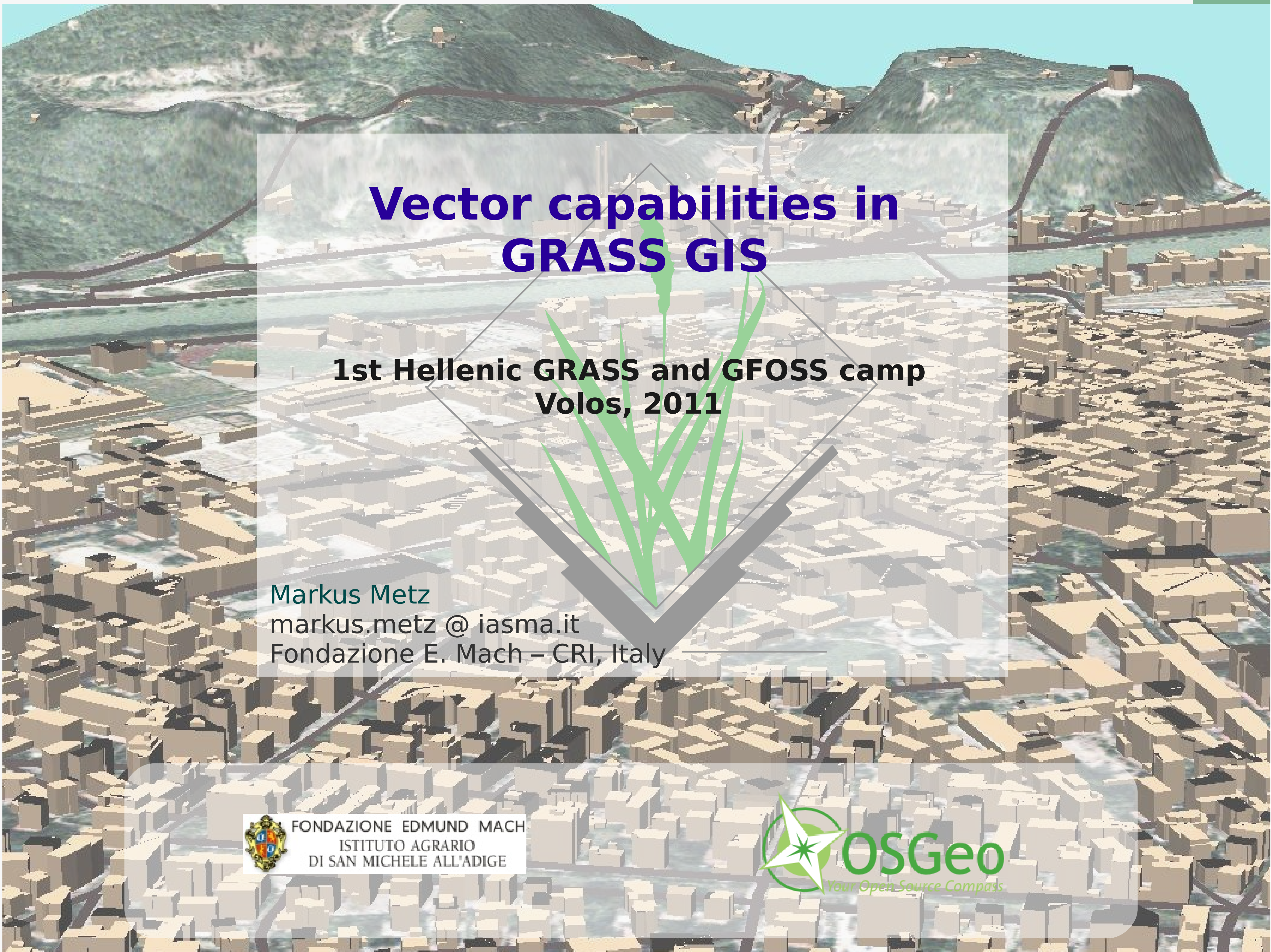


Vector capabilities in GRASS GIS

1st Hellenic GRASS and GFOSS camp
Volos, 2011

Markus Metz
markus.metz @ iasma.it
Fondazione E. Mach – CRI, Italy

FONDAZIONE EDMUND MACH
ISTITUTO AGRARIO
DI SAN MICHELE ALL'ADIGE

OSGeo
Your Open Source Compass

Presentation outline

- An introduction to vector topology
- Vector features in GRASS GIS
- Vector boundary operations
- Vector network analysis

Vector Topology

Non-topological vectors

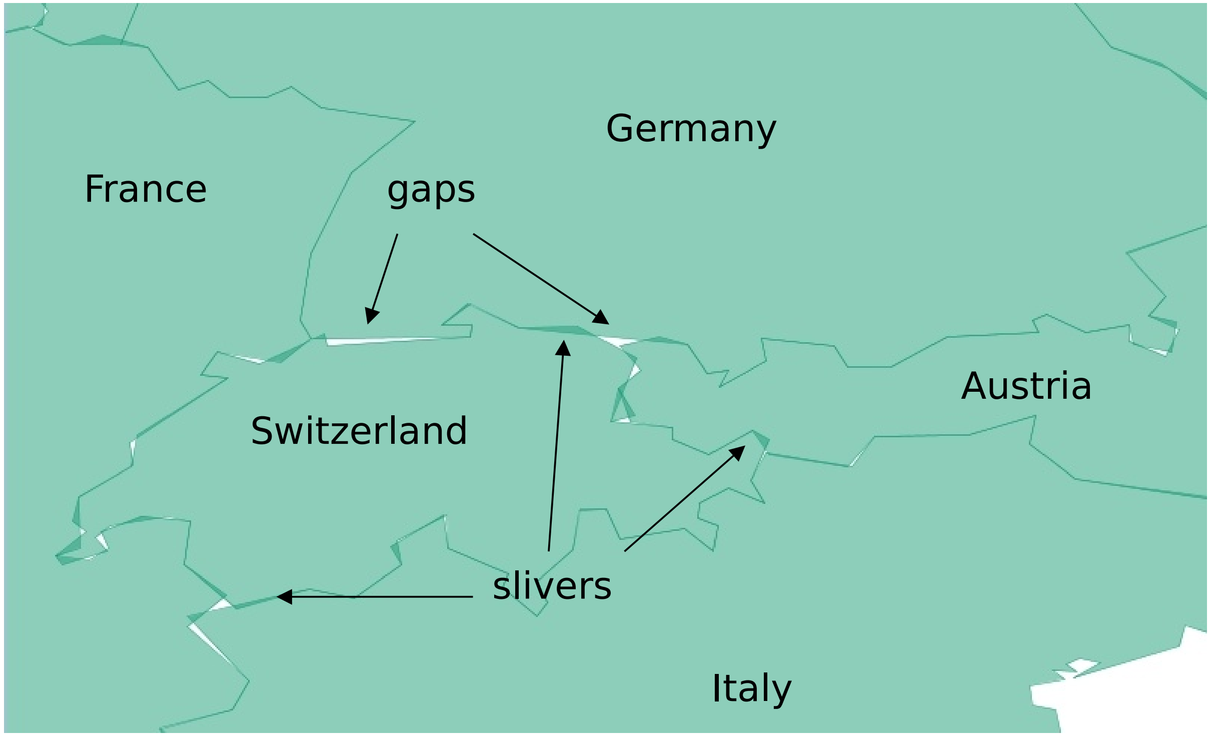
E.g. OGC Simple Features, ESRI shapefiles

