Gis2Graph algorithm

1 Description of the Gis2Graph algorithm

The proposed algorithm has the main purpose of **representing a city (or an area of a city) as a graph**, in particular by representing the road network of an area of the Italian territory in relation to some points of interest which could be buildings, bus stops, stations or any other set of points within that area.

The first phase of the algorithm concerns the collection of the geographical information about the city; in particular streets, crossroads and points of interest. The second concerns the transformation of the collected data into nodes and edges of a graph.

In this way, **a connection between points of interest is obtained based on the roads to be traveled to reach one point from another**.

Il grafo, oltre a rappresentare la rete stradale, si occupa di associare ad ogni punto di interesse la strada che ne permette l'accesso e di ottenere, quindi, una connessione tra i punti di interesse basata sulle strade da percorrere per raggiungere un punto da un altro.

1.1 Data Collection

As said, to create a graph of the area of a city it is necessary to get the geometrical information about the three most important components of a city: the **streets**, the **crossroads** and the **points of interest (POIs)**. The streets must be represented by linestrings, the crossroads by points, and the points of interest could be either represented by polygons or points.

There are three different ways to acquire this spatial data: by using the OpenGeoData Abruzzo database, by using the OpenStreetMap database and by importing personal POI shapefiles. The main method is by letting the algorithm exploit the overpass API to obtain the geographical elements made available by OpenStreetMap. Section 2 explains in detail how to choose input data via a configuration file.

Each point of interest is associated with the stretch of road closest to it, that is, the last one you would cross to reach it or the first one to start from it. In particular, the most plausible route is identified among the others available and, if not directly connected to the point of interest, **the remaining route is estimated**. For example, it may happen that for a building there is a small gravel road that leads from the main road to the entrance but which has not been mapped: in this case a line is estimated and generated which will represent the gravel road leading to the entrance.

When data is being collected, it must be taken into account that there are information that can be added as they are used in the particular case study. They could be also specified in the configuration file.

The oriented object implementation involves the creation of the City class which uses the Building, Crossroad, Waiting Area, Census Area and Street

classes which manage the loading of information from spatial data. The city can be described as a python class as shown in the figure X.

1.2 Transformation into a graph

The second phase of the algorithm is the transformation of the collected data into a graph. In particular, A CityToGraph class it's used to transform the city into a graph. For the construction of the edges it is useful to define the concept of *point of interest* as any possible starting or ending point in a route. These points are all those nodes of the graph that do not represent crossroads.

1.2.1 Description of the graph

The generated graph $G = (V, E)$ is a graph with a set of nodes $V = V_P \cup V_C$ where V_P is the set of the nodes representing the POIs and V_C is the set of nodes representing the crossroads and a set of edges $E = E_C \cup E_P$ where E_C is the set of the streets that connect a crossroad to another and E_P is the set of the *street-segment* that connect a POI to another or a POI to a crossroad.

Every node $v \in V$ has the attributes:

- **nodeType**: it can assume the values *"crossroad"* or *"point of interest"*;
- **geom_id**: represented as a string it represents the original id of the geometry, it is given from OSM or by the shapefiles;
- **pos**: expressed by lat/lon coordinates it represents the entrance to access to the POI; it is equal to the geometry if the POI is represented by a point, but if the POI is represent by a polygon it is a point that leans on its border;
- **id**: expressed by lat/lon coordinates it is equal to the **pos** attribute;
- **other_fields**: expressed by a dictionary, it is used to save all the other information taken from the shapefiles or OSM;

If v is a Points of Interest (i.e $v \in V_P$) it has also the attribute:

- **listOfTypes**: expressed by an array of strings, it indicates the types of POI, for example *"building"*, *"bus station"*, *"waiting areas"*, and so on; it is an array because for example a building could be also a school, they are chosen by the user in the configuration file as shown in 2, in particular a POI can have more than one type if other shapefiles which intersects with the geometries representing original POIs are imported, for example schools need to have geometries which intersect with the geometries of the buildings;

