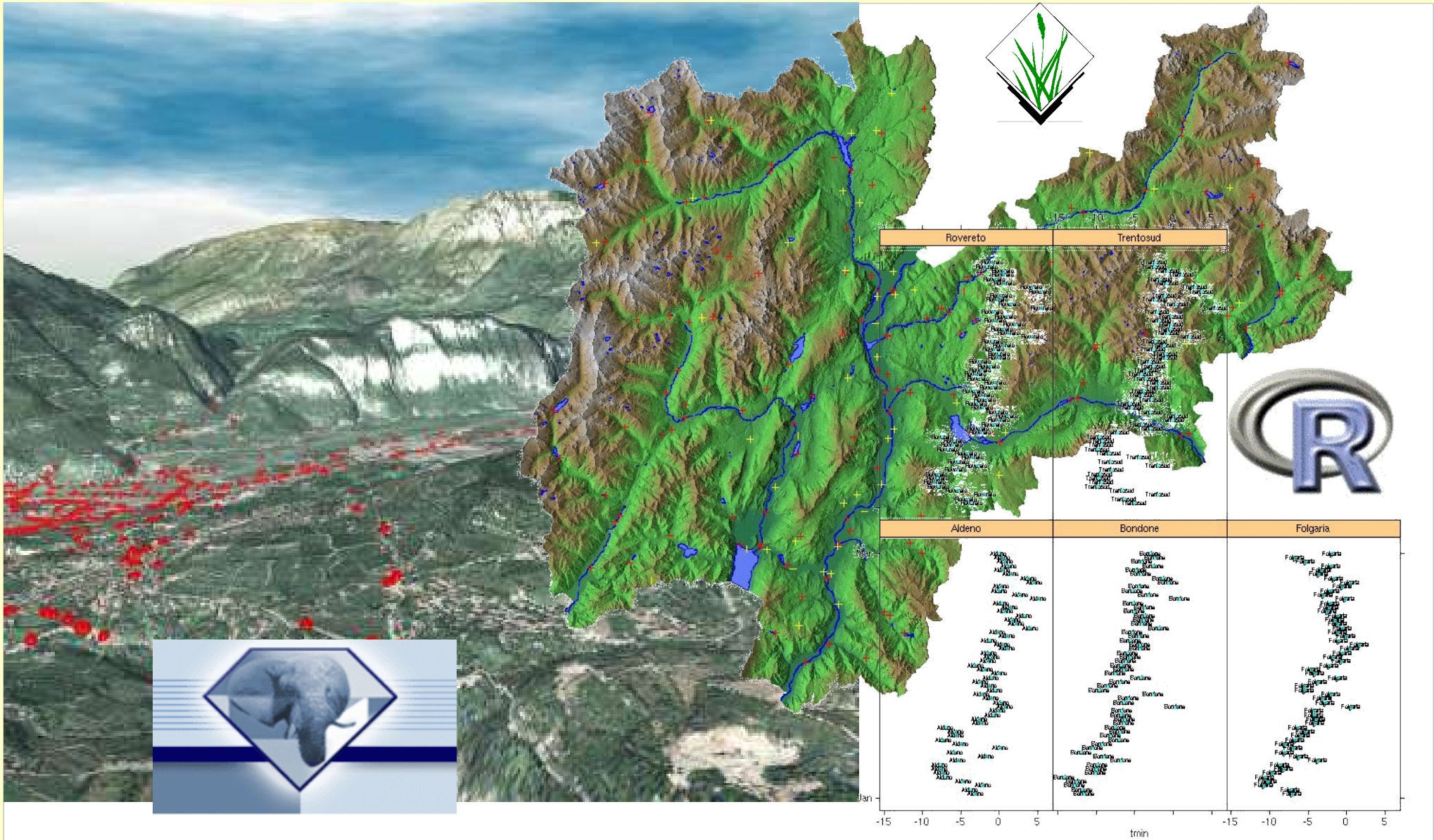
Advanced Data Bases
Part 4: Vector network analysis