Geometry types: points, lines, polygons

-> replicated boundaries for adjacent areas

Faster computations, but extra work for maintenance

Non-topological polygons generalized



Vector Topology

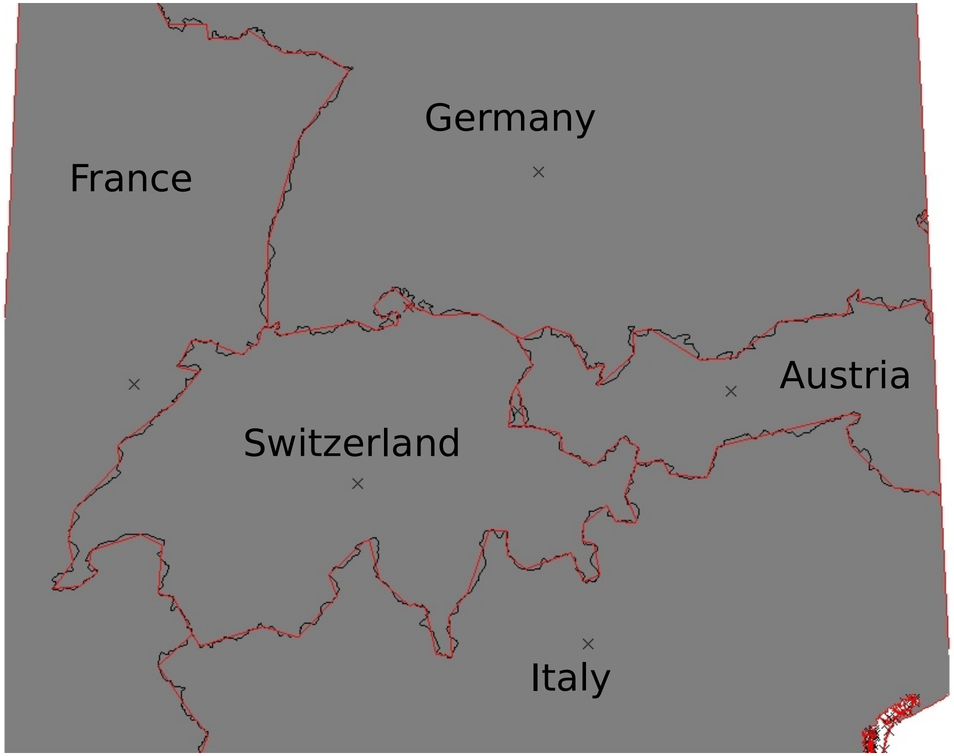
True vector Topology

Areas are constructed from boundaries

Boundaries are shared between adjacent areas

Slower computations, but less (nearly no manual) maintenance

Topological
boundaries generalized



Vector Topology

True vector Topology is implemented in e.g.

TNTmips

MApping Device –
Change Analysis Tool
(MAD-CAT)

GRASS GIS

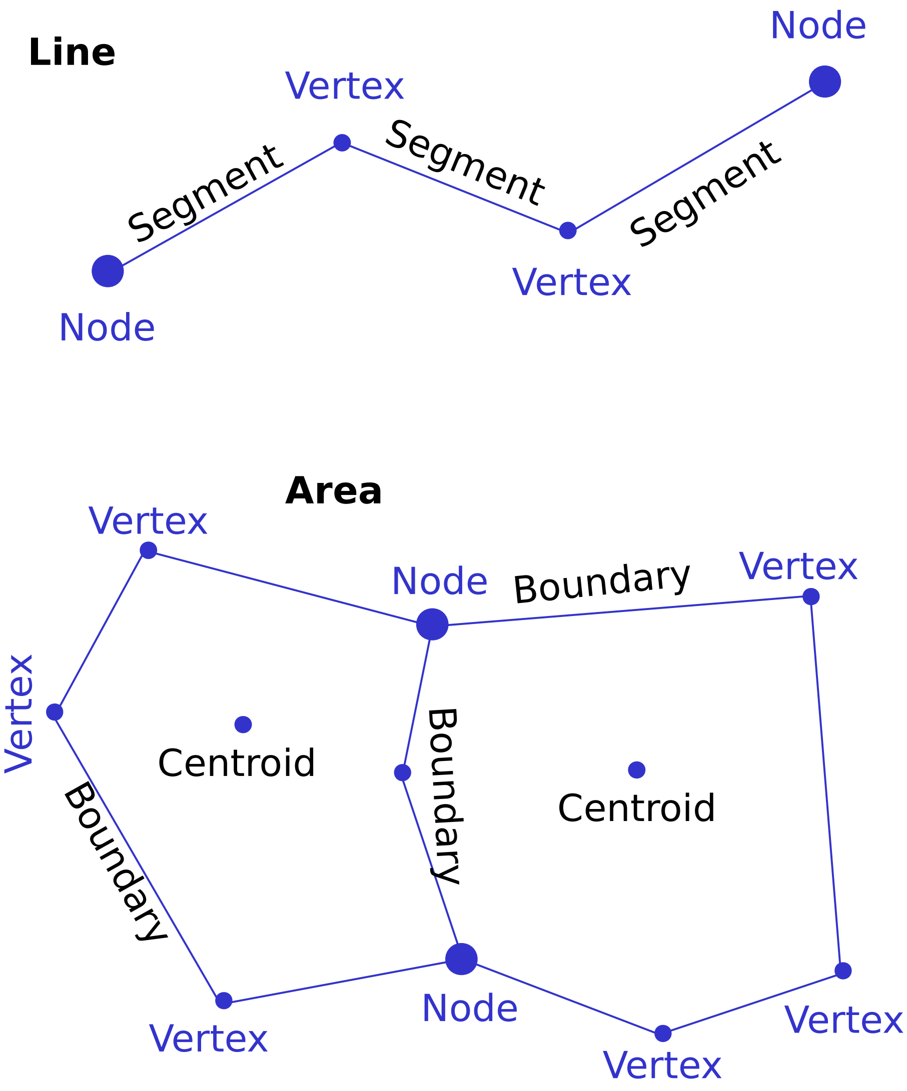
GRASS Vector model

Vector geometry types

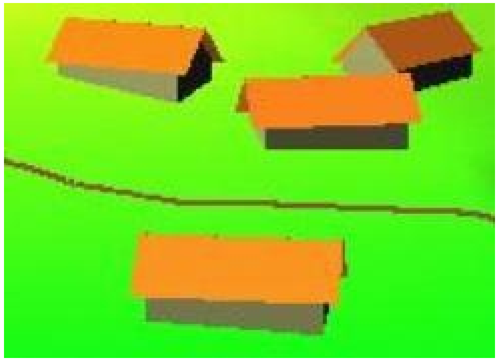
- Point
- Centroid
- Line
- Boundary
- Area (boundary + centroid)
- face (3D area)
- [kernel (3D centroid)]
- [volumes (faces + kernel)]

not in all GIS!

