Developing Graph Theoretic Analysis Tools in FOSS4GIS: An Experiment in OpenJUMP with a Specific Focus on Space Syntax

Burak Beyhan, PhD,
Mersin University, Faculty of Architecture,
Department of City and Regional Planning

21st – 23rd May 2012
FOSS4G-CEE & Geoinformatics, Prague 2012

Introduction:

FOSS movement has opened the door for GIS community to develop and share quality geoinformatics software programs even better than those developed by proprietary firms, as it was the case for the earlier paths along which the movement made it possible for the other communities to develop quality software programs in different fields.

Today even considerably simple FOSS4GIS programs are not any more simple GIS viewers. In terms of advanced GIS functions, properties and processes, they are becoming more and more competitive compared with the closed source software programs (Steiniger and Bocher, 2009; Beyhan, Belge and Zorlu, 2010).

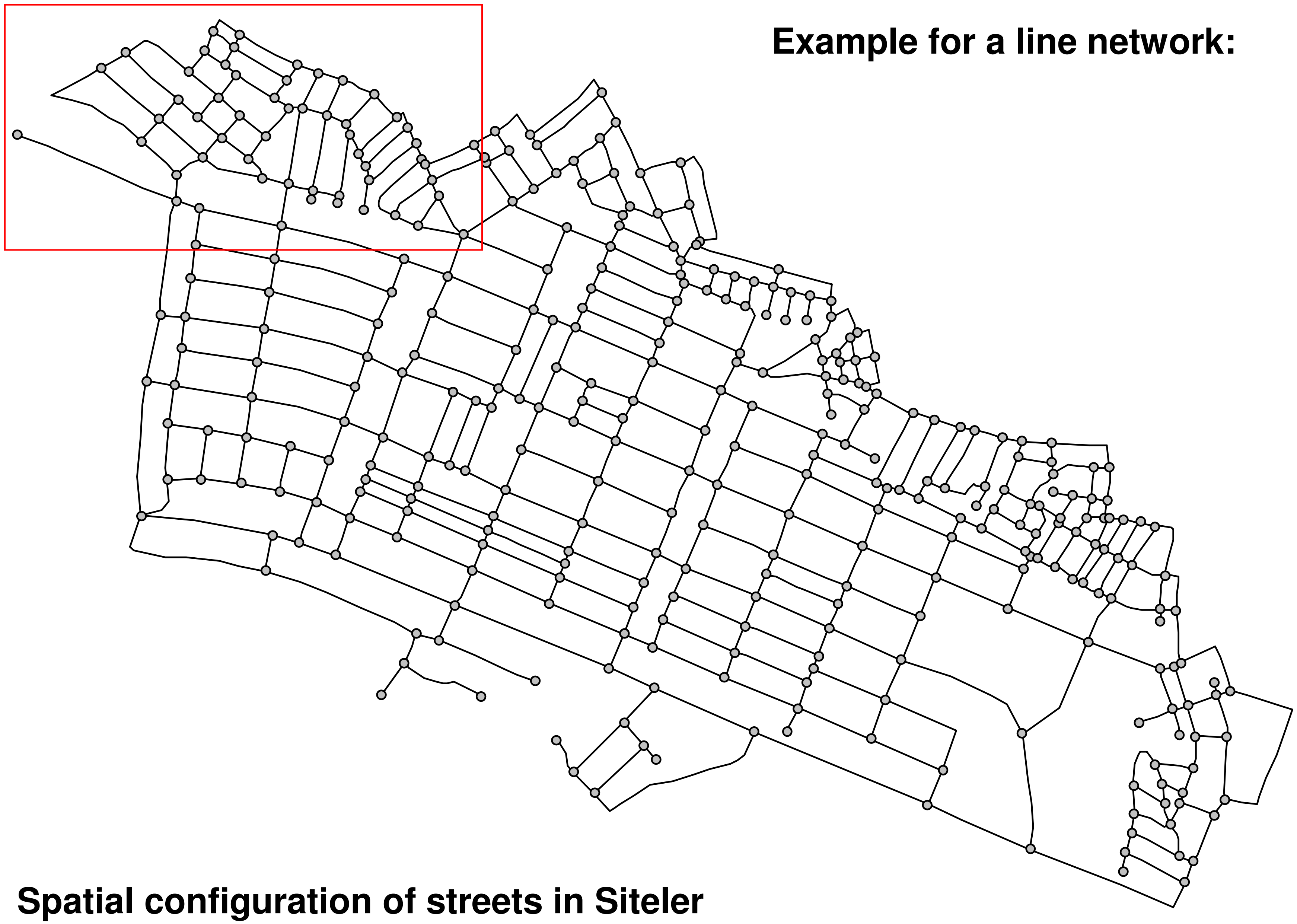
There exists a rich collection of libraries for further development of FOSS4GIS. Moreover, they are becoming increasingly available to larger group of users all over the world. FOSS4GIS save us to reinvent the wheel and makes it possible to build upon the experience of the others.

Aim of the study:

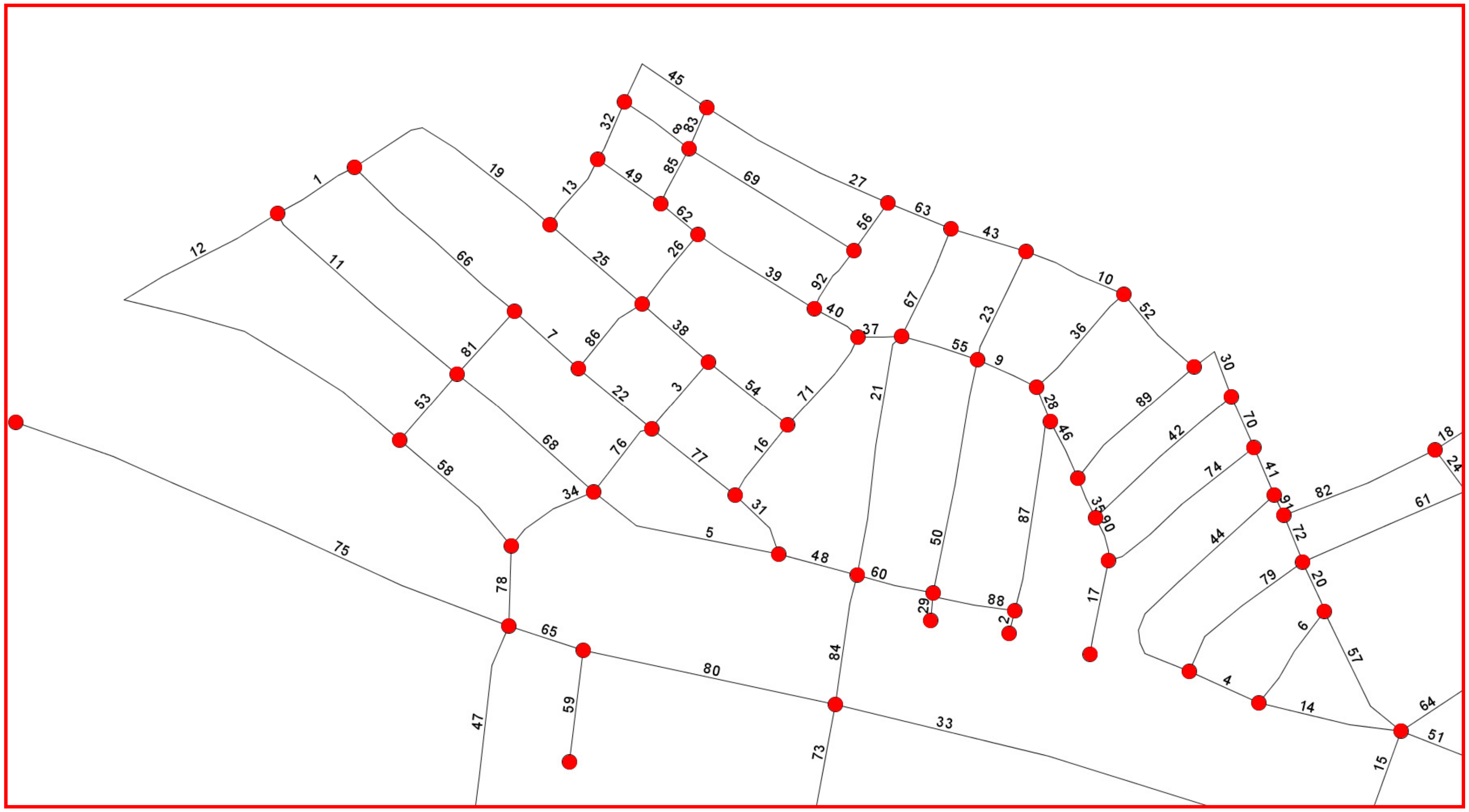
- The aim of this study is to reveal the scripting opportunities available in OpenJUMP for the analysis of spatial configurations with particular reference to Space Syntax Analysis (SSA).

As a graph theoretical method of analysis, in SSA the urban space is described by using an adjusted graph in which lines (such as streets) are represented as nodes, and the interconnections between them are shown as edges linking nodes.

