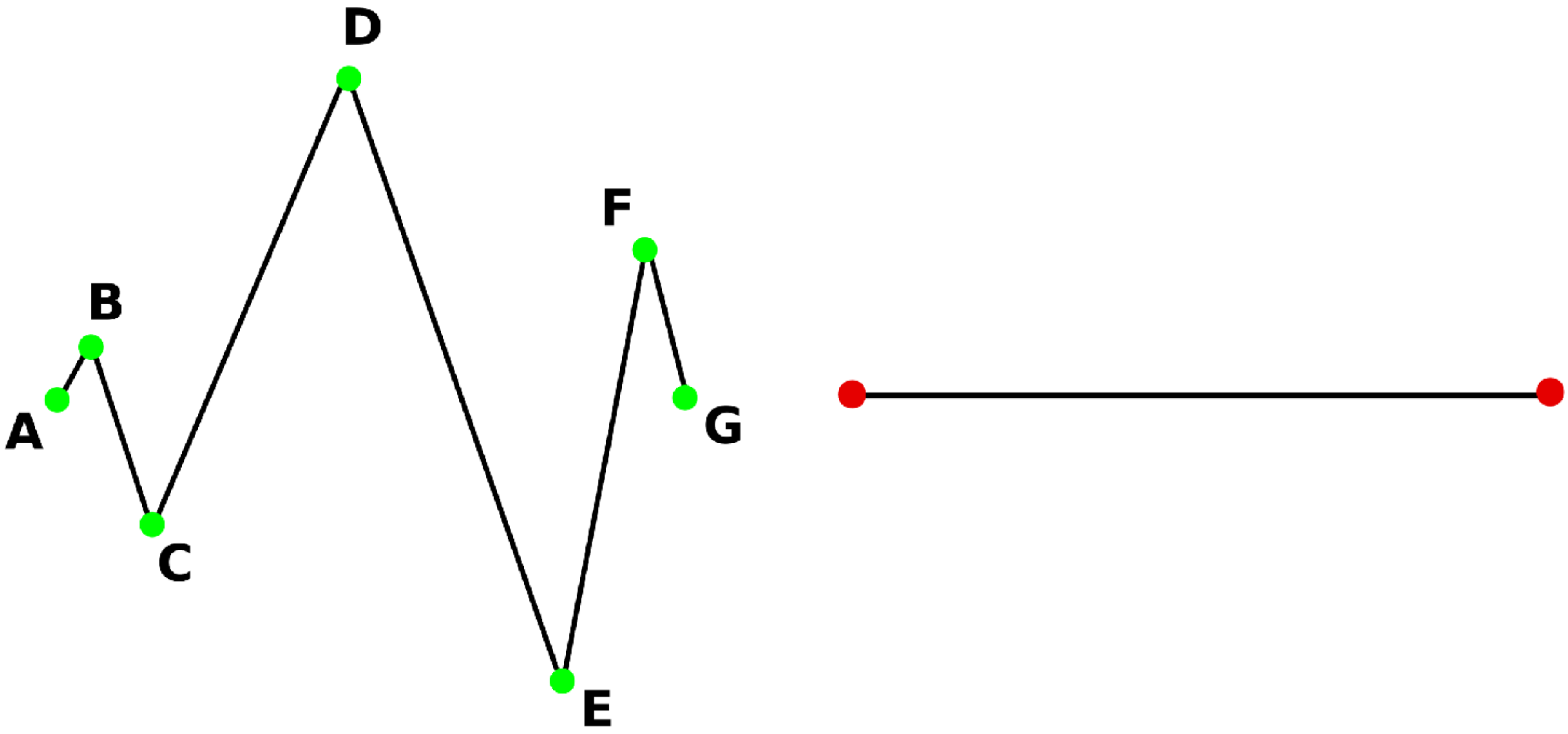


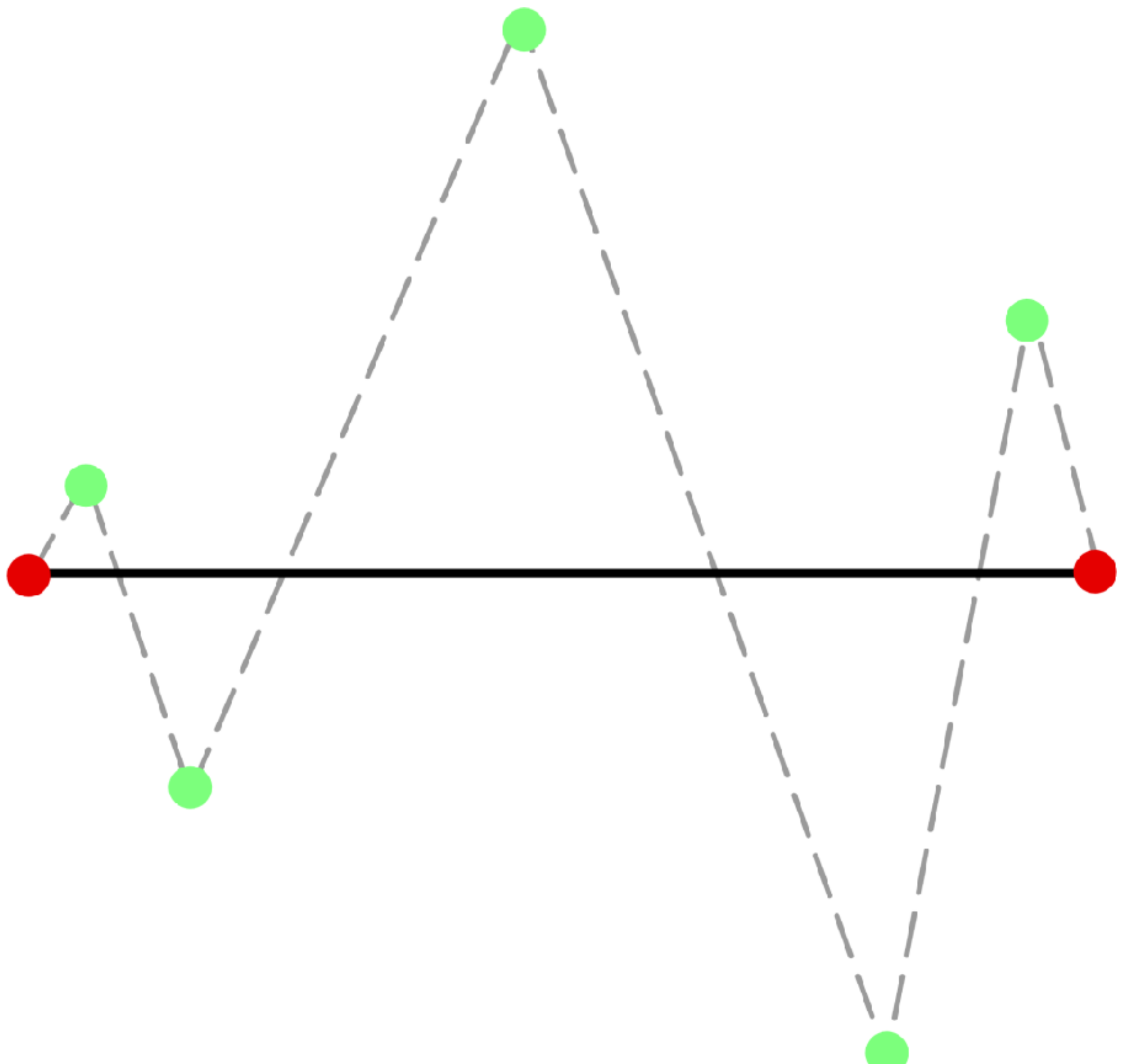


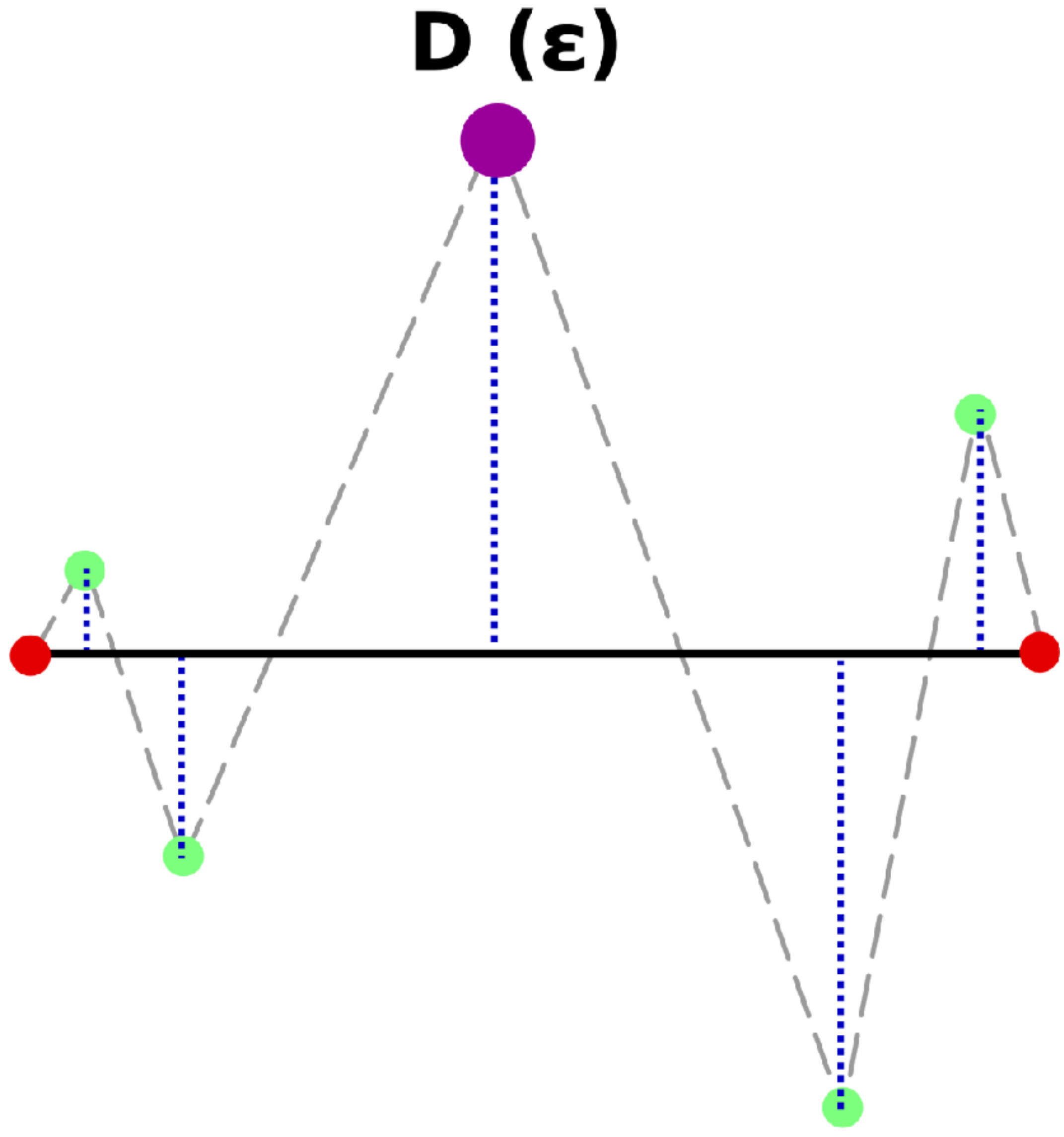
Background

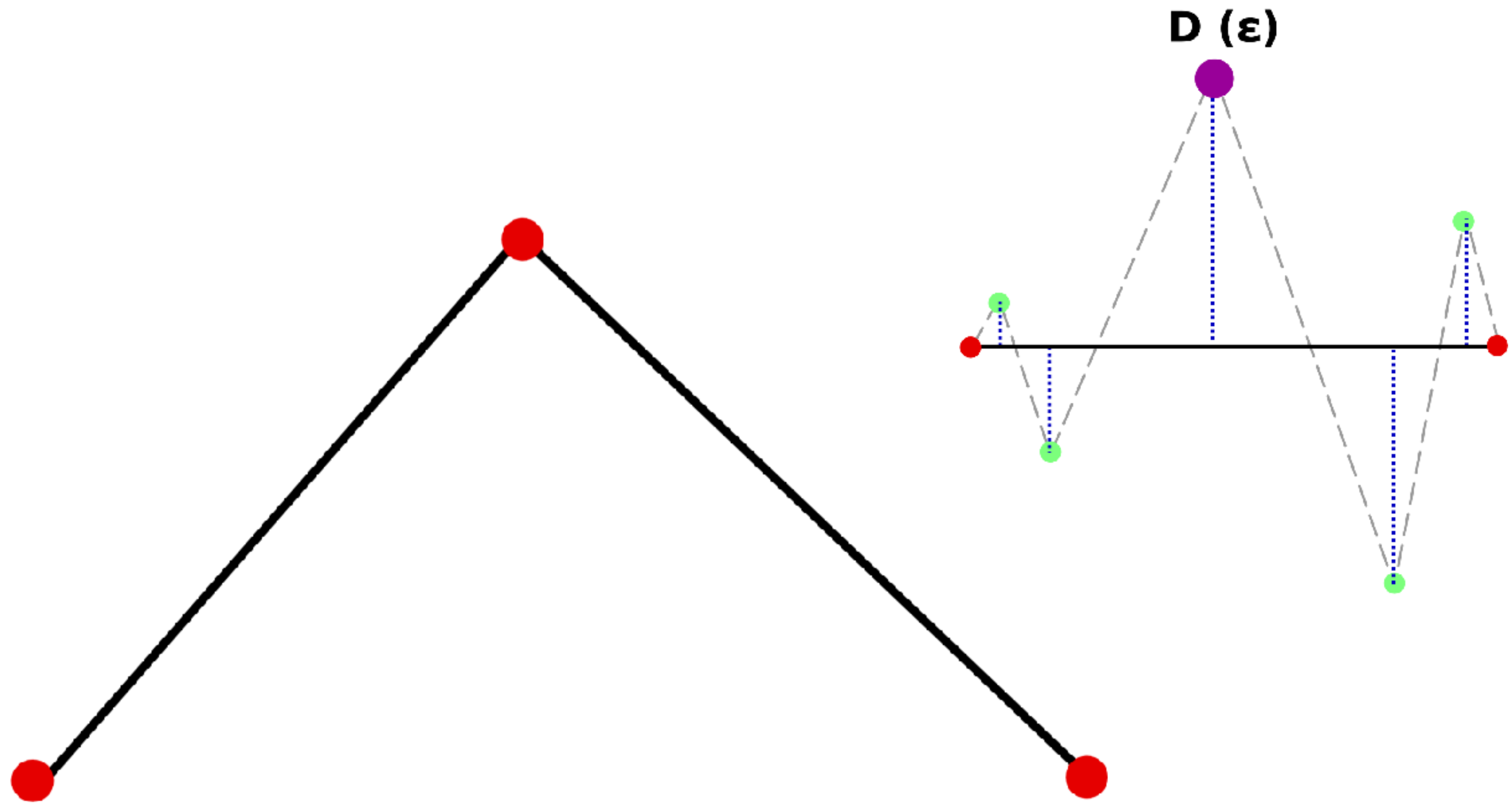
- GAP-edge forest (of trees)
  - Douglas Peucker algorithm [Douglas-Peucker, 1973]
  - BLG-tree data structure for storing result [Van Oosterom, 1990]
    - Binary Line Generalization
    - Tree structure
- Store edge information:
  - Generalized version & Non-generalized version in one tree
  - Merging of edges (with references) : forest