Geometry is **true** 3D when: x, y, z



Faces



Use of Spatial Index

GRASS Vector model

Vector geometry types

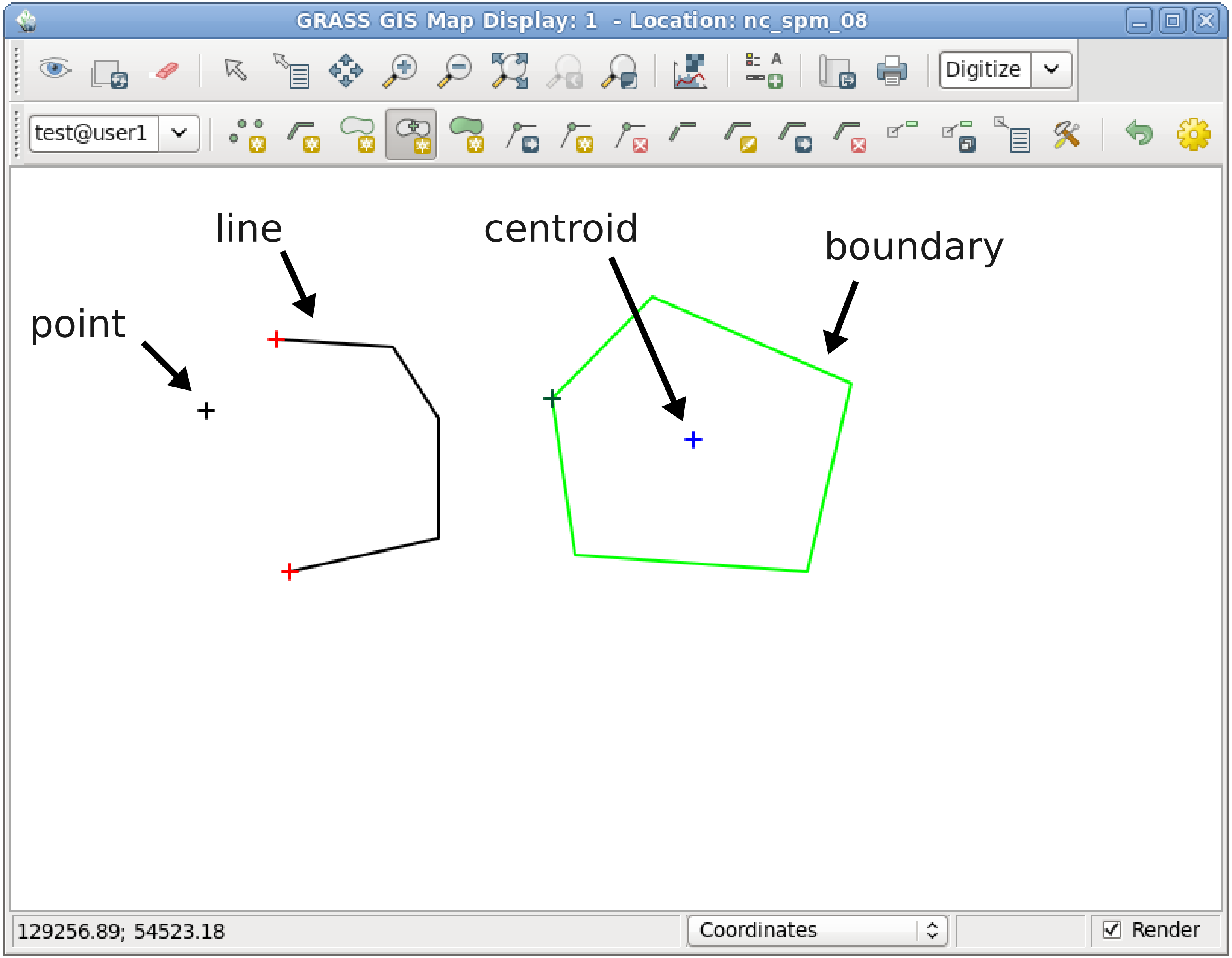
Basic geometry types, can be edited

- Point
- Centroid
- Line
- Boundary

A GRASS vector can contain a combination of several different types

GRASS Vector model

Vector geometry types



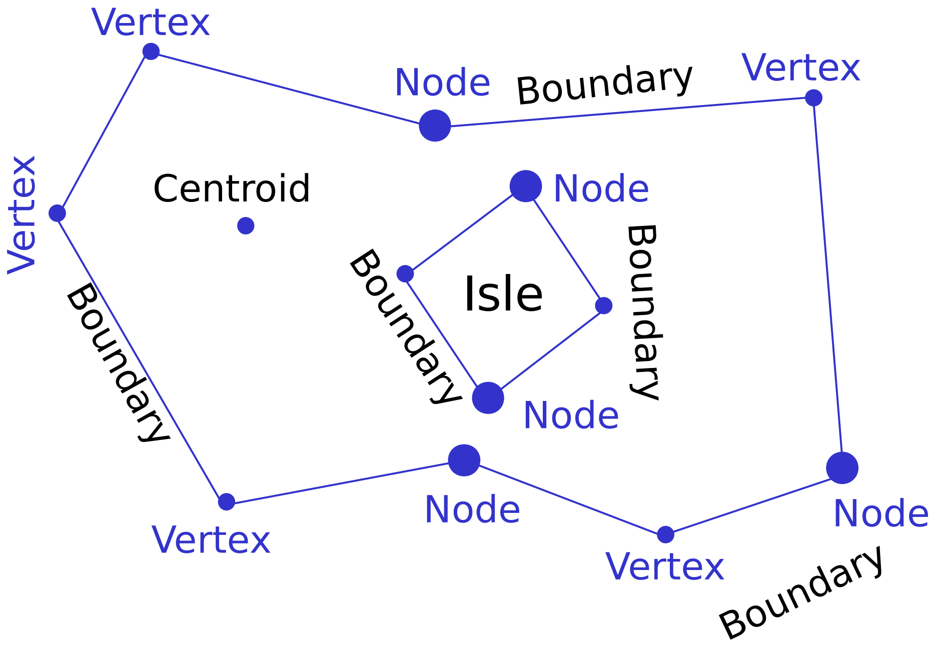
GRASS Vector model

Vector geometry types

Derived geometry types, constructed from basic types

- Area (closed ring of boundaries + centroid)
- Isle (closed ring of boundaries, no centroid)
- Node (at both ends of lines/boundaries; equal to points/centroids)

Isles and Nodes are not visible to users



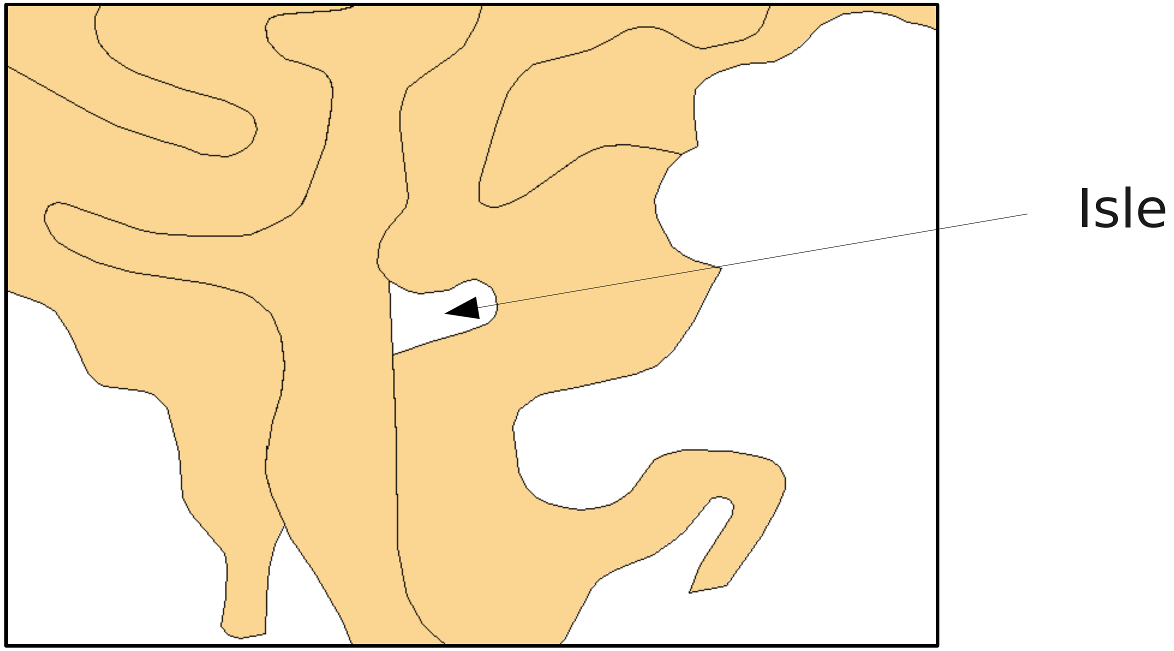
GRASS Vector model

Vector geometry types

Derived geometry types, constructed from basic types

- Area (closed ring of boundaries + centroid)
- Isle (closed ring of boundaries, no centroid)
- Node (at both ends of lines/boundaries; equal to points/centroids)

