Getting started with the NRLP tool

Instructions for installing software, setting up Neo4J, running a basic simulation, exploring the database, and making advanced changes to the parameters

I. Install required software

1. Install Neo4J's free community version for individuals: https://neo4j.com/download/

2. Install Python 2.7, including numpy. Anaconda is a good Python platform that comes with numpy: https://www.continuum.io/downloads

If you already have a basic Python installation, install numpy with pip from the command line:

```
> pip install numpy
```

3. Install py2neo, version 2.0.8. Note that the software currently will not run with later versions of py2neo, so be sure to specify the version in installation.

With Anaconda:

```
> conda install -c mutirri py2neo=2.0.8
```

With pip:

```
> pip install py2neo==2.0.8
```

II. Download model files

1. Download the tool NRLP.py from this website and save it to a folder of your choice

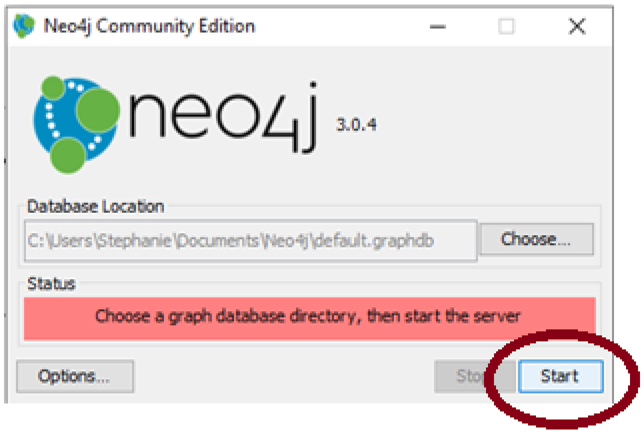
2. Download the three input files rivers.txt, structures.txt and links.txt and save them to the same folder as this script. The model files must be in the same file as this script to run the program.

3. Download the style sheet "belmontstyle.grass" and save it somewhere accessible (the same folder is recommended, but not required).

III. Set up Neo4J

If this is your first time running a simulation only:

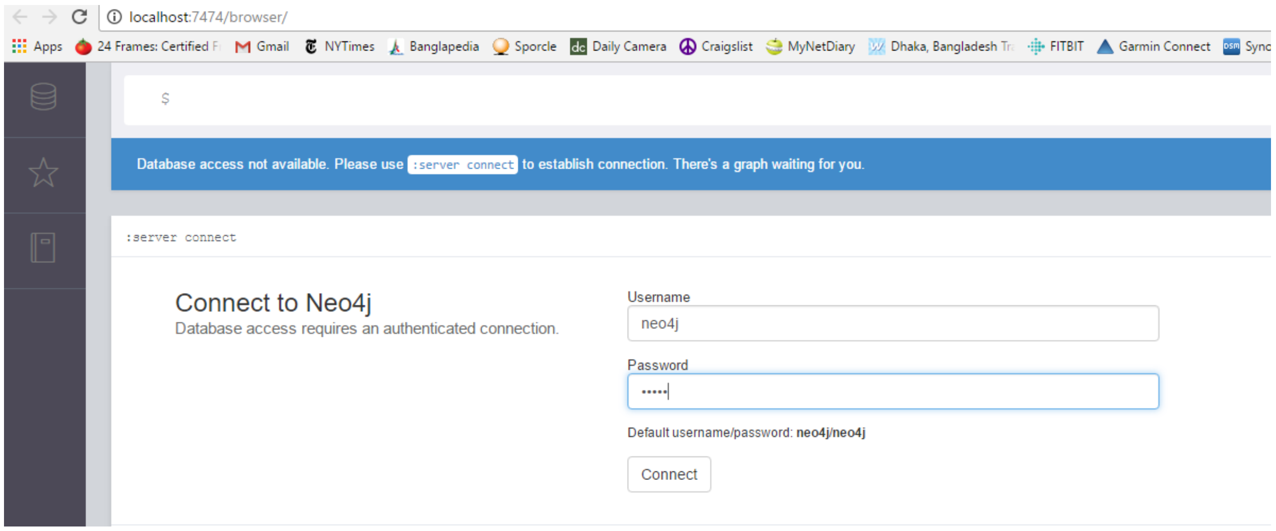
1. Start Neo4J (on Windows, from the Windows Start Menu.) Use the default database location and click "start":



