FlowTools

Report by Daniel Nüst (daniel.nuest@uni-muenster.de)
for the seminar *R+OSGeo* held by E. Pebesma[1], T. Hengl[2] and V. Olaya[3]

Summer semester 2009
Effective 15/10/09

1. Introduction

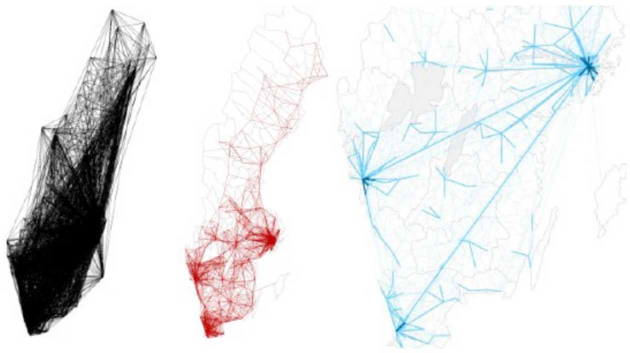
Flow maps are a well-known visualization technique within the cartographic community. They represent movement of material or abstract things (migration of people, traded goods) between two points in space. The encoded information comprises strength, direction and sometimes even course of the flow between locations. Albeit their powerful ability to remove visual clutter from these often complicated patterns, there are no advanced automatic algorithms for flow maps in professional geographic information systems. The biggest problem is the routing of the lines that connect the multiple pairs of points, because that route is normally not known or irrelevant (e.g. with abstract flows like money transactions). The lines should be clearly distinguishable and their course should look sensible or natural to the reader.
The most „beautiful“ examples, and thus more effective in carrying their message, remain hand-crafted, like the first flow maps by Minard[4] (e.g. Minard’s map of French wine exports for 1864[5]). Others are special animations, like flight pattern visualizations by Aaron Koblin[6], that come from a information visualization realm rather than geographic information science.

Some possible tools to create flow maps are:

- *Tobler’s Flow Mapper*[7] and *Flow Data Model Tools (FDMT) for ArcGIS*[8] can be used to create lines or arrow shaped visualisations of flow. Waldo Tobler developed an interactive standalone software to create flow maps which was ported to VBA macros for ArcGIS by Alan Glennon. Both projects are currently not under active development.
- *Flowmap 7.3*[9] *is a standalone software that offers extensive analysis features for flow maps, for example using a road network as a basis for travel cost calculation.*

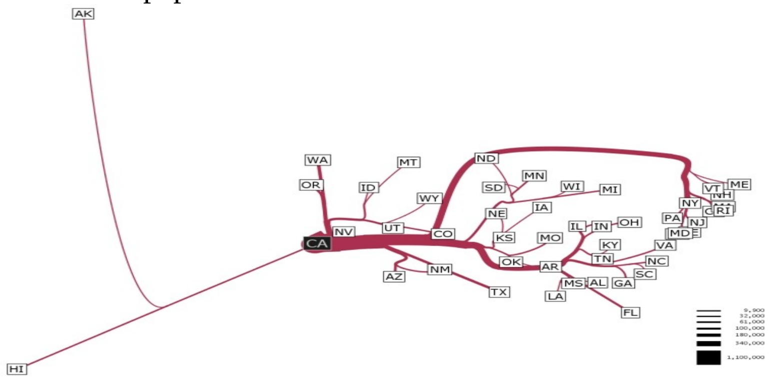
1 http://ifgi/de/professoren/3-professoren/3-pebesma-edzer-prof-dr
2 http://spatial-analyst.net/
3 http://www.volaya.es/
4 http://en.wikipedia.org/wiki/Charles_Joseph_Minard
5 http://en.wikipedia.org/wiki/File:Minard’s_map_of_French_wine_exports_for_1864.jpg
6 http://www.aaronkoblin.com/work/flightpatterns/index.html
7 http://www.csiss.org/clearinghouse/FlowMapper/
8 http://dynamicgeography.ou.edu/flow/
9 http://flowmap.geo.uu.nl/

- *Manual/halve-automatic creation of straight lines between source and target of the flow* – see examples below (own figures of commuters between Swedish communes created with MapInfo). The leftmost map shows the clutter that a large dataset with a lot of different sources and targets can create. The two maps to the right show filtered flows (“at least n-many people”) and a visualisation of flow strength by colour and line thickness.