Every edge $e \in E$ has attributes:

- **edgeType**: it can assume the values *"street"* or *"street_segment"*

- **width**: expressed by floats, it indicates the width of the street in meters, if the shapefiles of the streets are taken from the *opendata.regione.abruzzo.it* they can assume the values 3, 5 or 7 which respectively are an estimation of the three available informations "Width less than 3.5 meters", "Width between 3.5 meters and 7.0 meters" adn "Width greater than 7.0 meters".
- **length**: expressed by floats, it indicates the length of the street in meters, in case $e \in E_P$, the length is an estimate;
- **other_fields**: expressed by a dictionary, it is used to save all the other information taken from the shapefiles or OSM, in case $e \in E_P$ this value is the same of the street associated with it.
- **id_1** and **id_2**: expressed by lat/lon coordinates, they represents the identifier of the two nodes of the edge, in particular they need to represent the coordinates of a crossroad or of an entrance to access a POI;
- **start_point** and **end_point**: expressed by lat/lon coordinates, they are the same of the ids but they are useful to identify the direction of a street in case in **other_fields** there are some attributes that depends on it, for example the slope.

1.2.2 Detailed and abstract graphs

When generating the graph a more generic one is generated, too. In particular, we defined as **detailed graph** the one containing all the information just described and **abstract graph** the one that contains only the connection between crossroads, it is given by the graph induced by the nodes V_C. We recommend using this graph as it is more efficient for running the optimal path search algorithms. That is because the detailed graph contains many more nodes than the abstract graph: computing a path from a point of interest to another in G_D can however be fastened by exploiting G_A. To find a route between two points of interest we can distinguish three steps: departure, search and arrival. In the departure step, the detailed graph is explored in order to move from the starting point of interest (the origin) to the nearest crossroad that brings closer to the destination. In the search step, the algorithm uses G_A to find a route from the crossroad found in the departure step to a crossroad that is as close as possible to the destination (i.e., one of the endpoints of the street that contains it). Finally, in the arrival step one moves from a crossroads to the destination, so it is again necessary to use G_D. Using this method, the CPU time of the routines used in the elaboration of the evacuation plan can be substantially reduced.

2 How to use configuration file

To run the algorithm you have to compile the configuration file in the listing 1 according to your preferences.

The configuration file is a JSON where you can choose the parameters and the settings that you prefer.

At first, you can choose between two ways to collect streets and crossroads:

- If the area of interest is in Abruzzo, it is possible to use the shapefiles provided by **OpenGeoData Abruzzo** [1]. The shapefiles of OpenGeoData Abruzzo are divided in sections representing an area in Abruzzo of approximately 9.5 square kilometers (see the *"opengeodata_sections"* attribute of the configuration file).
- Otherwise, for any city or area in Italy it is possible to use the OSM elements by indicating the name of the city or the polygons representing the area of interest (see the *"city_name"* and *"wkt_area"* attributes).

In both cases the algorithm will automatically extract the necessary information from the chosen database.

For the points of interest there are more alternatives that can be used together:

- If you have used the OpenGeoData Abruzzo, you can also add the points of interest provided from their database. In particular you can choose between the following OpenGeoData names:
  - CR01G_EDIFICI: buidings;
  - CR359G_UN_VOL_SUP: same buildings but differentiated by area of sediment;
  - CR207G_EDIFICI_MINORI: minor buildings (as huts or shacks);
  - CR351G_SP_ACQ: lakes;
  - CR25G_VERDE: gardens;
  - CR27G_ALBERO: trees;
  - CR227G_ND_AAC: springs and fountains;
  - CR366G_MN_ARR_POS: church crosses;
  - CR236G_ATTR_SP: sport facilities;
  - CR09G_GALLERIA: tunnels;
  - CR311G_PONTE_SEDE: bridges
  - CR03G_TRALICCI: pylons;
  - CR200G_CAVA: quarries;

  You can indicate some of them in the *points_of_interest_from_open_data* attribute;
- You can use OpenStreetMap database: for now it is possible only to extract buildings or house numbers but to ensure that the algorithm can be applied in different usage scenarios, it is important an integration of other possible POIs available on Openstreetmap such as bus stops and other interesting elements available on OSM. You can indicate *"building"* or *"house numbers"* in the *points_of_interest_from_OSM* attribute.

