3Models Framework: An application to support evolution of computational biological models

By

Chooi Guan Lim

June 21 2006

version 1.0

Contents

1. Program Overview

The 3Models framework allows an artificial genome (AG) model to be mapped (or transformed) onto a network which can then be mapped onto a lineage tree, and subsequently, evolved over time. Changes can therefore be traced at each level originating from the AG level. In this framework, the AG model is based on Reil's work and the network model is based on a random boolean network. The framework comes with an application called 3Models.jar. The main class of the application resides in *FrameworkApplication.java* and is part of the 3modelsframework.application package. This manual talks about this particular application and how it works. An assumption of biological models is required. The application is a simple prototype which can be used as a reference to develop a more visually pleasing application. Because it is only a prototype, there may be minor issues when incorrect input is provided however this should not be a cause for concern. This application requires that Java 1.4.2 or above to be installed within the system.

2. Walkthrough

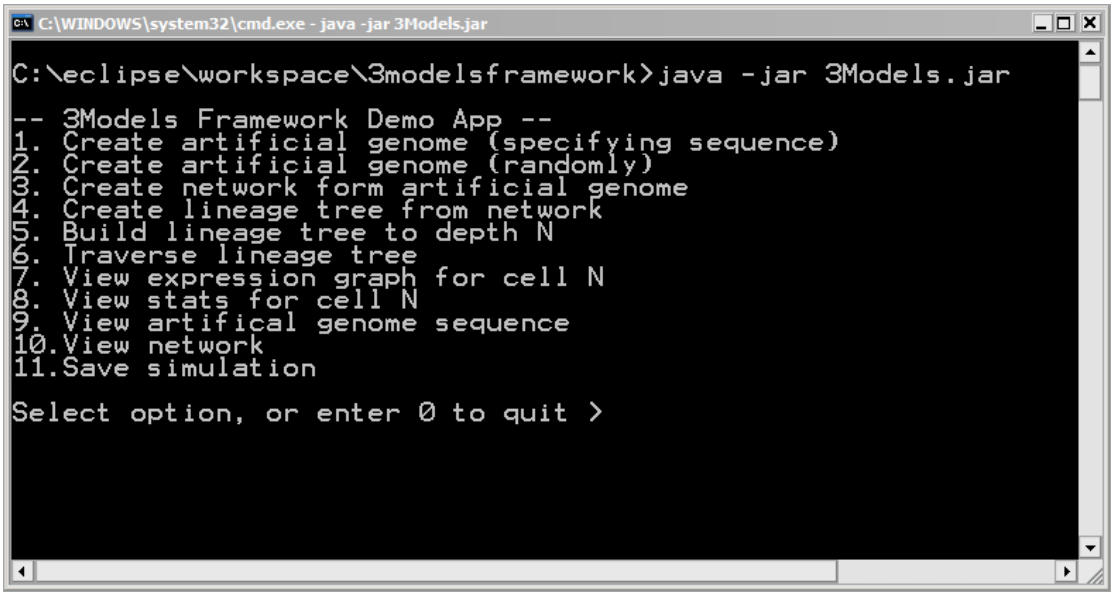
This section provides a quick walkthrough of the application. After following this walkthrough, you will be familiar with how the application works.

2.1. Interface

The application is a simple command line based interface. To run the application simply type

```
java –jar 3Models.jar
```

The following menu should appear upon execution.

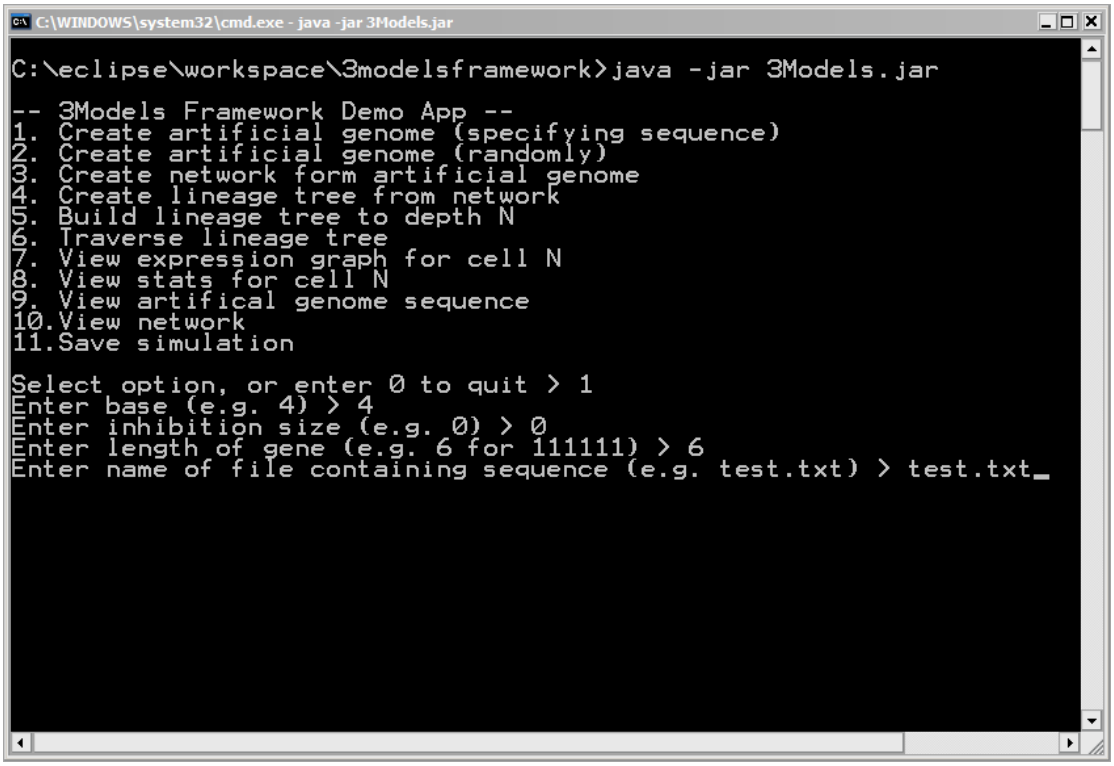


You can see that there are 11 menu options available, most of which are self explanatory. The next section will explore these options in further depth.