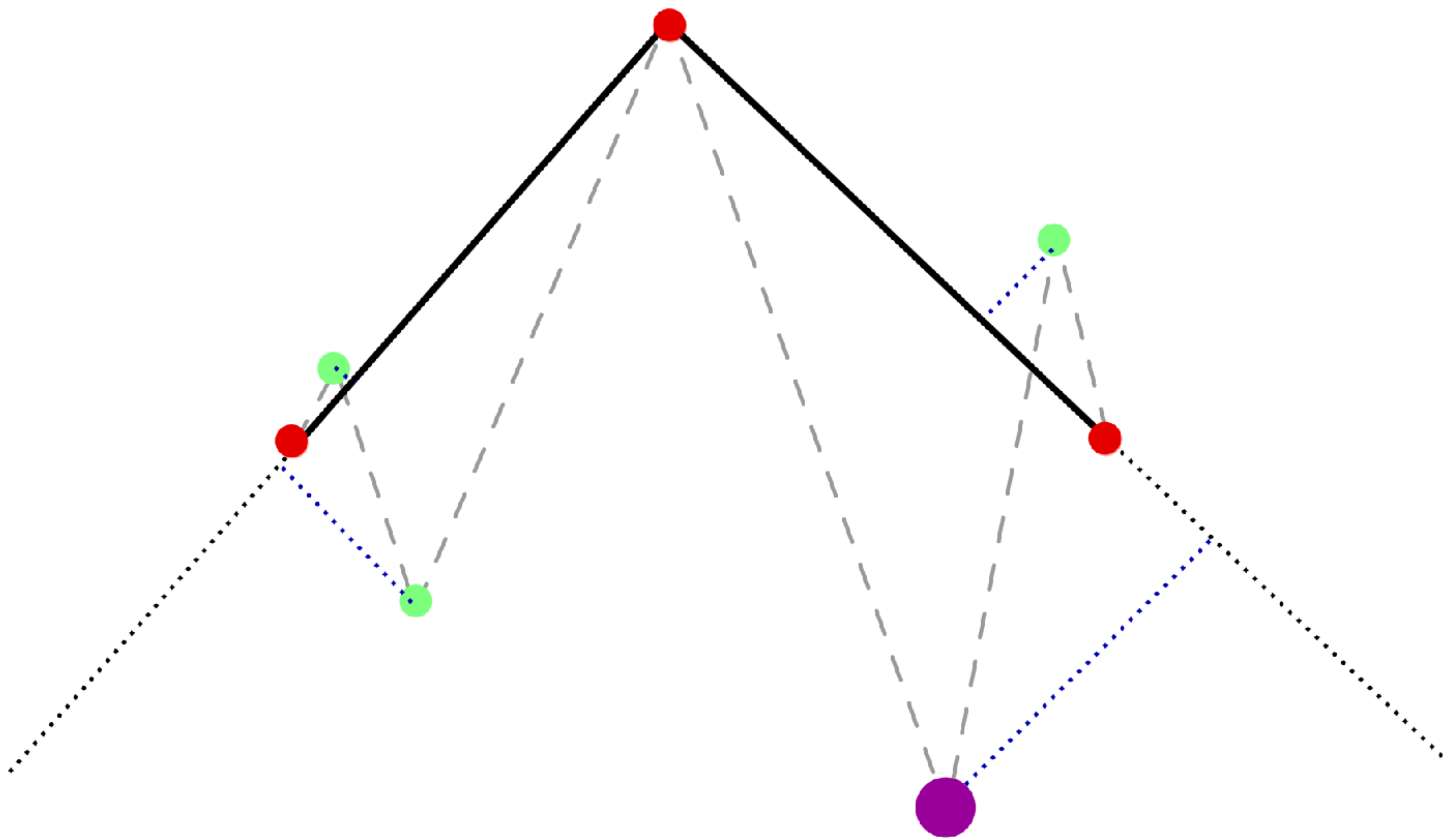
Douglas-Peucker Example

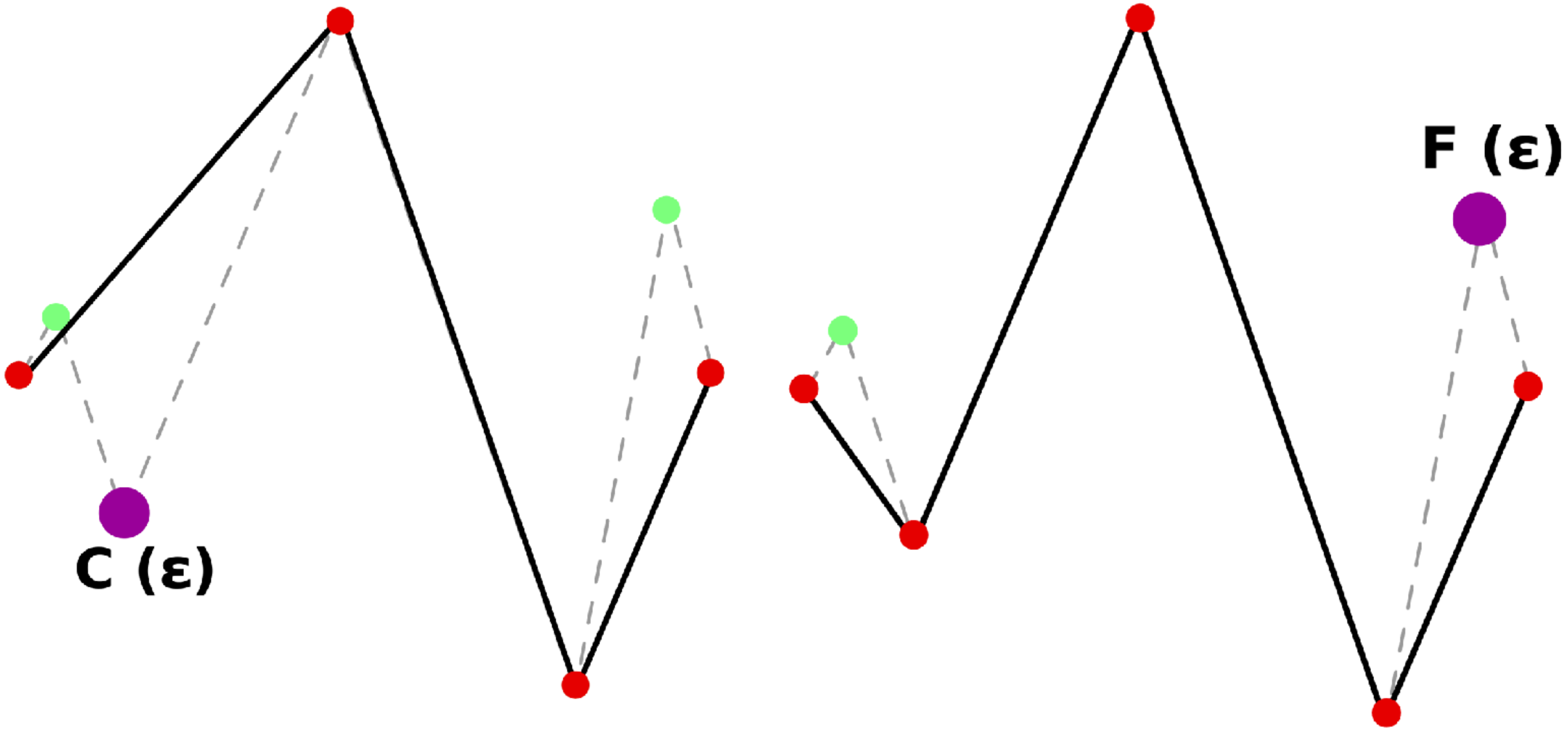





D (ε)


D (ε)


E (ε)

