RENESENG 4th Year Deliverables

Report for the Roadmap Visualisation of Supply Chains

March 2018

Dr. Nathanial Cooper
Marie Curie ITN Experienced Researcher
ER 3.2

Imperial College London, Chemical Engineering, London

Report for the Roadmap Visualisation of Supply Chains

Specification & Functionalities

The purpose of this algorithm is to assist in the roadmap visualisation of biomass supply chains. It provides a method to easily view and visually understand the locations and transportation pathways recommended by the biomass supply chain optimisation. This is beneficial as it provides a useful tool for immediate intuitive assessment of the biomass supply chain optimisation, through visual inspection. It also can be used to create representations of competing optimisation pathways to understand how differences in constraints or inputs can affect the resulting transport map through direct comparison.

The roadmap visualisation extracts and displays two important data sets from the optimisation:

First, it extracts and displays the location of each type of refinery that the supply chain optimisation recommends. It provides a different point layer with a uniquely coloured circular marker for each different type of refinery used. The points representing each refinery are placed at the centroid of the cell within which they are located. A point layer with a uniquely coloured square marker is also generated for the demand centres.

Second, it extracts and displays the transport pathways of specified materials from generation to termination. The visualisation provides a different line layer with uniquely coloured arrow markers for each different type of material plotted. This can be done for any biomass, intermediary or product (other than power as this is not modelled as physically transporting between locations). The pathways are represented as arrows between each step, starting at the originating cell's centroid and ending with the arrow head at the destination cell's centroid. The arrows vary in thickness relative to the amount of material transported. Arrow thickness is scaled such that the minimum flow rate of the specified material is set to the minimum arrow size, and the maximum flow rate of the specified material is set to the maximum arrow size. Thus, the arrow thickness is useful as an approximating of transport volume for a given material, but not to compare between materials, as the transport volumes vary too much for meaningful comparison.

As indicated, for both of these visualisation aspects the displayed data is separated into individual layers, one for each subtype of data displayed (i.e. each refinery type or each material type). These layers can be activated or deactivated within the software GUI for improved readability. It should also be noted that the unique colours assigned to the point layers and the line layers do not correspond to one another – a blue point layer does not relate to a blue arrow layer. These are simply overlapping colour schemes.

The roadmap visualisation algorithm is designed to be used in conjunction with an open source geographic information software known as Quantum GIS or QGIS. This is a software that displays, analyses and modifies geographic data, and can handle a variety of raster and shapefile data (including most data types produced by ArcGIS). The software is available for free under a GNU General Public License from the website https://www.qgis.org. The algorithm is written in Python, which is the required language for algorithms in QGIS. The algorithm was developed using QGIS 2.18.7, and written in a variant of Python 2.7, called PyQGIS – the version of python used by this version of QGIS. More recent versions of QGIS have not been tested for stability, but hopefully will be tested in the future.

Currently the algorithm and user guide are held by Imperial College London and are available upon request. The source code is also available on GitHub, through user *nathanialcooper*, with project name *Biomass-Supply-Chain-Roadmapping* or similar. Currently, the link to the repository is https://github.com/nathanialcooper/Biomass-Supply-Chain-Roadmapping. The code is available under the same license as the QGIS software, the GNU General Public License, v3. There may be updated versions of the Specifications & Functionality report, and the user guide there as well. There may also be a plugin released for QGIS in the future.

How to include and use the algorithm in QGIS is covered in the user guide.

To produce the visualisations indicated about (refinery location and material transport), a variety of input information is required.

1) The output file (.lst) from the GAMS biomass supply chain optimisation. This file contains all information about the results of the optimisation, and so must be included. It must be selected through the standard windows file menu.

2) A shapefile (.shp) or other vector file of the grid corresponding to the cells used in the GAMS analysis. This is a grid overlay of the country or region of interest and represents the way in which the area of interest and its resources was divided for the optimisation. This can be selected from the list of currently loaded shapefiles, or from a file.

3) The field of the vector grid corresponding to cell ID number. Shapefiles can have a variety of information associated with them beyond just geographic location information. This is stored in fields. Cell ID can be stored here. This is selected from a dropdown menu, which is populated based on the fields available in the selected shapefile.

4) The text name of the variable from the GAMS optimisation used to identify the presence or absence of a refinery. All cells in the GAMS optimisation have a user-defined variable to identify whether there is a refinery. This variable name must be provided. This is a text entry.