2.2. Running a simulation

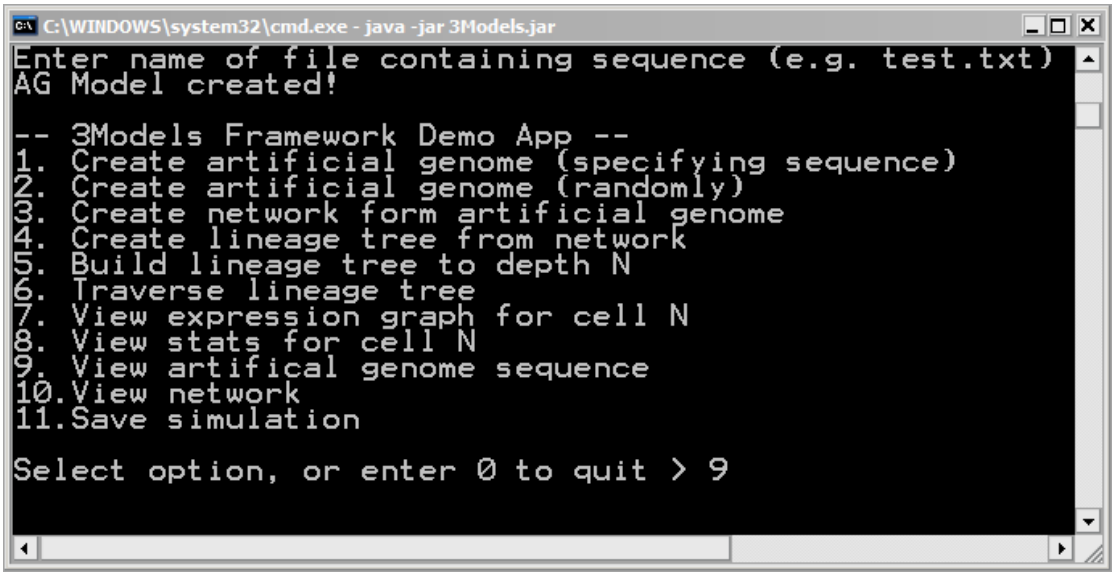
To start a simulation, select option 1 or 2 from the menu first. For this walkthrough, the sequence will be specified. This sequence can be found in the same directory as the application in a file named *test.txt*. The next screenshot shows option 1 being selected. In the software, an artificial genome must be created first, followed by the network model, and then the lineage tree model

2.2.1. Create an AG model

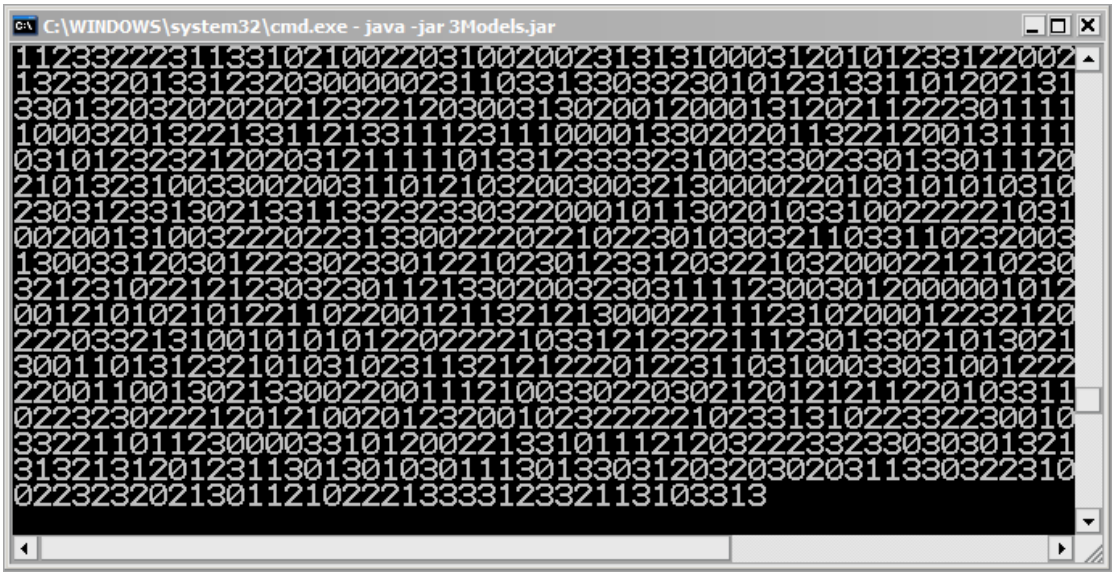


The AG base that is used is a set of digits from 0 to 3, so a base of 4. The network that we are creating has no inhibitions, so we set this to 0. If we use a value of 1, a random number will be picked between 0 and 3 and if the last digit of the gene contains this number, then the gene will be inhibited. Likewise if we use a value of 2, then 2 random numbers will be used. The random seed for a specified sequence is fixed at 0, however if you want a random sequence to be generated, this seed can be specified. The final option is to specify the length of the gene, in our sequence we want this to be a length of 6. Finally we specify the name of the file which contains the sequence. This file contains the sequence that we want to load in our simulation.

Upon hitting the enter key, the sequence is loaded and the AG model created. This is shown in the next screen.