Example for a line network:

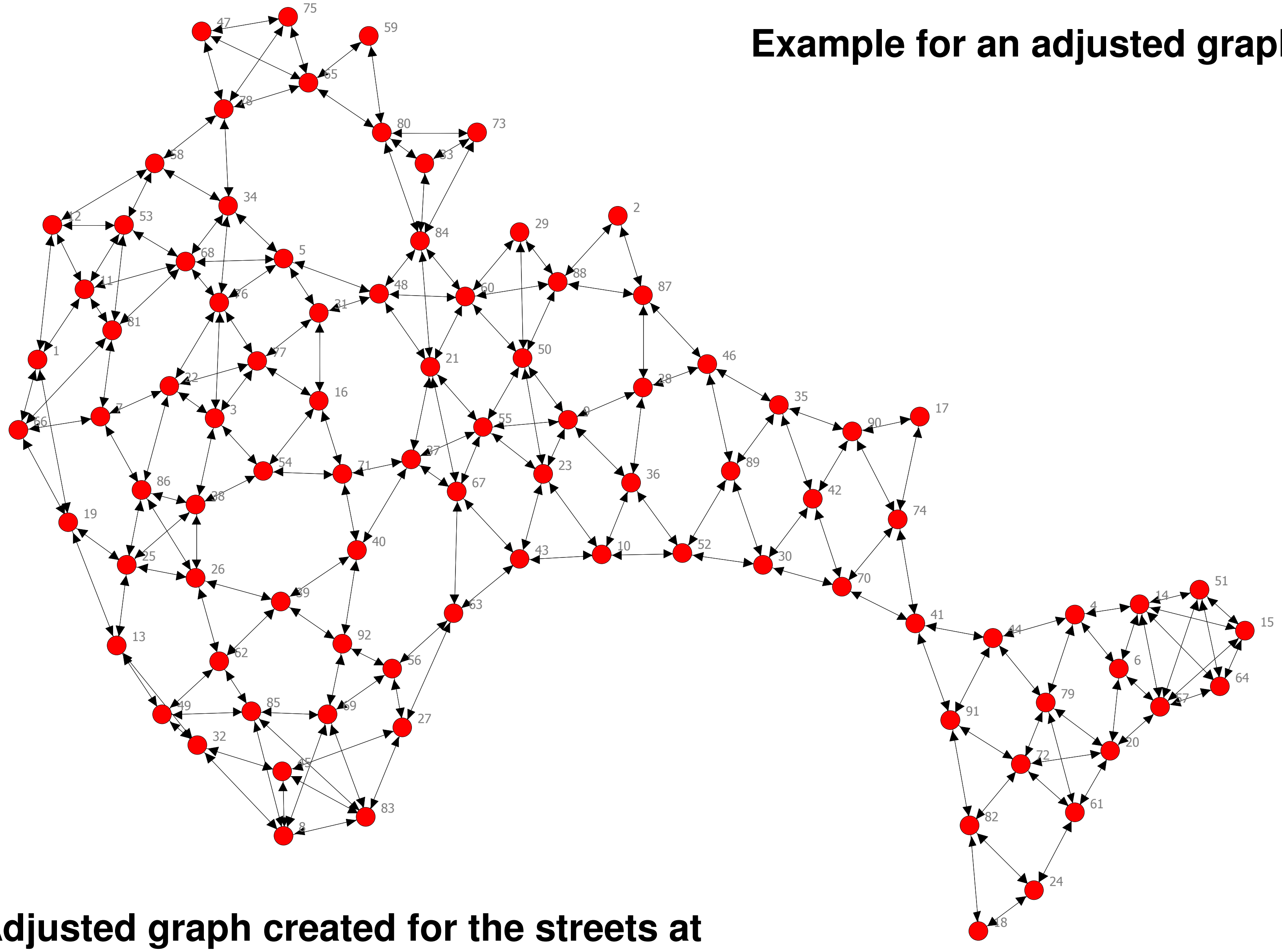


Spatial configuration of streets in Siteler



Streets at the north-east of Siteler

Example for an adjusted graph:



Adjusted graph created for the streets at the north-east of Siteler

Content of the presentation:

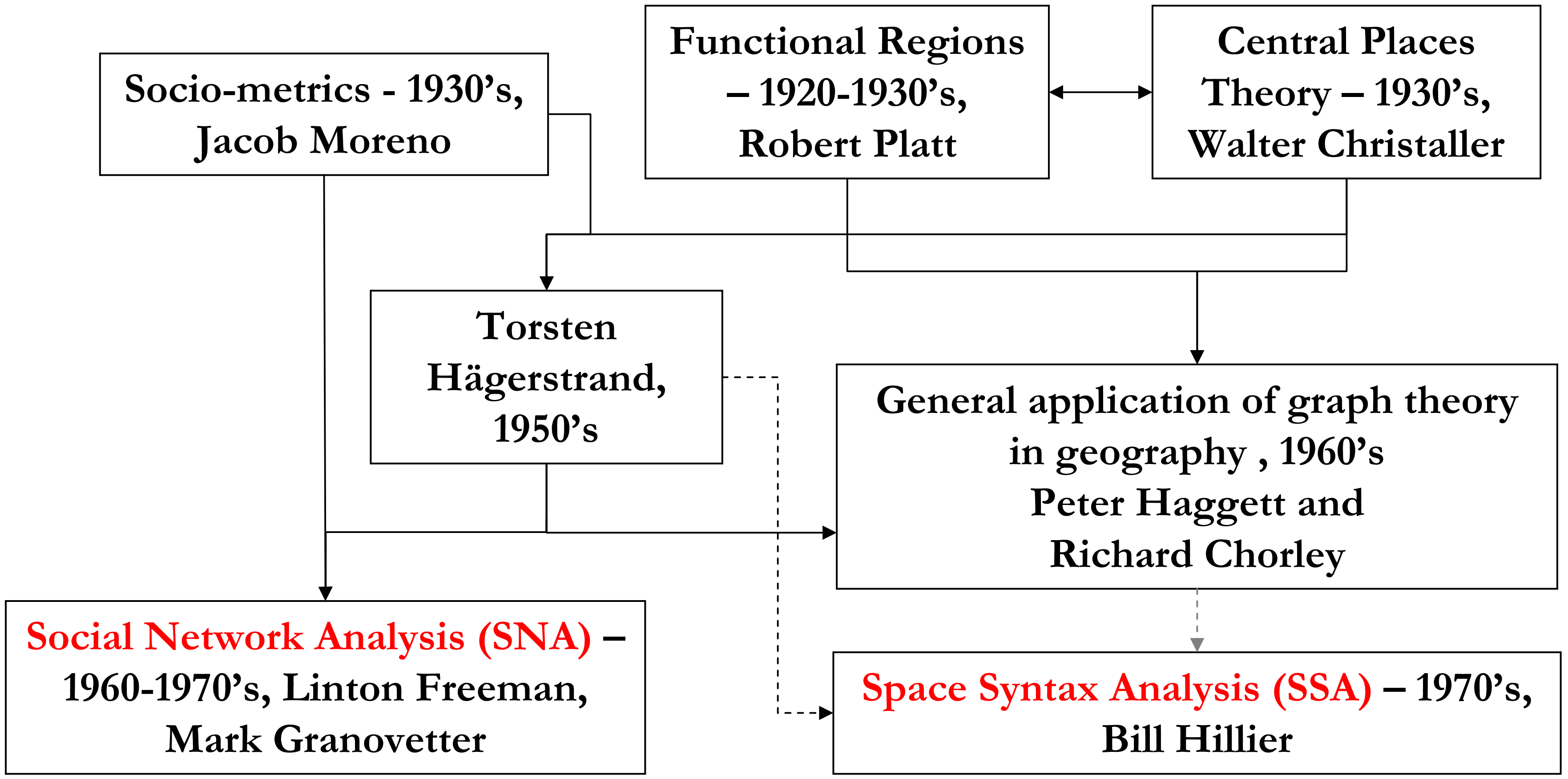
- A simple historical sketch of the graph theoretical approaches to the analysis of social netwoks and spatial configurations,
- Conceptualization and operationalization of SSA in GIS,
- Stages of the creation of a plugin for SSA,

A simple historical sketch for the socio-spatial methods of analysis based on graph theory:

It is strange to notice that Leonhard Euler’s well known essay on the impossibility of finding a solution for the Konigsberg seven-bridge problem which is assumed to be the first publication about graph theory stems from a spatial question. Thus, one can easily trace the employment of graph theory in spatial studies to the origin of the respective theory. Yet, it can be argued that one of the earlier applications of graph theory to the analysis of spatial configurations of Basic Spatial Units (BSU) can be observed within the field of regional science and geography in order to delimit the planning regions by drawing on functional regions.

On the other hand, a systematic application of graph theory in social studies was initiated by Jacob Moreno in 1930s under the name of “sociometrics” that has in later years become known as Social Network Analysis (SNA). It is important to notice that one may find the earlier considerations or implications of the employment of SNA for the analysis of spatial processes in the innovation diffusion studies of Hägerstrand. In his study on “Innovation Diffusion as a Spatial Process” Hägerstrand was very much influenced by Moreno.

A simple historical sketch for the socio-spatial methods of analysis based on graph theory:


Socio-metrics - 1930's, Jacob Moreno
Functional Regions – 1920-1930's, Robert Platt
Central Places Theory – 1930's, Walter Christaller
Torsten Hägerstrand, 1950's
General application of graph theory in geography , 1960's Peter Haggett and Richard Chorley
Social Network Analysis (SNA) – 1960-1970's, Linton Freeman, Mark Granovetter
Space Syntax Analysis (SSA) – 1970's, Bill Hillier

Conceptualization and operationalization of SSA in GIS:

In GIS Space Syntax Analysis may follow two different paths in its inception;

- studies based on vector databases;
    - **polygon networks** (convex spaces - Hillier and Hanson (2003)) (polygons are created according to an intuitive visual criteria and final map is composed of contiguous areas);
    - **line networks** (there are three options in the construction of line networks; (1) axial (based on vistas), (2) segment and (3) named line models) (Hillier and Hanson, 2003, Jiang and Claramunt, 2004; Tomko et al., 2008; Beyhan, 2010),
- studies based on grid databases;
    - creating **visibility index**,
    - application of SSA and SNA tools to visibility network, ...