C:\Users\Stephanie\Documents\Neo4j\default.graphdb

Choose a graph database directory, then start the server

2. Click "browse to http://localhost:7474". Your browser should open and Neo4J will request a username and password for the database. Enter the default, neo4j and neo4j.



```
$
```

Database access not available. Please use `:server connect` to establish connection. There's a graph waiting for you.

```
:server connect
```

Connect to Neo4j

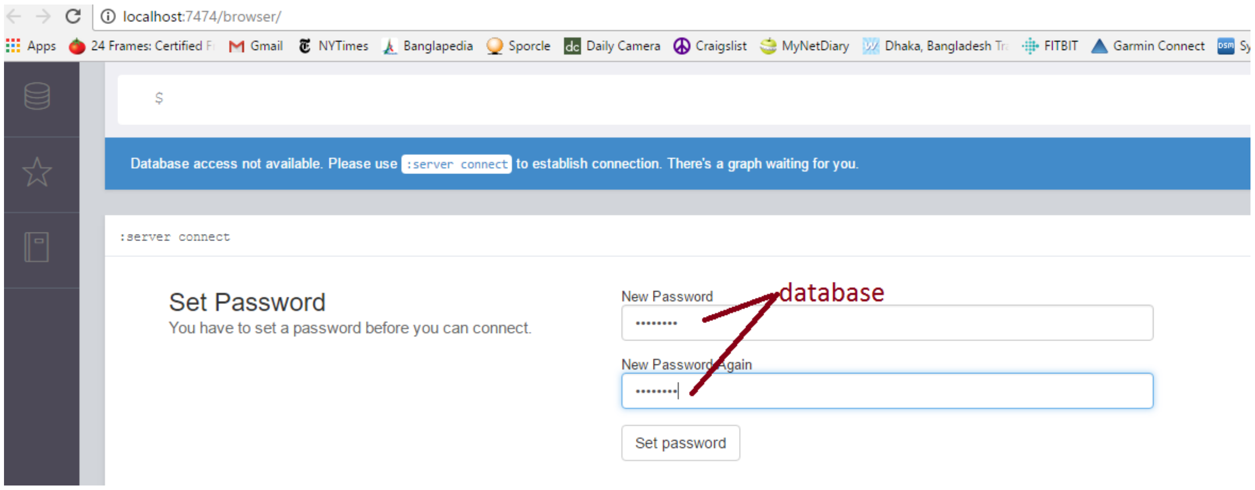
Database access requires an authenticated connection.

Username: neo4j

Password: •••••

Default username/password: **neo4j/neo4j**

3. Change your password to “database” (all lowercase, no quotes). If you choose to use a password besides database, you will need to modify the authentication line in the python script – so don’t chose a different password unless you feel comfortable making that modification. This will only be your password for this model run, not for your Neo4J account in general.



```
$
```

Database access not available. Please use `:server connect` to establish connection. There's a graph waiting for you.

```
:server connect
```

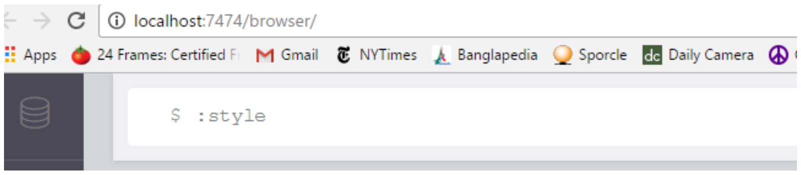
Set Password

You have to set a password before you can connect.