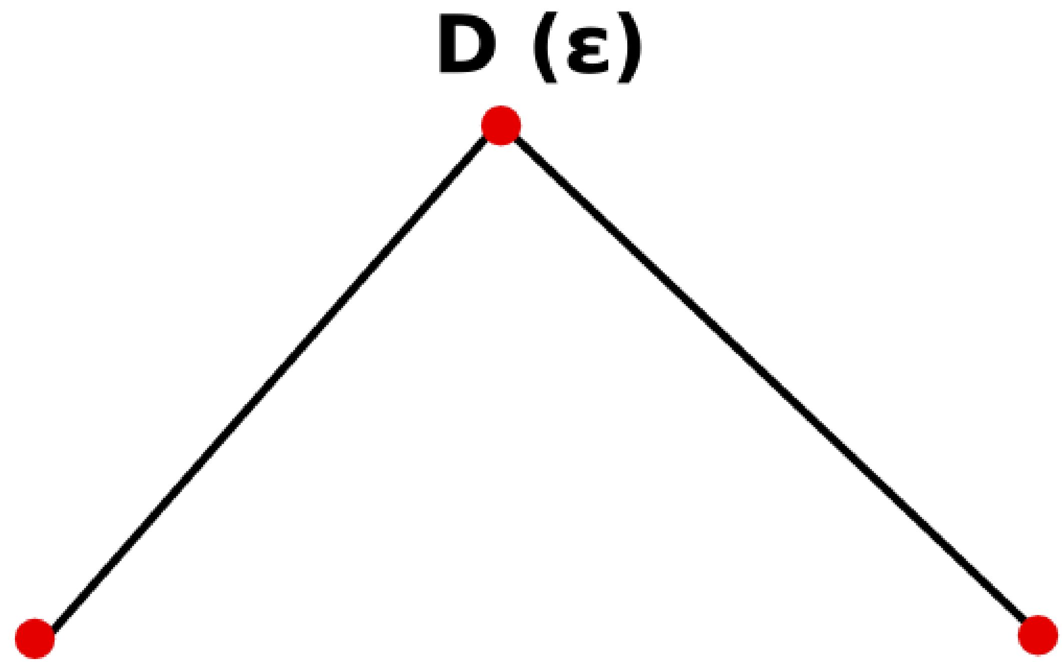


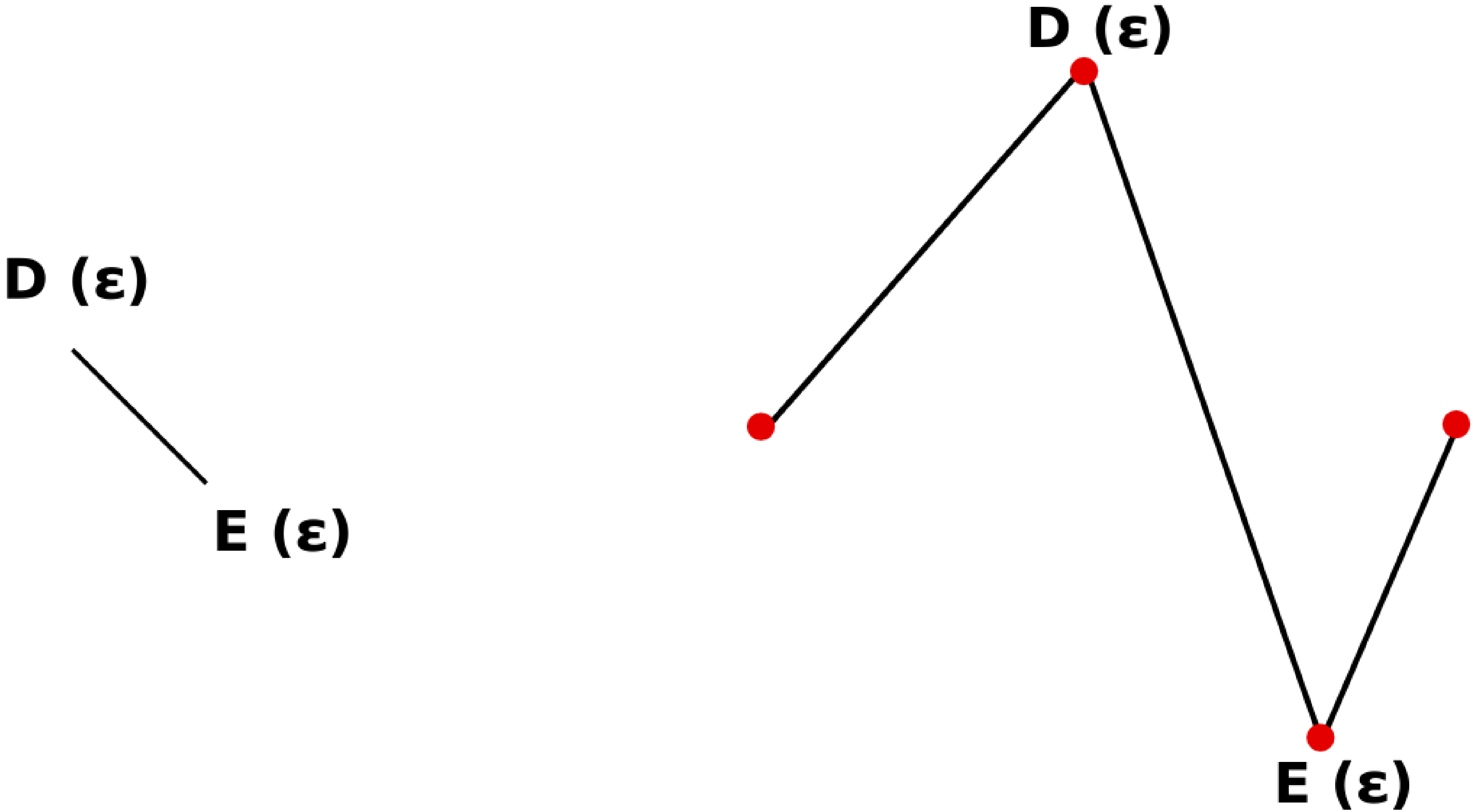
F (ε)
C (ε)

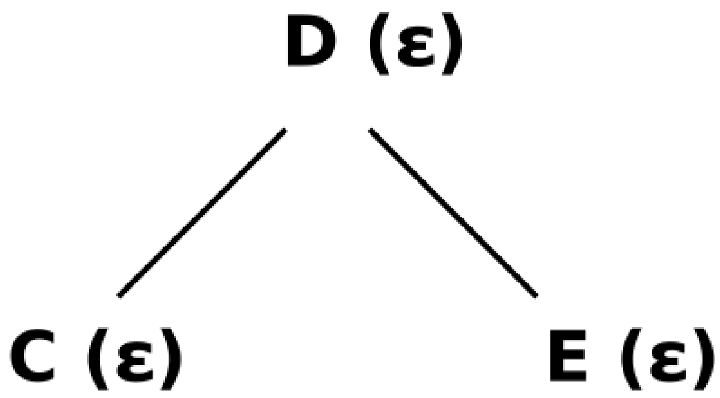
Store D-P result in tree : BLG

- Calculation intensive algorithm
  - Each time calculate all distances to all remaining points