Isles and Nodes are not visible to users



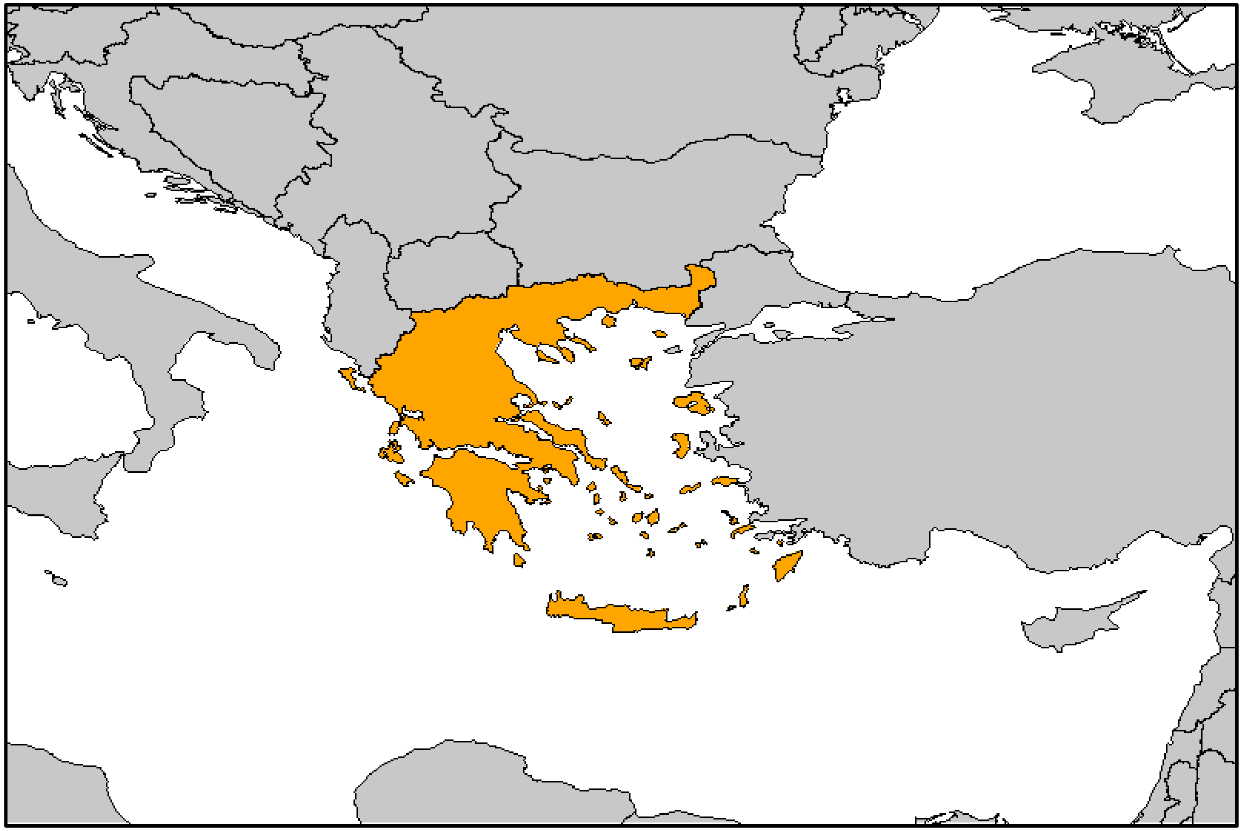
North Carolina, soils_wake@PERMANENT

GRASS Vector model: Categories

Basic geometry types can have categories

Unique categories: unique id

Shared categories equivalent to e.g. Multipolygon



GRASS Vector model: Categories

Reclassification

Converting unique categories to shared categories

```
v.reclass in=world_boundaries out=world_boundaries_country \
column=country

# unique categories
v.db.select map=world_boundaries columns=cat where="country = 'Greece'"
cat
1327
... [48 more category values]
1431

# grouped by country
v.db.select map=world_boundaries_country columns=cat \
where="country = 'Greece'"
cat
77
```