- *Flow Maps – Automatic Generation and Visualization in GIS* by Birgit Pieke and Antonio Krüger [1] is a short paper based on a diploma thesis written at ifgi[10]. It presents an algorithm for automatic flow map layout that reduces intersections of edges. The algorithm is implemented as an ArcGIS extension.

These tools lack an intelligent routing for the lines. It should be intelligent in the sense that a good (understandable) visualization is created even if connections exist. Moreover they are not available as free and open source software. Also a hierarchical structuring of the flow that can simplify the flow patterns is not utilized. The output maps are also visually not really appealing. These are exactly the issues that are addressed by the tool *Flow Map Layout* (later referred to as *FML*) by Phan et al. [2]. Their technique uses „graph layout algorithms that minimize edge crossings and distort node positions“ to create a nice looking hierachical flow map while maintaining relative positions. See figure from the paper's website:



„Figure 1: A flow map of migration from California from 1995-2000, generated automatically by our system using edge routing but no layout adjustment.“

The freely available version of *FML* is limited to only one layer of data, meaning only one source (or sink) position for a group of target positions (or sources). Advanced versions with routing of several graphs that even share nodes were developed by the author as well.

10 http://ifgi.uni-muenster.de/

2. The Project

An original contribution to the R+OSGeo (R and Open Source geographic science) community is realized as the final couse assignment. The code of *Flow Map Layout* is written in Java and available under a BSD license[11], which is compatible with GNU General Public License[12]. Based on the algorithm that is implemented in *FML* some first flow mapping features are added to the SEXTANTE[13] algorithm library.

The code is published publically on the project's website[14] as a zip file and in a public repository (see section Code and Libraries). The licensing, documentation and availability of the project aim at encouraging others to engage themselves in developing more flow algorithms for SEXTANTE, developing my algorithms further or trying out open source software in general.

A comment has to be made about working with gvSIG and SEXTANTE. A guide is given with *SEXTANTE Programming Guide* [3] to set up an Eclipse-workspace[15] to develop algorithms with SEXTANTE and gvSIG. Sadly, the guide was partly incomplete and a few (partly undocumented) errors had to be solved. The building process across many projects was not trivial to apply. At two points, extensions stopped working and time was lost in getting the components running again. This resulted in loosing quite a few hours of working time to task not belonging to the actual project. The combination of gvSIG and SEXTANTE is a powerful tool and it's great to have all the original source code to see the inner workings – but it is a challenging set-up as well.

2.1 Algorithms

First, a simple version is implemented using direct straight lines between source and destination (see Manual/halve-automatic creation in the introduction). It simply creates lines between two points and attaches the given weight, which can be of any type as it is not processed at all, to it. These lines are stored in an output shape file. The weight can be used to visualize the strength of the flow along that line and thereby even filter out selected flows. This cartographic visualization is out of the scope of this project.
Second, the *FML* implementation is used as a library to create a more sophisticated routing of the lines. This graph is then transferred to lines in a a shape file, also attributed with a numeric weight. Naturally, the node adjustment feature of *FML* is deactivated to keep correct positions during the process. Because of the limitations of shape files regarding Bezier curves, a sampling method that creates many short lines to imitate the curve using a `FlatteningPathIterator`[16] is used. The properties of the curve sampling can be set via two numeric variables: maximum sampling flatness (the maximum allowable distance between the control points and the flattened curve) and maximum point limit (the maximum number of recursive subdivisions allowed for any curved segment). The sampling itself can even be deactivated by unchecking the corresponding box in the algorithm menu.
Multiple flows can be created using the layering option of any GIS utilizing SEXTANTE.

11 http://en.wikipedia.org/wiki/BSD_licenses
12 http://en.wikipedia.org/wiki/GNU_General_Public_License
13 www.sextantegis.com
14 http://ifgiweb.uni-muenster.de/~d_nues01/flowtools/
15 http://www.eclipse.org
16 http://www.j2ee.me/javase/6/docs/api/java/awt/geom/FlatteningPathIterator.html

2.2 Data Format

FML uses a database or a simple text file with comma separated values as input. The latter contains one source (stored along with coordinates and name) and many destinations (stored with coordinates, name and value of the flow). This approach could be supported by providing tabular input to the given algorithms. A more GIS like adapted approach was taken instead. This project's SEXTANTE algorithms require two shape files. The first file contains only the source point. The second file contains destination points with a flow size respectively a weight as an attribute. All points need to have an identifier attribute. Of course source and destinations can be used vice versa. The weight is assumed to be sensible already because it's processing lies out of the scope of this project. Preprocessing by the user, like standardization, might be required for a successful result. For simplicity the data is assumed to be in geographical coordinates (latitude and longitude) so that projected input points might require a conversion beforehand.
To test the algorithms a set of example shape files is created manually. They are based on the centres of some postcode areas in Münster, Germany. The files are located in the data folder in the project's repository.