New Password: ••••••••

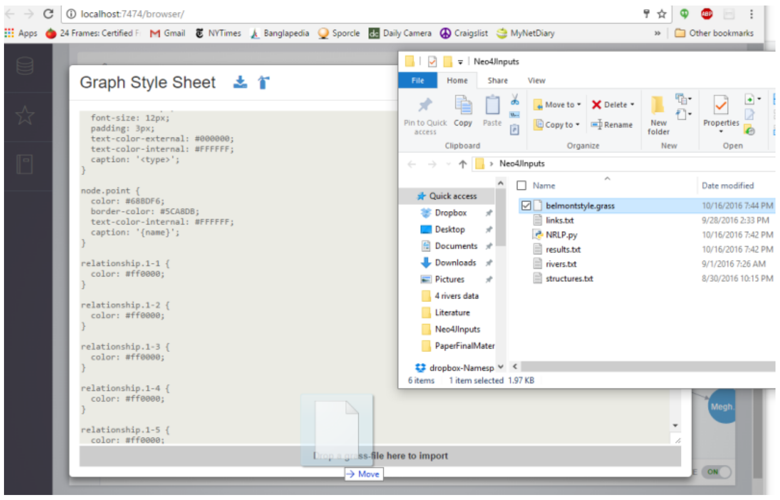
New Password Again: ••••••••

4. Set the style sheet by typing “:style” (you need the colon) into the command window and hitting enter.



```
$ :style
```

A Graph Style Sheet should pop up. Drag the file “belmontstyle.grass” from your input files folder onto the button “Drop a grass file here to import.” Release your mouse when the file is hovering above the button. This step is not required, but is recommended to improve ease of interpreting the model results once the simulation is run. It will restyle all graphs built in this database to have red link canals and blue rivers, with river mouths larger than other nodes.



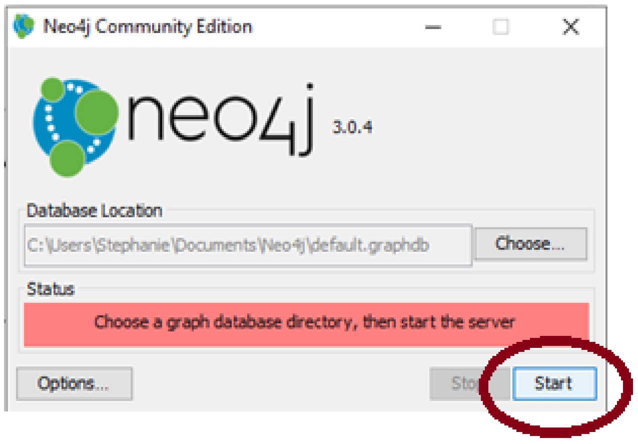
Graph Style Sheet

```
  font-size: 12px;
  padding: 3px;
  text-color-external: #000000;
  text-color-internal: #FFFFFF;
  caption: '<type>';
}

node.point {
  color: #68BDF6;
  border-color: #5CA8DB;
  text-color-internal: #FFFFFF;
  caption: '{name}';
}

relationship.1-1 {
  color: #ff0000;
}

relationship.1-2 {
  color: #ff0000;
}

relationship.1-3 {
  color: #ff0000;
}

relationship.1-4 {
  color: #ff0000;
}

relationship.1-5 {
  color: #ff0000;
```

Name	Date modified
belmontstyle.grass	10/16/2016 7:44 PM
links.txt	9/28/2016 2:33 PM
NRLP.py	10/16/2016 7:42 PM
results.txt	10/16/2016 7:42 PM
rivers.txt	9/1/2016 7:26 AM
structures.txt	8/30/2016 10:15 PM

IV. Run a basic simulation with all link canals set to “off”

1. If Neo4J is not already running, start Neo4J (in Windows, from the start menu) and click “start”:


Neo4j Community Edition
neo4j 3.0.4
Database Location
C:\Users\Stephanie\Documents\Neo4j\default.graphdb
Choose...
Status
Choose a graph database directory, then start the server
Options...
Sto
Start

2. Open a command line (cmd in Windows, or a terminal in Mac/Unix), and navigate to the input files folder:

```
> chdir Desktop\Neo4JInputs
```

(on Windows), or

```
> cd Desktop/Neo4JInputs
```

(on Mac/UNIX).