Markus Neteler, ITC-irst, 2004

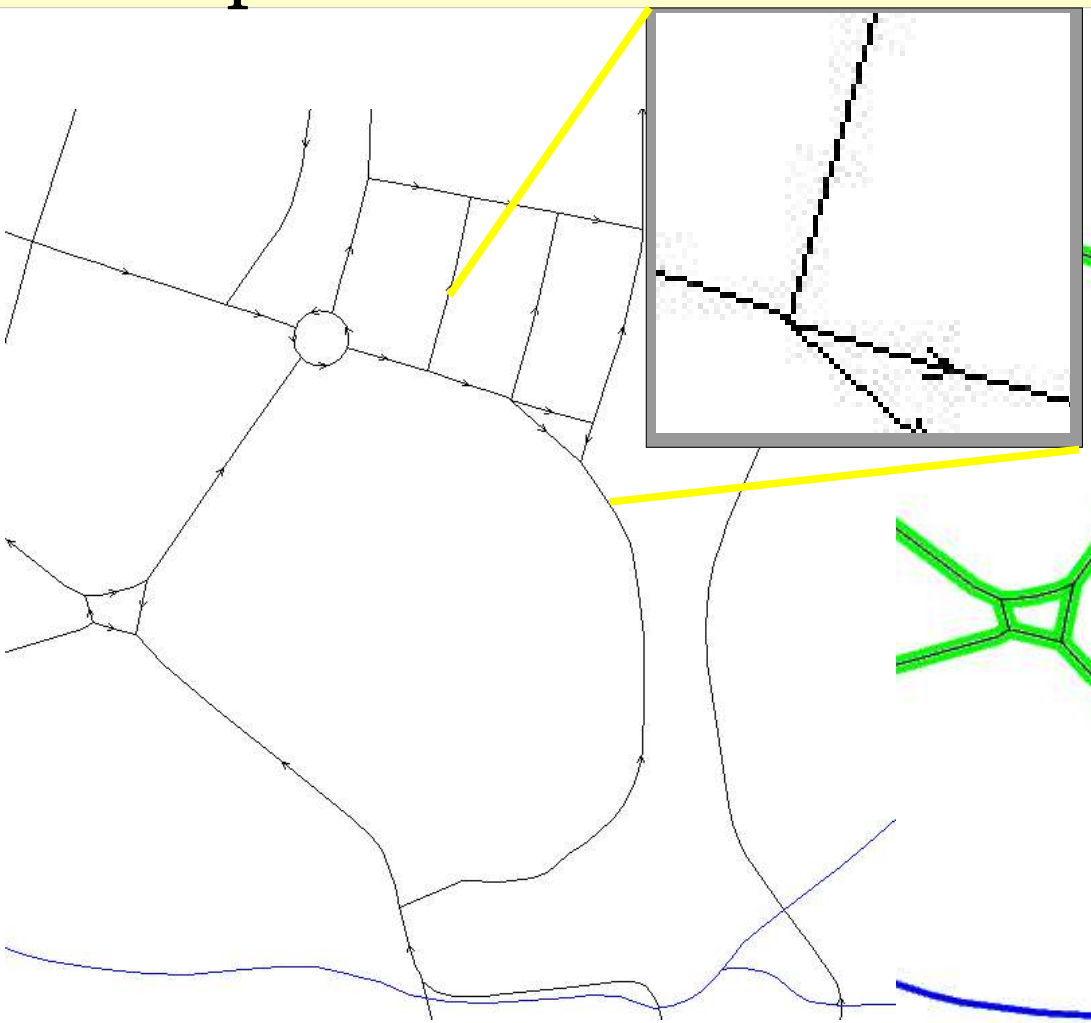


Markus Neteler, ITC-irst 2004

Outline: Vector Network Analysis

- Finding shortest paths on vector networks (path)
- Creation of a cyclic connecting of given nodes (Traveling salesman problem)
- Calculation of optimal connection of network nodes (Minimum Steiner tree)
- Subdivision of a network into subnetworks (Iso distances)
- Allocation of subnets for nearest centers (Subnet allocation)