- Store result in tree
  - No need for computation
  - Get wanted level of detail of line from tree
- Binary Line Generalization tree
  - example with previous line

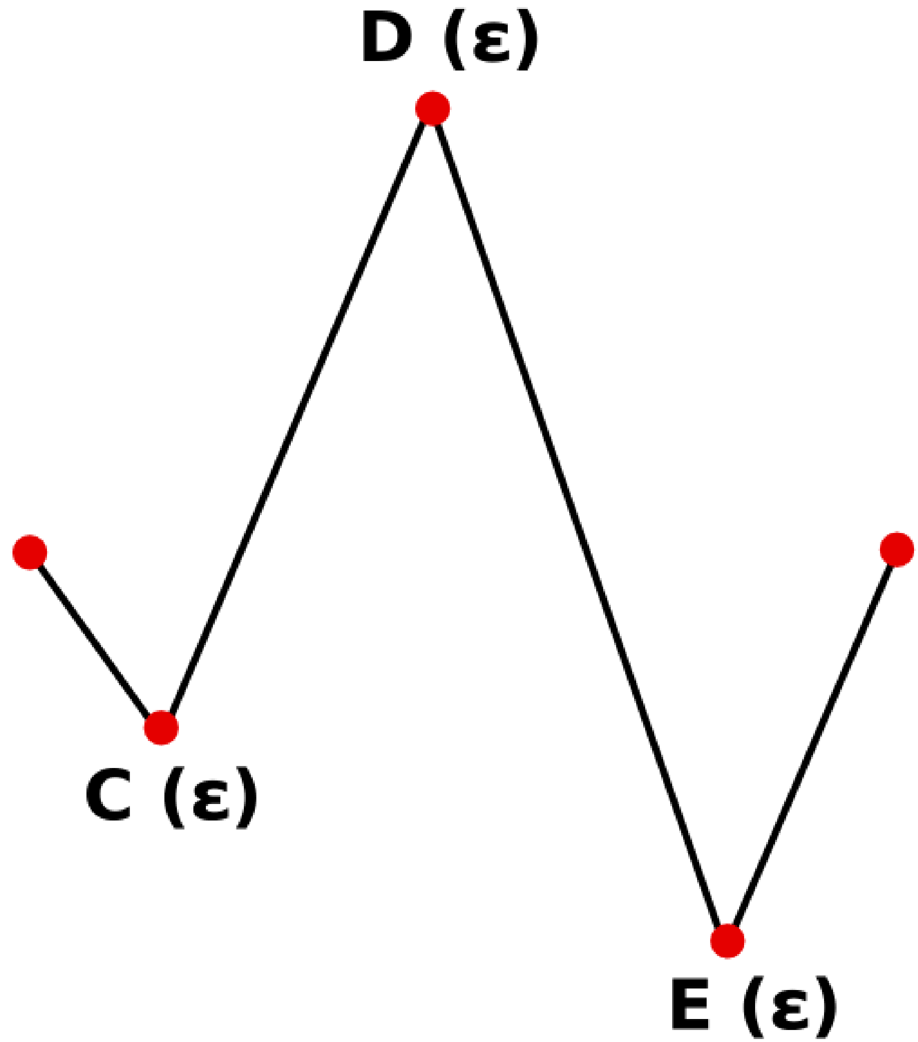
D (ε)