Conceptualization and operationalization of SSA in GIS:

SSA plugins designed in this study is planned to be operational mainly for the analysis of line networks. Options available for the construction of line networks can be listed as below (Beyhan, 2010);

• Segment approach; dividing streets into their constituent parts by producing different nodes for each segment involved in any street,

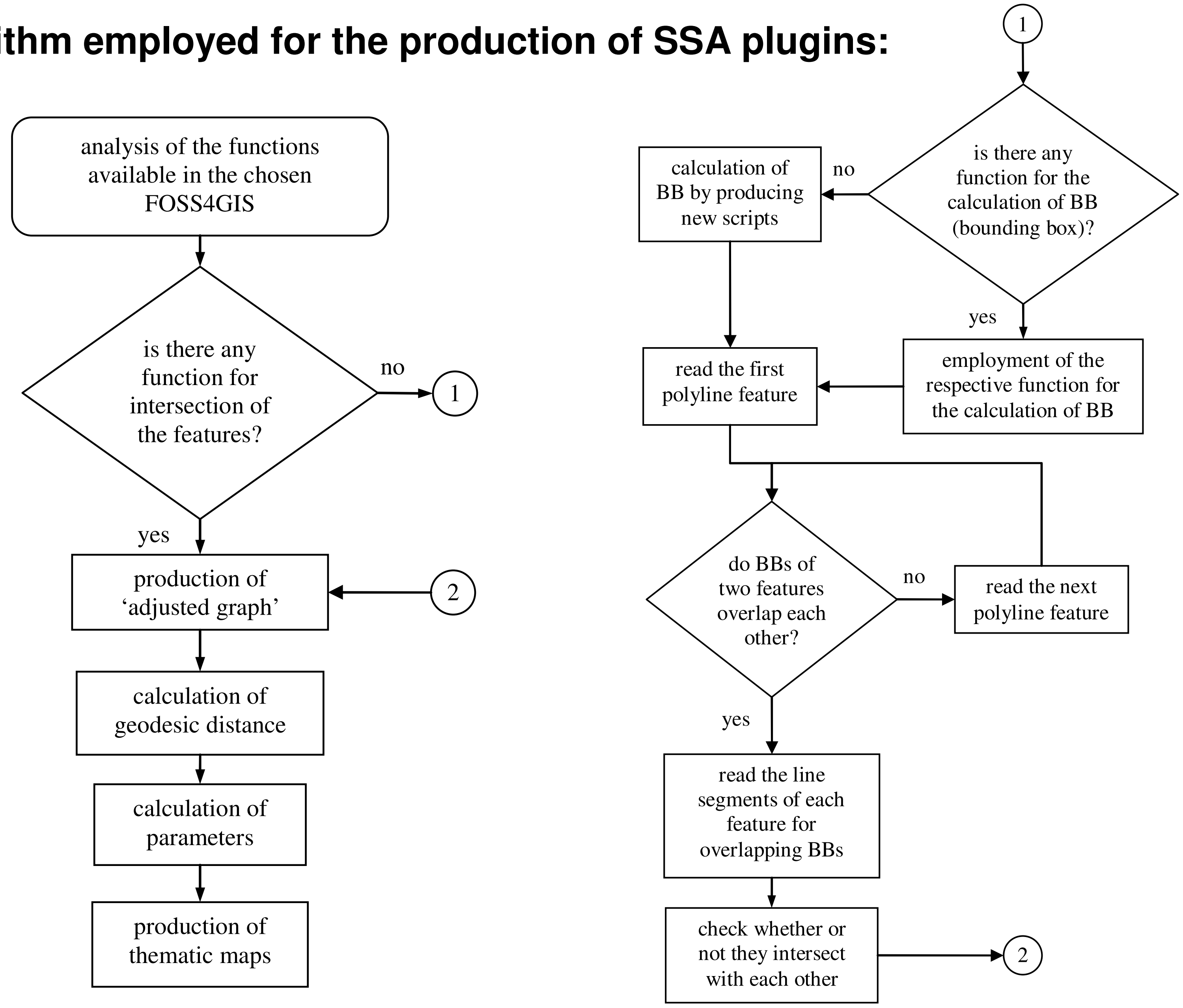
• Axial lines; the second option which employs axial lines is more traditional in SSA studies and actually based on the visibility pattern of the streets. In this option, streets are partitioned into axial lines in such a way that as long as possible each street is represented by a set of longest but fewest lines along which one can easily see both ends of the respective lines.

• Named streets; the third and last option takes the streets themselves as the basic spatial unit of analysis without disaggregating them into segments or axial lines. Named streets approach puts its emphasis on the fact that in daily life people usually refer to streets by employing their names not the particular segments that actually constitute the respective streets.

Stages of the study:

- Construction of the core algorithm used in the plugins for the creation of adjusted graphs,
- Calculation of widely used measures of space syntax by drawing on the interconnections between SSA and Social Network Analysis (SNA),
- Scripting opportunities and options in OpenJUMP and FOSS4GIS,
- Creation and testing of scripts by employing scripting platforms and console applications,
- Design of GUIs for space syntax plugin in Java, BeanShell and Jython.

Algorithm employed for the production of SSA plugins:



analysis of the functions available in the chosen FOSS4GIS
is there any function for intersection of the features?
no
1
yes
production of 'adjusted graph'
2
calculation of geodesic distance
calculation of parameters
production of thematic maps
1
is there any function for the calculation of BB (bounding box)?
no
calculation of BB by producing new scripts
yes
employment of the respective function for the calculation of BB
read the first polyline feature
do BBs of two features overlap each other?
no
read the next polyline feature
yes
read the line segments of each feature for overlapping BBs
check whether or not they intersect with each other
2

Source: Beyhan (2011).

Parameters in SSA in Connection with SNA;

Connectivity (ci); degree centrality which is based on the simple counting of the ties incoming or outing from a node in a network is called connectivity in SSA.

Total Depth (tdi) of a node is the sum of the geodesic distance between the respective node and the other nodes. It actually corresponds to the farness parameter in closeness centrality in SNA.

Mean Depth (mdi) of a node is the total depth divided by the total number of other nodes (the average length of the shortest paths between the respective node and every other node).

Local Depth of a node is its radius (usually) three depth.

Global Integration (gi) of a node actually correlates with the closeness centrality which is a measure reflecting the average length of the shortest paths to all other nodes of the graph (‘closeness centrality’ of a node in SNA can be defined as the reciprocal of ‘mean depth’).

Local Integration of a node is its radius (usually) three integration.

Formulas for basic space syntax parameters:

$$c_i = k$$

where *ci* is the connectivity value for basic spatial unit (BSU) *i*, *k* is the number of other BSUs connected to BSU *i*.

$$td_i = \sum_{j=1}^{n} d_{ij}$$

where *tdi* is the total depth value for BSU *i*, *dij* is the geodesic distance between BSU *i* and BSU *j*, and *n* is the total number of BSUs.

$$md_i = \frac{td_i}{n-1}$$

where *mdi* is the mean depth value for BSU *i*, *tdi* is the total depth value for BSU *i*, *n* is the total number of BSUs.

$$g_i = \left(\frac{\dfrac{2(md_i - 1)}{k-2}}{\dfrac{2(n(\log_2((n+2)/3)-1)+1))}{(n-1)(n-2)}} \right)^{-1}$$

where *gi* is the global integration value for BSU *i*, *mdi* is the mean depth value for BSU *i.*

Source: Hillier and Hanson (2003).

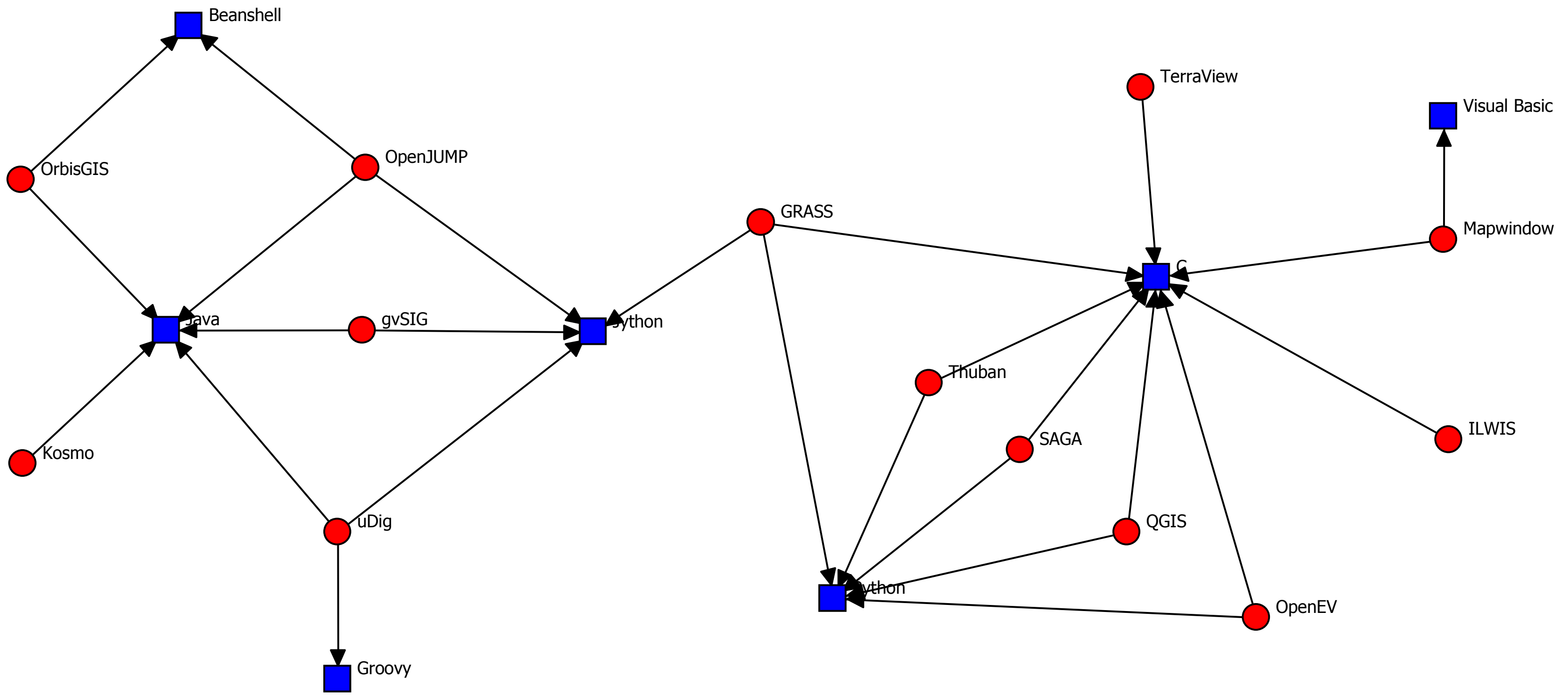
In OpenJUMP there are three different options to create plugins:

• employment of a **java** scripting platform (such as Eclipse platform) that is capable of compiling jar files,

• **BeanShell** scripts as plugins (it is easier for beginners to employ BeanShell console in order to see how scripts work and once the script is produced it can be called as a plugin via Menu in OpenJUMP),