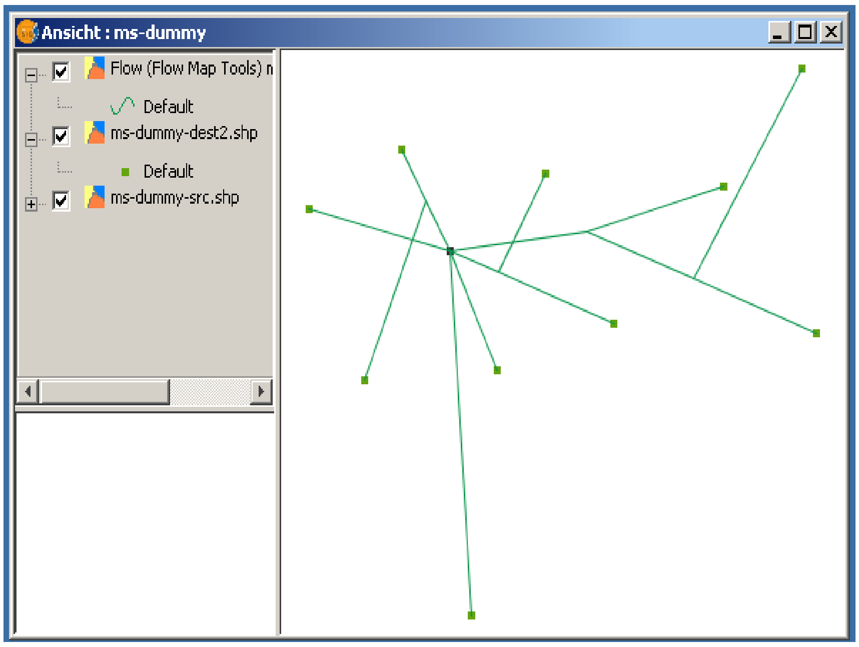
2.3 Problems

The goals of the project were not reached completely. The first, simple algorithms could easily be implemented. The second algorithm however is not successfully executed.

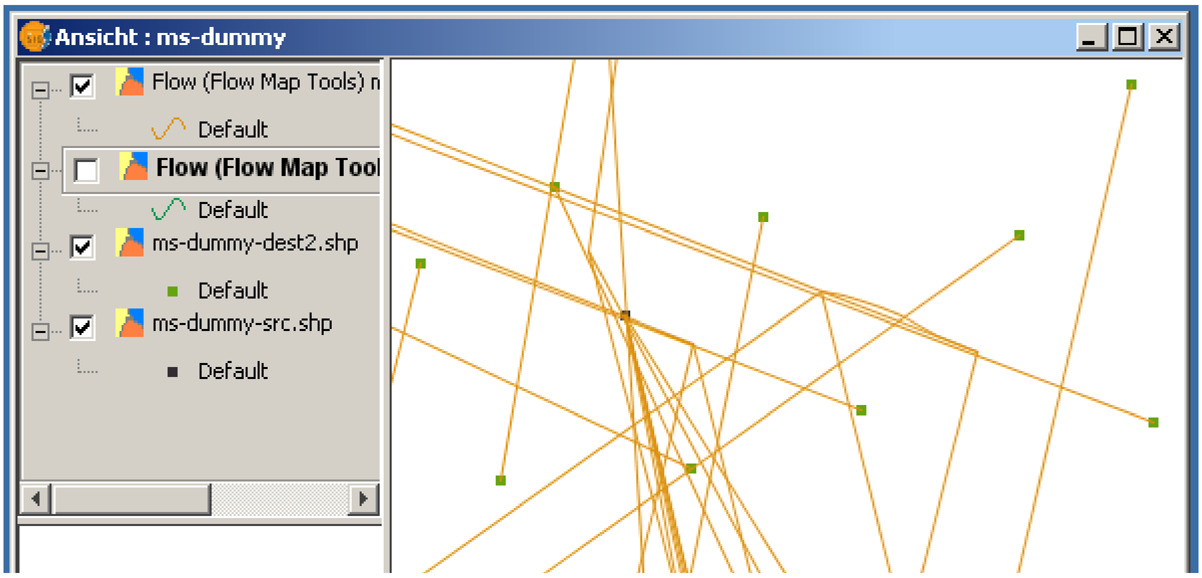
The problem lies in the calculation of the cubic curves. Erroneous values (i.e. NaN, not a number) are set for the control points of the Java class `CubicCurve2D`. These errors could be resolved with fewer destination points (see screenshot below). The root problem might be a rounding error problem, because the used dummy points lie relatively close to each other. This problem was irritating at first because the *FML* algorithm seemed to be running fine. The code used Java assertions to check for illegal values. That feature had to be activated in the virtual machine at first.