GRASS Vector model: Layers

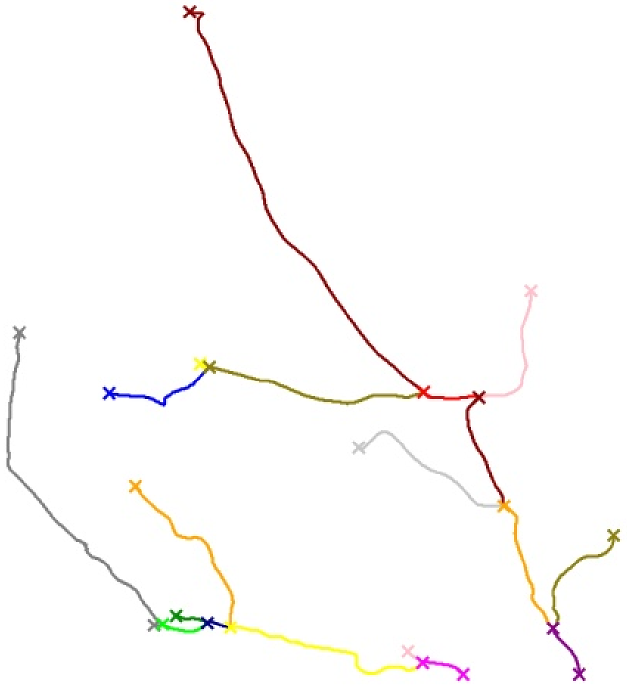
Layers ~ thematic groups

Each layer can have its own attribute table

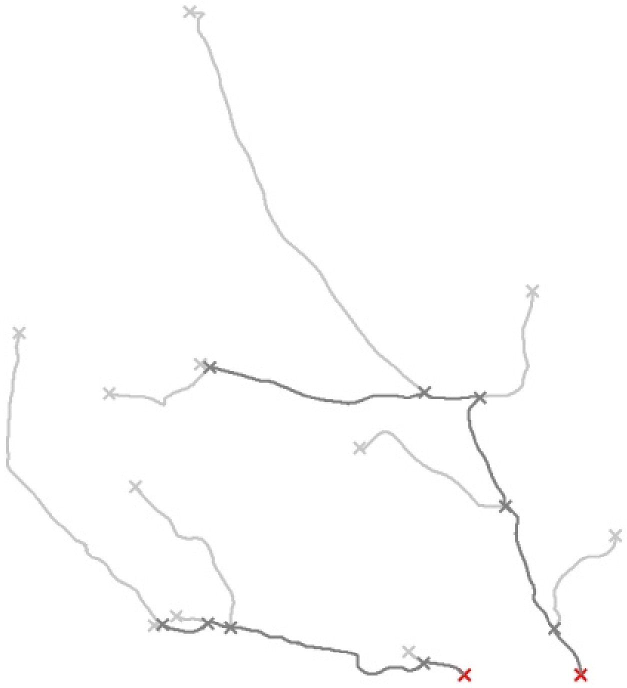
Example: river networks

Layer 1: unique stream ID

Layer 2: categories for stream head, intermediate stream, outlet

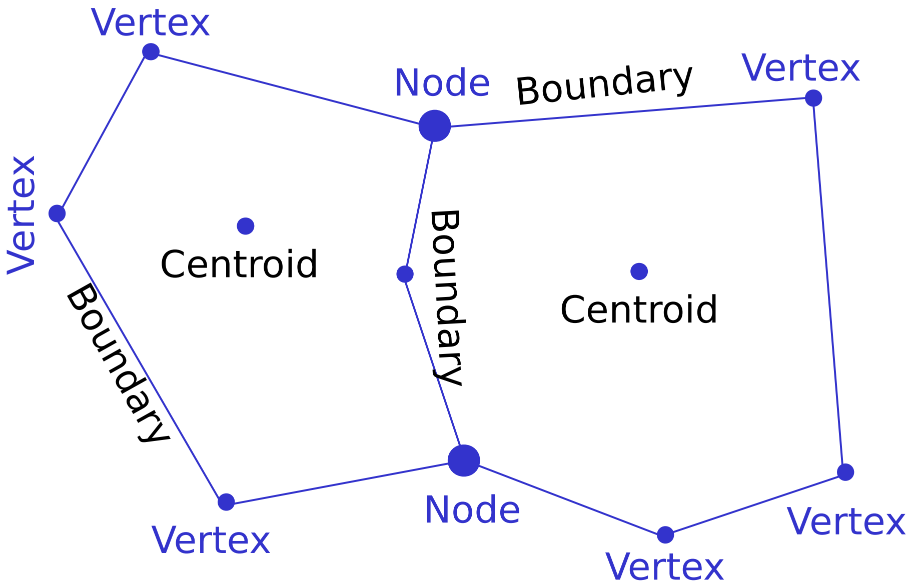


Layer 1: unique id



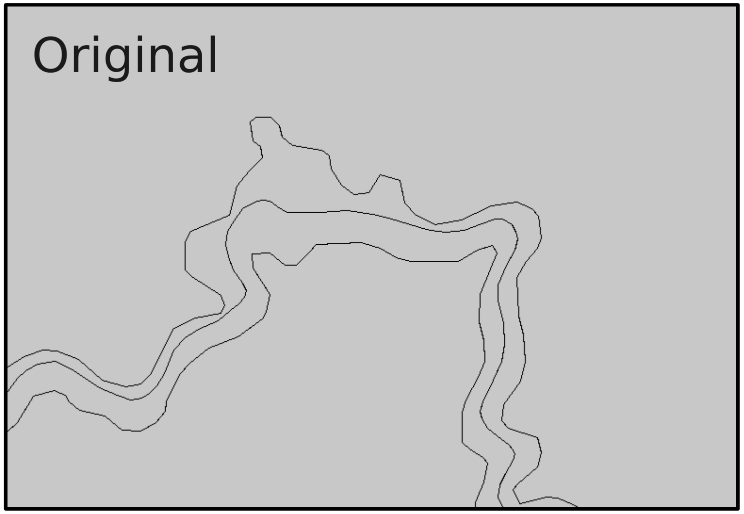
Layer 2: stream type

Vector boundary operations in GRASS GIS

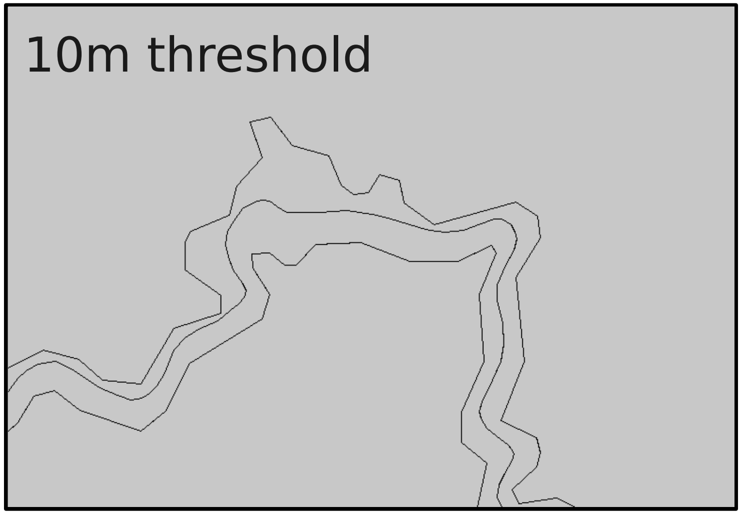


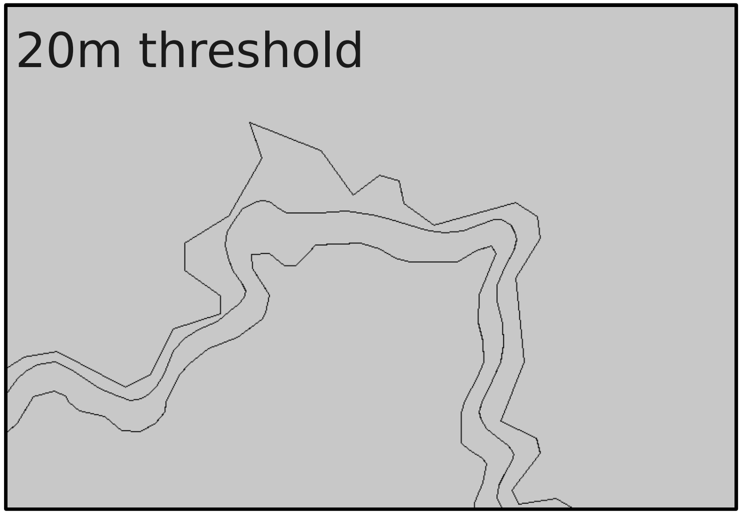
Vector boundaries: smoothing

North Carolina: boundary_county



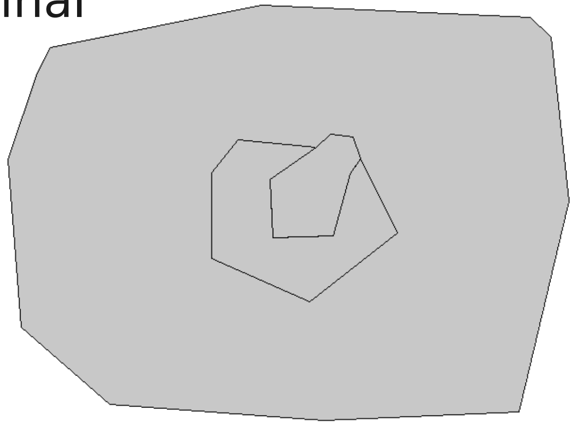
```
v.clean in=boundary_county \
out=boundary_county_smooth_10 \
tool=prune thres=10.00
```





