Using the SpatiaLite's Routing facility

Open Street Map support:

If you really are interested in Routing, you can't ignore the **Open Street Map** [*OSM*] project: http://www.openstreetmap.org/

OSM make freely available an impressive collection of world-wide road networks under the **Creative Commons SA** license. Data quality and accuracy may widely vary (depending on the country), but all this represent anyway an useful resource you cannot miss to explore.

You can dowload OSM data from two sites:

- http://downloads.cloudmade.com/
  - world-wide datasets
- http://download.geofabrik.de/osm/europe/
  - most interesting for European datasets.

OSM datasets are available for download under several formats and different flavors: anyway the main (and most general and comprehensive) format is identified by an **.osm** suffix, and actually is an XML derivative.

Loading an OSM dataset into some SpatiaLite DB:

In this tutorial we'll use the dataset available for download at:
http://download.geofabrik.de/osm/europe/luxembourg.osm.bz2

Please note: several OSM datasets are really huge (and thus, requiring lot of time to be parsed and loaded into the SpatiaLite DB.
So we'll use the Luxembourg dataset simply because it's one of the smallest available, but still representing a quite complex and realistic road network.

The file you've just downloaded is compressed using the **bizp2** algorithm: inflating the original **.osm** file isn't at all difficult, because lots of tools are available for both Windows and Linux.

Step #1:

You should create a new (empty) DB using the **spatialite-gui** tool: this preliminary step is required in order to properly initialize the DB, creating any metadata table and feeding the *spatial_ref_sys* table. Please, name this DB as: **luxembourg.sqlite**
Once you've created the DB you can quit **spatialite-gui**.

Step #2:

Now you should open a command shell, invoking the CLI tool **spatialite_osm**

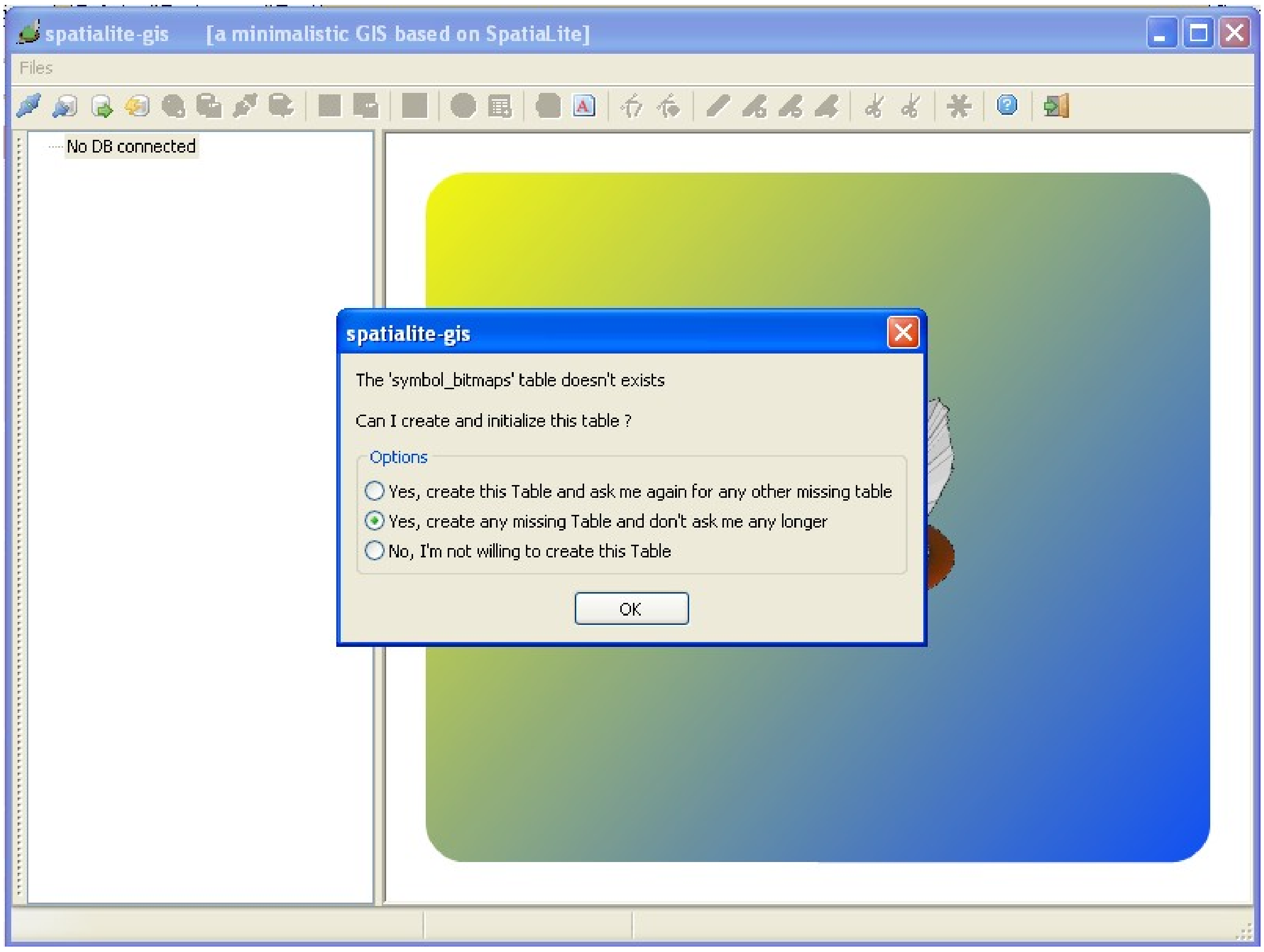
```
C:\>spatialite_osm -o luxembourg.osm -d luxembourg.sqlite -T roads -m
SQLite version: 3.6.20
SpatiaLite version: 2.4.0
using IN-MEMORY database
Loading OSM nodes ... wait please ...
        Loaded 327249 OSM nodes
Verifying OSM ways ... wait please ...
        Verified 12361 OSM ways
Disambiguating OSM nodes ... wait please ...
        Found 14 duplicate OSM nodes - fixed !!!
Loading network ARCs ... wait please ...
        Loaded 23869 network ARCs
Dropping temporary table 'osm_tmp_nodes' ... wait please ...
        Dropped table 'osm_tmp_nodes'
Dropping index 'from_to' ... wait please ...
        Dropped index 'from_to'
exporting IN_MEMORY database ... wait please ...
        IN_MEMORY database succesfully exported
VACUUMing the DB ... wait please ...
        All done: OSM graph was succesfully loaded
C:\>
```