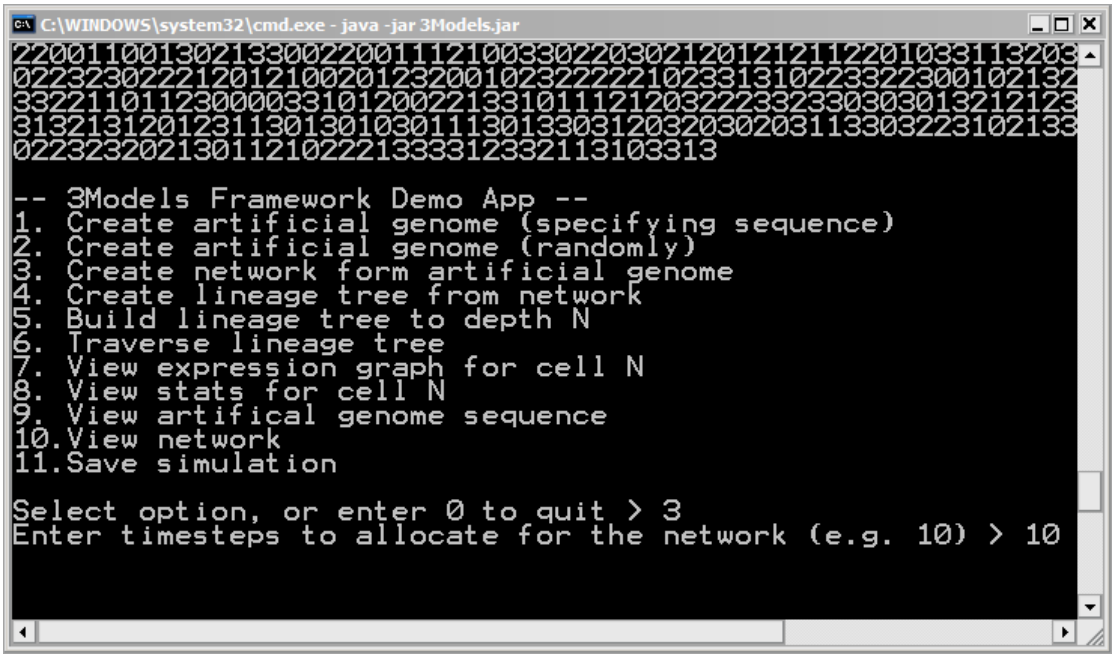
The creation of the AG model can be verified by selecting option 9. Part of the sequence is shown in the next screen.



Once the AG model has been created, the next step would be to create the network model from the AG model. This is discussed in the next section.

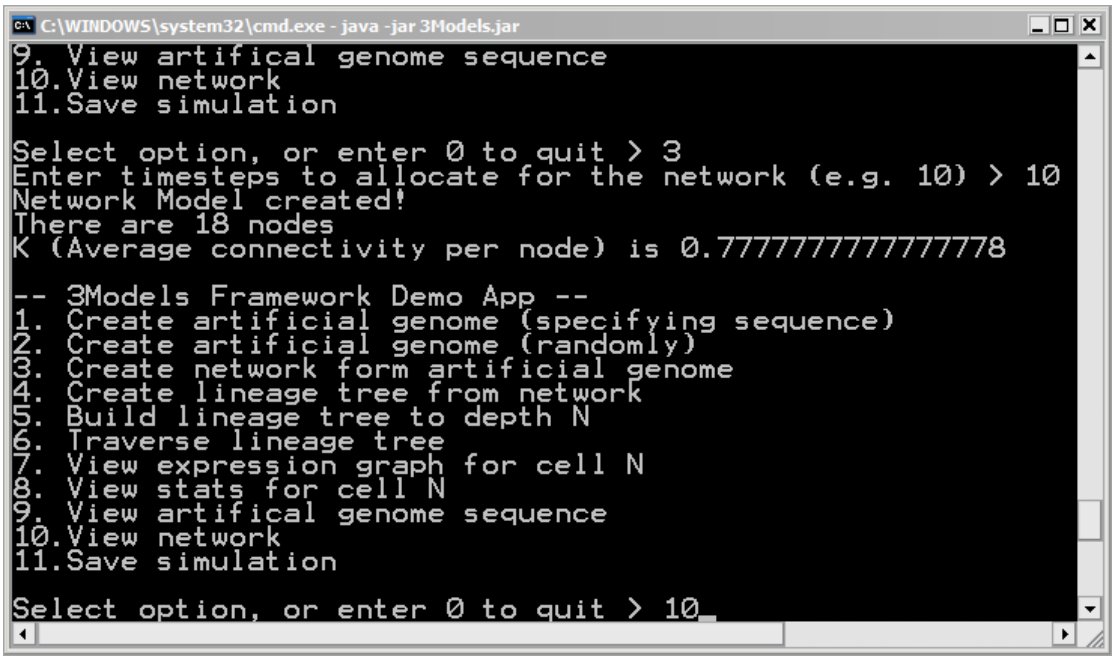
2.2.2. Creating the network model

The AG model must be created first prior to creation of the network model. The network model is created by selecting option 3 from the menu. This is shown in the next screen.