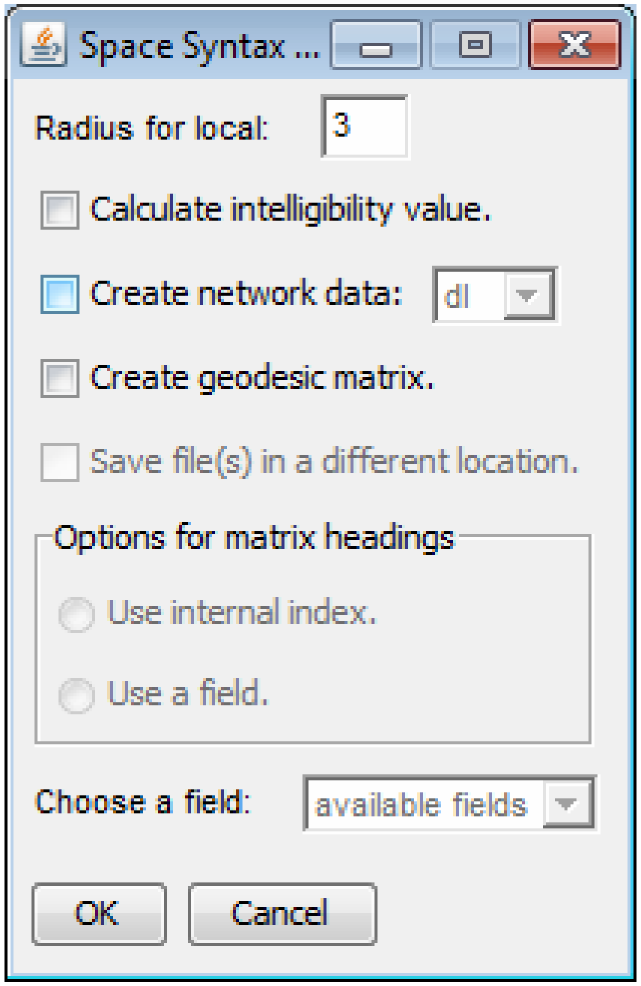
• **Jython** console and plugin option (although compared with jar plugins Jython plugins are slow, they work faster than BeanShell plugins).

Scripting options in FOSS4GIS:



GUIs for SSA plugin:

BeanShell and Jython:



Space Syntax ...

Radius for local: 3

Calculate intelligibility value.

Create network data: dl

Create geodesic matrix.

Save file(s) in a different location.

Options for matrix headings

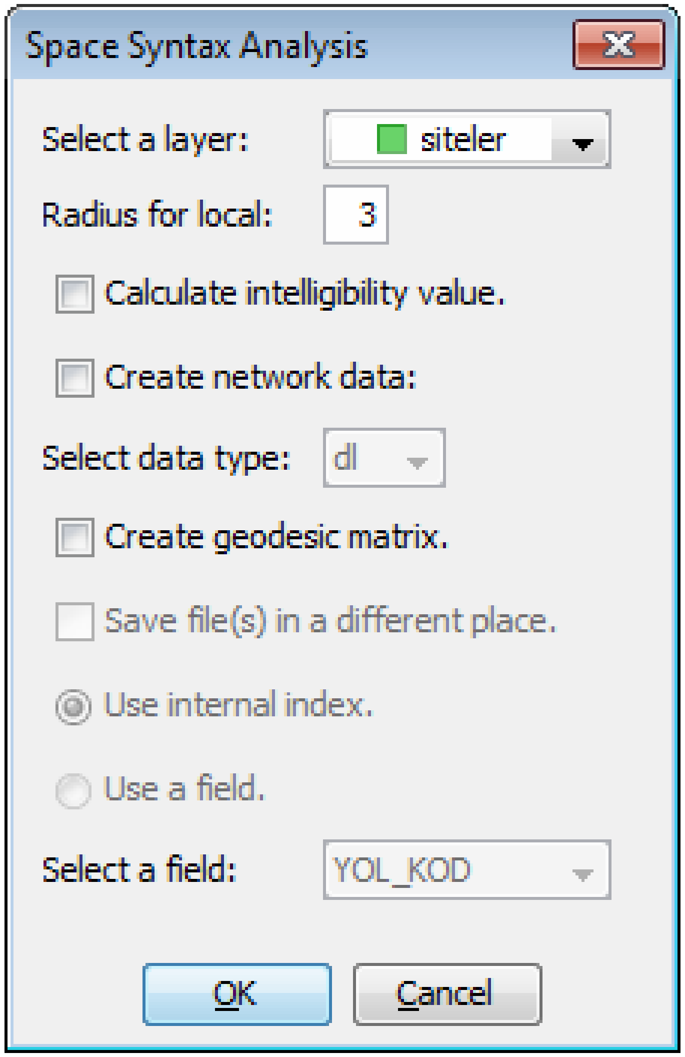
Use internal index.

Use a field.

Choose a field: available fields

OK Cancel

Java:



Space Syntax Analysis

Select a layer: 🟩 siteler

Radius for local: 3

Calculate intelligibility value.

Create network data:

Select data type: dl

Create geodesic matrix.

Save file(s) in a different place.

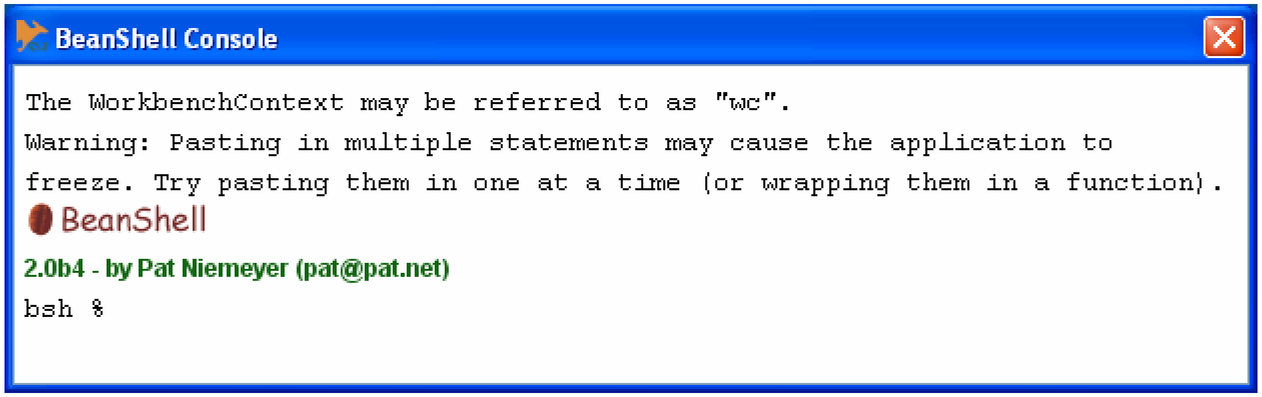
(●) Use internal index.

Use a field.

Select a field: YOL_KOD

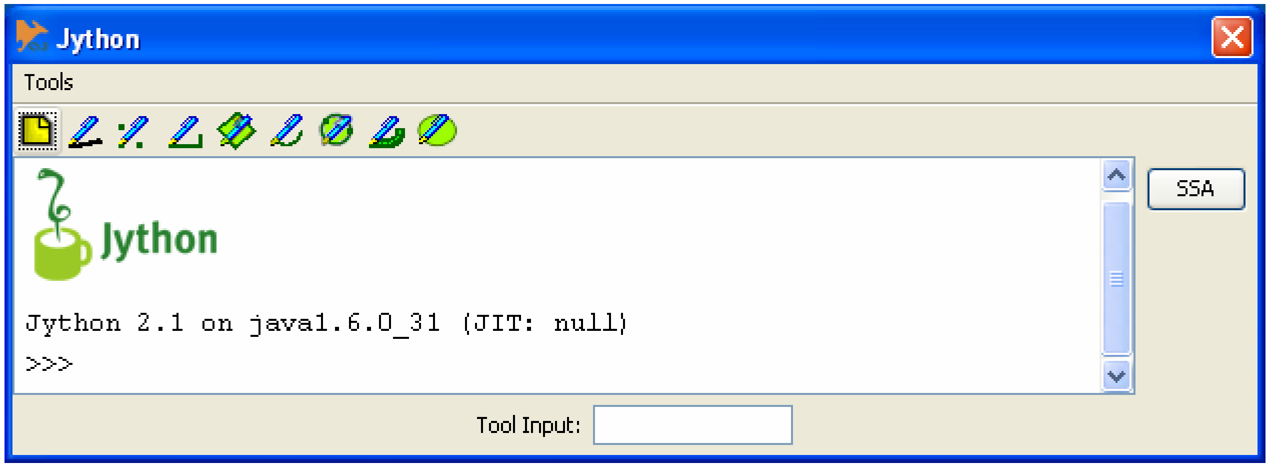
OK Cancel

BeanShell Console in OpenJUMP:



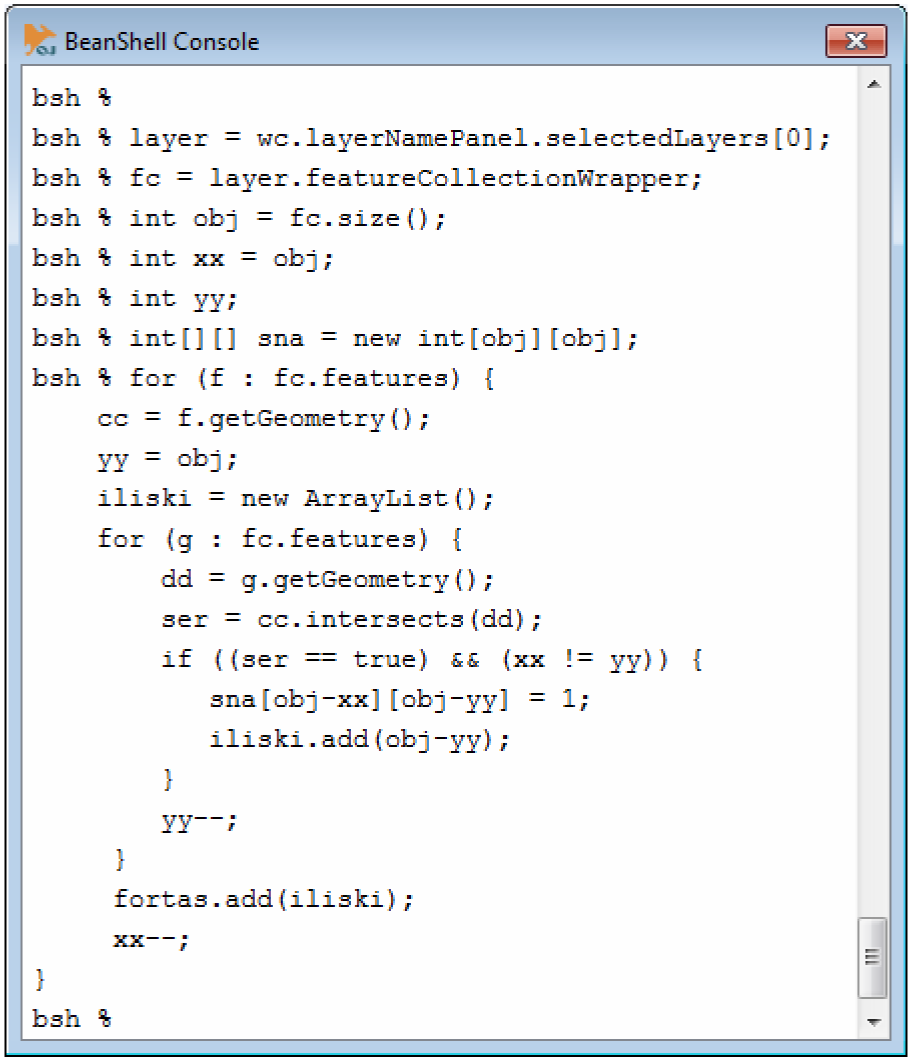
```
The WorkbenchContext may be referred to as "wc".
Warning: Pasting in multiple statements may cause the application to
freeze. Try pasting them in one at a time (or wrapping them in a function).
bsh %
```