All right, the OSM dataset **luxembourg.osm** has just been loaded into the table **roads** contained into the SpatiaLite DB named **luxembourg.sqlite**.

Please note: we've used an IN-MEMORY database [**-m**] in order to speed-up the loading process (that is quite complex and time consuming).

Using the interactive Routing facility:

Once we've loaded the OSM dataset into the *luxembourg.sqlite* DB we can launch the **spatialite-gis** tool, connecting this DB.

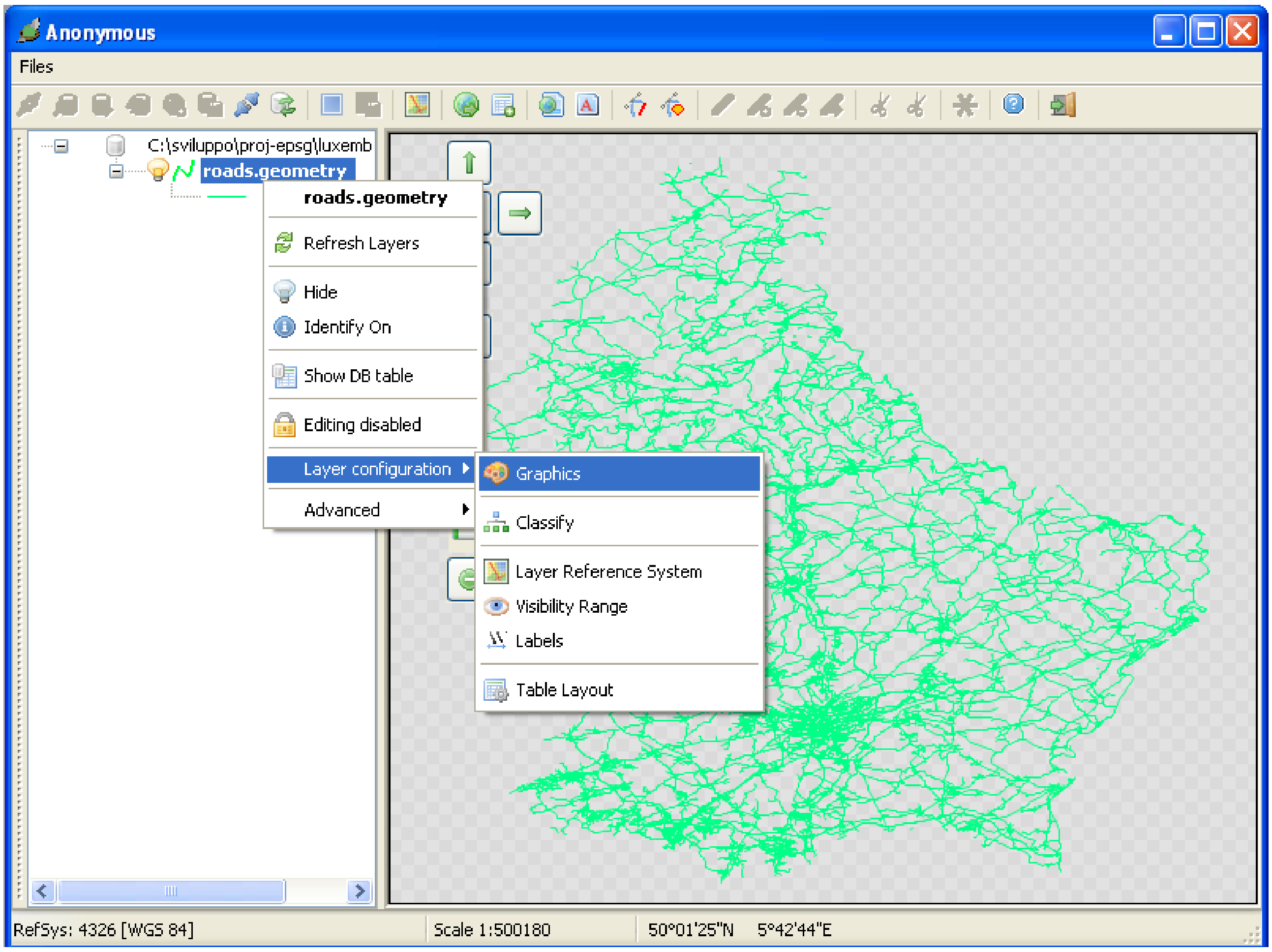


The 'symbol_bitmaps' table doesn't exists

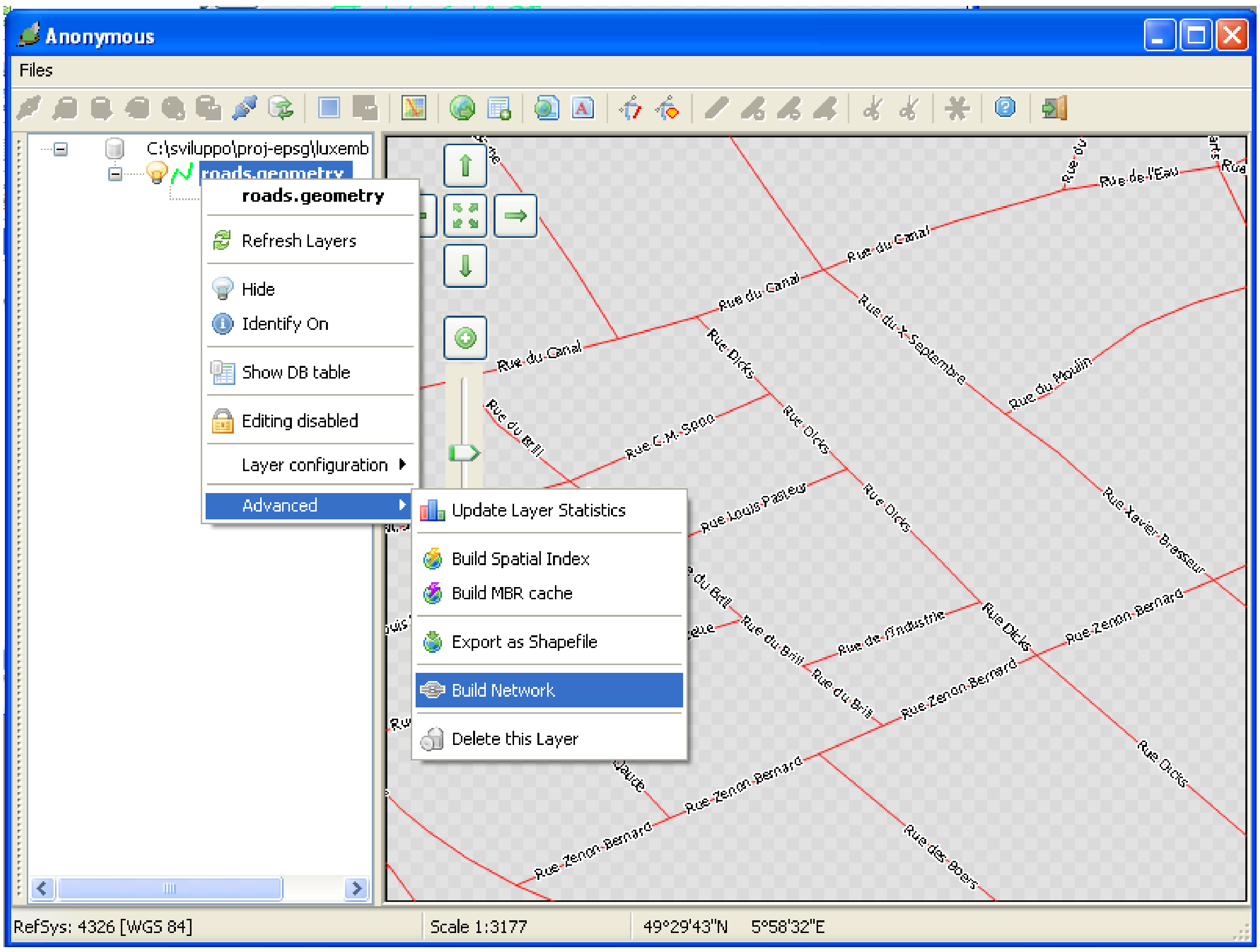
Can I create and initialize this table ?