But even if the curves are calculated, then the lines take long ways around the points. This again might be caused by the close proximity of the input points. Deeper investigation of the layout algorithm itself is required of to resolve these issues.

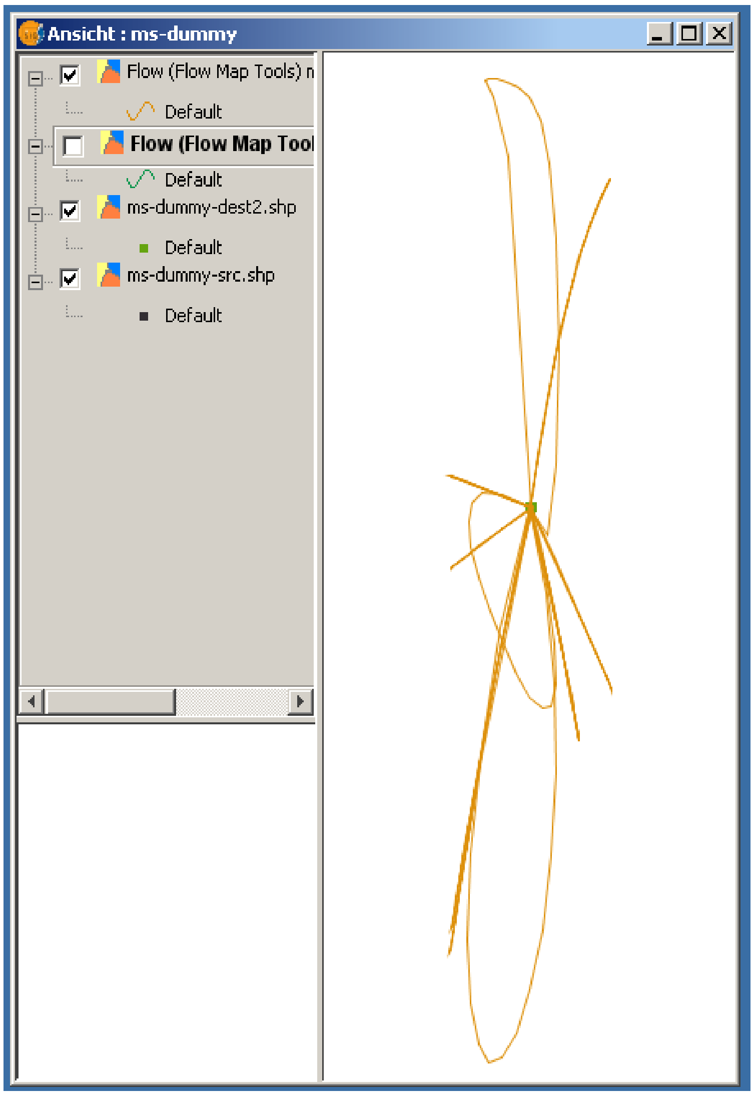
Screen shot of *FML* map without sampling the curves (see the binary tree structure of the graph):



A result with maximum sampling flatness 1.0E-5 and point limit 4. Close up:



Overview showing the whole line layer (see visible sampling of the edges in the second map which disappears with higher point limits):



2.4 Future Work

Some targets for future work, other than fixing the problems mentioned before, are

- creating additional flow algorithms in flow tools, not limited to visualization of flows.
- accepting tabular data as input (e.g. in the original *FML* format).
- testing further with larger and with actual (!) datasets.
- adding translations.
- adding maven support (i.e. a pom.xml document) for convenient development.
- integrate processing (classification etc.) for the flow variable.

3. Code and Libraries

Flow Map Layout is available under a quite non-restrictive BSD License. The libraries used by *FML* are available under GPL v2[17] or less restrictive licenses. Due to that they are included in lib folder in the project. These libraries are:

- epsgraphics.jar is licensed under GPLv2.
- j3dcore.jar, vecmath.jar are licensed under GPLv2 with the “classpath exception”[18].
- prefuse.jar[19] and prefusex.jar are licensed under BSD license[20].

