

Tutorial

for setting up GBC calculation on Windows

This tutorial shows how to set up the implementation for calculation of the GBC described in **Rami Puzis et al** "*Fast algorithm for successive computation of group betweenness centrality*".

There are two methods how to access the methods:

- via server
- via libraries

Both methods are described below.

Requirements:

- AlgServer
- Installed Java (tested 1.6 and 1.7)
- Java IDE (e.g. [Netbeans](#) or [Eclipse](#))

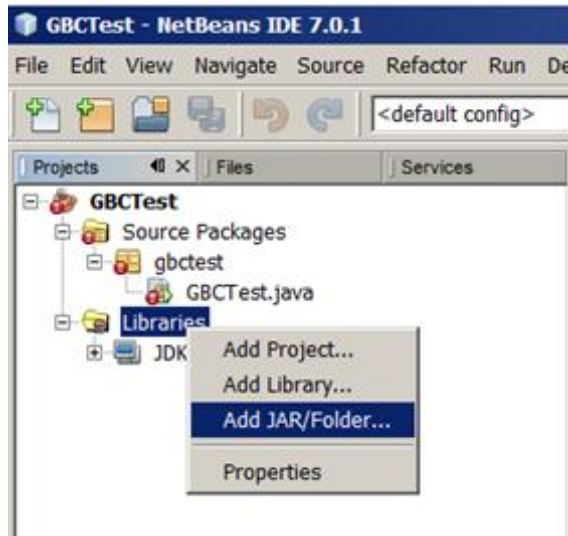
Additionally for method a:

- Any library or implementation for an XMLRPC client (e.g. [Apache xmlrpc](#))

This tutorial uses Apache *xmlrpc* with *Netbeans 7.0.1* (via server) and *Eclipse 3.7.2* (via libraries) on *Windows 7*.

Step 2:

Start your IDE and create a new standard Java project (*Netbeans: Java Application*). First you need to import the libraries provide by *Apache xmlrpc* to implement an xmlrpc client.



Right click on Libraries and choose “Add JAR/Folder...”

A file choosing window pops up. In *xmlrpc* version 3.1.3 you only need to choose the following 3 .jar files:

- *xmlrpc-client-3.1.3.jar*
- *xmlrpc-common-3.1.3.jar*
- *ws-commons-util-1.0.2.jar*

If you want don't worry about a little overhead and to be 100% sure you can also choose all 5 .jar files.

Step 3:

In this step you need to implement an *xmlrpc* client to contact the gbc server. In your main method you need to create the following code:

```
XmlRpcClientConfigImpl config = new XmlRpcClientConfigImpl();
config.setServerURL(new URL("http://127.0.0.1:8080/AlgorithmsServer"));
XmlRpcClient client = new XmlRpcClient();
client.setConfig(config);
```

This piece of code creates a new config for a client, creates the client and combines the config with the client. Be aware to input the right location of the server – Especially if you changed the port in the *serverProps.xml* configuration file earlier. In this example the server is running on localhost. The name should always be *AlgorithmsServer*.

Step 4:

Choose an accurate test file. Attached are two example files: `seltestfile.sel` and `nettestfile.net`. For further information please refer to the *Pajek Manuel* (.net format).

The test files must be inside `AlgServer/data/`.

Step 5:

In the last step the example code is created. First of all you need to be aware of the following handlers:

Handler name	Replaces ...
SE	<code>server/structuralEquivalence/StructuralEquivalenceController</code>
Brandes	<code>server/shortestPathBetweenness/BrandesController</code>
GBC	<code>server/shortestPathBetweenness/GBCController</code>
BCC	<code>server/shortestPathBetweenness/FasterBCCController</code>
Group	<code>server/group/GroupController</code>
MobilityBC	<code>server/mobility/CongestionAwareBCCController</code>
FasterGRBC	<code>server/rbc/FasterGRBCCController</code>
GRBC	<code>server/rbc/GRBCCController</code>
FasterSRBC	<code>server/rbc/FasterSRBCCController</code>
SRBC	<code>server/rbc/SRBCCController</code>
ContributionVRBC	<code>server/rbc/ContributionVRBCCController</code>
StatefullVRBC	<code>server/rbc/StatefullVRBCCController</code>
VRBC	<code>server/rbc/VRBCCController</code>
RationalVRBC	<code>server/rbc/Rational/RationalVRBCCController</code>
RationalStatefullVRBC	<code>server/rbc/Rational/RationalStatefullVRBCCController</code>
RationalContributionVRBC	<code>server/rbc/Rational/RationalContributionVRBCCController</code>
RationalGRBC	<code>server/rbc/Rational/RationalGRBCCController</code>
Closeness	<code>server/closeness/ClosenessController</code>
GCloseness	<code>server/closeness/GroupClosenessController</code>
Degree	<code>server/degree/DegreeController</code>
GDegree	<code>server/degree/GroupDegreeController</code>
RWB	<code>server/randomWalkBetweenness/RWBController</code>
SK	<code>server/saritKraus/SKController</code>
Dfbnb	<code>server/dfbnb/DfbnbController</code>
Network	<code>server/network/NetworkController</code>
TM	<code>server/trafficMatrix/TrafficMatrixController</code>
Execution	<code>server/execution/ExecutionController</code>
Server	<code>server/AlgorithmsServer</code>
TrastBC	<code>server/sato/TrastBCCController</code>
GreedyClustering	<code>server/clustering/GreedyClusteringController</code>
BudgetedGreedyClustering	<code>server/clustering/BudgetedGreedyClusteringController</code>

If you want to use any methods from the packages/classes in the right column you always have to use the left substitute (as a parameter in your *xmlrpc client* to send to the *xmlrpc server*).

First you need to create a network. You can use `.net` for *Pajek* files or `.sel` for files containing simple edge lists in form `<vertexID1><spaces><vertexID2><line end>`. Returned is a network ID.

```
// server.network.NetworkController.importNetwork
Object[] params = new Object[]{"selftestfile.sel", ""};
Integer netID = (Integer) client.execute("Network.importNetwork", params);
System.out.println("NetID is:" + netID);
```

Second you check if the import worked properly by requesting the number of vertices (=nodes). The number of vertices should be exactly how many are defined in the input file.

```
//network.NetworkController.getNumberOfVertices(NetID)
params = new Object[]{netID};
Integer result = (Integer) client.execute("Network.getNumberOfVertices", params);
System.out.println("Vertices:" + result);
```

Third you are creating an algorithm for the given netID:

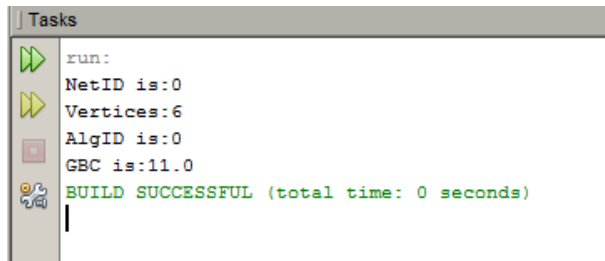
```
//server.shortestPathBetweenness.GBCController.create
params = new Object[]{netID, "", false, false};
Integer algID = (Integer) client.execute("GBC.create", params);
System.out.println("AlgID is:" + algID);
```

Last you calculate the GBC, whereas the second parameter defined the vertices for which you calculate the GBC. Be aware when you import networks other than `.net` the vertices are indexed in ARBITRARY order. You usually don't care about indices as vertices are anonymous in most studies but if you do (or if you are writing tests) make sure to work with *Pajek* format only. Also note that vertices always have zero-based indexing when you communicate with the server.

```
//server.shortestPathBetweenness.GBCController.getGBC
params = new Object[]{algID, new Object[]{0}, new Object[]{}};
Double gbc = (Double) client.execute("GBC.getGBC", params);
System.out.println("GBC is:" + gbc);
```

If the returned GBC is -1.0 something went wrong. Be aware this could also be already an issue when trying to import a network! The error handling is still quite immature.

