Graph extension for OpenJUMP

v 0.1 (2010-04-27) by Michaël Michaud

1 General concepts

Graph Extension is a set of plugins bundled as a jar file to be placed in OpenJUMP plugin directory (default plugin directory is lib/ext).

Graph extension has two dependencies :

- **JGraphT** is a graph library implementing many graph algorithms, a few of which are used by OpenJUMP graph extension
- **jump-jgrapht** includes glue-code between JGraphT and OpenJUMP feature model. jump-jgrapht has been bundled in a separate jar, so that other extensions can easily use other JGraphT algorithms.

1.1 Jump-jgrapht glue-code

The way jump-jgrapht build a graph from a feature collection is very simple : it uses linear features as edges and LineString end-points as nodes. It test connectivity using equals() method between first and last coordinate of each LineString. That means that

1. graph construction only makes sense for linear networks
2. *feature collection is supposed to be correctly noded*. If two features intersect at a point strictly interior to one of the feature (rather than at their extremity), the graph will not connect these features together.

TODO : Currently, jump-jrapht includes code to create *undirected* graphs from a feature collection. There is no code to create directed graphs yet. It would be interesting to add this feature, but there is no unique way to modelize a directed graph in a GIS (one can use natural orientation of LineStrings, single-attribute-based orientation, orientation defined by several attributes...).

1.2 Graph extension plugins and common options

Every plugin from graph extension is installed in the Plugins menu of OpenJUMP, in a submenu called... Graph.

Most options have a tooltip giving more details about option signification.

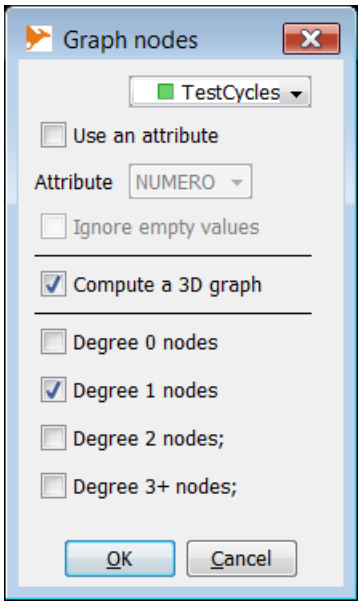
Some options are common to several plugins :

- 3D graph : if selected, the graph will be computed in 3D. This means that two lines which are connected in the plan but have end-points at different altitudes will not be connected in the graph.
- Use attribute : if selected, the plugin will allow to select an attribute, and will compute a different graph for each set of features having the same attribute value.
- ignore empty : this option is linked to the previous one. If ignore

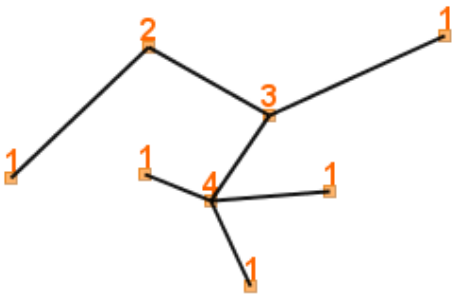
empty is selected, features with null value and features with empty string will be excluded from the graph. This feature can be used, for example, to build one graph by river and to exclude unnamed rivers.

2 Graph nodes PlugIn

This PlugIn computes all the nodes of a graph. Options will allow to output :



- degree 0 nodes : isolated nodes. It makes no sense for a pure linear network. It could represent puntal features isolated from the network in a mixed geometry layer, but it is not implemented yet.
- degree 1 nodes : pendant nodes, or end-nodes. Nodes connected to one edge only. Very useful to find interruptions in a network.
- degree 2 nodes : nodes linking two linear features exactly.
- degree 3+ nodes : nodes where 3 edges or more connect together. They represent intersections or forks in a network. It is possible to know the exact degree of each node in the output node layer (see here after).



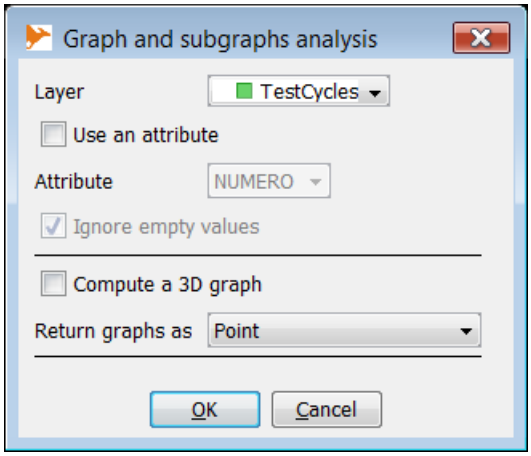
Common options are also available to compute a 3D graph rather than a 2D one, or to compute a distinct graph for each distinct value of a user-defined attribute.

The result of this PlugIn is a unique layer named after the original layer name followed by "-nodes", and with one attribute containing the degree of each node.

Tip : you can easily create separate layers for each degree using Edit > Extract > Create separate layers for each attribute value.

3 Connected components PlugIn

This plugin analyzes the connectivity of a graph. If the graph is not fully connected, several connected subgraphs are computed.