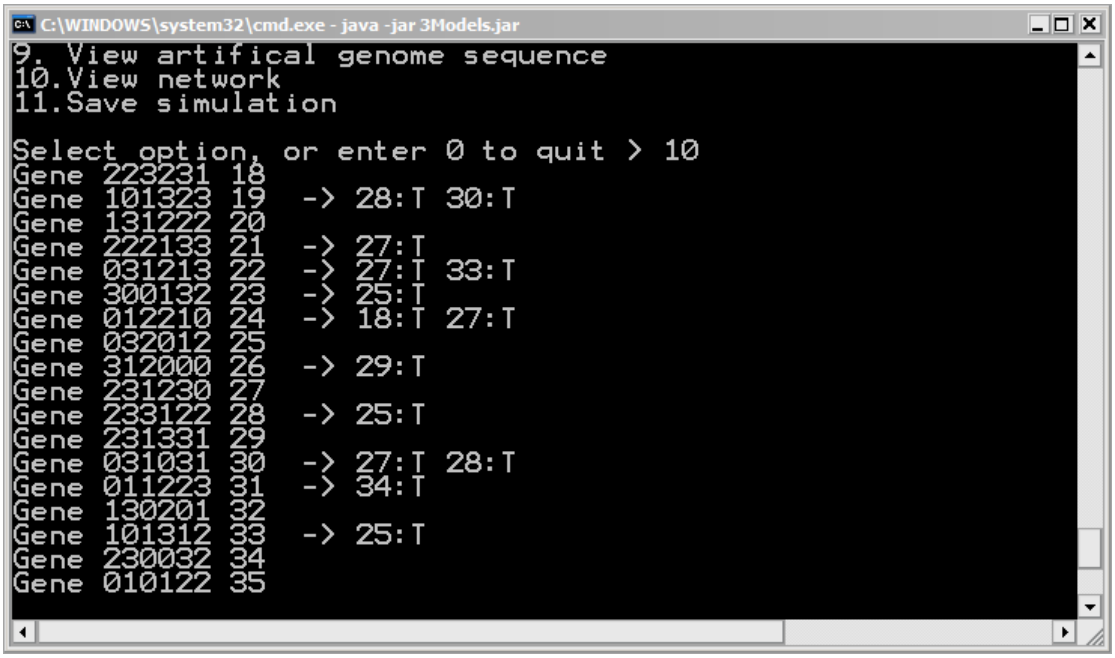


Timesteps has to be allocated for the network in order for the gene expression graph to be created. For our AG model, a timestep of 10 is more than enough. This number is up to you to decide.

Once the timesteps parameter is entered, the network is created. For the particular AG model, 18 nodes were created and there is an average connectivity of 0.78 for each node.



The network can be viewed by selecting option 10. The results are shown in the in the next screen.

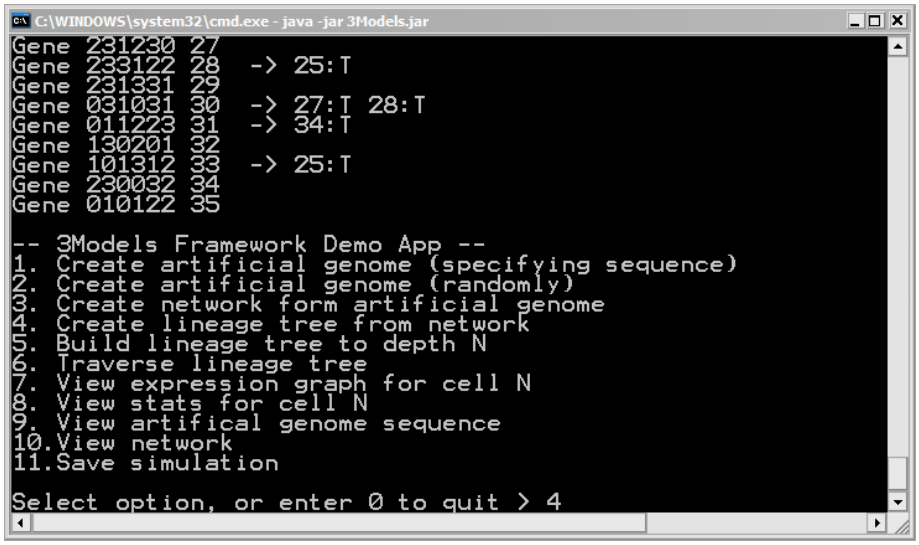


The full listing of the 18 network nodes is shown here. The output is described as follows. Gene “223231” has an id of 18, and does not regulate any other genes. Gene “101323” has an id of 19, and regulates (i.e. the - > arrow) the gene with id 28 (which is gene “233122”) and the gene with id 30 (which is gene “031031”). The links from gene 19 regulating genes with id 28 and 30 are set to a status of true, which means they are active and not inhibited (remember earlier that we specified inhibition to be 0). If this was changed to “28:F 30:F”, it would mean that the links are inhibitory links. This same explanation applies to the rest of the genes. Altogether from the above, there are 4 genes which regulate 2 genes each, 6 genes which regulate 1 gene each, and the 8 genes which do not regulate other genes.

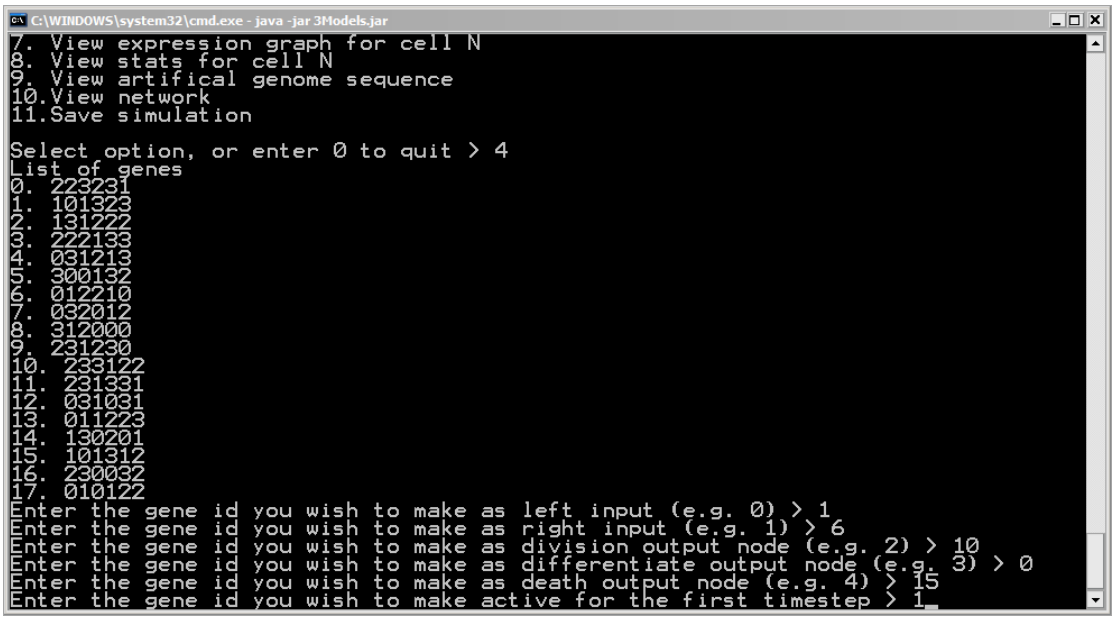
The lineage tree can now be constructed. This is discussed in the next section.