3. Run the script. We are going to run it with an optional argument that sets all link canals to “off” for this initial test.

```
> python NRLP.py --canals OFF
```

This runs the simulation with all link canals set to “off,” and the software will calculate water discharge changes for 10 points (Hooghly mouth, Farakka Barrage on the Ganga, Hardinge Bridge on the Ganga, Bahadurabad gauging station on the Brahmaputra, the Ganga/Brahmaputra/Meghna mouth at the Bay of Bengal, the Mahanadi mouth, the Godavari mouth, the Krishna mouth, the Penna mouth, and the Kaveri mouth). If everything is set up correctly, you should see a list of rivers printing to the console. When the simulation is done, a new file should appear in the folder: “results.txt.” This file should give expected annual and monthly changes in water discharge for the points specified. All of these values should be “0” on your initial run, because all link canals are set to “off.”

V. Explore the results in the database

Neo4J is a graph database program with its own query language, Cypher. The Cypher commands in this document are only meant get you started exploring the connected graph database of India's rivers and the NRLP link canals. For a complete tutorial of Cypher, see Neo4J's materials on their website.

1. If you haven't already, open your browser and navigate to http://localhost:7474/browser/ . Note that if Neo4J is running, this tab should have opened when you clicked "start."

2. In the command window, type the following:

```
$ MATCH (n) RETURN (n)
```

and hit enter. This will display everything in the database - the full river and link canal system built by the last simulation.

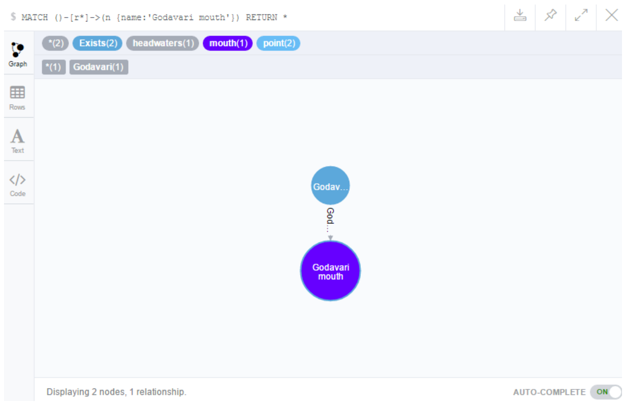
Because the simulation had all link canals set to "off," you should only see rivers. The rivers flow to their mouths; arrows indicate direction of flow. There is no spatial information in this database, so the locations of the rivers on the screen does not correspond to their locations in real life. This is just a backend representing connectivity and flow. Flow directions have been hard-coded into the database. If you'd like, you can click on nodes and drag them around to reposition them on the screen. Note that you can blow up the database window by clicking the small arrow in the top right corner. You will need to shrink it again before entering any new commands.

Click on any of the nodes (light or dark blue circles) to see its monthly change in water discharge. All values should be zero, since all link canals were set to "OFF." Many rivers consist of just two nodes – a headwaters and a mouth – and the rivers are not connected together because there are no link canals in the simulation.

3. Let's explore connectivity by querying the database. We are going to ask which nodes contribute to a certain node of interest – in this case, the Godavari mouth. Do this by typing:

```
$ MATCH ()-[r*]->(n {name:'Godavari mouth'}) RETURN *
```

This is equivalent to asking the database "which nodes are connected to the node called Godavari mouth?" The answer will display on the screen – in this simulation, only the Godavari headwaters node is connected to the river mouth.



```
$ MATCH ()-[r*]->(n {name:'Godavari mouth'}) RETURN *
```

Displaying 2 nodes, 1 relationship.