5) The text name of the variable from the GAMS optimisation used to identify the flow of material between two cells. The GAMS optimisation has a user-defined variable to identify material flows between cells. This variable name must be provided. This is a text entry.

6) The cell ID numbers for demand centres used in the optimisation. The GAMS optimisation had a defined location for where products are demanded, defined by the user. This information must be provided for visualisation purposes. This is a text entry, with entries separated by a comma with no spaces.

7) The name of materials for which the user wants the roadmap visualisation. There are a variety of materials in the optimisation, and only some of them are relevant or may need to be tracked. This is a text entry, with entries separated by a comma with no spaces.

8) The name of transport methods used in the GAMS optimisation. There may be a variety of transportation methods available to employ in the optimisation. The names of these transportation methods must be provided for parsing of the results. This is a text entry, with entries separated by a comma with no spaces.

The outputs of the algorithm are the refinery and material transport layers indicated in the opening of this document. These layers appear in the QGIS software, overlaying the vector file of the grid (should this be set to viewable). These layers can then be used however the user sees fit once they have been created. For example, they can be saved for future use, or could alternatively be used in an image. The user could also incorporate these layers into maps of the region that are exported for later viewing or insertion into a report, using Map Composer or Print Composer.

Algorithm Methods

The algorithm is currently a script that is run from the Processing Toolbox of QGIS. The description of how to do this is provided in the User Guide. PyQGIS scripts are interpreted rather than compiled, so once the code has been inserted, it can be run immediately, provided the necessary inputs are available.

The program imports several key packages, including the qgis.core and PyQt4.core packages, as well as a colour package. These are necessary for handling and creating the new output layers, and assigning the correct morphology to them.

The algorithm goes on to parse the GAMS output file to sort information into vectors / matrices for future use. It does this by reading the output file line by line until it finds a “start” line corresponding to the beginning of a section of output from one of the specified variables. Once it reaches one of these “start” lines, it parses the data until it reaches the end of the data section, signified by the presence of a new variable’s “start” line, at which point it reverts to looking for a new “start” line for one of the relevant variables.

Parsing the information provided in the refinery locations variable is tricky, in that the output is designed to be a table that is human readable, but slightly less convenient for a machine. The table is structured with refinery types listed down the left side, and cell numbers listed along the top. A “1.000” is present at the intersection of elements which have a refinery of the given type, otherwise the space is blank. The parsing is aided, however, by the fact that the table is fixed width, and so the table contents can be tokenized, and the positions of the “1.000” tokens can be matched up with the cells values listed along the top of the table. This information can then be registered in a vector of vectors, where each subvector relates to one refinery type. The vector is used later to create the point layers. The vector is necessary as the table can span multiple line blocks, if the number of cells with refineries is too numerous.

Parsing the information provided in the flow rate variable is slightly more straightforward. There is a material and an origination cell listed on one line, and then a transport method, destination cell and transport volume listed on the subsequent line. Here, the algorithm simply needs to check whether the material and transport method used are of the types specified by the user. If they are, then the origination / destination cell pair, and the monthly average amount of material transported between the two, are registered in a vector of vectors, where each subvector relates to one material type.

The layers that are generated for each refinery type or material type must be easily differentiable from one another. The easiest way of doing this is by varying the colour of the layer between each type. For the point layers, each refinery type and the demand centres have a different colour for their points. For the arrow line layers, each material type has a different colour for the arrows. It is challenging to create visually distinct colours reliably when using the RGB colour model, but it is more straightforward when using the Hue Saturation Value colour model. Using this, a set of evenly spaced points between 0 and 1 can be selected for

Hue, while Saturation and Value are set to 1, generating visually unique pure colours. These can then be translated to values in the RGB colour model (what is used in PyQGIS) using the colorsys package.

To actually create the refinery location point layers, the vector of vectors with refinery locations is iterated through. In the subvectors, which each match a refinery type, the centroids of the cells corresponding to the cell ID numbers listed are added as points to the new point layers. Each refinery type is given its own colour from the list of generated colours, and is then added to the project, so the user can view it.

To actually create the material transport line layers, the vector of vectors with material transport pairs is iterated through. Each subvector corresponds to a single material type. Within each subvector, each transport pair is examined, where the centroid of the origination cell is used as the starting point for the arrow and the centroid of the destination cells is used as the arrow head point. The arrow size is then set so that the largest transport volume between two cells is the largest arrow size, and the smallest transport volume between two cells is the smallest arrow size, with all other arrows sizes scaled in between based on the transport volume. Each material type is given its own colour from the list of generated colours and is then added to the project, so the user can view it.