The newly implemented SEXTANTE `GeoAlgorithms` are located in the package `de.ifgi.sextante.flowTools`. Adaptations of classes from *FML* are in the subordinated package `flowmap`. The code is available as an Eclipse Java-project in a Subversion[21] repository at the address `http://svn.xp-dev.com/svn/FlowTools/`.

The **installation procedure** is as follows:

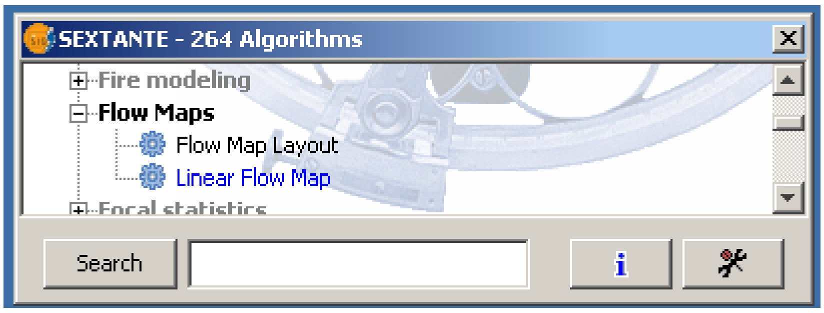
- Follow the instructions in [3], in the section *Setting up Eclipse to work with SEXTANTE and gvSIG*.
- Import the FlowTools project folder to your workspace (checkout via Subversion or import zip file in Eclipse).
- Adjust the following properties in the ant build-file to your needs: `extensions-dir`, `gvSIG-lib-dir`.
- Run the build file, start gvSIG and try out the algorithms with the provided dummy data (see Example Workflow).

4. Example Workflow

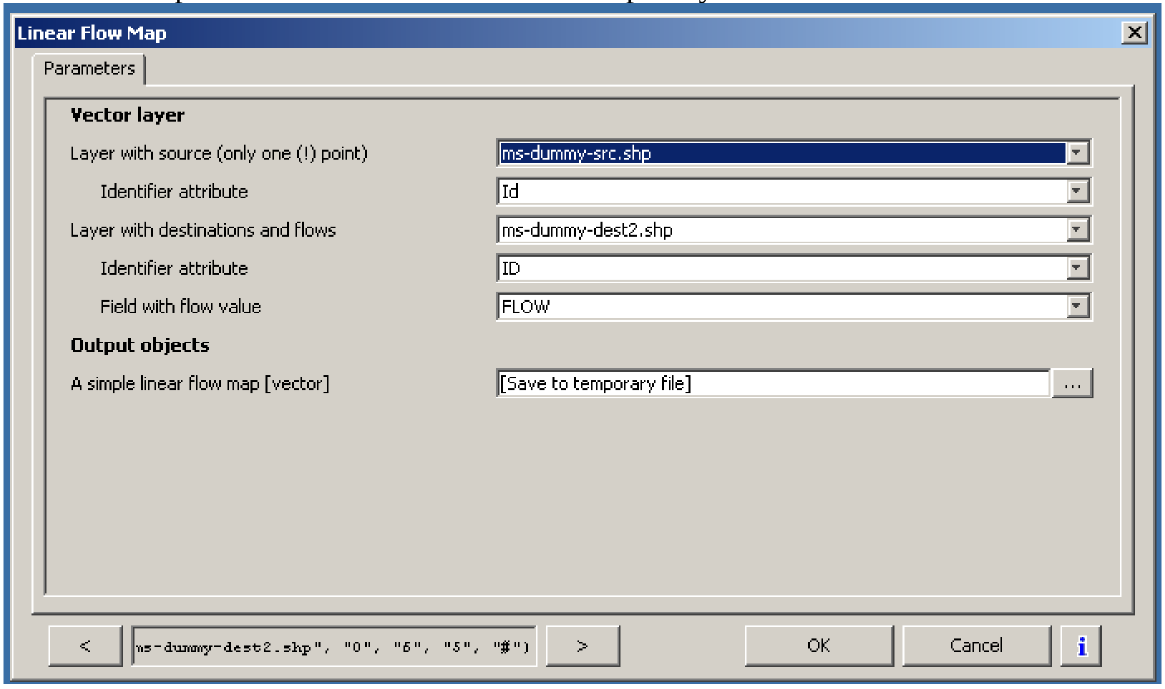
This workflow is based on dummy data that can be found in the data directory in the project folder (`ms-dummy-dest2.shp` for destinations and `ms-dummy-src.shp` for source of the flow). Due to the encountered problems with the *FML* variant the linear flow map algorithm is presented.