Jython Console in OpenJUMP:



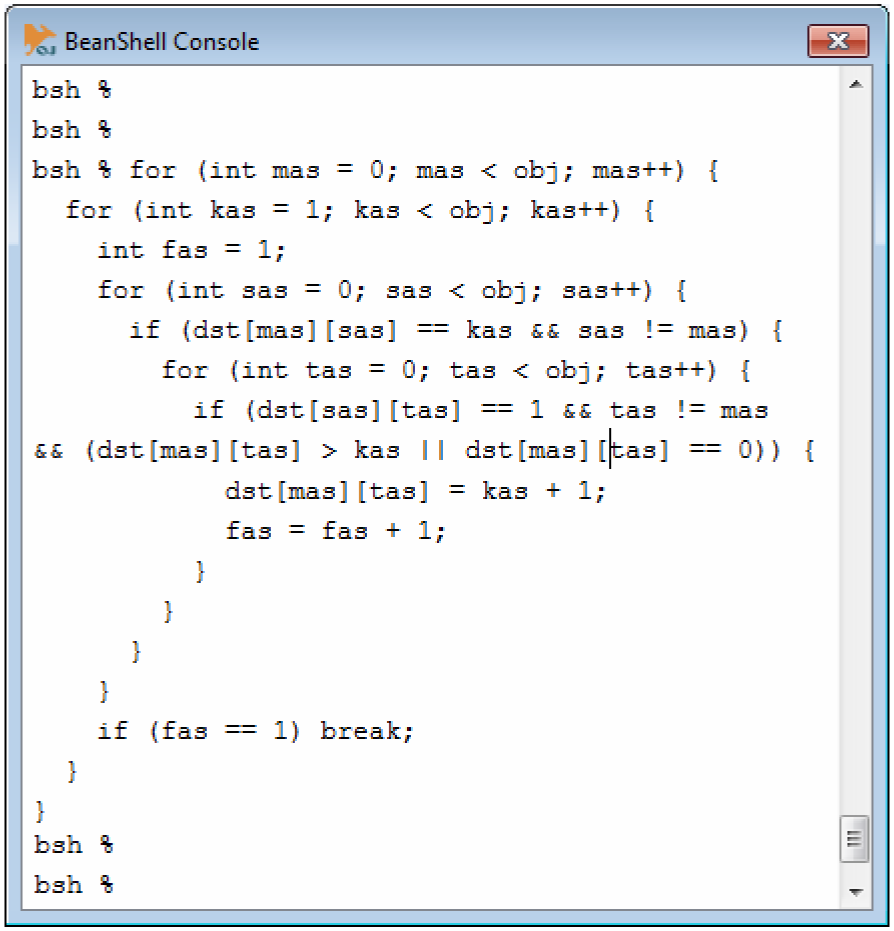
```
Jython 2.1 on java1.6.0_31 (JIT: null)
>>>
```

Construction of adjusted graph:



```
bsh %
bsh % layer = wc.layerNamePanel.selectedLayers[0];
bsh % fc = layer.featureCollectionWrapper;
bsh % int obj = fc.size();
bsh % int xx = obj;
bsh % int yy;
bsh % int[][] sna = new int[obj][obj];
bsh % for (f : fc.features) {
    cc = f.getGeometry();
    yy = obj;
    iliski = new ArrayList();
    for (g : fc.features) {
        dd = g.getGeometry();
        ser = cc.intersects(dd);
        if ((ser == true) && (xx != yy)) {
           sna[obj-xx][obj-yy] = 1;
           iliski.add(obj-yy);
        }
        yy--;
     }
     fortas.add(iliski);
     xx--;
}
bsh %
```

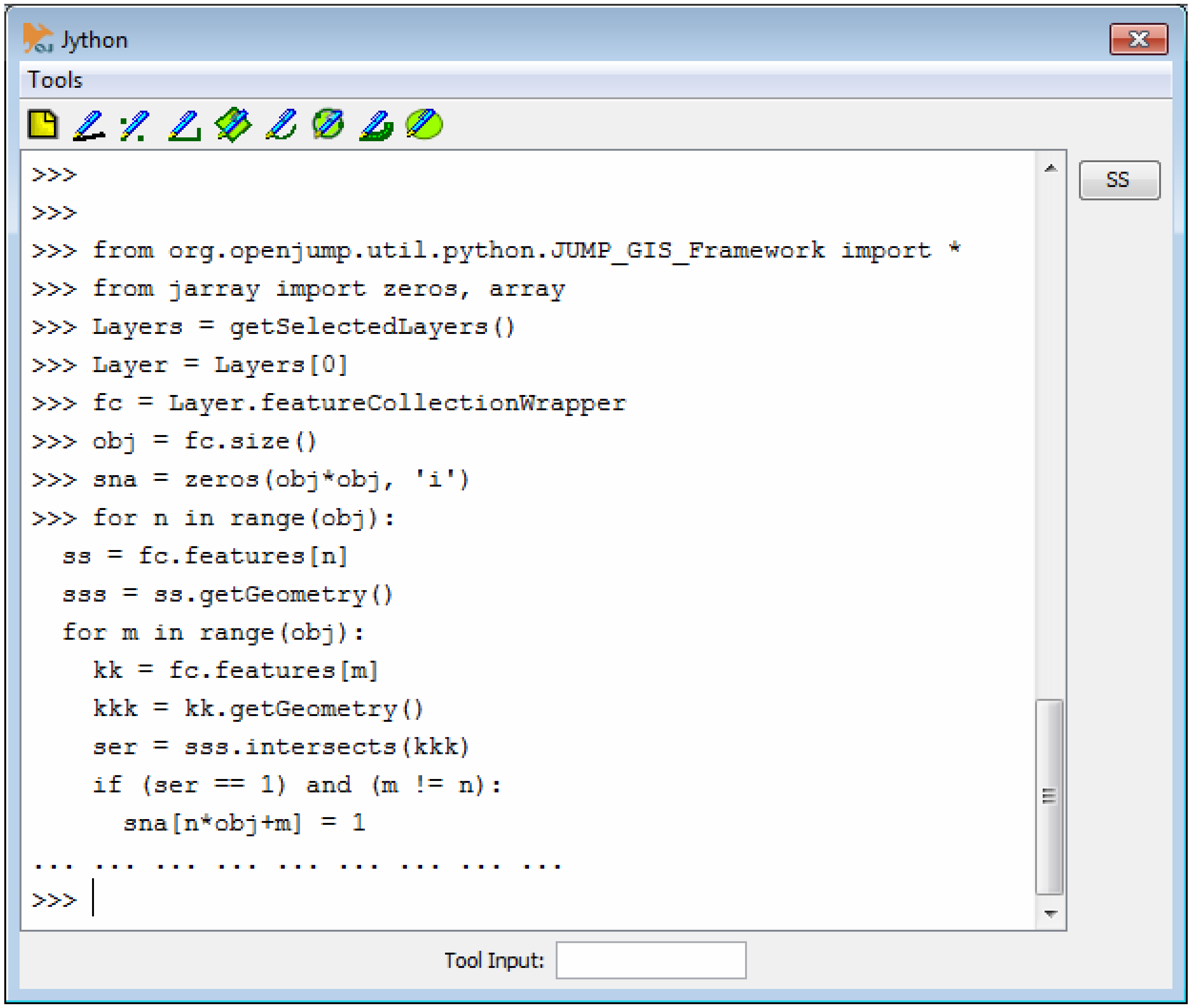
Calculation of geodesic:



```
bsh %
bsh %
bsh % for (int mas = 0; mas < obj; mas++) {
  for (int kas = 1; kas < obj; kas++) {
    int fas = 1;
    for (int sas = 0; sas < obj; sas++) {
      if (dst[mas][sas] == kas && sas != mas) {
        for (int tas = 0; tas < obj; tas++) {
          if (dst[sas][tas] == 1 && tas != mas
&& (dst[mas][tas] > kas || dst[mas][tas] == 0)) {
            dst[mas][tas] = kas + 1;
            fas = fas + 1;
          }
        }
      }
    }
    if (fas == 1) break;
  }
}
bsh %
bsh %
```