2.2.3. Creating the lineage tree model

The network model must be created first prior to creation of the lineage tree model. The network model is created by selecting option 4 from the menu. This is shown in the next screen.



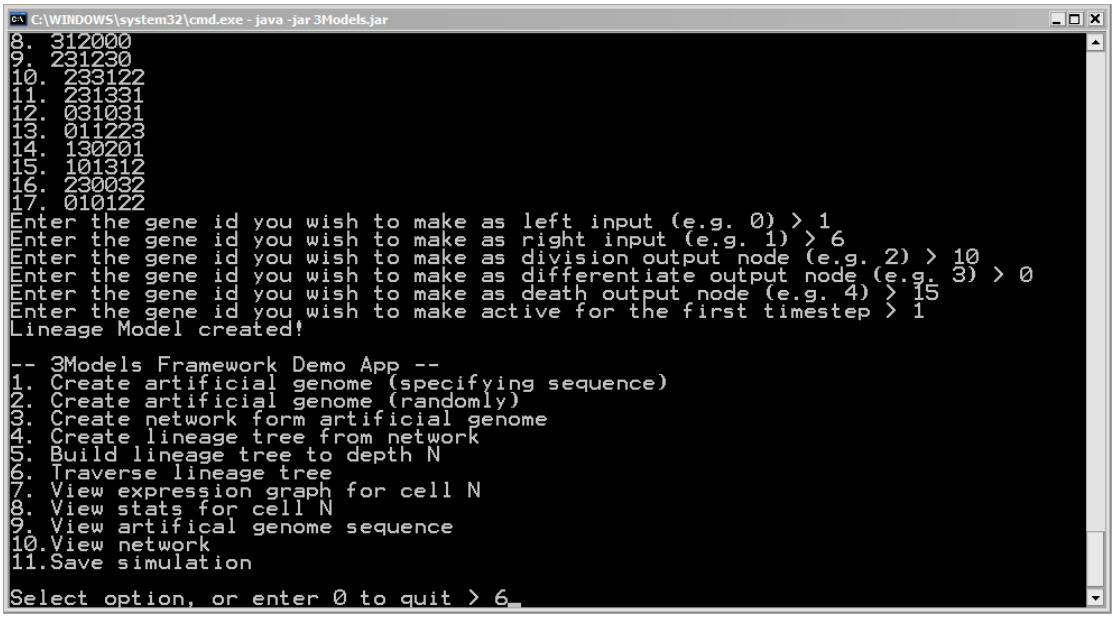
Once the enter key is pressed, the following screen will appear.



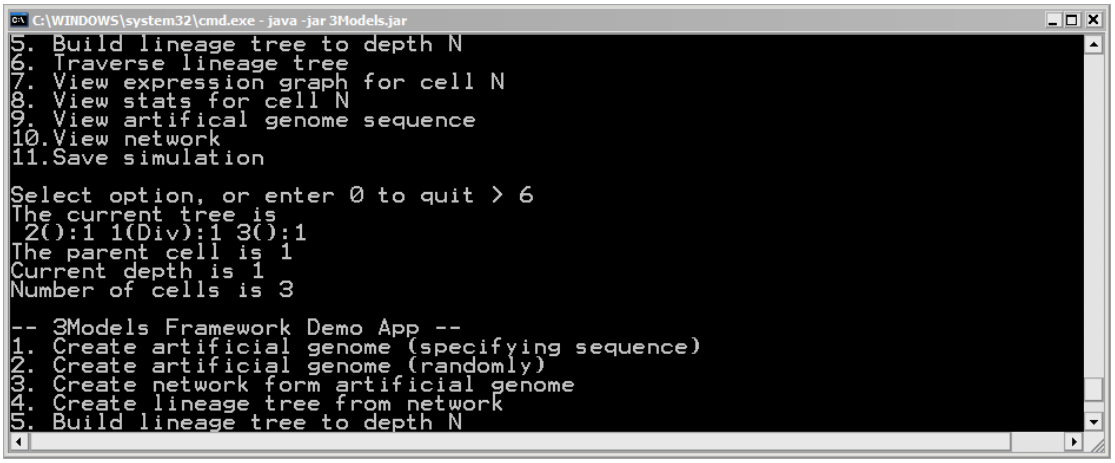
In order to create a lineage tree, you have to define a gene (or node in the network model) to be a left input, and another gene to be a right input. The gene that serves as a left input is turned on (or made active) for the left child of a parent node when it divides or dies, and likewise the gene that serves as a right input is turned on for the right child of a parent node when it divides or dies. In addition, you have to define a gene that causes division, a gene that causes differentiation and a gene that causes death. From the above screen, we are setting gene 1 which is “101323” to be a left input gene, gene 6 which is “0122210” to be a right input gene, gene 10 which is “233122” to be a gene that causes division, gene 0 which is “223231” to be a gene that causes differentiation and gene 15 which is “101312” to be a gene that causes death. Finally, gene 1 is set to be the node that is turned on for the first timestep.

In the default model, if the node that controls division is active before cells that cause differentiation and death, then the cell will divide. However if this node is not active, then the next node that will be inspected will be the node that controls differentiation. Finally if the node that causes differentiation is not active, then the node that causes death will be examined. If neither of these are active, then the cell remains a precursor cell.