Options
- Yes, create this Table and ask me again for any other missing table
- [✓ Yes, create any missing Table and don't ask me any longer]
- No, I'm not willing to create this Table

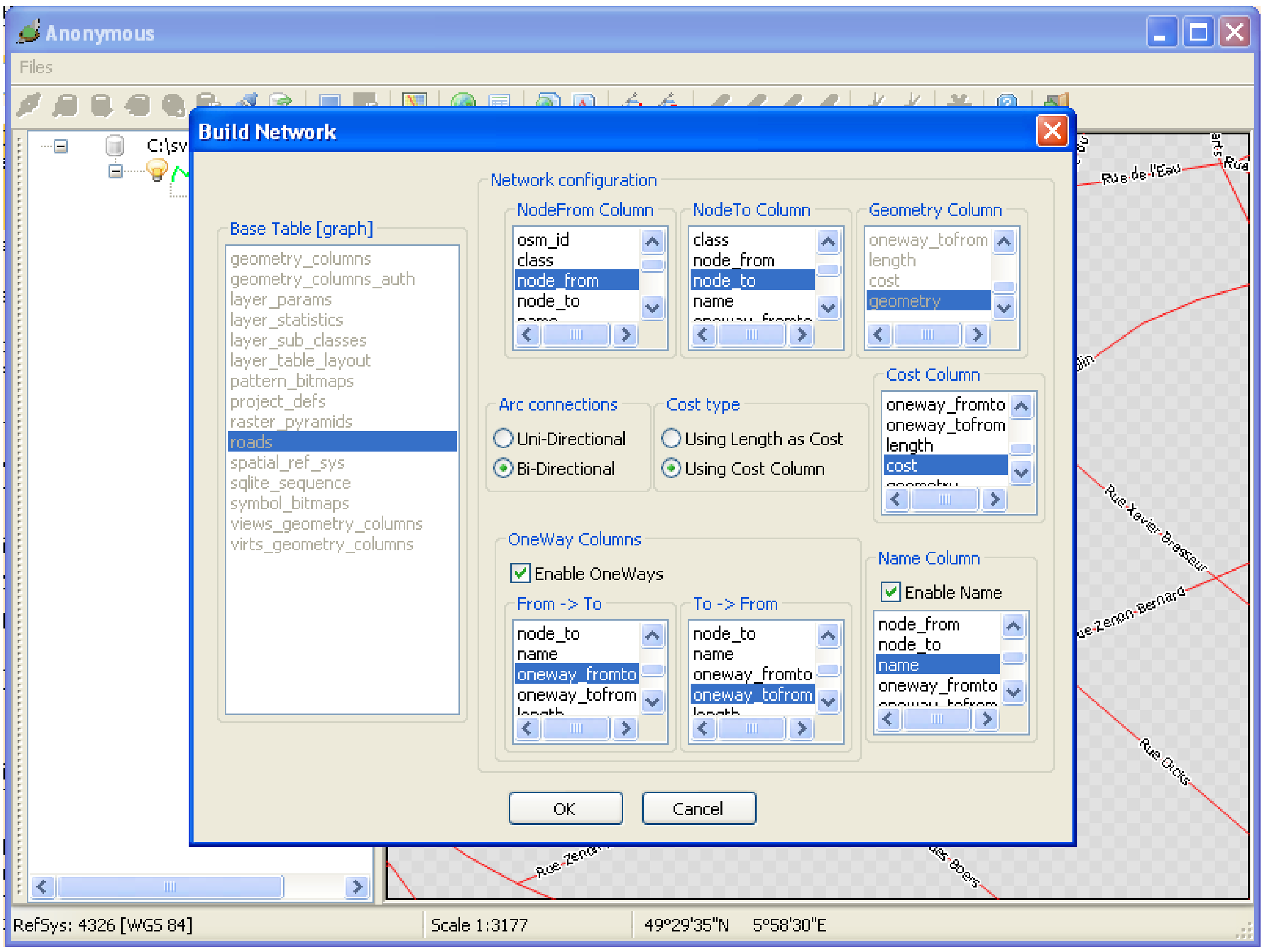
Step #1: create any missing GIS specific auxiliary table



Step #2: set up a more convenient graphic rendering and activate the Labeling feature.



Step #3: now we can build a full feathered Network. *Please note*: since now we simply have an ordinary Linestring layer, not yet a Network.



Build Network

Base Table [graph]
- geometry_columns
- geometry_columns_auth
- layer_params
- layer_statistics
- layer_sub_classes
- layer_table_layout
- pattern_bitmaps
- project_defs
- raster_pyramids
- [✓ roads]
- spatial_ref_sys
- sqlite_sequence
- symbol_bitmaps
- views_geometry_columns
- virts_geometry_columns

Network configuration

NodeFrom Column: osm_id, class, [✓ node_from], node_to, name

NodeTo Column: class, node_from, [✓ node_to], name, oneway_fromto

Geometry Column: oneway_tofrom, length, cost, [✓ geometry]

Arc connections: Uni-Directional [✓ Bi-Directional]

Cost type: Using Length as Cost [✓ Using Cost Column]