- You can also import your personal POI shapefiles if they are polygons or points by indicate them in the *imported_points_of_interest* attribute.
- You can use your personal POI shapefiles to add information to already existing POIs: for example you may add information of a shapefiles for "schools" to the already imported shapefiles of "buildings". You can indicate this in the *imported_points_of_interest* attribute.

2.1 Description of each attribute

Going into the details, the configuration file has the following attributes to indicate preferences.

- **city_name**: it is the name of the city you want to map, it can assume any value you prefer if you are using the attributes "wkt_area" and "opengeodata_sections" but if they are empty it will be used to extract the information from OSM so, in that case, it is important to write correctly the name of the city;
- **wkt_area**: it is used when you want to extract the OSM elements of a particular area, it is the geometry representing the area you want to map, it must be a polygon and written in Well-known text (WKT), it is used to extract the information from OSM;
- **working_path**: it is the path where are saved the shapefile you want to use (? - mi sa lo possiamo toglie);
- **opengeodata_sections**: it is used if you want to extract streets and crossroads from OpenGeoData Abruzzo db, it is an array with the ids of the sections you want to map and to find the ids of the area in which you are interested you can use the ArcGIS map that you find the bibliography [2];
- **points_of_interest_from_open_data**: it is used if you want to extract points of interest from OpenGeoData Abruzzo db, it is a key-value pair where the key represents the name you want to give at that point of interest and the value is the name given by OpenGeoData (for example *"Building": "CR01G_EDIFICI"*);
- **points_of_interest_from_OSM**: it is used if you want to exctract points of interest from OSM db, it is a key-value pair where the key represents the name you want to give at that POI and the value can be "building" or "house number" (for example *"Address": "house number"*);
- **imported_points_of_interest**: it is used to import personal POI shapefiles, it is a key-value pair where the key is the name you want to give to that POI and the value has three other key-value pairs where the keys are:

  - "path": that want has value the path of the shapefiles;
  - "add_to_feature": the name of the already existing POI if you want to add information to it (otherwise it is an empty string);
  - "suffix": the suffix you want to give to the new attributes if you are adding the information (otherwise it is an empty string);

- **saving_shapefiles_path**: it is the path where you want to save the shapefiles of the exctracted data (streets, crossroads and POI from OpenGeoData and OSM);
- **graph_name**: it is the name you want to give to graph;
- **saving_graph_path**: it is the path where you want to save the graph;
- **saving_graph_formats**: it is an array with all the formats in which you want to save the graph, you can choose between "gpickle", "adjacency list" and "json".

In listing 1 it is shown the template of the configuration file.

2.2 Some examples

2.2.1 OpenGeoData Abruzzo: buildings in Capitignano

Suppose we are interested in the buildings of Capitignano and we want to use the OpenGeoData Abruzzo database. The first step is to find the ids of the sections containing the interested area as shown in figure 1 and we suppose we are interested in section 348081 and 348082. In this case the configuration file will be as shown in listing 2

2.2.2 OSM: house numbers in L'Aquila

Suppose we are interested in the house numbers of a particular area in L'Aquila and we want to use OSM database. In this case we need to know the geometry of the area and indicate it in well-known text in the *"wkt_area"* attribute of the configuration file. A simple way to generate the right wkt of a geometry, is to use online sites that allows you to draw a geometry and then to extract it in different formats including the well-know text. **??**

References

[1] D.B.T.R. Regione Abruzzo from OpenGeoData, `http://opendata.regione.abruzzo.it/content/dbtr-regione-abruzzo-scala-15000-edizione-2007-formato-shp`.