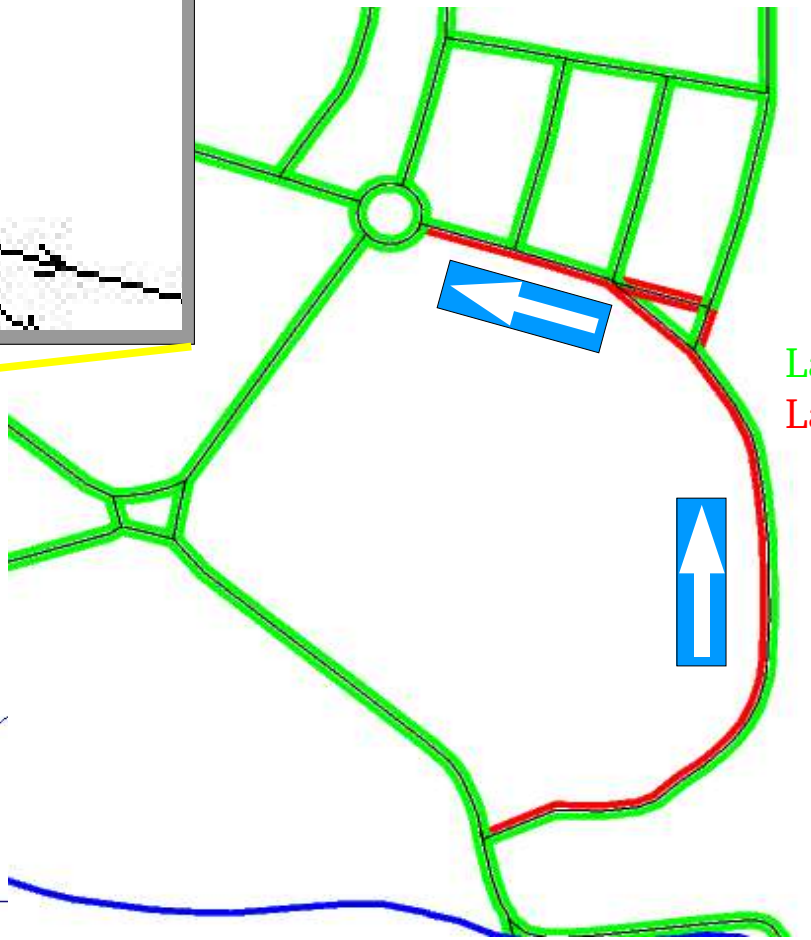
Shortest path



Vector network with directions

ps.map

Lane open
Lane closed

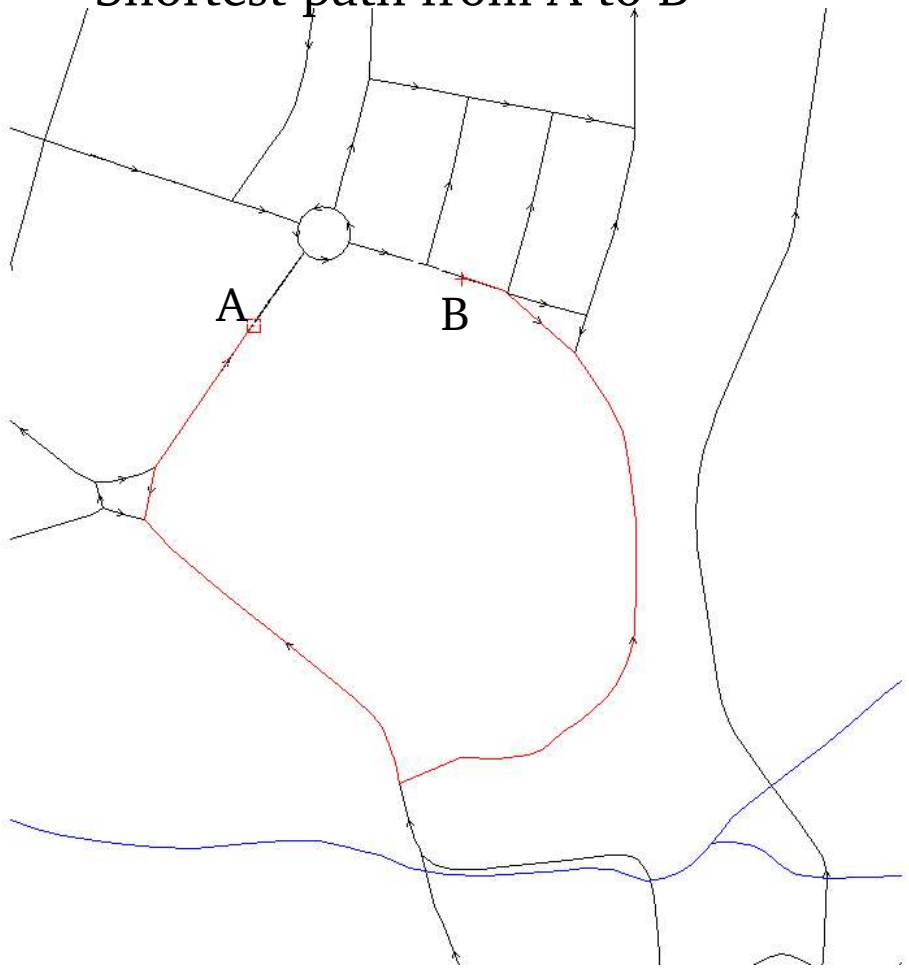


For each direction an attribute column
Value -1 closes direction (one way street)