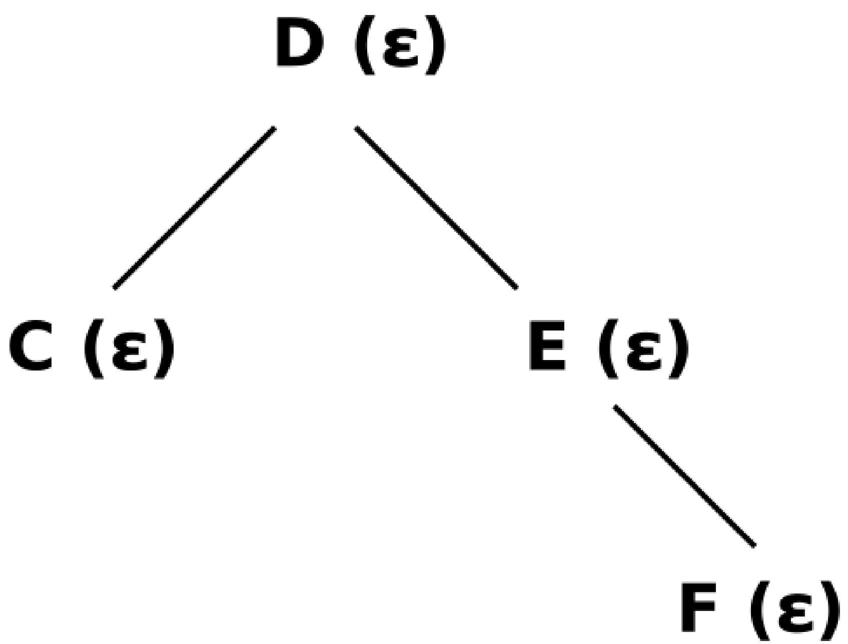

D (ε)
E (ε)
D (ε)
E (ε)