4. Turn the link canals on and re-build the database. Go back to the command line (cmd in Windows, or terminal in Mac/Unix) and type:

```
> python NRLP.py --canals ON
```

This will build every link canal in the NRLP plan. It may take up to a minute to run the simulation. Changes in water discharge for the ten points of interest will print to the console and to the results.txt file again, but this time there should be values representing the monthly and annual water discharge changes given full implementation of the NRLP. We can query the database now to see what the river system of India looks like with the NRLP in place:

```
$ MATCH (n) RETURN (n)
```

It is highly recommended to expand the result window using the arrow in the top right corner to view the entire system (you will have to shrink the window again to enter new commands). Rivers are still blue, but now link canals have been constructed and appear red in the results window. New dams and barrages appear pink. You can click on any link canal to see its operating specifications. For any node – not just the nodes printed to the results file – you can click to see the monthly and annual changes in water discharge for that point.

5. Now let’s return to the question of connectivity for the Godavari River. Repeat the query from above, asking which nodes contribute to the Godavari river mouth in the all-link-canals scenario:

```
$ MATCH ()-[r*]->(n {name:'Godavari mouth'}) RETURN *
```

You should now see the astounding number of nodes that will contribute a portion of their flow to the Godavari river mouth given full implementation of the NRLP. These nodes represent points along rivers, (all from new watersheds), from which at least some quantum of water could be expected to travel to the Godavari river mouth if all link canals are constructed.

VI. Advanced usage: changing model simulation parameters

In this section, you will learn to:

(a) Build a user-defined subset of link canals,
(b) Print different nodes to the output files,
(c) Change the operating specifications of the dams and link canals, and create new dams and/or link canals, including linking new rivers together.

a. Building a user-defined subset of link canals

To build a user-defined subset of link canals, open the "links.txt" input file. Here are listed all of the link canals in the links database of Higgins et al. (2017). Note that while all link canals are listed, #s 27-29 are not currently implemented in the Neo4J tool because they do not related to the deltas studied in Higgins et al. (2017). Adding functionality for these West Indian links is a topic of future work.

To turn a subset of link canals on, simply set the keywords "ON" and "OFF" in front of each link accordingly. It is not recommended to use find-and-replace to globally change these keywords, as this can change the names of structures (e.g., "Sone Barrage" changes to "SOFFe Barrage" and breaks the link canal in the simulation.

Then, rebuild the database from the command line, with no canals argument this time:

```
> python NRLP.py
```

New results will print to the file, and the Neo4J database at http://localhost:7474/browser/ can be inspected and queried as before.

Caution: while changing the model keywords is simple in practice, it is listed under "advanced usage" because there are currently no fail-safes built in to prevent unrealistic link canal subsets from being built. For example, it is possible to build link canal 1.3 in the model without building sections 1.1, 1.2, 1.4 or 1.5. In reality, this would never occur. Similarly, it is possible to build the Subarnarekha-Mahanadi link canal (#15) without building the compensating link canal from Farakka Barrage to the Subarnarekha (#14). This scenario would result in more water being pulled from the Subarnarekha than exists in the river. Users should therefore take care to build only realistic groups of link canals if the data are to be used for scientific purposes.

b. Printing different nodes to the output file

Any node in the database can have its annual and monthly water discharge changes printed to the results file. To print different nodes to the output file, open the python script NRLP.py and change the node names in line #23. Node names must be formatted exactly as they are written on the nodes displayed in the Neo4J workplace. Exact node name formatting can also be found in the "structures.txt" and "rivers.txt" files.

c. Changing the operating specifications of the dams and link canals.

Modifying or adding a new a link canal:

To change the operating specifications of the link canals, modify the numbers in the "links.txt file. Parameters and units are given in the header rows.

As an example, we will modify link canal #10. In the default files downloaded from the website, link canal #10 is:

ON;10;Daudhan Dam;Barwa Sagar Reservoir;1074;68;366;49;591;.181,.107,.004,.004,.004,.006,.076,.038,.067,.108,.214,.191;;T

Explanation of parameters:

ON: The link canal is turned on in the simulation
10: The link canal's ID number from Higgins et al. (2017)
Daudhan Dam: The link canal will offtake water from the Daudhan Dam on the Ken River. Operating specifications for the 'Daudhan Dam' structure can be found in "structures.txt'
Barwa Sagar Reservoir: the link canal will outfall into Barwa Sagar Reservoir on the Betwa River.
1074: The canal is set to remove 1074 MCM (10^6 cubic meters of water) from the Ken River per year
68: 68 MCM per year will be lost to evaporation
366: 366 MCM per year will be lost to enroute irrigation
49: 49 MCM per year will be lost to enroute domestic and industrial usage
591: 591 MCM per year will outfall into the Barwa Sagar Reservoir. Outfall should equal offtake minus losses, however, note that there is currently no fail-safe to ensure that the user has set the parameters this way. Therefore, it is theoretically possible to direct the model to outfall more water than is taken from the offtake river.
.181,.107,.004,.004,.004,.006,.076,.038,.067,.108,.214,.191: These 12 numbers denote assumed monthly operating specifications. The values must total to 1, however, note that there is currently no fail-safe to check and make sure that values equal to 1. The values together reflect the proportion of the 1074 MCM that will be transferred in each month Jan-Dec.
None: The second to last value, "additional required structures," is empty in this scenario. In other link canals, it lists additional structures such as re-regulating dams that the database should build in conjunction with the main offtake and outfall structures or points.
T: "T" means that this canal operates as a direct transfer – that is, water is directly transferred according to the specified proportions each month. Most link canals are set to operate this way. The other option for the parameter, "D," indicates a "dam" scenario, in which water is stored during the monsoon season and released during the dry season. In that scenario, the model handles the transfer as follows: the offtake (dammed) river sees a reduction of water according to the inverse of the specified proportions. In other words, water is reduced in the offtake river when the proportion is "0" for the month. This represents the water being stored in the dam, but not delivered to the outfall river. The outfall river sees the outfall as specified by the proportions.

To change the operating specifications for link canal #10, simply change any of its parameters or values. For example, to connect the link canal to Farakka Barrage instead of Barwa Sagar Reservoir, change "Barwa Sagar Reservoir" to "Farakka Barrage." To double the annual offtake of water, change "1074" to "2148." To simulate a hotter climate by increasing evaporation loss, change "366" to a higher number. Change the 12 proportion values to change how the water is moved throughout the year.

Note that you can also build a new link canal simply by adding another line to the links file. Fill in the parameters and values as above. If the new link canal does not utilize structures (dams, barrages, or offtake and outfall points) that already exist in the database, and/or if the new link canal connects rivers that do not already exists in the database, these will have to be added to the structures.txt and/or rivers.txt file. Adding structures and rivers is very straightforward and is explained in the next section.

Adding a new river

In the rivers.txt file, each row has two values. The first value of each row is a river name. The second value is a list of all nodes through which the river passes. The first node typically the river headwaters, "Rivername headwaters." Next are all confluences through which the river passes, named as "incoming-parent." For example, the confluence where the Kosi enters the Ganga is named "Kosi-Ganga." The last confluence is the river mouth, typically called "Rivername mouth." If the river does not end at a mouth, it ends at its confluence with the parent river (e.g., the Kosi ends at "Kosi-Ganga.")

To add a new river to the file, simple add a new row to rivers.txt. Fill in the headwater node, confluences, and mouth node accordingly. For example, to add the Netravati to the model, a line would be added to rivers.txt:

Netravati;Netravati headwaters,Netravati mouth

To add the Gomti River, a tributary of the Ganga that was not included in the model, a line would be added to rivers.txt:

Gomti;Gomti headwaters,Gomti-Ganga

In this case, the Ganga line would also have to be modified to include its new confluence with the Gomti:

*Ganga;Ganga headwaters,Yamuna-Ganga,**Gomti-Ganga,**Ghaghara-Ganga,Son-Ganga,Gandak-Ganga,STG-Ganga,Kosi-Ganga,Mahananda-Ganga,Farakka Barrage,Hardinge Bridge,Brahmaputra-Ganga,Meghna-Ganga,Ganga mouth*

The confluence must be added in the correct location relative to the existing confluences in the database (confluences are listed for each river from upstream to downstream).