Shortest path: d.path, v.net.path

Vector network with directions

Shortest path from A to B



Shortest path based on attributes defining one way roads

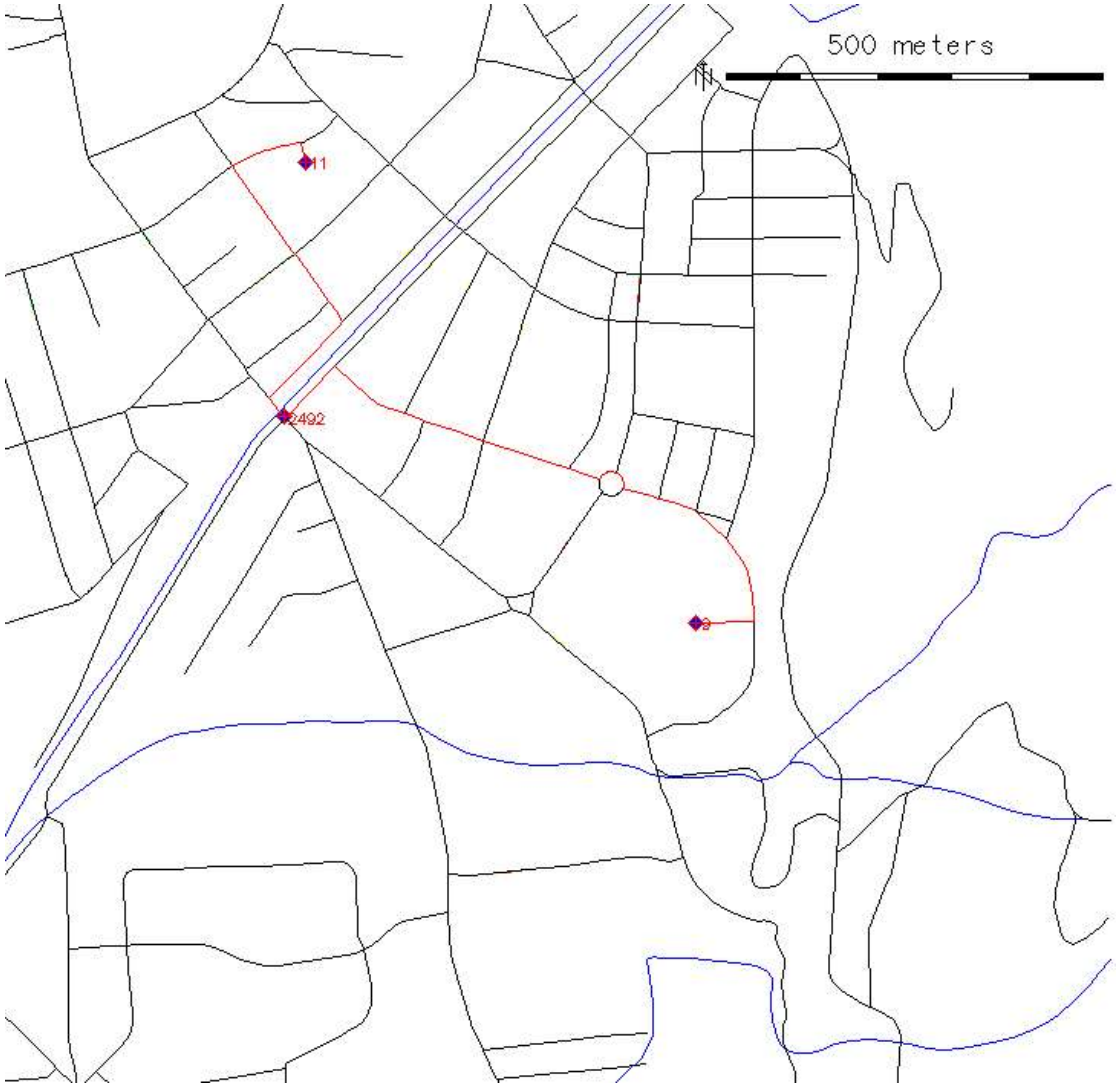
Traveling salesman problem: v.net.salesman