- Start *gvSIG* and add the shape files `ms-dummy-dest2.shp` and `ms-dummy-src.shp` as layers to your project.
- Navigate to the *Flow Map* algorithms.

17 http://www.jibble.org/licenses/gnu-license.php
18 https://java3d.dev.java.net/#Licenses
19 http://prefuse.org
20 http://prefuse.org/license-prefuse.txt
21 http://subversion.tigris.org



- Select *Linear Flow Map*.
- Select the respective files and attributes of the input layers.


Linear Flow Map

Parameters

Vector layer

Layer with source (only one (!) point)	ms-dummy-src.shp
Identifier attribute	Id
Layer with destinations and flows	ms-dummy-dest2.shp
Identifier attribute	ID
Field with flow value	FLOW

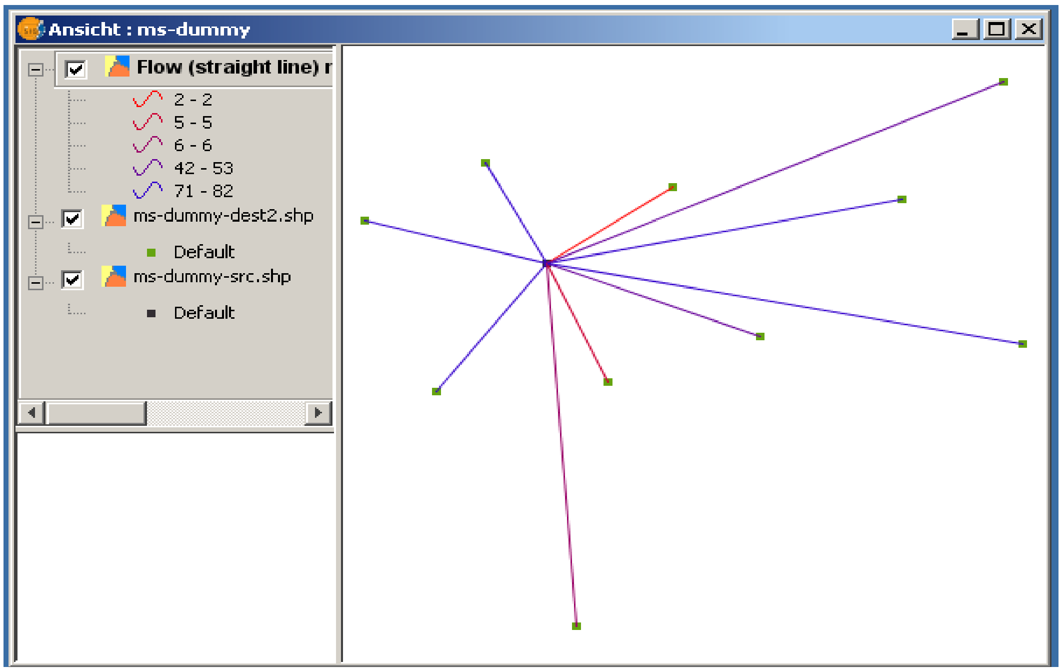
Output objects

A simple linear flow map [vector]	[Save to temporary file]

```
ms-dummy-dest2.shp", "0", "6", "5", "#")
```

- Click “Ok”.

The following screen shot shows a visualization of the flow strength (field FLOWSIZE) by colour of the lines:



5. References and Literature

[1] Pieke B, Krüger A. Flow Maps – Automatic Generation and Visualization in GIS. *Proceedings of GI-Days 2007*, IfGIprints 30, Münster; 261-265.
Download: http://www.gi-tage.de/archive/2007/downloads/acceptedPapers/pieke.pdf

[2] Phan D, Xiao L, Yeh R, Hanrahan P,Winograd T. Flow map layout. *Proceedings of the 2005 IEEE Symposium on Information Visualization*, 2005 (Mineapolis, MN), IEEE Computer Society, Washingon, DC, USA, 2005; 219–224.
Homepage: http://graphics.stanford.edu/papers/flow_map_layout/

[3] Olaya V. *SEXTANTE Programming Guide*, Version 1.0 Rev. May 19, 2009
Download: https://svn.forge.osor.eu/svn/sextante/docs/LaTeX/en/ProgrammingSextante/ProgrammingGuide.pdf

6. License

22 http://creativecommons.org/licenses/by-nc-sa/3.0/