Modifying or adding a new structure (dam/barrage/offtake/outfall point):

To modify a dam, open the structures.txt file. Parameters are given in the header rows. As an example, here is the line for the Chisapani Dam:

Chisapani Dam,Ghaghara,Ghaghara headwaters,Sarda-Ghaghara,12000,Proposed,Dam,

Explanation of parameters:

Chisapani Dam: The structure name
Ghaghara: The river on which the structure will be built
Ghaghara headwaters: The node upstream of the structure
Sarda-Ghaghara: The node downstream of the structure
12000: The dead storage of the structure, if it is a dam (this water will be permanently removed from the river in the simulation)
Proposed: The structure is proposed as part of the NRLP. The other option, "exists," is for existing structures that are proposed to be utilized as part of the NRLP (for example, Farakka Barrage).
Dam: The structure type. Options are Dam, Barrage, Offtake, or Outfall. Offtake is for an offtake location without a specified new dam or barrage, and Outfall is for an outfall location directly into a river.

To modify a dam's dead storage, for example, simply change that value in structures.txt. To build a new structure, add a row to the structures.txt file. To find the closest upstream node, build the model with all canals on:

```
> python NRLP.py --canals ON
```

Examine the database in Neo4J to find the two nodes between which you wish to put your new structure.

VII. Advanced usage (b): Putting it all together to add a new (fake) link canal

To put all of these examples together, we will build a new (fake) link canal into the model to transport water from the Gomti to the Son. The Gomti is a tributary of the Ganga that was not originally included in the model.

1. Open rivers.txt. Add the Gomti to the model by adding the line:

Gomti;Gomti headwaters,Gomti-Ganga

to the bottom of rivers.txt. Next, modify the Ganga line to add the Gomti-Ganga confluence:

*Ganga;Ganga headwaters,Yamuna-Ganga,**Gomti-Ganga,**Ghaghara-Ganga,Son-Ganga,Gandak-Ganga,STG-Ganga,Kosi-Ganga,Mahananda-Ganga,Farakka Barrage,Hardinge Bridge,Brahmaputra-Ganga,Meghna-Ganga,Ganga mouth*

Save rivers.txt and close.

2. Open structures.txt. We will add a new dam on the Gomti river, with a dead storage of 3000 MCM. Because this is the only structure on the Gomti river, we know it will fall between the nodes *Gomti headwaters* and *Gomti-Ganga.* Add the dam by adding the line to the end of structures.txt:

Gomti High Dam,Gomti,Gomti headwaters,Gomti-Ganga,3000,Proposed,Dam,

We also need to add an outfall location on the Son river for our new link canal. We will put it far upstream of the Son, between the headwaters and the proposed Kadwan Dam. Dead storage is 0 because it is a simple outfall point. Add a second line to structures.txt:

Gomti-Son Outfall,Son,Son headwaters,Kadwan Dam,0,Proposed,Outfall

Save structures.txt and close.

3. Open links.txt. We will add our link canal and specify that it operates as a "Dam," so that water is stored from July-October and delivered to the Son evenly during the rest of the months. 2000 MCM will be transferred per year, with 100 transmission (evaporation) loss, 500 enroute irrigation use, and 30 enroute domestic and industrial use. 13700 will therefore be delivered to the Son per year.

To represent this link, add a line to links.txt:

ON;30;Gomti High Dam;Gomti-Son Outfall;2000;100;500;30;1370;.125,.125,.125,.125,.125,.125,0,0,0,0,.125,.125;;D

Save links.txt and close

4. Run the model to rebuild the database

```
> python NRLP.py
```

5. Navigate to http://localhost:7474 in your browser to view the Neo4J database.

6. Enter the following command to view the Ganga basin with the new canal:

```
$ MATCH ()-[r*]->(n {name:'Farakka Barrage'}) RETURN *
```

Your new link canal should be visible with ID # 30 in the database. The results.txt file should also show that Ganges river discharge is reduced by 630 MCM per year, as we specified (2000 is removed from the Gomti, but 1370 is returned to the Son and transported to the Ganga before it reaches Farakka Barrage). Monthly discharge values in cubic meters per second are also slightly altered.