The next screen shows that the lineage tree being created.

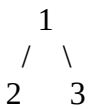


The current lineage tree can be traversed using option 6. This is an inorder traversal of the lineage tree. The output of this traversal is shown in the next screen.



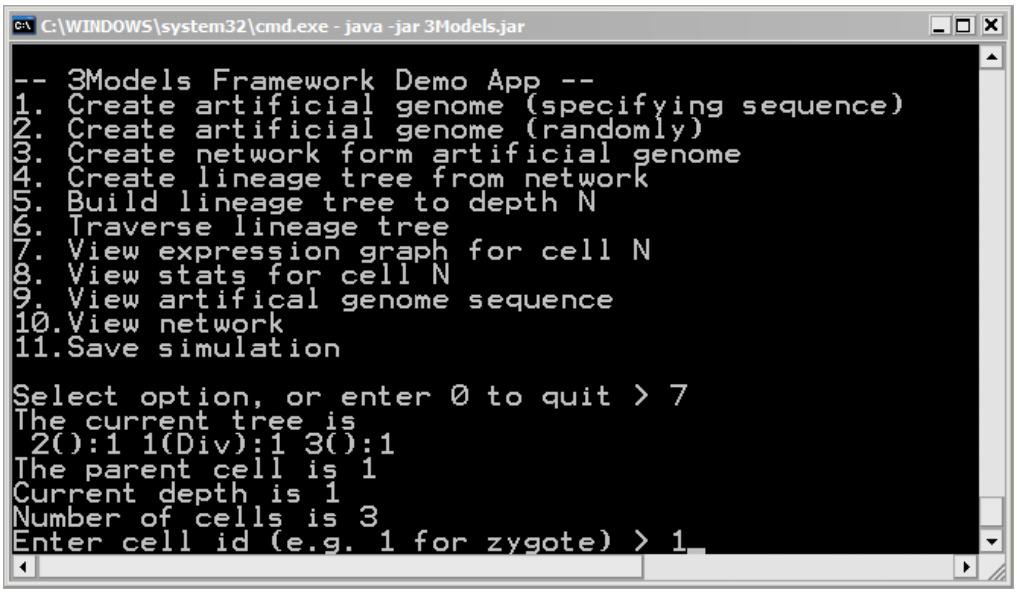
The output shown is 2():1 1(Div): 3():1
This means that 2 is a precursor cell because there is nothing in between the brackets. A (D) means the cell has differentiated. A (Div) means the cell has divided and an (X) means the cell has died. The digit to the right of the colon is the parent. Therefore 2 is a precursor cell whose parent has an id of 1. Cell 1 is the zygote as the parent is the same as itself, i.e. 1. Cell 3 is also a precursor cell and its parent is cell 1.

If this were shown in a tree it would appear as follows.

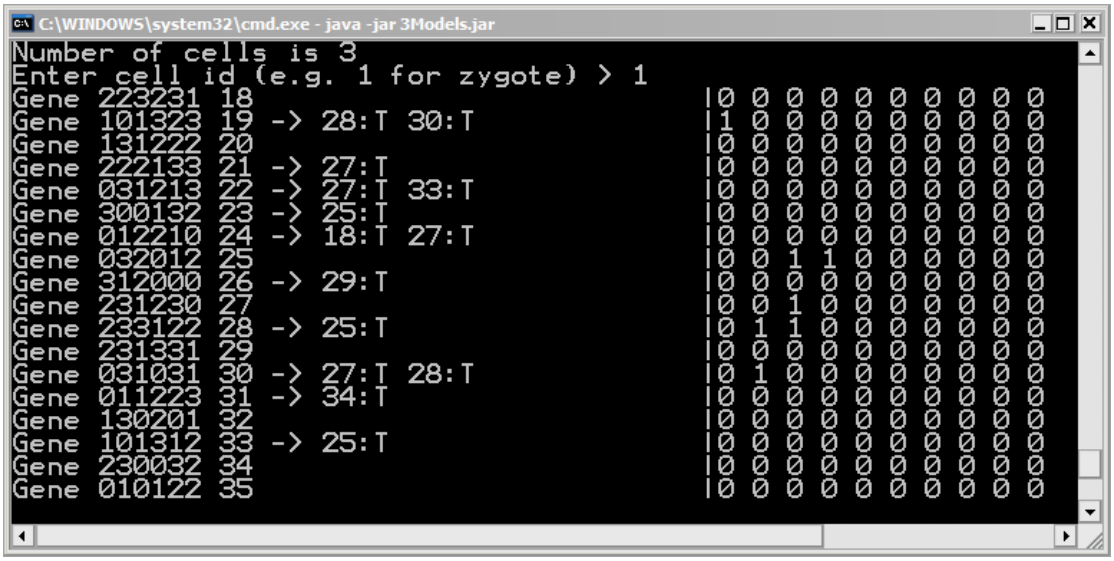


The current depth of the tree is also shown, which is 1 and the total number of cells which is 3.

The expression graph for cells can also be viewed. To view the gene expression graph for the zygote, i.e. cell 1, the option to select would be 7. This is shown in the next screen.