Output should look like this:



The source code of the example *xmlrpc client* is also attached.

Possible errors:

- Windows UAC, try to start the server as administrator
- Files in the data folder whose names include the name of the you are trying to import e.g. Copy of test.net and test.net
- Different port number in the serverProps.xml and in the your implemented xmlrpc client
- The file you are using must be within the /data/ directory
- Check the Javadoc for any problems cause by wrong method calls

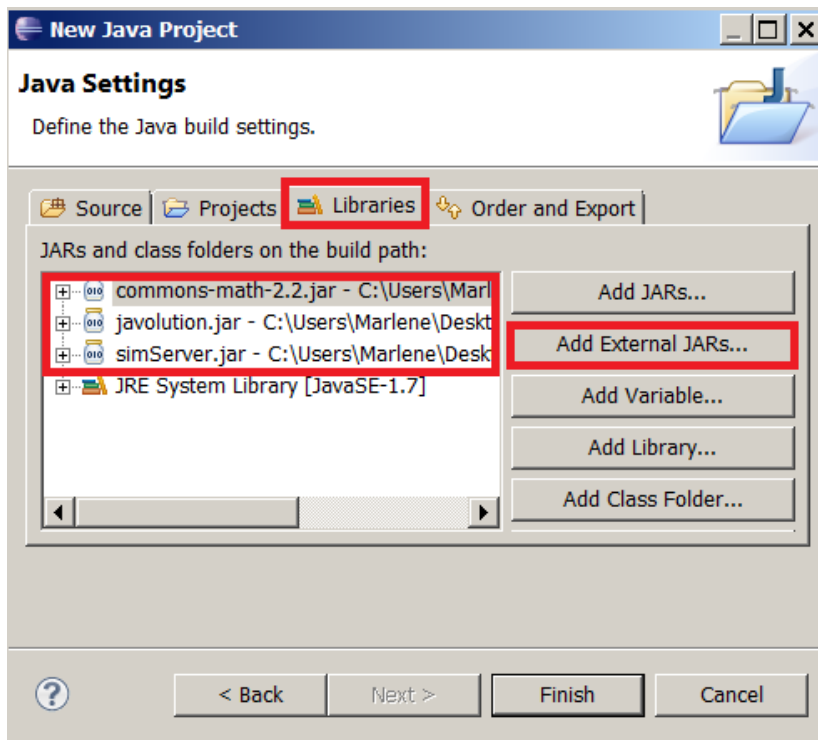
B – via libraries

The second approach does not make use of the *xmlrpc server*. Instead you access the classes directly via included libraries.

Step 1:

Start your IDE and create a new standard Java project (*Eclipse: Java Project*). First you need to import the libraries. In the new project settings choose Libraries and click on the button “Add External JARs...” to add the following .jar files:

- *simServer.jar*
- *lib\javolution.jar*
- *lib\commons\commons-math-2.2.jar*

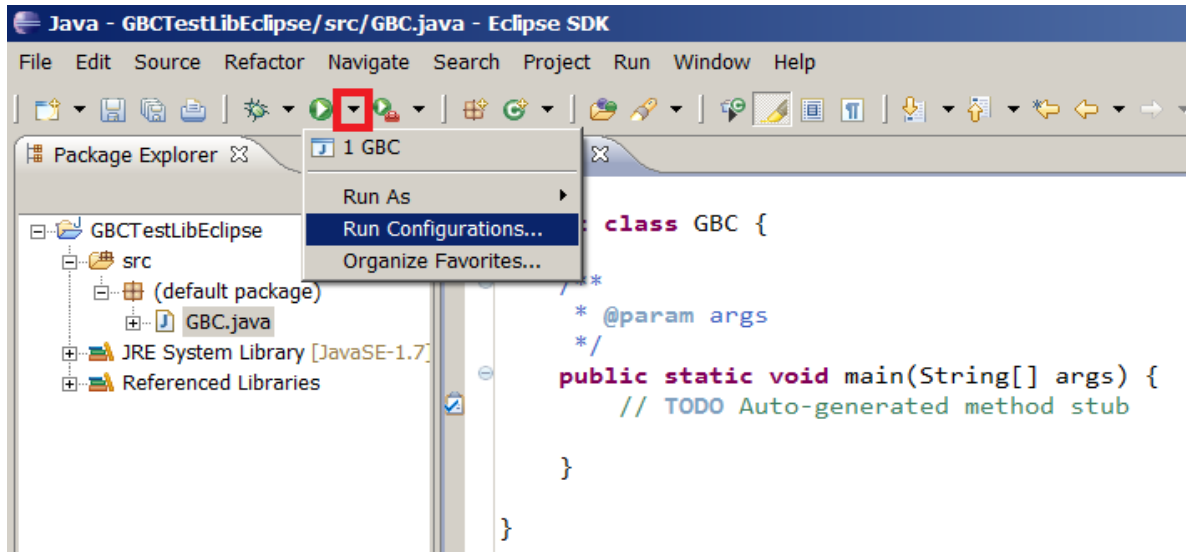


If you are having problems try to add:

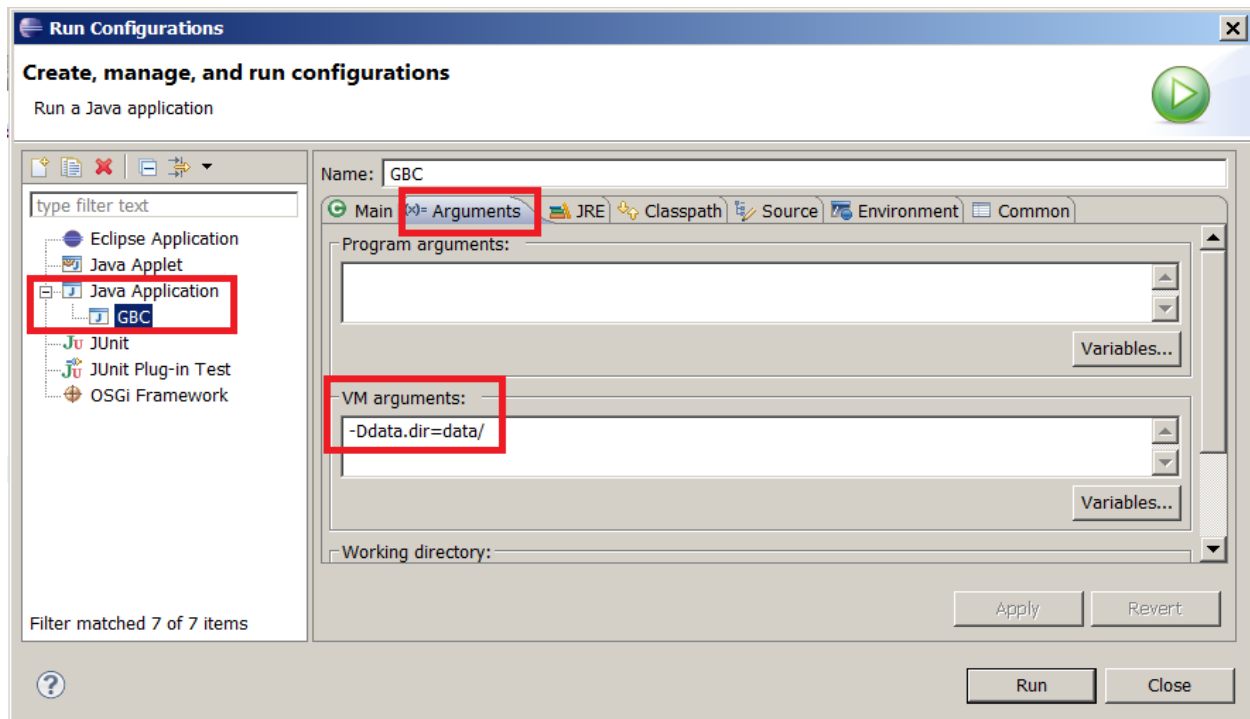
- *lib\Opus5.jar*
- *lib\Jama-1.0.2.jar*
- *lib\jlink\JLink.jar*

Step 2:

Add a Java test class and a corresponding run configuration. After creating a Java class click the little arrow next to the green button on the top panel and choose “Run Configurations...”. A new window should open.