Based on vector length

Can be also based on speed or other attributes, single column

Example:
Salesman who has to deliver a product in one day to several shops

Optimal connection: v.net.steiner



Based on vector length

Can be also based on speed or other attributes, single column

Example:
High-speed internet connection between hospitals along road network

Subnetworks by Iso-distances: v.net.iso

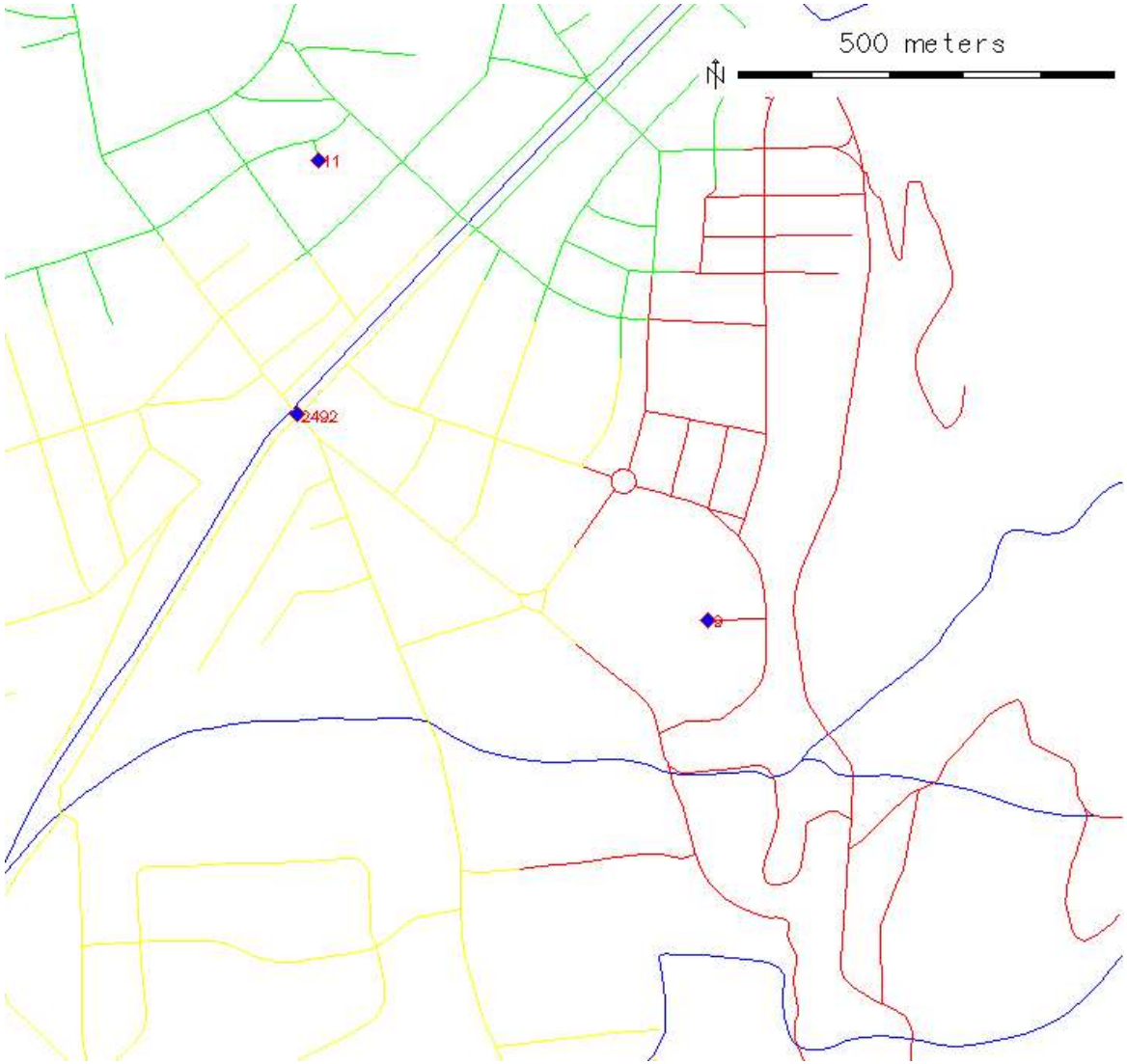
Iso-subnetwork

Distances:
500m green
1000m yellow
2500m red

Can be also based on speed or other attributes

Example:
Reachability of surroundings from given starting points

Allocation of subnets: v.net.alloc



Based on vector length

Can be also based on speed or other attributes, single column

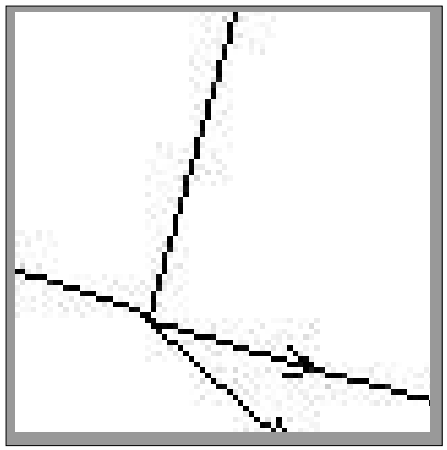
Example: responsability areas for local police stations, dividing the area into subbasins

Excercise: Required data sets

- Street network 'grafo'
- Hospital coordinates 'ospedali'
- Import with `v.in.ogr`

Excercise: Shortest path with one way streets

Street network 'grafo'



1. copy the map 'grafo' to a new name into your mapset:
   g.copy vect=grafo,grafo_USERID

2. Display manager, load 'grafo_USERID'

   Display vector directions

3. Check Attribute table for potential useful columns
   -> nothing

   Try 'd.path' module for simple shortest path:
   d.path grafo_USERID

Excercise: Shortest path with one way streets

Table preparation I: Edit attribute table with OpenOffice

- we add two columns to describe possible speed in each direction.
- we add 1 column to describe the vector length

Start OpenOffice without parameters, then:

File -> New -> Text Document (only needed to see 'Tools' menu!)
Tools -> Data Sources
-> New Data Source
[General]
* Database type: dBase
* Data source URL [Browser: ...]
-> select directory from
grassdata/<LOCATION>/<MAPSET>/dbf/
[Tables]
* check if the table 'grafo_USERID' is visible
-> OK button

Excercise: Shortest path with one way streets

Table preparation II with OpenOffice

View -> Data
* Sources
-> select your data source (probably "Data source 1")
* Tables
* your table 'grafo_USERID'
* click right mouse button on table
-> Edit table
-> add three columns:
* "forward", type "Decimal", 2 decimal places
* "backward", type "Decimal", 2 decimal places
* "length", type "Decimal", 2 decimal places

-> Save the table.

'Exit' from OpenOffice.

Alternate method to generate table

Create new table without OpenOffice:

```
#remove old one:
echo “drop table grafo” | db.execute

echo “create table grafo ( cat integer, forward double,
      backward double, length double)” | db.execute

# Check table link to map with:
v.db.connect -p grafo_USERID
```

Editing attributes in a new (!) table

```
#generate new rows for each vector category (ID):
v.to.db  grafo_USERID option=cat

echo “select * from grafo_USERID” | db.select
# ... should report a row for each vector
```

Excercise: Shortest path with one way streets

Editing attributes in the updated table

Bulk-assign speed attributes (we don't want to edit 2000 lines manually):

```
echo "select * from grafo_USERID" | db.select

echo "update grafo_USERID set forward=50 where forward=0" | db.execute
echo "update grafo_USERID set backward=50 where backward=0" | db.execute

echo "select * from grafo_USERID" | db.select
```

Bulk-assign vector length in kilometer:

```
v.to.db grafo_USERID option=length col1=length units=kilometers
echo "select * from grafo_USERID" | db.select
```

Excercise: Shortest path with one way streets

Graphical modification of attributes in the map

Edit the vector network to set speed or define one way streets:

d.vect mygrafo disp=shape,dir

d.m: OPTIONS -> "TOGGLE FORM MODE" to switch on/off editing

d.what.vect
- speed: enter value (positive number)
- one way street: -1

Be sure to have visible the vector directions (d.vect, d.m) because the 'forward' and 'backward' column must be set appropriately.