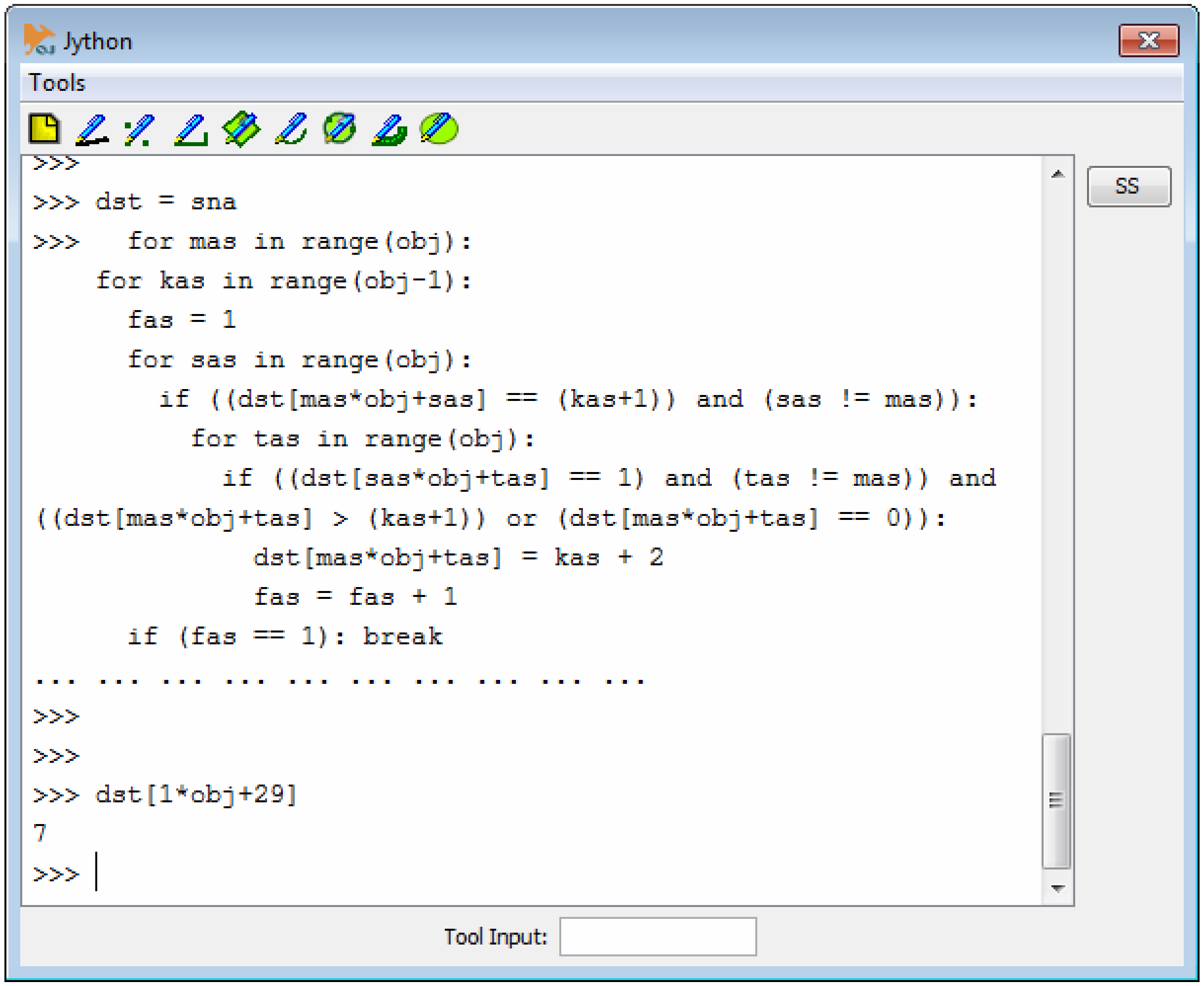
To run the script as a plugin, save it as a ".bsh" file and copy to "…\ext\BeanTools"

Construction of adjusted graph in Jython Console:



```
>>>
>>>
>>> from org.openjump.util.python.JUMP_GIS_Framework import *
>>> from jarray import zeros, array
>>> Layers = getSelectedLayers()
>>> Layer = Layers[0]
>>> fc = Layer.featureCollectionWrapper
>>> obj = fc.size()
>>> sna = zeros(obj*obj, 'i')
>>> for n in range(obj):
  ss = fc.features[n]
  sss = ss.getGeometry()
  for m in range(obj):
    kk = fc.features[m]
    kkk = kk.getGeometry()
    ser = sss.intersects(kkk)
    if (ser == 1) and (m != n):
      sna[n*obj+m] = 1
... ... ... ... ... ... ... ... ...
>>>
```

Calculation of geodesic in Jython Console:



```
>>>
>>> dst = sna
>>>   for mas in range(obj):
    for kas in range(obj-1):
      fas = 1
      for sas in range(obj):
        if ((dst[mas*obj+sas] == (kas+1)) and (sas != mas)):
          for tas in range(obj):
            if ((dst[sas*obj+tas] == 1) and (tas != mas)) and
((dst[mas*obj+tas] > (kas+1)) or (dst[mas*obj+tas] == 0)):
              dst[mas*obj+tas] = kas + 2
              fas = fas + 1
      if (fas == 1): break
... ... ... ... ... ... ... ... ... ...
>>>
>>>
>>> dst[1*obj+29]
7
>>> 
```

In order to calculate geodesic distance between the nodes in a connected graph, algorithmically four nested loops and a simple counter are employed in the scripts given above. The first (mas) and third (sas) loops functions respectively as the row and column counters required in order to read the values in the adjacency matrix. The second loop (kas) is employed as an accumulator index. If the value of a non-diagonal cell identified by the values of the first and third loops is equal to the accumulator index that takes values beginning from 1 signifying that the nodes indicated by the row and column numbers of the respective cell is adjacent to each other, another loop (tas) is initiated in order to detect the nodes contiguous to the node indicated by the value of the third loop and adjacent to the node specified by the value of the first loop. In other words, the first run of the fourth loop (tas) for each couple of the adjacent nodes specified by the values of mas and sas (regularly incremented in the first and third loops) detects the nodes located within a geodesic of 2 from the node which is identified by the values of the first loop. If such kind of a cell is identified, the value of the cell identified by the values of first and fourth loops is set to the value of the accumulator index (kas) plus 1. In order to shorten the time period spent for the calculation of geodesic between the node specified by the value of the first loop and the other nodes in the graph, a counter named fas is employed.

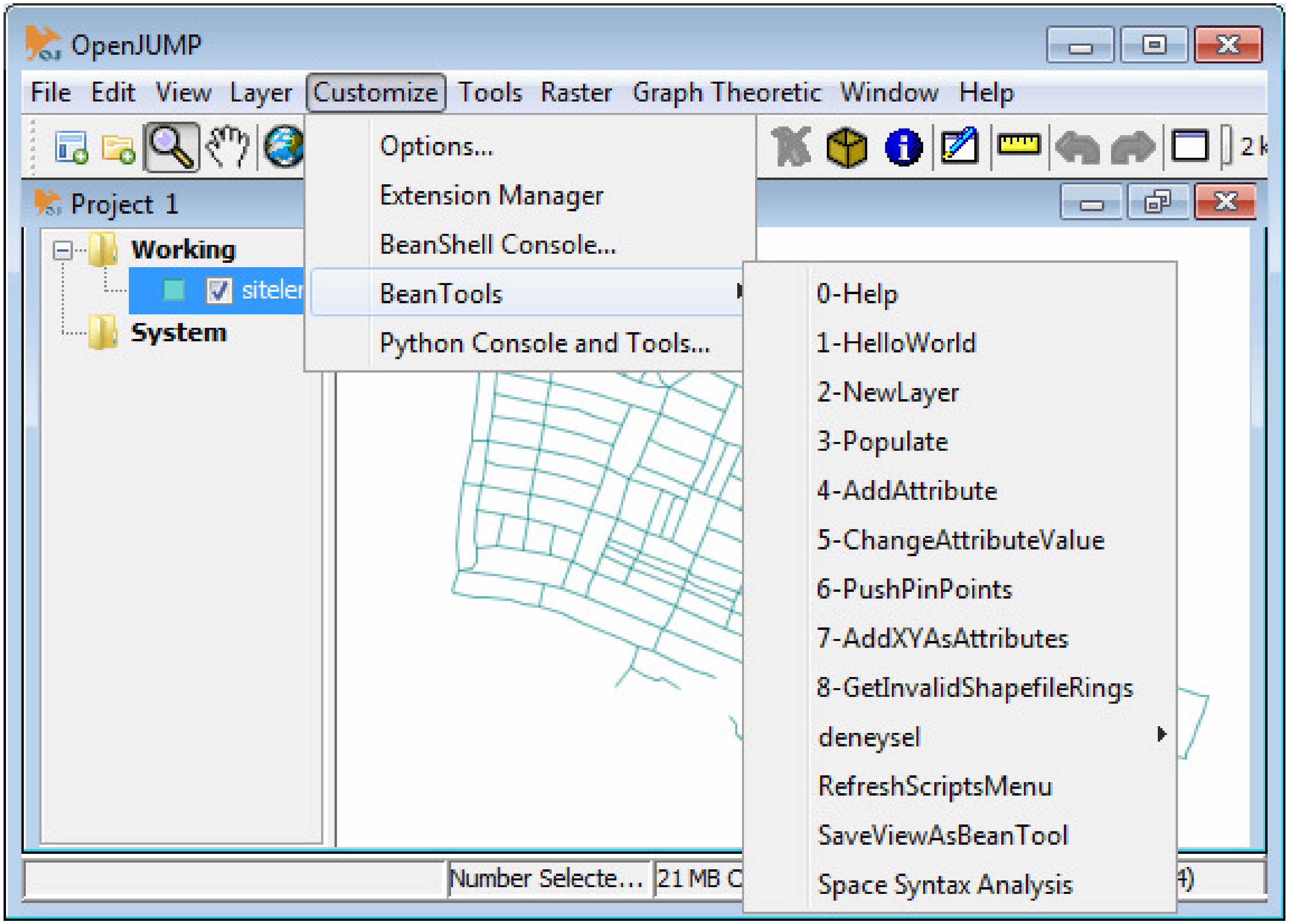
After identification of all the nodes located within a geodesic of 2 from the node indicated by the value of mas, the value of the second loop is incremented by 1 in order to find the nodes located within a geodesic of 3 from the respective node. If all the geodesics between the first node indicated by the value of mas and the other nodes are calculated, the counter fas breaks the second loop and resets the accumulator index for the calculation of the geodesics between the second node indicated by the value of mas and the other nodes. The respective procedure is repeated till the matrix showing geodesic between the nodes in the graph is constructed. Some conditionals are also used in order to prevent the miscalculations in the algorithm.