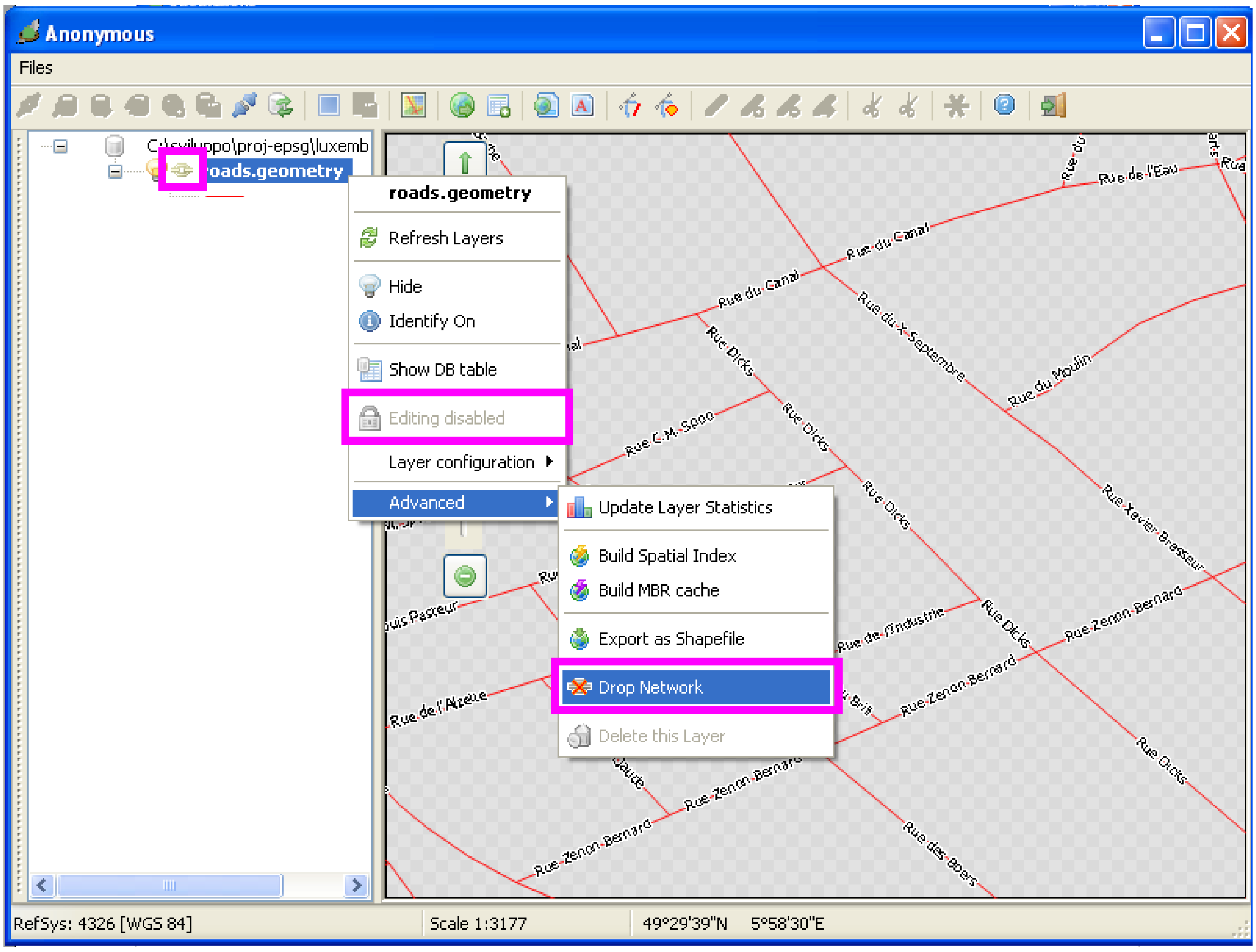
Cost Column: oneway_fromto, oneway_tofrom, length, [✓ cost], geometry

OneWay Columns
- ✓ Enable OneWays
- From -> To: node_to, name, [✓ oneway_fromto], oneway_tofrom, length
- To -> From: node_to, name, oneway_fromto, [✓ oneway_tofrom], length

Name Column
- ✓ Enable Name
- node_from, node_to, [✓ name], oneway_fromto, oneway_tofrom