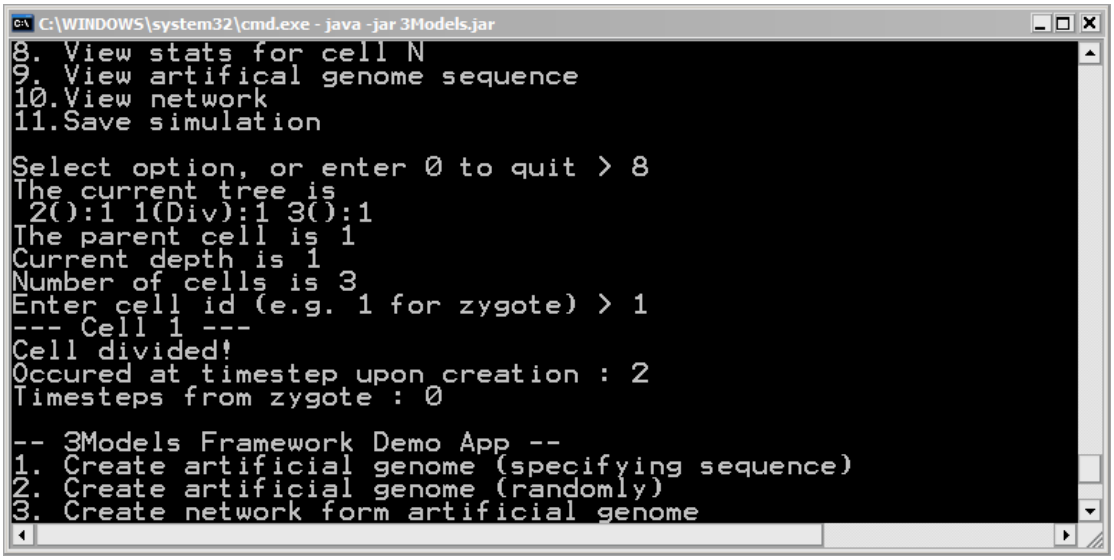


The tree is traversed inorder and displayed along with the details mentioned previously. Once the enter key is pressed, the following screen is displayed for cell 1.



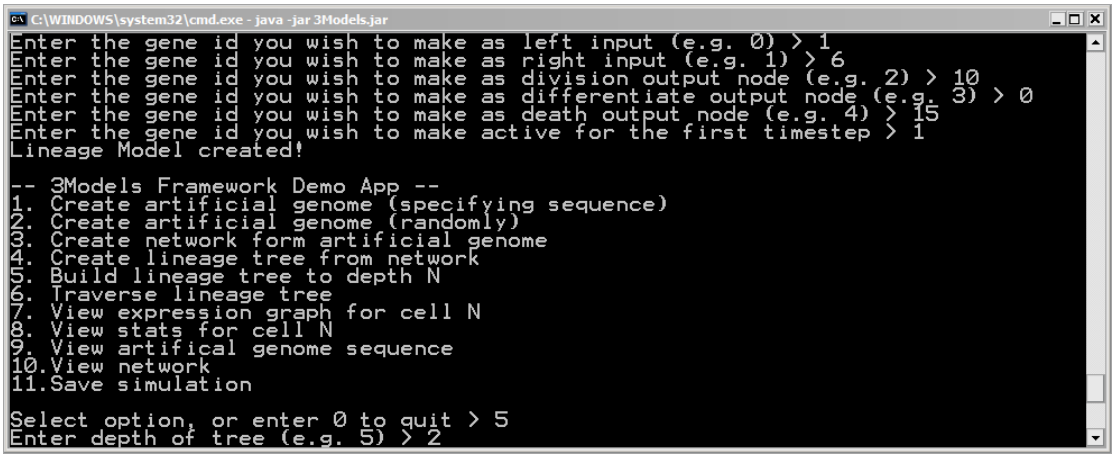
The above screen shows the gene expression graph for the zygote. At timestep 1 it can be seen that gene 101323 was turned on (i.e. a 1). At the next timestep, timestep 2, gene 233122 (i.e. id 28) was active which caused the zygote to divide. This gene was earlier set to be the gene that causes division when active.

Further information about the cell can be found in option 8 which is shown in the next screen.

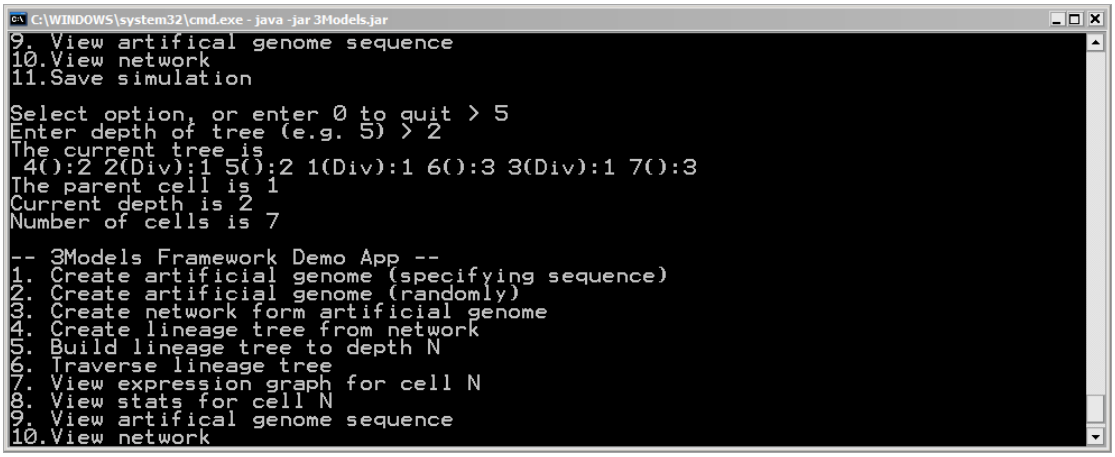


The tree is traversed inorder and displayed along with the details mentioned previously. The information for cell 1 shows that the cell has divided at timestep 2. The total timesteps from the zygote is 0, since cell 1 in this case is the zygote itself.

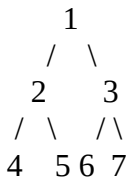
The lineage tree can be built up to a certain depth or until all cells have either differentiated or died, whichever comes first. This option is available as option 5, as shown in the next screenshot.



The depth specified is 2, which results in 7 cells being created in the model. This is shown in the next screen.