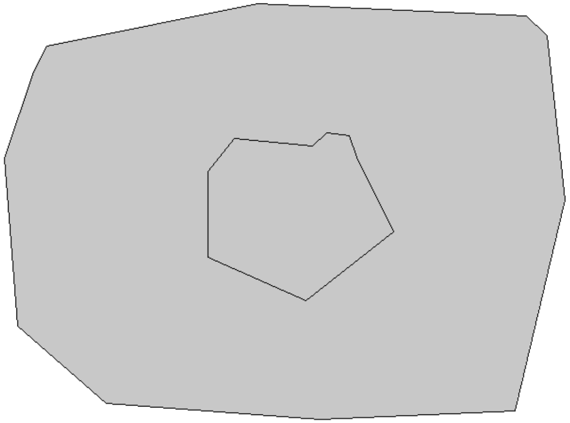
Vector boundaries: removing small areas

Original

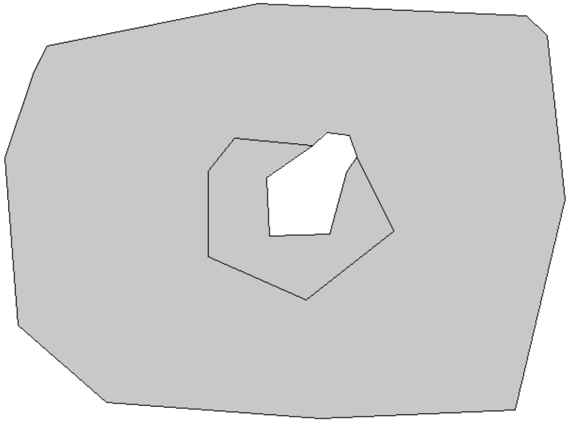


Removing the smallest area in the center

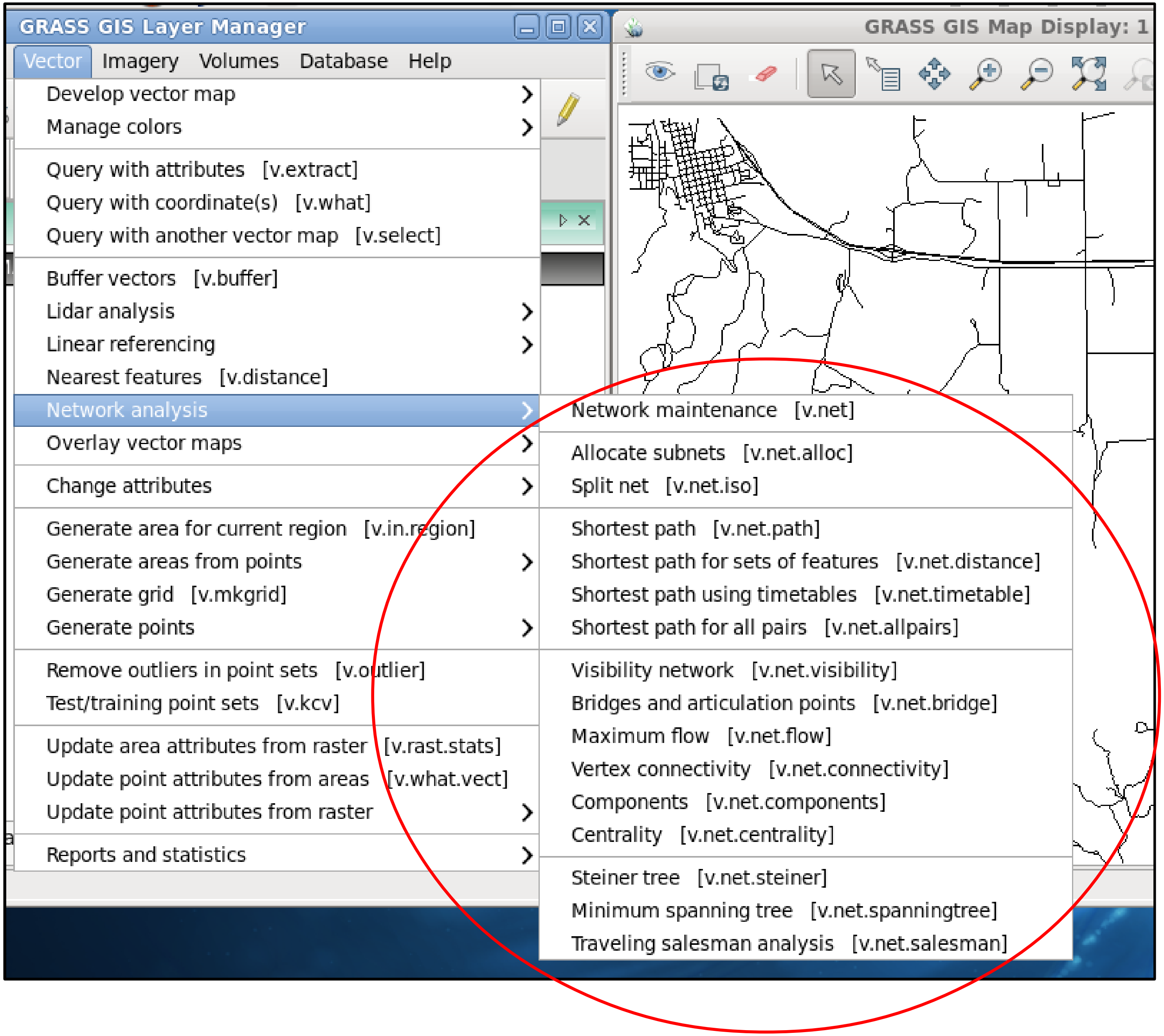
topological



non-topological

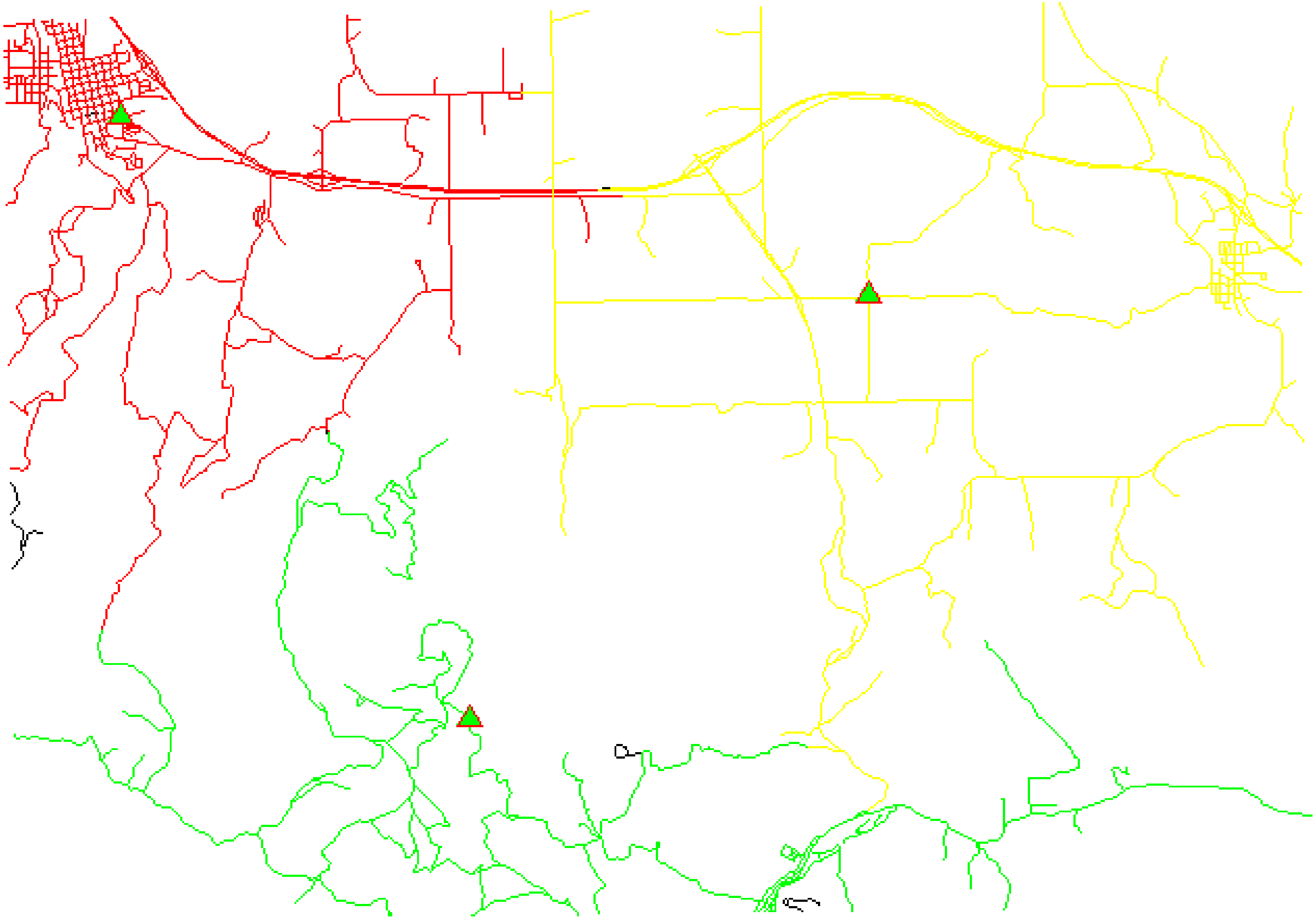


Vector network analysis in GRASS GIS


GRASS GIS Layer Manager
Vector
Imagery
Volumes
Database
Help
Develop vector map
Manage colors
Query with attributes [v.extract]
Query with coordinate(s) [v.what]
Query with another vector map [v.select]
Buffer vectors [v.buffer]
Lidar analysis
Linear referencing
Nearest features [v.distance]
Network analysis
Overlay vector maps
Change attributes
Generate area for current region [v.in.region]
Generate areas from points
Generate grid [v.mkgrid]
Generate points
Remove outliers in point sets [v.outlier]
Test/training point sets [v.kcv]
Update area attributes from raster [v.rast.stats]
Update point attributes from areas [v.what.vect]
Update point attributes from raster
Reports and statistics
Network maintenance [v.net]
Allocate subnets [v.net.alloc]
Split net [v.net.iso]
Shortest path [v.net.path]
Shortest path for sets of features [v.net.distance]
Shortest path using timetables [v.net.timetable]
Shortest path for all pairs [v.net.allpairs]
Visibility network [v.net.visibility]
Bridges and articulation points [v.net.bridge]
Maximum flow [v.net.flow]
Vertex connectivity [v.net.connectivity]
Components [v.net.components]
Centrality [v.net.centrality]
Steiner tree [v.net.steiner]
Minimum spanning tree [v.net.spanningtree]
Traveling salesman analysis [v.net.salesman]
GRASS GIS Map Display: 1