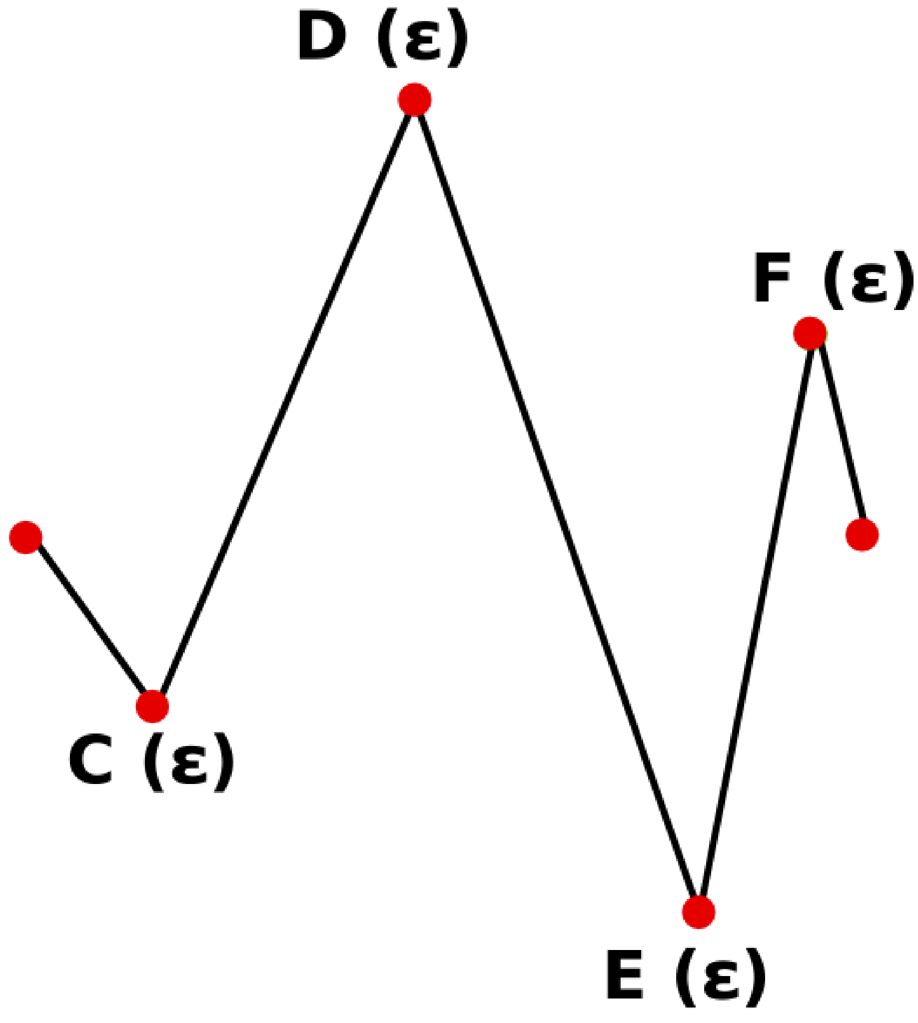


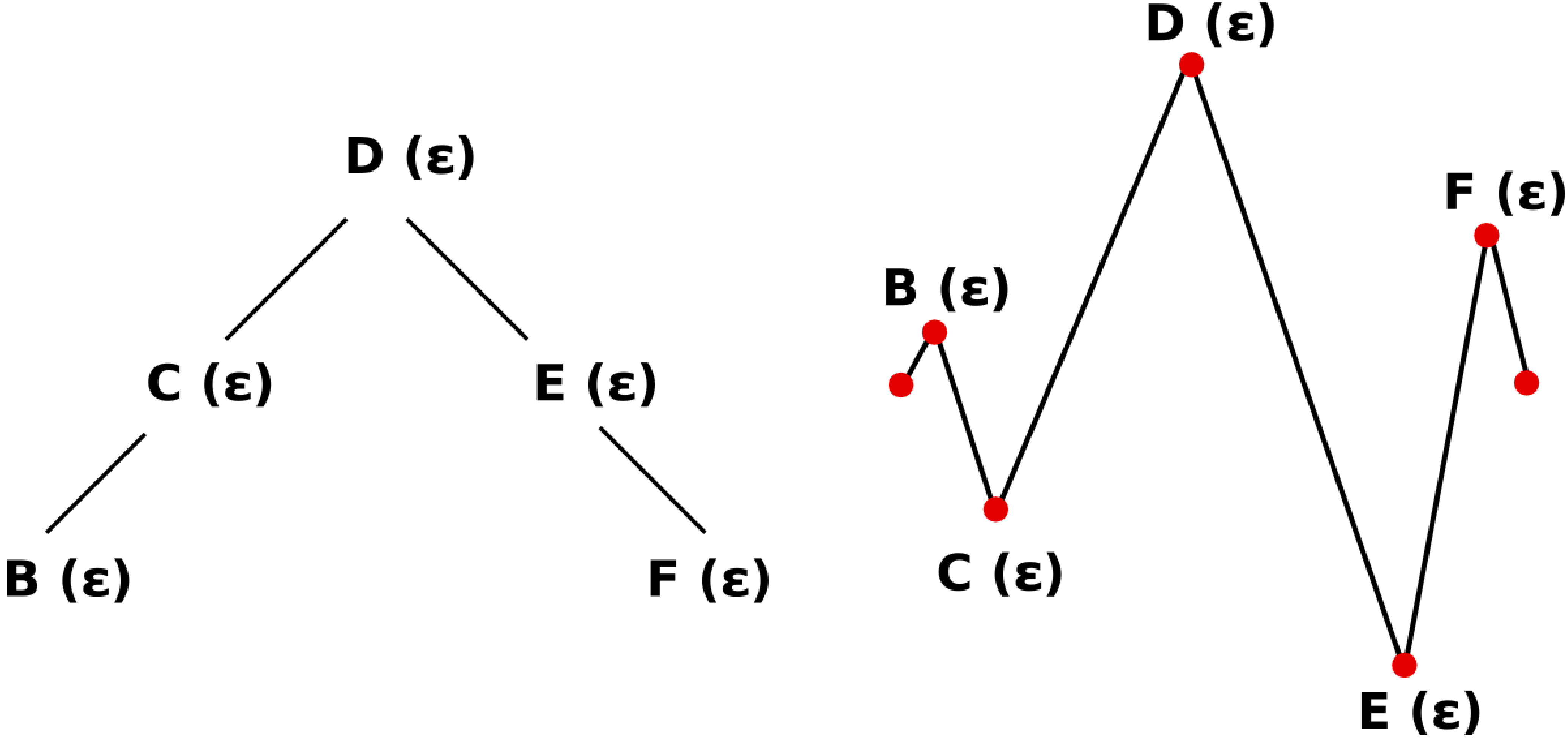
D (ε)
C (ε)
E (ε)



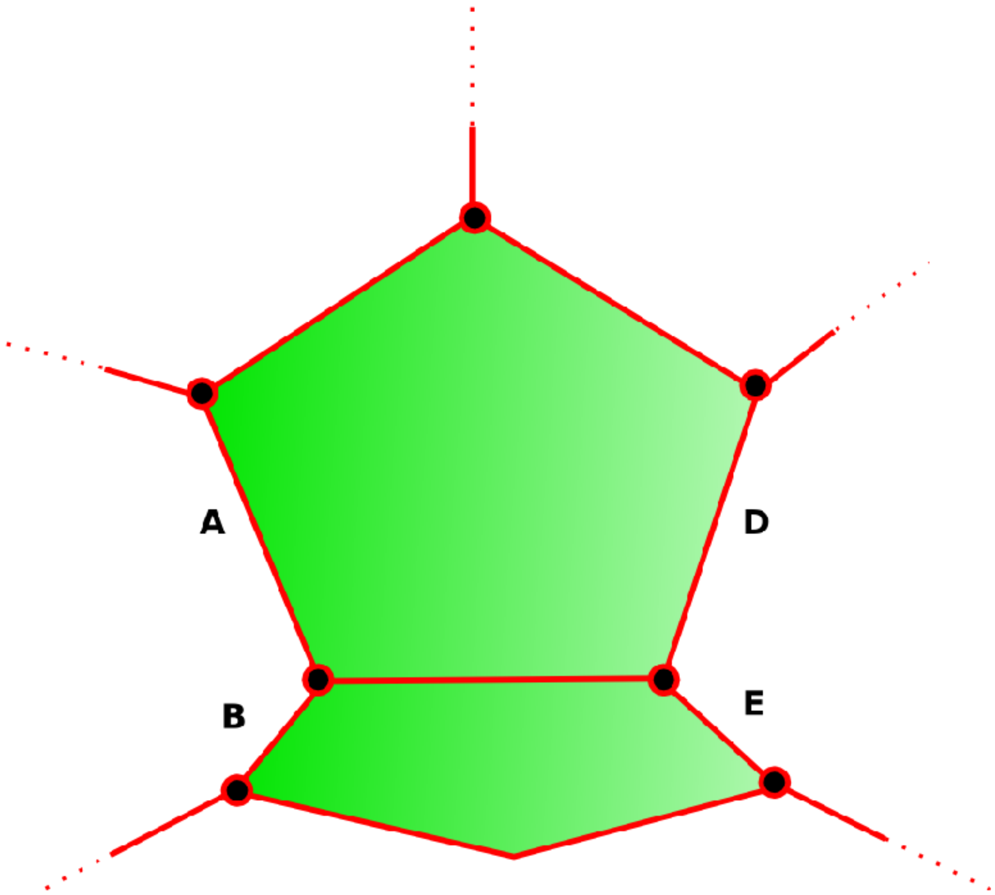
D (ε)
C (ε)
E (ε)


D (ε)
C (ε)
E (ε)
F (ε)