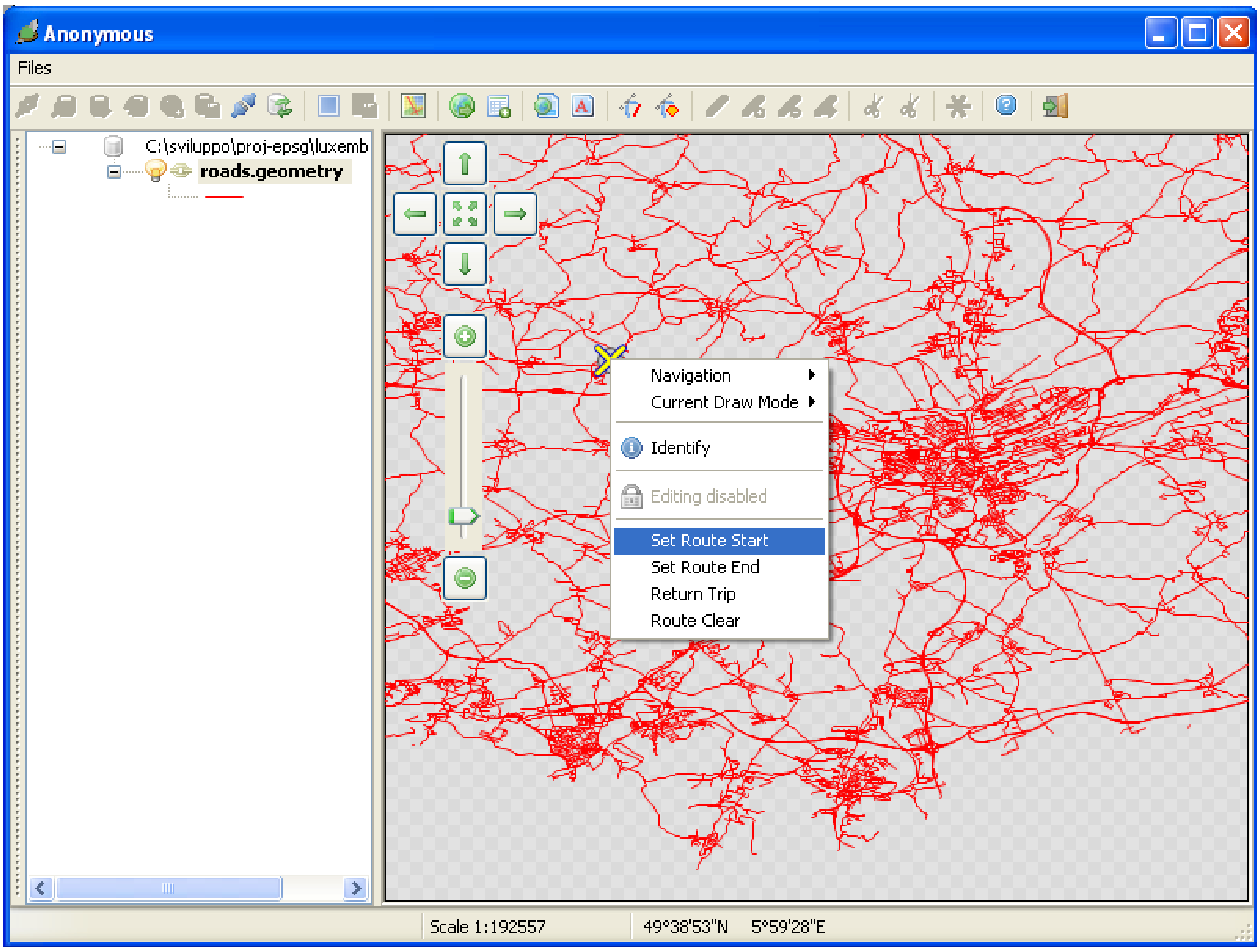
OK Cancel

<u>Step #4:</u> apply the above settings and then confirm.