The plugin outputs two layers. One describing original graph(s) characteristics and one for the connected components. Attributes for each layer are :

Original graph(s)

TestCycles-graphs (1 Feature)

...	FID	Connected Subgraphs	Features	Pendant Vertices	Length
	79	15	63	30	708.8372873659017

- Connected subgraphs : number of connected subgraphs in the graph
- Features : total number of features in the graph
- Pendant nodes : total number of pendant nodes (or end-nodes)
- Length : total length of the graph

Connected subgraph(s)

TestCycles-subgraphs (15 Features)

...	FID	Connected Subgraph	Features	Pendant Vertices	Length
	64	1/15	2	1	40.70948508132716
	65	2/15	4	2	50.52770330511273

- Connected subgraph : number of the connected subgraph as 2/3 for the second (unordered) connected subgraph of a graph
- Features : number of features in the connected subgraph
- Pendant nodes : number of pendant nodes (or end-nodes)
- Length : total length of the connected subgraph

Geometries representing graphs in those two layers maybe one of the following :

- Point : the whole graph or subgraph is represented as a single point (getInteriorPoint JTS function)
- MultiLineString : the whole graph or subgraph is represented as a MultiLineString which is a combination of all the linestrings of the graph (or subgraph)

- Simplified MultiLineString : the graph (or subgraph) is represented as a MultiLineString where each original LineString is replaced by a simplified LineString composed of two points : initial coordinate and end coordinate.

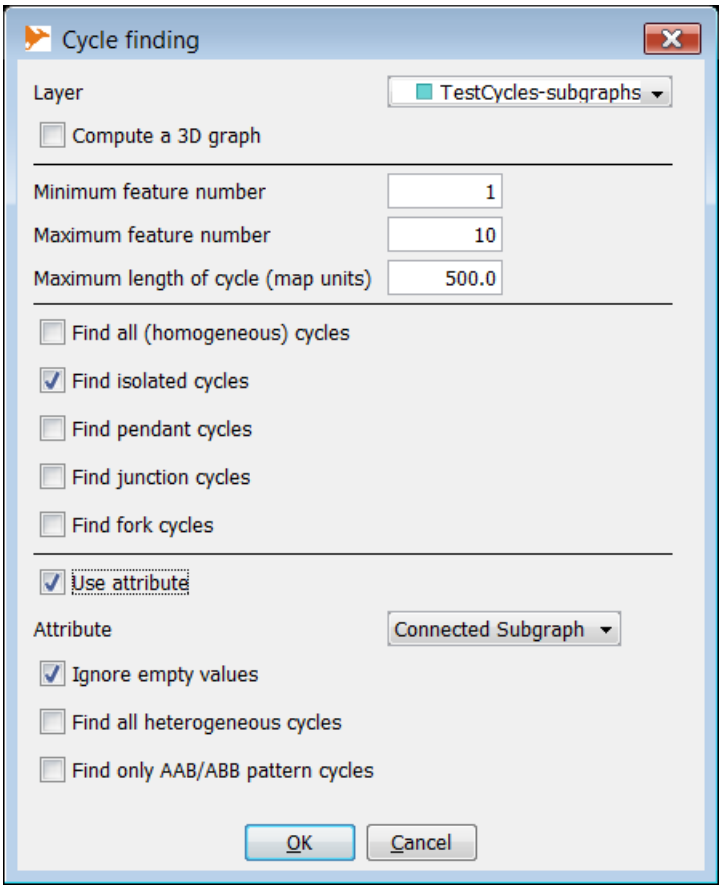
4 Cycle detection PlugIn

Cycle detection finds simple cycles in a layer representing a network. It can be used, for example, to find traffic-circle in a road network, anastomosis in a river network...

Number of cycles in a network can be very high, even if the plugin only returns only simple cycles.

You can limit the number of returned cycles using the following parameters :

- minimum number of features : minimum number of features composing the cycle. You may want, for example, select every cycle but the one made of a single feature (LinearRing).
- maximum number of features : maximum number of features composing the graph.
- maximum length : maximum length of the cycle. Used to remove large cycles. The unit is the map unit (beware maps in decimal degrees)



Other options make it possible to select cycles depending on their "degree" :

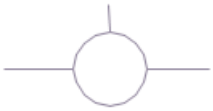
- isolated cycles (degree 0) : there is no edge entering or leaving the cycle. If an attribute is used (see below), this option means that the cycle has homogeneous attribute values and

that there is no entering or leaving edge with the same attribute value

- pendant cycle (degree 1) : there is only one edge connected to the cycle. If an attribute is used (see below), this option means that the cycle has homogeneous attribute values and that only one edge with the same attribute value is connected to the cycle

- junction cycle (degree 2) : there is exactly two edges connected to the cycle and not belonging to it. If an attribute is used (see below), this option means that the cycle has homogeneous attribute values and that exactly 2 edges with the same attribute values are connected to the cycle :

- fork cycle (degree 3+) : there is three edges or more connected to the cycle. If an attribute is used (see below), this option means that the cycle has homogeneous attribute values and that at least 3 edges with the same attribute values are connected to the cycle :



There is a third section to select cycles with heterogeneous attributes. One can choose to return :

- all cycles with heterogeneous attributes (cycles with at least two different values, including null)
- only cycles including two degree 3 nodes, one with incident edges having attribute values A, A and B and the other incident edges having values A, B and B. This pattern, shown in the figures here after, often shows continuity problems due to wrong attribute values.