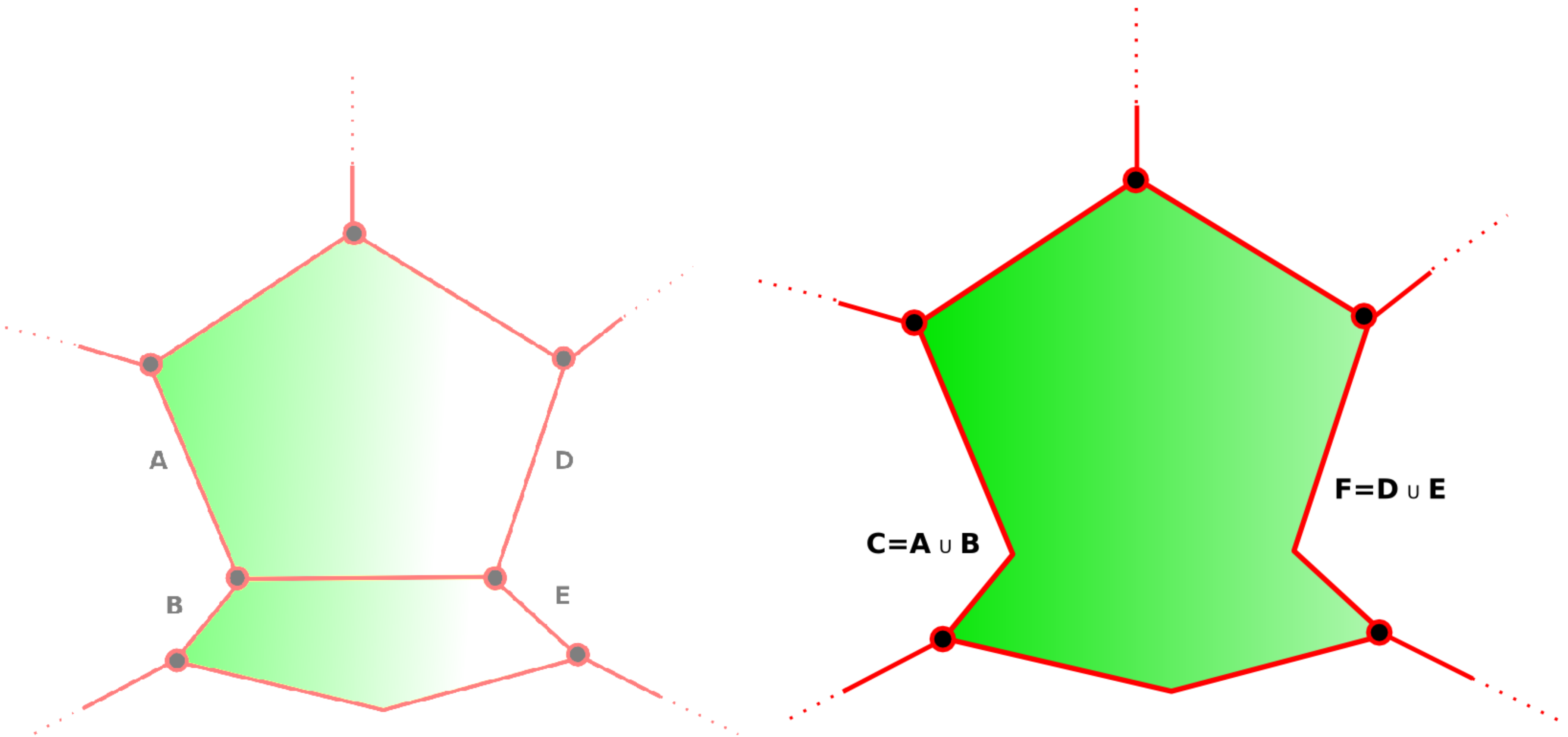

D (ε)
F (ε)
C (ε)
E (ε)


D (ε)
C (ε)
E (ε)
B (ε)
F (ε)
D (ε)
F (ε)
B (ε)
C (ε)
E (ε)

Combine into forest : Union edges



Combine into forest : Union edges



Background

- *What are the good things about having these data structures inside the database for generalization?*
  - Non-redundant storage
  - Makes it possible to use generalized data, avoiding need of real time computation:
    - Calculate once, store result, query, retrieve & visualize
  - Test bed for other research:
    - Suitable for progressive refinement (first send coarse data, then refine, until given criteria) & interface design [De Vries & Van Oosterom, 2006]

Results

- *How can the GAP-face tree and GAP-edge forest be implemented in an object-relational DBMS*
  - Spatial data types
  - Programming interface required
  - Extensible database : custom data type
  - Indexing: B-tree (numeric data) & R-tree (spatial data [in 2d and 3d!])

Results

- Implementation:
  - Tables & Indexes
  - Algorithm for filling tables (= building GAP-face tree)
  - Binary Line generalization:
    - Data type for holding BLG-trees + filling algorithm
    - Algorithm for recursively merging BLG-trees
    - Algorithm to get geometry from BLG-tree at certain depth
  - Reconstruction of geometry of topologically stored faces
  - Middle layer: database – Google Earth

Results

- *What problems occur when implementing the data structures inside the database?*
  - Minor changes needed in some algorithms
  - No indexing: no performance, due to huge amounts of data:
    - kadedge table (original): 178,815 entries
    - tgapedge table (GAP edge forest): 3,258,262 entries
  - Mapping 'importance' ↔ 'scale' ↔ 'BLG(ε)' is not linear: unknown relation, differ per dataset?
  - BLG might be self intersection: prohibit with constraints in calculation (known):
    - But, also with merging of BLG's!
- However, it works...


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
N
W
E
S
Image © 2006 TerraMetrics
© 2005
Google
Pointer 51°48'41.07" N 5°14'59.34" E
Streaming |||||||||| 100%
Eye alt 14060 ft


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


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

To do

- *How can we assess the performance of the implemented structure?*
  - First solve how to use the data structure efficiently, especially mapping between:
    - Scale
    - Importance
    - BLG tree: ε

- Write thesis

Conclusions

- *What do we learn from an implementation of the GAP-face tree and GAP-edge forest in an object relational management system?*
  - Approach is feasible
  - Possible to implement – with some minor adjustments to the theory as described in [Van Oosterom, 2005]
  - Calculate once, visualize & query real time
  - No redundancy of geometry

Questions / Discussion

Now (or later...)

- Martijn Meijers

- http://www.gdmc.nl/martijn/
- b.m.meijers@student.tudelft.nl
- +31-6-29 25 28 82