Use 'SUBMIT' button to set the values for the vector in the DBMS.

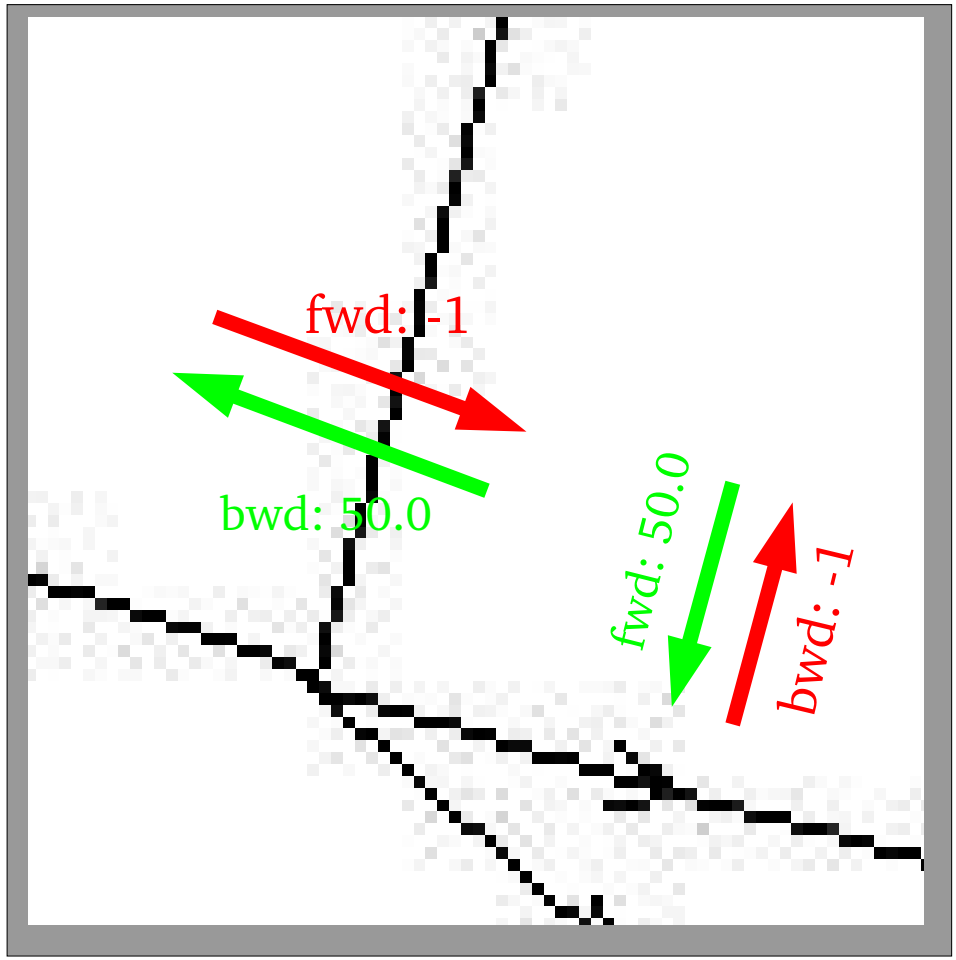
Excercise: Shortest path with one way streets

Graphical modification of attributes in the map

One way streets:
d.what.vect
- speed: enter value (positive number)
- one way street: -1

Forward is indicated by
the direction arrow on the line.
Check it for every line!