Vector network analysis in GRASS GIS

Vector network analysis in GRASS GIS



Network analysis

General concept of a network graph

- Arcs connected by nodes
- Forward/backward costs assigned to each arc (oneway road)
- Starting point(s)
- Ending point(s)

Cost definition examples

- Distance ➔ shortest path
- Travelling time ➔ fastest path
- Travelling costs (fuel, train ticket, etc) ➔ cheapest path

Network analysis: traveling salesman

Distances as costs

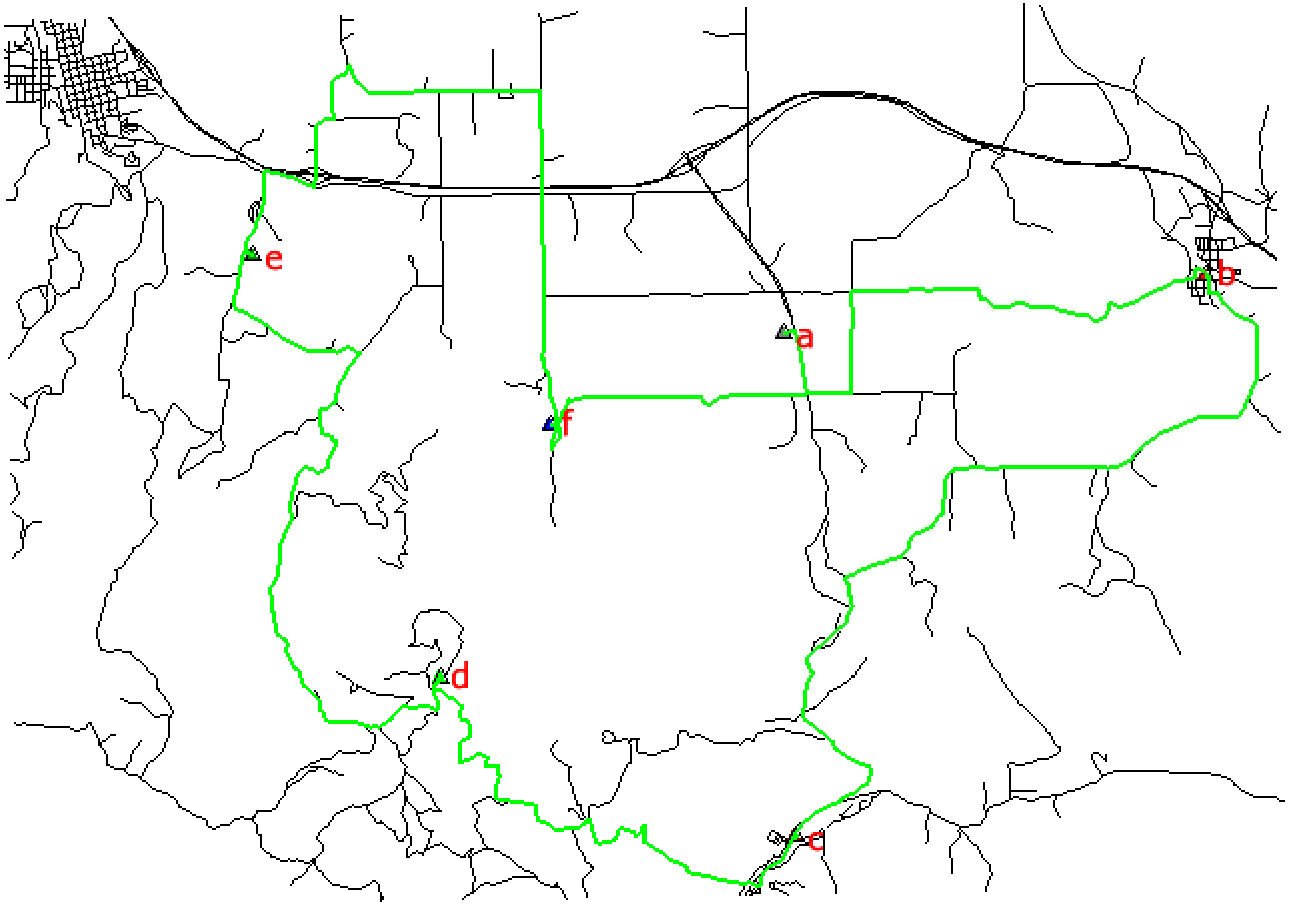
Spearfish example

```
# we want to vist 6 locations on our trip

echo "1|601653.5|4922869.2|a
2|608284|4923776.6|b
3|601845|4914981.9|c
4|596270|4917456.3|d
5|593330.8|4924096.6|e
6|598005.5|4921439.2|f" | v.in.ascii cat=1 x=2 y=3 out=centers \
col="cat integer, east double precision, \
north double precision, label varchar(43)"

# prepare network
g.copy vect=roads,myroads
v.net myroads points=centers out=myroads_net op=connect \
thresh=500

v.net.salesman myroads_net ccats=1-6 out=mysalesman_length
```

Network analysis: traveling salesman

Distances as costs

Result



Network analysis: traveling salesman

Traveling time as costs

```
# create unique categories for each line in layer 2
v.category in=myroads_tmp out=myroads opt=add cat=1 layer=2

# add new table for layer 2
v.db.addtable myroads layer=2 col="cat integer, label
varchar(43),length double precision,speed double precision,cost
double precision"

# copy road type to layer 2
v.to.db myroads layer=2 qlayer=1 opt=query qcolumn=label
columns=label

# create lines map connecting points to network (take care of
layers)
v.net myroads points=centers out=myroads_net op=connect
thresh=500 alayer=2 nlayer=1
```

Network analysis: traveling salesman

Traveling time as costs

Road type	Speed limit
Interstate	75 mph
Primary highway, hard surface	75 mph
Scondary highway, hard surface	50 mph
Light-duty road, improved surface	25 mph
Unimproved road	5 mph

Network analysis: traveling salesman

Traveling time as costs

```
# define traveling costs as length in miles divided by speed
limit in miles per hour:

v.to.db map=myroads_net layer=2 type=line option=length
col=length unit=miles

# set speed limits in miles / hour
v.db.update myroads_net layer=2 col=speed val="5.0"
v.db.update myroads_net layer=2 col=speed val="75.0"
where="label='interstate'"
v.db.update myroads_net layer=2 col=speed val="75.0"
where="label='primary highway, hard surface'"
v.db.update myroads_net layer=2 col=speed val="50.0"
where="label='secondary highway, hard surface'"
v.db.update myroads_net layer=2 col=speed val="25.0"
where="label='light-duty road, improved surface'"
v.db.update myroads_net layer=2 col=speed val="5.0"
where="label='unimproved road'"
```