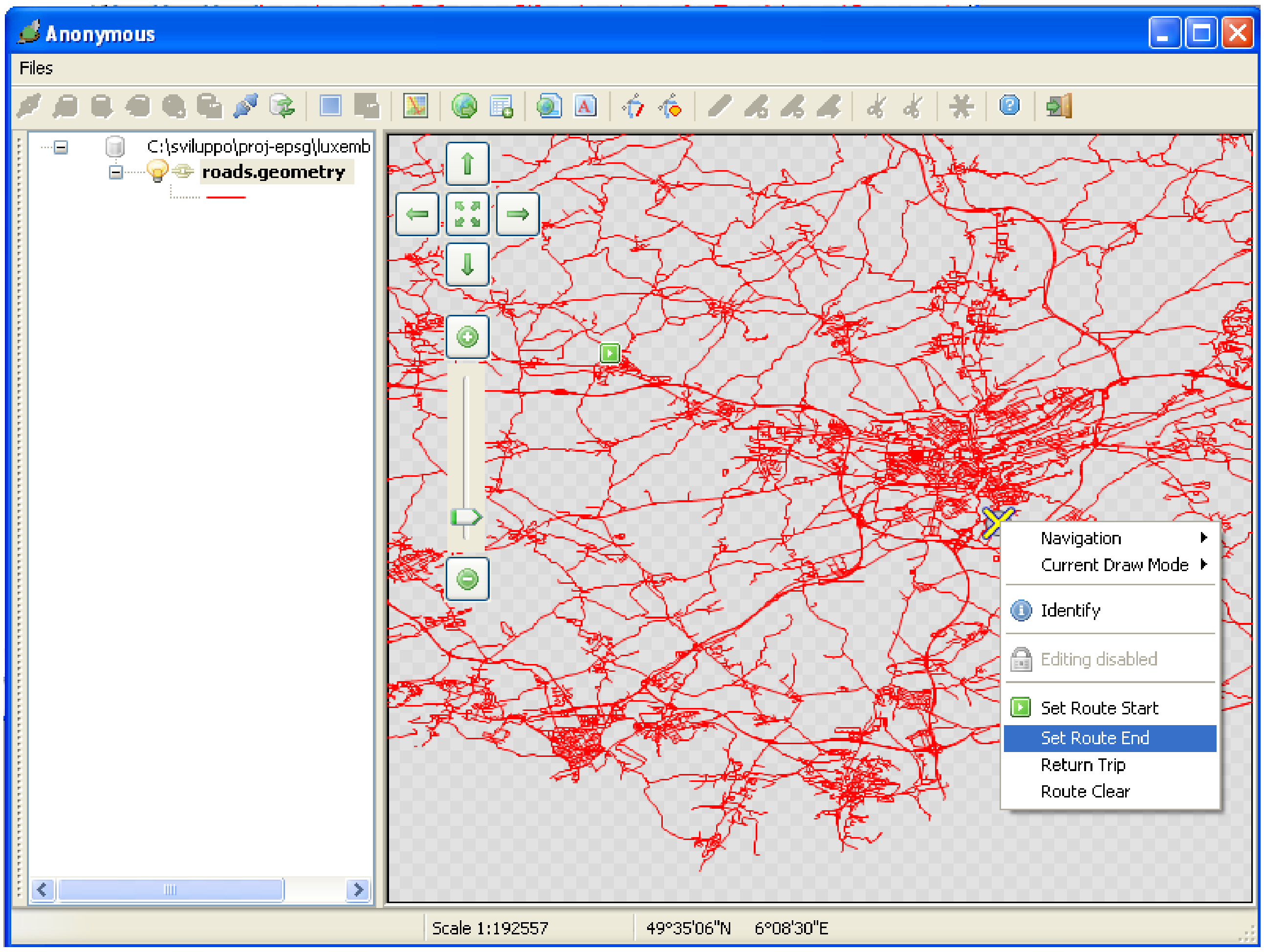


Please note:

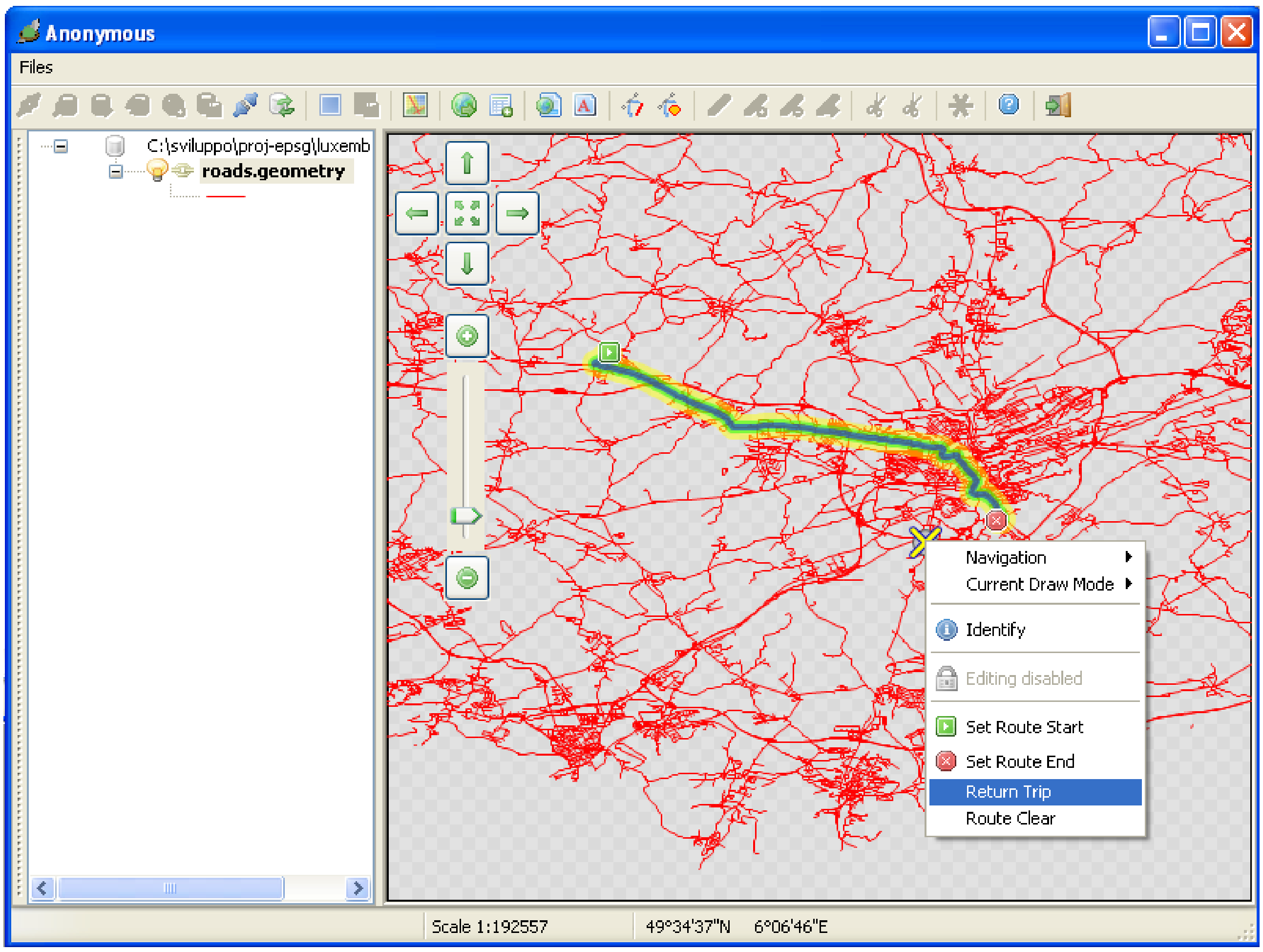
- a Network is identified as a special Layer, assuming the *link* symbol
- a Network always is **read-only**: consequently no Edit operation is allowed
- you can revert back to the original plain Linestring Layer simply dropping an existing Network



Step #5: when a Network Layer is activate, moving the mouse over the map will show a cross marker each time a Network Node is intercepted.
You can then use the context menu (by pressing a *right-click*) and fix the Start or End nodes for the Route you wish to set.

