**FiberNet Documentation V1.0**

**Sean Freeman, Jagadeesh Potluri**

**Repository Folder Structure:**

3\_clustering: Output of our 3 clustering, including the Steiner points of the Steiner tree.

3\_clustering-nosteiner: Output of our 3 clustering, excluding the Steiner points

Fiber Layouts: All of our fiber layouts, including our final layout for the entire neighborhood, fiber\_layout-v5-steiner.pdf.

Fibernet Cost: Our excel cost sheet, used to calculate our final costs to deploy the network.

Python-Code: All python code used in the project, described below.

QGIS: Our QGIS project file for the shapefiles with the map, with how we got to that point described below.

Steiner\_Tree: Our work before and after utilizing GeoSteiner to calculate our Steiner Tree, described below.

Concorde-work: Our work using Concorde TSP Solver, including input and output to Concorde.

Htmlcov: Our “coverage” HTML output.

Shapefiles: The shapefiles utilized in this project

Sqlite\_osm\_data: The Sqlite Open Street Maps data, described below.

**The below steps need to be followed in creating shape files.**

1. Install QGIS software.
2. In QGIS, go to Vector->OpenStreetMap->Download data. OSM file will be created.
3. Next, go to Vector->OpenStreetMap->Import topology from XML, give osm file as input.

**Optional Step**: Incase if you need to change DB name, you can modify in “Connection name” field.

Click ok.

Now OSM DB will be created.

1. Next, go to Vector->OpenStreetMap->Export topology to SpatiaLite.

Give DB file created as input

Export Type:

Points

Polylines (open ways) -> for Roads

Polygons (Closed ways) -> for houses.

Depending on your need choose the appropriate one.

Click on “Load from DB”. And check the fields which ever needs to be present in shape files DB.

Click “Ok”.

1. Now Shapefile is created in your directory which contains filenames as “.dbf”, “.prj”,”.qpj”,”.shp”,”.shx” files.

**The below steps need to be followed to view Shapefile and DB data**

1. Double click on “.shp” file to open Shapefile.
2. To check database for corresponding Shapefile. On the left side, In the “Layers” section find out the Shapefile name.

Right click on it-> Open Attribute table. Now DB will be opened with a list of fields and its values.

**The below steps need to be followed to filter DB data for houses/roads and to create shape files for Roads and Houses**.

1. On the left side, In the “Layers” section find out the Shapefile name.
2. Right click on it-> Properties. Click on Query Builder.
3. Construct the query as you need.
4. Once query is noted. Click ok.
5. Based on the query Shapefile will be modified.

Filter Condition applied:

1. For houses filter based on building attribute which contains only “houses”.
2. For roads filter based on highway attribute which contains only cycleway, path, primary, primary\_link, residential, secondary, secondary\_link, territory, territory\_link and unclassified.

Now save your file. Shapefile will be saved which contains filenames as “.dbf”, “.prj”,”.qpj”,”.shp”,”.shx” files.

Repeat these steps for Roads as well as houses. At the end of these steps you will have 2 set of Shapefiles. One Shapefile is for Roads and another is for Roads.

**The below steps need to be followed to add new layer to existing layer.**

1. Goto Layer->Add Vector Layer-> Select Dataset(.shp file) that needs to be added.

Click open.

New layer will be added to the existing layer.

To save the layers. Click save and give a filename. It will be stored as “.qgs” file

**Python file descriptions:**

Following packages needs to download. Place all these in your working directory

1. Shapefile library - http://pyshp.googlecode.com/svn/trunk/shapefile.py
2. Matplotlib-1.4.0 - http://matplotlib.org/downloads.html
3. SciPy - http://www.scipy.org/install.html

bindhousetoint.py (uses Shapefile Library): Used to bind houses to intersection points. What this means is that given a list of intersections, say where a driveway intersects a road, returns the corner point of the house that is closest to this intersection point. The house data is from a shapefile.   
Call by: python3.4 bindhousetoint.py <intersection point list> <house shapefiles>

clusterpoints.py (uses Matplotlib): Used to cluster houses using K-Means clustering, using Lloyd’s Method. Given a list of house points, it will put them into the specified number of clusters and output the points to a different file for each cluster. Finally, it will plot the clusters.  
Call by: python3.4 clusterpoints.py <inputfile> <number of clusters>

convertfromgeo.py (uses Shapefile Library): Used to convert GeoSteiner output (which returns a list of x,y coordinates) into a latitude, longitude list.   
Call by: python3.4 convertfromgeo.py <geosteiner output> <output file>

converttoconcorde.py (uses shapefile library): Used to convert lat, lon list into both geosteiner input and Concorde TSP input.   
Call by: python3.4 converttoconcorde.py <lat lon points> <output file>

get\_average\_points.py (uses Shapefile library): Used to get center points of houses from a shapefile (gathered using the above description of QGIS).   
Call by python3.4 get\_average\_points.py <shapefile> <output file>

plot\_clustering.py (uses shapefile library and Matplotlib): Used to plot a clustering on a map of the roads and houses.   
Call by: python3.4 plot\_clustering.py <road shapefile directory> <house shapefile directory> <clustering directory (including trailing /)> <output file>

plot\_mst.py (uses shapefile library and Matplotlib): Used to calculate and plot a minimum spanning tree of the given points.   
Call by: python3.4 plot\_mst.py <road shapefile directory> <house shapefile directory> <Latitude, longitude points file> <output file>

shapefile.py: The Shapefile Library, described above.

unittests.py: The unit tests for these files, used by nose.

**Our steps for calculating the Steiner Tree and plotting it:**

* + - 1. Take the shapefile given of the houses and use get\_average\_points.py to find the terminals of the houses.
      2. Next, Use clusterpoints.py to cluster the set of latitudes and longitudes given by the above into a set of points.
      3. Next, use converttoconcorde.py to convert the output latitude, longitude list into a format that GeoSteiner can use for each cluster.
      4. Use GeoSteiner (<http://www.diku.dk/~martinz/geosteiner/>) to calculate the Steiner tree of the given points for each cluster.
      5. Use convertfromgeo.py to convert the GeoSteiner output into a working latitude, longitude list of points for each cluster.
      6. Use plot\_mst.py to calculate and plot the MST for each clustering (since the steiner points are already included, the MST is the steiner tree).