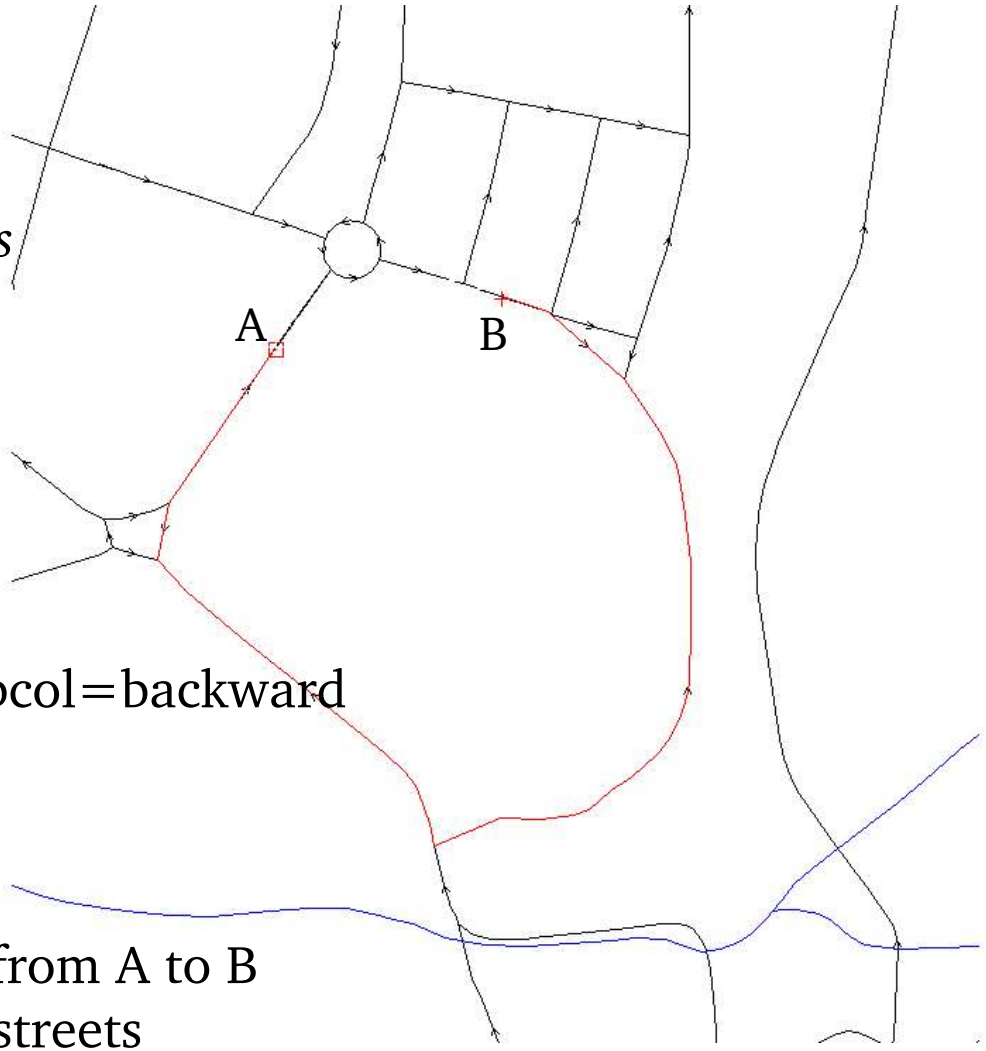
Block two directions by
-1 for both forward/backward.



Excercise: Shortest path with one way streets

Testing the new map:

d.path grafo_USERID
...should ignore the one way streets



But:
d.path mygrafo afcol=forward abcol=backward
... will respect one way streets

Shortest path from A to B
with one way streets

Excercise: Shortest path with one way streets

Command based networking:

If that all works, try v.net.path instead:
- input are the point category numbers
- set layer=1 if there is no extra table for nodes
- d.what.vect tells you the cats of the nodes

E.g., from node 9 to 12 (we add an ID as first number):

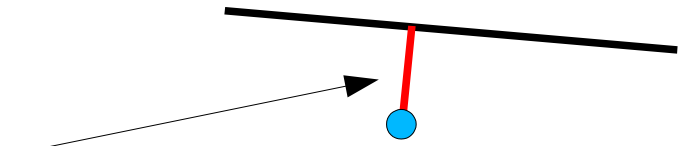
```
echo "1 9 12" | v.net.path mygrafo afcol=forward abcol=backward out=mypath layer=1
```

The result is saved to a new map.

Excercise: Alloc, Iso, Steiner, ...

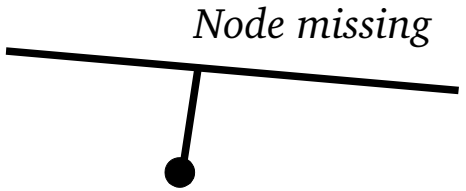
Data preparation:

We have to connect the hospitals to the street network:



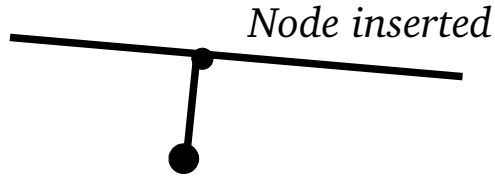
#create lines map connecting points to network
v.distance -p from=ospedali to=mygrafo out=connect upload=dist column=dist
d.vect connect col=red

#merge all maps into a new one: streets, connections, points:
v.patch mygrafo,connect,ospedali out=mygrafo_ospedali



#fix the topology: break lines, snap to nodes:
v.clean mygrafo_ospedali output=mygrafo_ospedali_clean tool=break,snap

g.remove vect=mygrafo_ospedali
g.copy vect=mygrafo_ospedali_clean,mygrafo_ospedali
g.remove vect=mygrafo_ospedali_clean