In order to increase the speed of the algorithm, the fourth loop is modified by employing a two dimensional array list (*fortas*) showing only the nodes adjacent to each other and another one dimensional array list (*iliski*) created from the first one and showing only the nodes that are adjacent to the node indicated by the value of the third loop (sas). The BeanShell scripts containing the respective modifications are given as below;

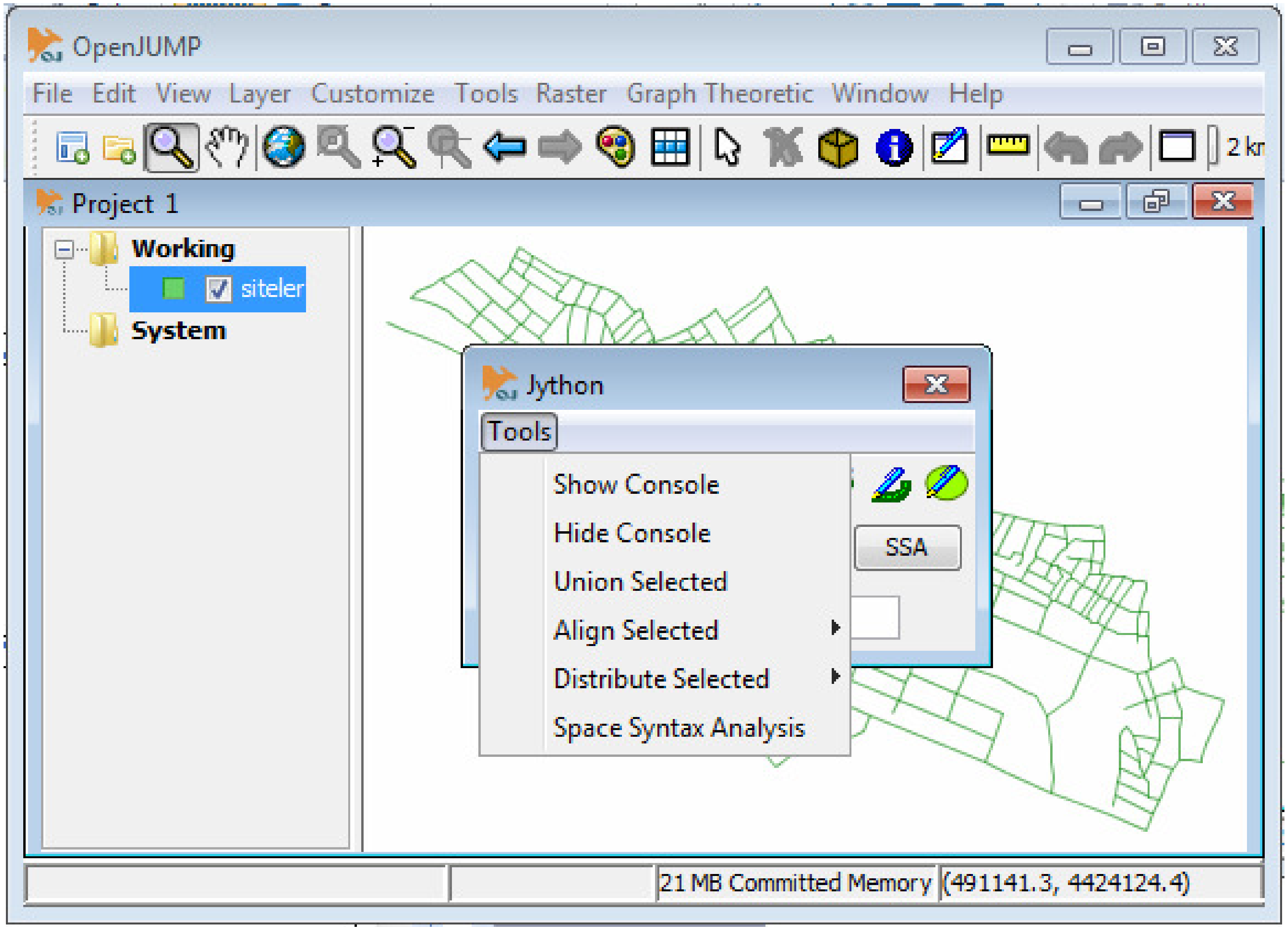
```
// construction of adjusted graph – for accelerated geodesic calculation

    fortas = new ArrayList();

    for (f : fc.features) {
            cntr = 0;
            cc = f.getGeometry();
            yy = obj;
            iliski = new ArrayList();
            for (g : fc.features) {
              dd = g.getGeometry();
              ser = cc.intersects(dd);
                if ((ser == true) && (xx != yy)) {
                  sna[obj-xx][obj-yy] = 1;
                  iliski.add(obj-yy);
               }
               yy--;
            }
            fortas.add(iliski);
            xx--;
    }

    dst = sna;
```

```
// calculation of geodesic – improved and accelerated script

for (int mas = 0; mas < obj; mas++) {
   for (int kas = 1; kas < obj; kas++) {
      int fas = 1;
      for (int sas = 0; sas < obj; sas++) {
         if (dst[mas][sas] == kas && sas != mas) {
            iliski = new ArrayList();
            iliski = fortas.get(sas);
            int sizeOfList = iliski.size();
            for (int sIndex = 0; sIndex < sizeOfList; sIndex++){
               int tas = iliski.get(sIndex);
               if (tas != mas && (dst[mas][tas] > kas || dst[mas][tas] == 0)) {
                  dst[mas][tas] = kas + 1;
                  fas = fas + 1;
               }
            }
         }
      }
      if (fas == 1) break;
   }
}
```

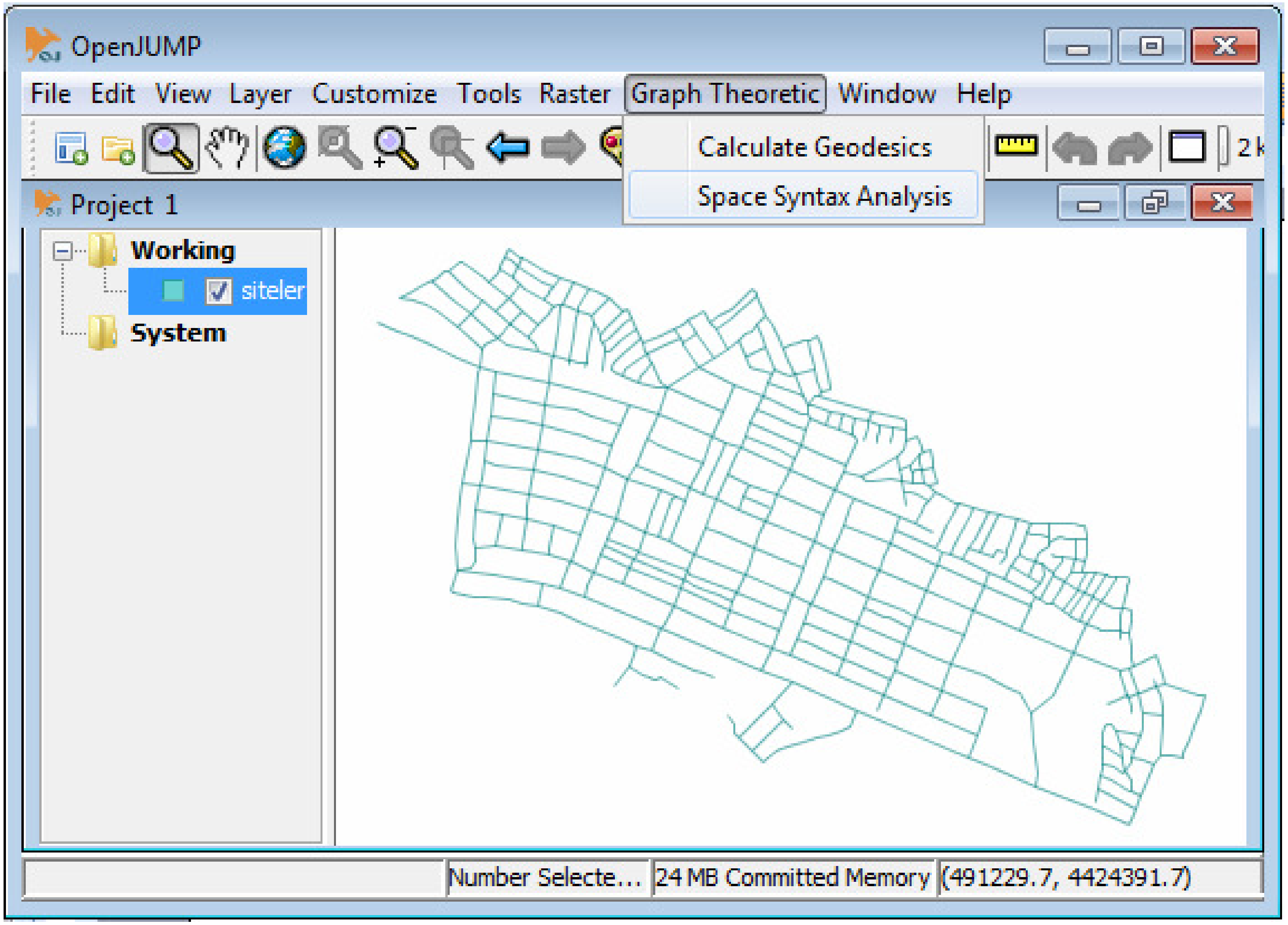
Running BeanShell SSA script as a plugin:



Jython SSA plugin:



Java SSA plugin:



For further analysis of the network data, two output options are available in SSA plugin;

- UCINET Data Language (dl) file,
- PAJEK’s Network (net) file.

If the adjusted graph is exported into any of the above formats, a series of graph theoretic parameters can be calculated not only in UCINET and PAJEK that are freely available for academic purposes, but also in other SNA software programs such as ORA, visone and socnetv.

For example, one can easily open the respective network in ORA and conduct further analysis (such as calculation of other measures of centrality including closeness and betweenness). The new parameters calculated in ORA can later be transferred to FOSS4GIS by saving the results to a csv file that can be linked to the original spatial data in GIS.