[2] , ArcGIS map with OpenGeoData sections, `https://www.arcgis.com/home/webmap/viewer.html?webmap=25d04135ab6d49d4a757cf135c7f48cf`.

```
{
    "city_name": "My city",
    "wkt_area": "POLYGON((42.350615 13.399994,
        42.35334 13.393940, 42.355443 13.396391,
        42.35495 13.401535, 42.354072 13.401517))",
    "working_path": "My/path",
    "opengeodata_sections" : ["000000","000001"],
    "points_of_interest_from_open_data":
            {"POI_Type": "OpenGeoData_NAME"},
    "points_of_interest_from_OSM" :
            {"POI_Type": "buildings"},
    "imported_points_of_interest" :
            { "School":
                  {
                    "path": "path/to/shapefile.shp",
                    "suffix": "string",
                    "add_to_feature": "POI_Type"
                  }, },
    "saving_shapefiles_path": "My/path",
    "graph_name": "MyCity_network_buildings",
    "saving_graph_path": "My/path",
    "saving_graph_formats":
            ["gpickle", "adjacency list", "json"]
}
```

Listing 1: Configuration file template

[3] , Geojson.io to find the wkt area, `https://geojson.io/`. https://geojson.io/

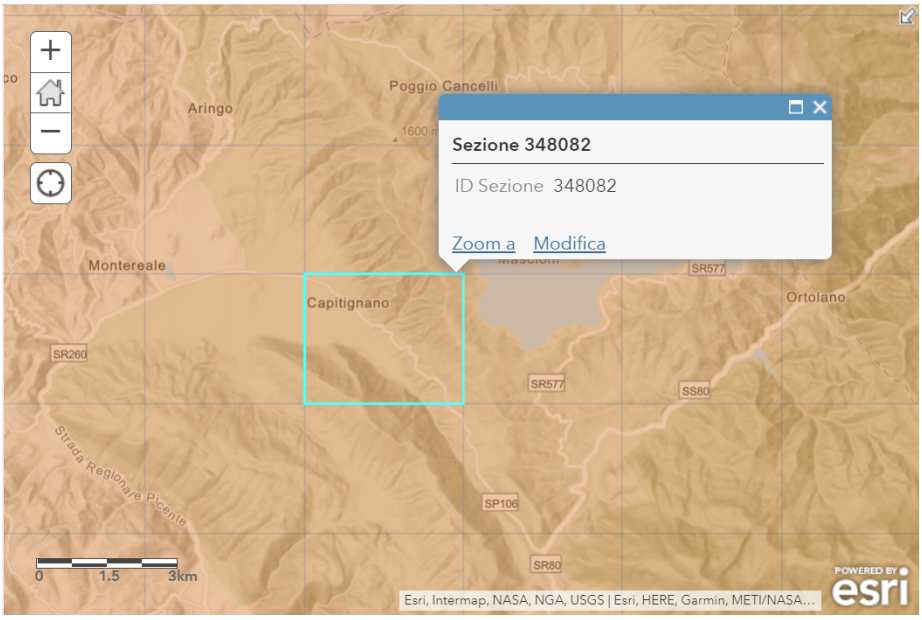

Figure 1: Finding on ArcGIS the ids of the interested sections of OpenGeoData

```
{
  "city_name": "Capitignano",
  "wkt_area": "",
  "working_path":
        "data/examples/OGD_Capitignano_buildings",
  "opengeodata_sections": ["348081", "348082"],
  "points_of_interest_from_open_data":
        {"Edificio": "CR01G_EDIFICI"},
  "points_of_interest_from_OSM" : {},
  "imported_points_of_interest" : {},
  "saving_shapefiles_path":
        "data/examples/OGD_Capitignano_buildings/shapefiles",
  "saving_graph_path":
        "data/examples/OGD_Capitignano_buildings/graphs",
  "graph_name": "OGD_Cap_graph",
  "saving_graph_formats":
        ["gpickle", "adjacency list", "json"]
}
```

Listing 2: Configuration file for Capitignano buildings with OpenGeoData