d.vect mygrafo_ospedali

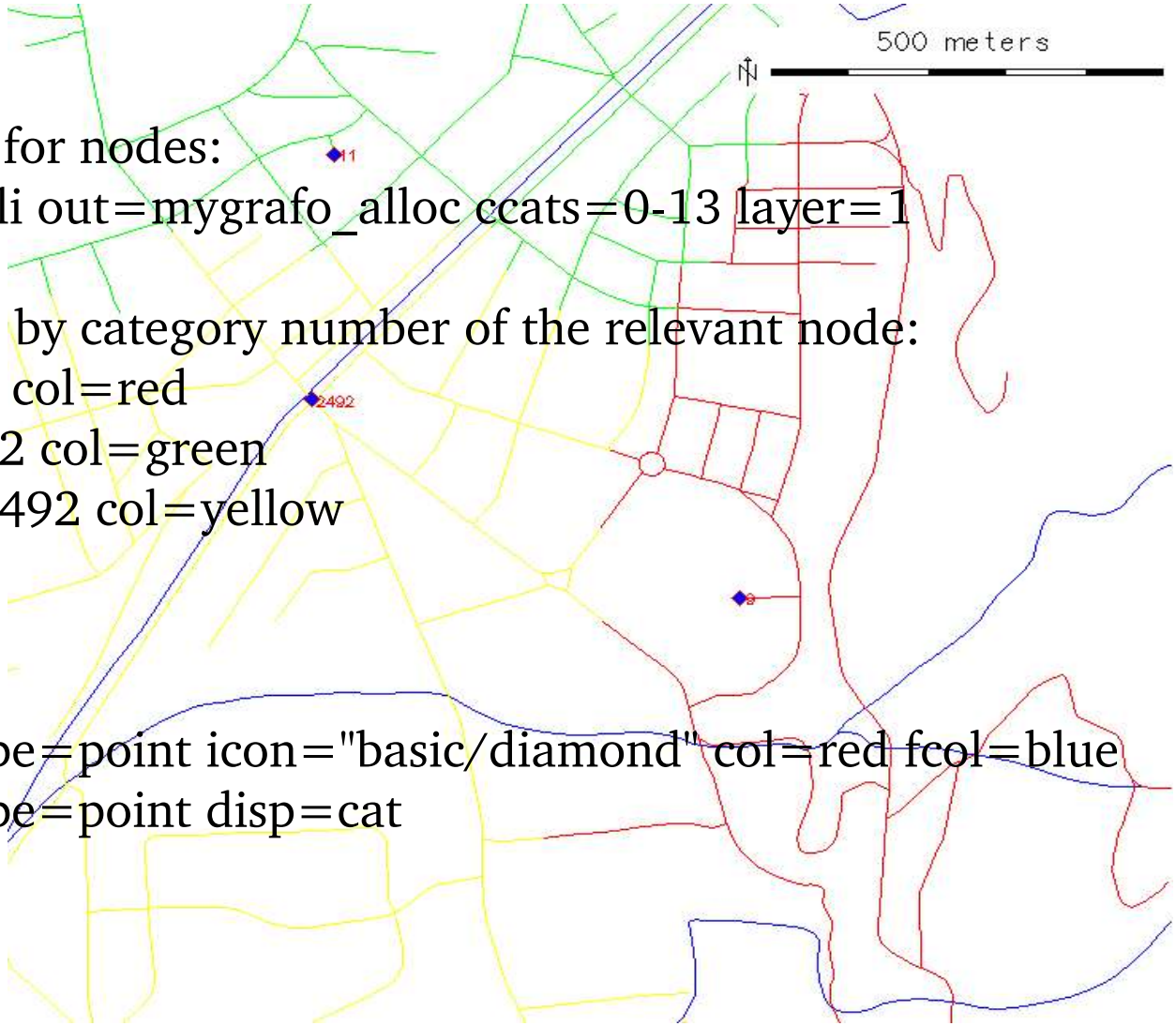
Excercise: Alloc, Iso, Steiner, ...

Data preparation II: Adding attribute table

```
# create new table:
 echo "create table mygrafo_ospedali ( cat integer, grafoid integer , ospid integer )" | db.execute

# link table to map:
 v.db.connect mygrafo_ospedali table=mygrafo_ospedali key=cat driver=dbf \
                              database='$GISDBASE/$LOCATION_NAME/$MAPSET/dbf'

# add category numbers to table (creates new row for each vector line):
 v.to.db mygrafo_ospedali type=line option=cat col1=grafoid

# let's look at the result:
 d.erase
 d.vect mygrafo_ospedali
 d.vect mygrafo_ospedali type=point col=red
 d.what.vect
```

Excercise: v.net.alloc

```
# nfield=1 if no extra table for nodes:
 v.net.alloc mygrafo_ospedali out=mygrafo_alloc ccats=0-13 layer=1

# the result is to be selected by category number of the relevant node:
 d.vect mygrafo_alloc cat=9 col=red
 d.vect mygrafo_alloc cat=12 col=green
 d.vect mygrafo_alloc cat=2492 col=yellow

# add some nice maps:
 d.vect idrfiu col=blue
 d.vect mygrafo_ospedali type=point icon="basic/diamond" col=red fcol=blue
 d.vect mygrafo_ospedali type=point disp=cat
 d.barscale -tm
```



Excercise: v.net.iso, v.net.steiner, ...

Try the same with v.net.iso, v.net.steiner etc.

The usage is similar – and enjoy the manual pages!

(Don't forget to send suggestions to the GRASS Development Team)