Centrality meausres available in ORA;

- **Authority Centrality**
- **Betweenness Centrality**
- **Bonacich Power Centrality**
- **Closeness Centrality**
- **Eigenvector Centrality**
- **Hub Centrality**
- **Information Centrality**

List of meausres available in ORA:

Access Redundancy
Actual Workload
Agent Socio Economic Power
Assignment Redundancy
Authority Centrality
Average Distance
Average Speed
Betweenness Centrality
Betweenness Network Centralization
Bonacich Power Centrality
Boundary Spanner
Burt Constraint
Capability
Classic SNA Density
Clique Count
Closeness Centrality
Closeness Network Centralization
Cognitive Demand
Cognitive Distinctiveness
Cognitive Expertise
Cognitive Resemblance
Cognitive Similarity
Column Breadth
Column Count
Column Degree Centrality
Column Degree Network Centralization
Column Redundancy
Communication
Communication Congruence
Communicative Need
Complete Exclusivity
Complexity
Connectedness
Correlation Distinctiveness
Correlation Expertise
Correlation Resemblance
Correlation Similarity
Density
Diameter
Diffusion
Effective Network Size
Efficiency
Eigenvector Centrality
Exclusivity
Fragmentation
Geodesic Distance
Global Efficiency
Hierarchy
Hub Centrality
In Degree Centrality
In Degree Network Centralization
Information Centrality
Interdependence
Interlockers
Inverse Closeness Centrality
Isolate Count
Knowledge Access Index
Knowledge Actual Workload
Knowledge Congruence
Knowledge Diversity
Knowledge Exclusivity
Knowledge Load
Knowledge Negotiation
Knowledge Omega
Knowledge Potential Workload
Knowledge Redundancy
Knowledge Task Completion
Knowledge Under Supply
Lateral Link Count
Link Count
Local Efficiency
Location Relevance
Minimum Speed
Network Levels
Node Count
Node Levels
Out Degree Centrality
Out Degree Network Centralization
Overall Task Completion
Performance As Accuracy
Personnel Cost
Pooled Link Count
Potential Boundary Spanner
Potential Workload
Radials
Reciprocal Link Count
Relative Cognitive Distinctiveness
Relative Cognitive Expertise
Relative Cognitive Resemblance
Relative Cognitive Similarity
Relative Expertise
Relative Similarity
Resource Access Index
Resource Actual Workload
Resource Congruence
Resource Diversity
Resource Exclusivity
Resource Load
Resource Negotiation
Resource Omega
Resource Potential Workload
Resource Redundancy
Resource Task Completion
Resource Under Supply
Row Breadth
Row Count
Row Degree Centrality
Row Degree Network Centralization
Row Redundancy
Sequential Link Count
Shared Situation Awareness
Simmelian Ties
Skip Link Count
Social Technical Congruence
Span of Control
Spatial Betweenness Centrality
Spatial Closeness Centrality
Spatial Degree Centrality
Spatial Eigenvector Centrality
Strict Knowledge Congruence
Strict Resource Congruence
Strong Component Count
Task Exclusivity
Total Degree Centrality
Total Degree Network Centralization
Transitivity
TriadCount
Upper Boundedness
Watts-Strogatz Clustering Coefficient
Weak Boundary Spanner
Weak Component Count
Weak Component Members

Source: ORA software (Version: 2.2.8.b), and Reminga and Kathleen (2003).

Concluding Remarks:

- SSA plugins developed by employing the different scripting opportunities available in OpenJUMP can successfully build adjusted graphs from a given vector layer and subsequently calculate the geodesic distance and the basic space syntax parameters,
- Plugins also provide users with an option to produce outputs that can be used in the software programs developed for the analysis of social networks.
- As exemplified in OpenJUMP, FOSS4GIS have a great potential for the employment and exploration of graph theoretical analysis through various scripting languages (pure Java, BeanShell and Jython) thanks to their flexible and extensible architecture.

Script for GUI in Jython plugin:

```
f = JFrame("Space Syntax Analysis")
f.setBounds(100,100,210,325)
f.setResizable(0)
tf1 = TextField("3")
CAF = Choice()
CAF.add("available fields")
CAF.setEnabled(0)
IV = JCheckBox("Calculate intelligibility value.", 0)
NT = JCheckBox("Create network data:", 0)
NW = Choice()
NW.add("dl")
NW.add("net")
NW.setEnabled(0)
GM = JCheckBox("Create geodesic matrix.", 0)
SF = JCheckBox("Save file(s) in a different location.", 0)
SF.setEnabled(0)
ino = ButtonGroup()
UII = JRadioButton("Use internal index.", 0)
UAF = JRadioButton("Use a field.", 0)
UII.setEnabled(0)
UAF.setEnabled(0)
ino.add(UII)
ino.add(UAF)
radioPanel = JPanel()
radioPanel.setLayout(GridLayout(2, 1))
radioPanel.add(UII)
radioPanel.add(UAF)
radioPanel.setBorder(BorderFactory.createTitledBorder(BorderFactory.cr
eateEtchedBorder(), "Options for matrix headings"))
radioPanel.setPreferredSize(Dimension(190, 80))
ll = Label("")
ll.setPreferredSize(Dimension(215, 5))
tamam = JButton("OK")
iptal = JButton("Cancel")
f.add(Label("Radius for local:"))
f.add(tf1)
f.add(IV)
f.add(NT)
f.add(NW)
f.add(GM)
f.add(SF)
f.add(radioPanel)
f.add(Label("Choose a field:"))
f.add(CAF)
f.add(ll)
f.add(tamam)
f.add(iptal)
f.setLayout(FlowLayout(0))
f.setAlwaysOnTop(1)
def dmpressed(x):
  dm = NT.getModel().isSelected() or GM.getModel().isSelected()
  SF.setEnabled(dm)
  UII.setEnabled(dm)
  UAF.setEnabled(dm)
  CAF.setEnabled(dm and UAF.getModel().isSelected())
  NW.setEnabled(NT.getModel().isSelected())
def ipressed(x):
  f.dispose()
def ufpressed(x):
  if UAF.getModel().isSelected():
     CAF.setEnabled(1)
     Layers = getSelectedLayers()
     Layer = Layers[0]
     fc = Layer.featureCollectionWrapper
     fs = fc.featureSchema
     CAF.removeAll()
     CAF.add("available fields")
     for atin in range(fs.attributeCount-1):
       CAF.add(fs.getAttributeName(atin+1))
  else:
     CAF.setEnabled(0)
def spaceSyntax(event):
  NT.actionPerformed = dmpressed
  GM.actionPerformed = dmpressed
  UII.actionPerformed = ufpressed
  UAF.actionPerformed = ufpressed
  tamam.actionPerformed = SS
  iptal.actionPerformed = ipressed
  f.show()
```

References:

Beyhan, B., Belge, B. and Zorlu, F. (2010) "Özgür ve Açık Kaynak Kodlu Masaüstü CBS Yazılımları Üzerine: Karşılaştırmalı ve Sistemli bir Değerlendirme (Free and Open Source Desktop GIS Software Programs: A Comparative and Systematic Evaluation)", Harita Dergisi, 143, 45-61, http://www.hgk.msb.gov.tr/dergi/makaleler/143_6.pdf

Beyhan, B. (2010) "Socio-spatial characteristics of labor mobility and innovation inside an industrial cluster: Some reflections from Siteler in Ankara", paper presented at the 50th Anniversary European Congress of the Regional Science Association International, Jönköping, Sweden, 37 pages.

Beyhan, B. (2011) "Developing Space Syntax Tools for Free and Open Source Software for GIS", in Proceedings of the 19th International Conference on Geoinformatics (Geoinformatics 2011), Shanghai, China, 6 pages, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5981111&tag=1

Borgatti, S.P., Everett, M.G. and Freeman, L.C. (2002) Ucinet for Windows: Software for Social Network Analysis. Harvard: Analytic Technologies.

Reminga, J. and Kathleen, C. (2003) Measures for ORA (Organizational Risk Analyzer).

Freeman, L.C. (2004) The Development of Social Network Analysis: A Study in the Sociology of Science, Vancouver: Empirical Press.

Hägerstrand, T. ([1953] 1967) Innovation Diffusion as a Spatial Process, Postscript and translation by Allan Pred, Chicago: The University of Chicago Press. Translated from Innovationsforloppet ur korologisk synpunkt, published in 1953 by C.W.K. Gleerupska, Lund, Sweden.

Haggett, P. and Chorley, R.J. (1969) Network Analysis in Geography, New York: St. Martin's Press.

Hillier, B. and Hanson, J. (2003) The social logic of space, Cambridge: Cambridge University Press.

Jiang, B. and Claramunt, C. (2004) A Structural Approach to the Model Generalization of an Urban Street Network, GeoInformatica, 8(2), 157-171.

Moreno, J.L. (1934) Who Shall Survive? A New Approach to the Problem of Human Interrelations. Washington D.C.: Nervous and Mental Disease Publishing.

Moreno, J.L. (1937) "Sociometry in Relation to Other Social Sciences", Sociometry, 1(1/2): 206-219.

OpenJUMP, Developer Documentation and How To, http://sourceforge.net/apps/mediawiki/jump-pilot/index.php?title=Developer_Documentation_and_HowTo

OpenJUMP, Scripting with BeanShell, http://sourceforge.net/apps/mediawiki/jump-pilot/index.php?title=Scripting_with_BeanShell

OpenJUMP, How to run OpenJUMP in Eclipse, http://sourceforge.net/apps/mediawiki/jump-pilot/index.php?title=How_to_run_OpenJUMP_in_Eclipse

Pajek Wiki, http://pajek.imfm.si/doku.php

Steiniger, S. and Bocher, E. (2009) "An overview on current free and open source desktop GIS developments", International Journal of Geographical Information Science, 23(10), 1345-1370.

Tomko, M., Winter S., Claramunt, C. (2008) Experiential hierarchies of streets, Computers, Environment and Urban Systems 32 (2008) 41–52.

plugins can be downloaded from the following webpage:

http://mekandizim.mersin.edu.tr/

Please refer to this presentation and the following studies if you employ SSA plugins in your studies;

Beyhan, B. (2011) “Developing Space Syntax Tools for Free and Open Source Software for GIS”, in Proceedings of the 19th International Conference on Geoinformatics (Geoinformatics 2011), Shanghai, China.

Beyhan, B. (2012) “A simple installation and user's guide for the plugins and scripts developed to conduct space syntax analysis (SSA) in FOSS4GIS: OpenJUMP, gvSIG, OrbisGIS, Quantum GIS, OpenEv, Thuban, MapWindow GIS, SAGA, and R Project”, http://mekandizim.mersin.edu.tr/.

Beyhan, B. (2012) “Plugins and Scripts Developed to Conduct Space Syntax Analysis in FOSS4GIS: OpenJUMP, gvSIG, OrbisGIS, Quantum GIS, OpenEv, Thuban, MapWindow GIS, SAGA, and R Project”, http://mekandizim.mersin.edu.tr/, forthcoming.