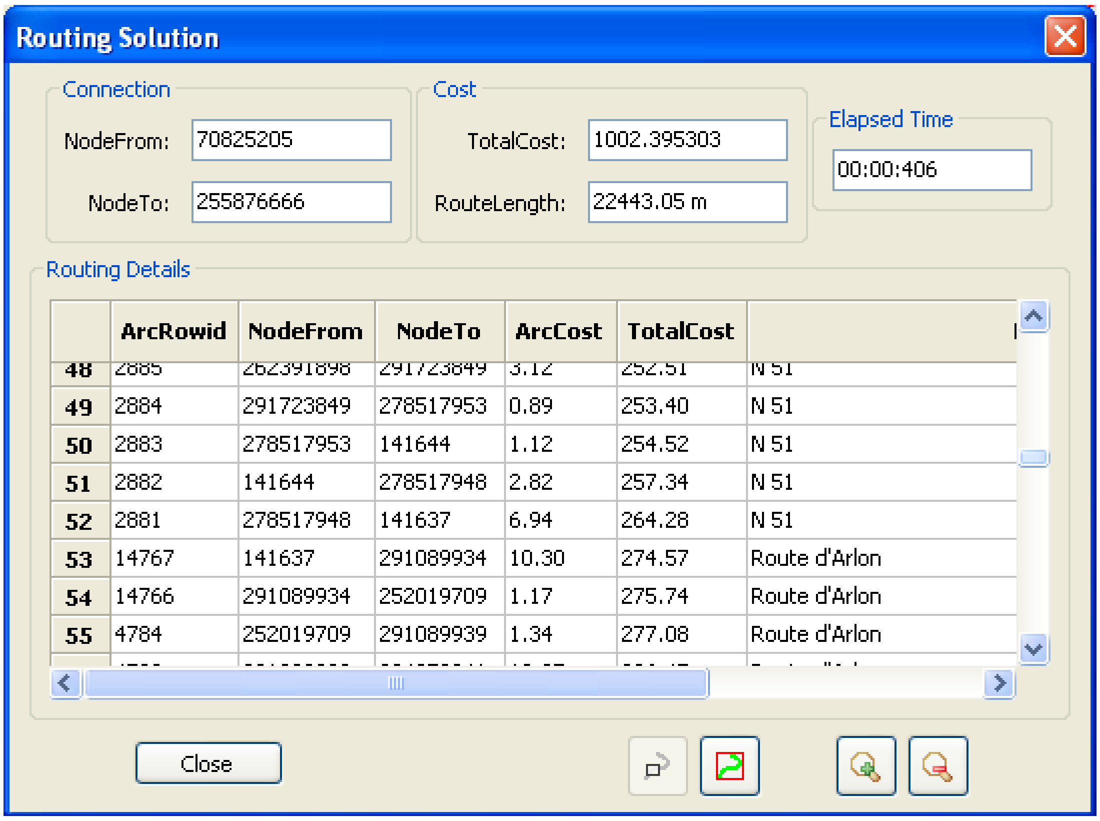


Step #6: then you have to repeat the above step in order to set the Route destination node.



Step #7: as soon as you identify both Route extreme Nodes, a Routing solution will be computed, and the resulting path will be shown over the map. Please note that:

- the error message **Unreachable destination** means that your Network is someway broken (missing nodes, wrong one-way settings and so on)
- you are allowed to set a different Start or End, still keeping untouched the opposite extremity of the Route
- you can ask to compute the return trip for the same extremities [yes, this is really useful to check one-ways]
- and you can clear at all any currently set Route.



Routing Solution

Connection

NodeFrom: 70825205

NodeTo: 255876666

Cost

TotalCost: 1002.395303

RouteLength: 22443.05 m

Elapsed Time

00:00:406

Routing Details

	ArcRowid	NodeFrom	NodeTo	ArcCost	TotalCost	
48	2885	262391898	291723849	3.12	252.51	N 51
49	2884	291723849	278517953	0.89	253.40	N 51
50	2883	278517953	141644	1.12	254.52	N 51
51	2882	141644	278517948	2.82	257.34	N 51
52	2881	278517948	141637	6.94	264.28	N 51
53	14767	141637	291089934	10.30	274.57	Route d'Arlon
54	14766	291089934	252019709	1.17	275.74	Route d'Arlon
55	4784	252019709	291089939	1.34	277.08	Route d'Arlon

Each time a Routing solution is computed, a panel will appear too, showing the complete Network Arc list required to get the solution.
This is really useful in order to check each single step in a very analytical way.
And you are allowed to perform zoom-in and zoom-out, to center map in order to show the complete route or a single selected arc.