Network analysis: traveling salesman

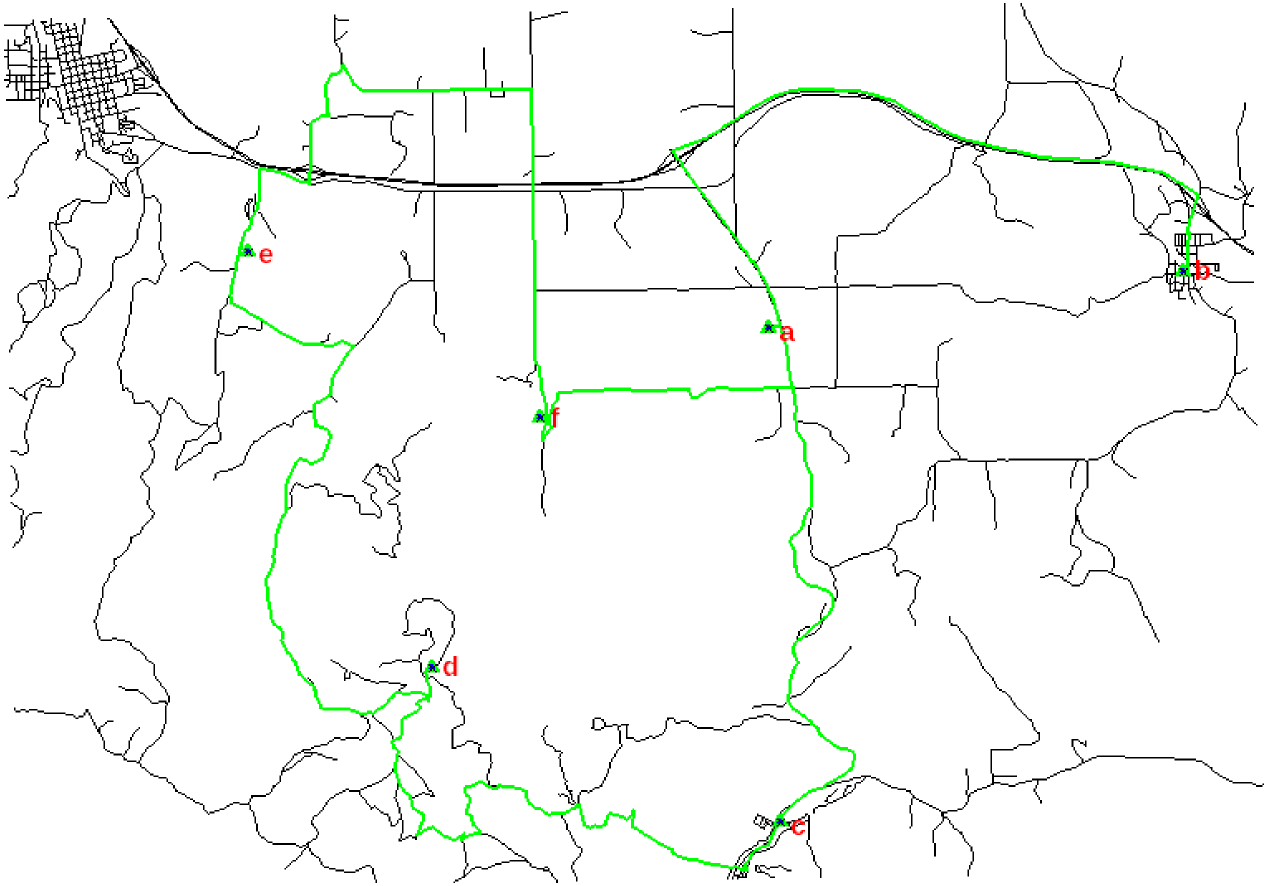
Traveling time as costs

```
# set costs as traveling time in hours
v.db.update myroads_net layer=2 col=cost val="length / speed"

# fastest path: traveling costs = length / speed
v.net.salesman myroads_net alayer=2 nlayer=1 acol=cost ccats=1-6
out=mysalesman_fastest
```

Network analysis: traveling salesman

Traveling time as costs

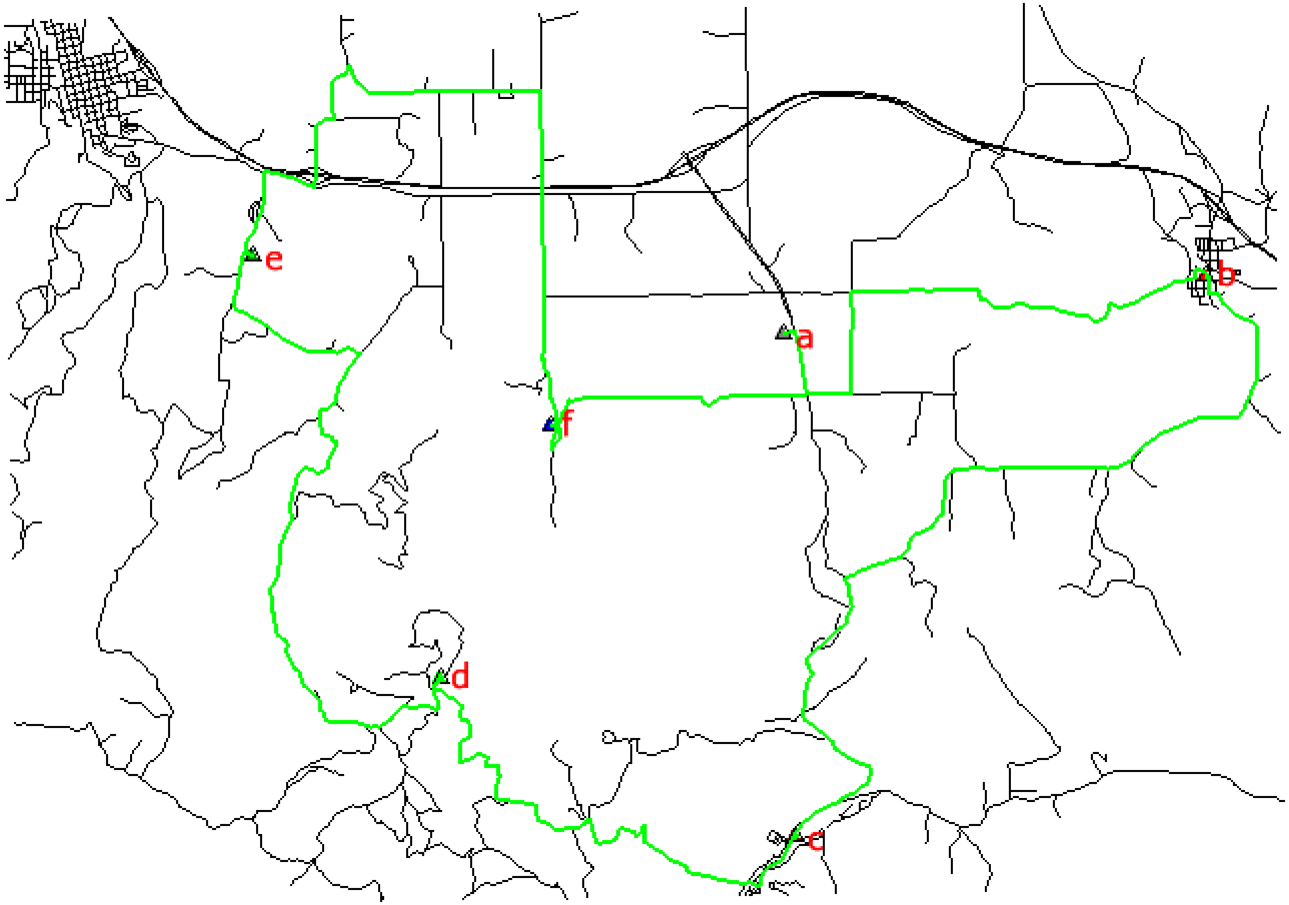
Result



Network analysis: traveling salesman

Distances as costs

Result



Thank you
for your attention

Markus Metz
markus.metz @ iasma.it
Fondazione E. Mach – CRI,
Italy