Double click on the left on "Java Application". A new run configuration with the name of your main class should be created. Switch to arguments and add "-Ddata.dir=data/" to the VM arguments.



Step 3:

Choose an accurate test file. Attached are two example files: `seltestfile.sel` and `nettestfile.net`. For further information please refer to the *Pajek Manuel* (.net format).

The test files must be the projects directory in a `/data/` folder. In this example the path would look like this: `...\workspace\GBCTestLibEclipse\data`

whereas `GBCTestLibEclipse` is the name of the project.

Step 4:

You are ready to implement your own code. First you need to create a network. You can use `.net` for *Pajek* files or `.sel` for files containing simple edge lists in form `<vertexID1><spaces><vertexID2><line end>`. Returned is a network ID. We also check that the import worked properly by requesting the number of vertices (=nodes) and edge. The number of vertices should be exactly how many are defined in the input file.

```
server.network.NetworkController nc = new server.network.NetworkController();
int netID = nc.importNetwork("nettestfile.net", "");
System.out.println("NetID: "+netID);
System.out.println("Number of Vertices: "+nc.getNumberOfVertices(netID));
System.out.println("Number of Edges: "+nc.getNumberOfEdges(netID));
```

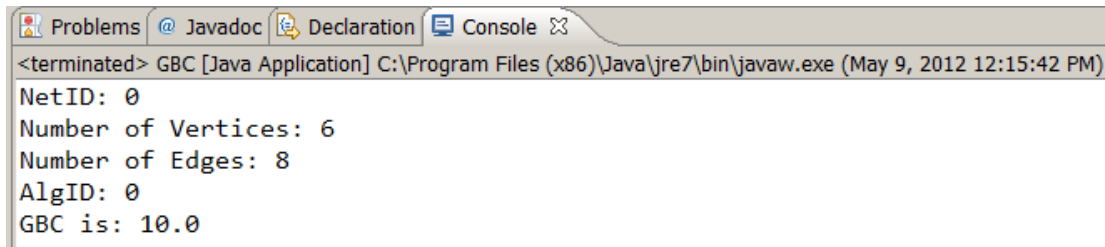
Second you are creating an algorithm for the given `netID`:

```
server.shortestPathBetweenness.GBCController gbcc = new
server.shortestPathBetweenness.GBCController();
int algID = gbcc.create(netID, "", false, false);
System.out.println("AlgID: "+algID);
```

Last you calculate the GBC, whereas the second parameter defined the vertices for which you calculate the GBC. Be aware when you import networks other than `.net` the vertices are indexed in ARBITRARY order. You usually don't care about indices as vertices are anonymous in most studies but if you do (or if you are writing tests) make sure to work with *Pajek* format only. Also note that vertices always have zero-based indexing when you communicate with the server.

```
double gbc = gbcc.getGBC(algID, new Object[]{}{3}, new Object[]{});
System.out.println("GBC is: " +gbc);
```

Output should look like this:



The screenshot shows the Eclipse IDE's Console window. The title bar includes tabs for 'Problems', '@ Javadoc', 'Declaration', and 'Console'. The console text is as follows:

```
<terminated> GBC [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (May 9, 2012 12:15:42 PM)
NetID: 0
Number of Vertices: 6
Number of Edges: 8
AlgID: 0
GBC is: 10.0
```

Again – the Java source files is also attached.

Possible errors:

- Files in the data folder whose names include the name of the you are trying to import e.g. Copy of test.net and test.net
- No /data/ directory in the (Eclipse) projects folder containing the test files
- Not all libraries are included
- Not choosing the right running configuration
- Check the Javadoc for any problems cause by wrong method calls