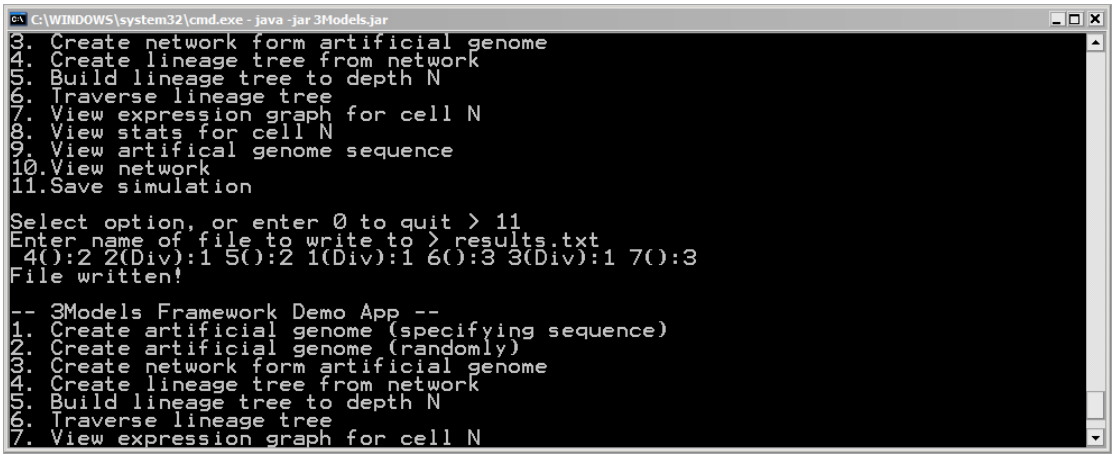


The tree looks like the following

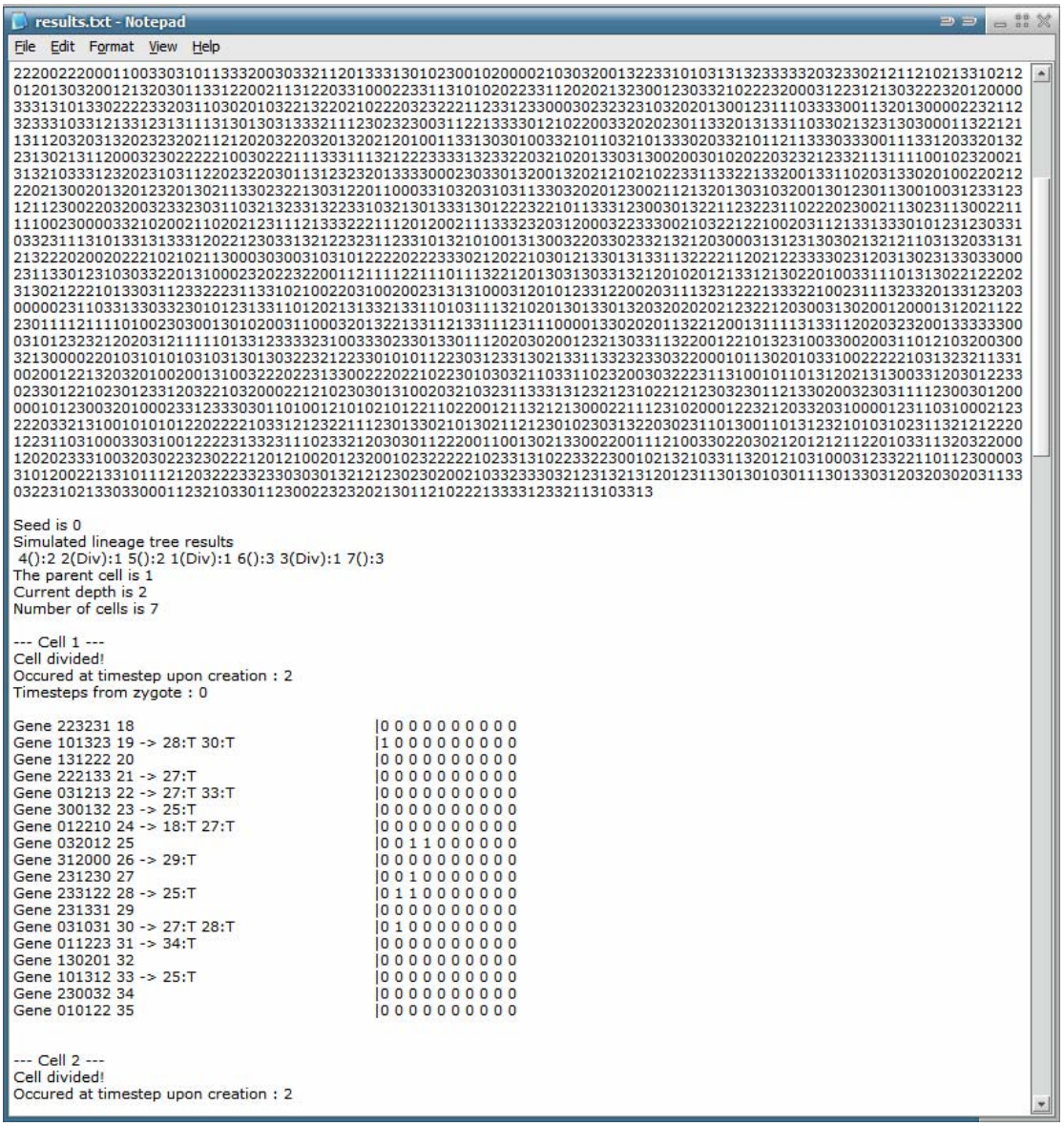


Cells 1, 2 and 3 have divided.

The results of the simulation can be saved using option 11. The next screen shows the results being saved in *results.txt*. This file is stored in the same directory as the application.



The file can now be opened in any text editor and have its results examined. The output of this file is shown in the next screen.



Gene expression graphs for all cells are generated along with useful statistics.

2.3. Running additional simulations

Once the lineage tree has been built, simulations can be run to create trees of increasing depth. A new lineage tree can also be generated from the current network if new inputs and outputs are to be changed, and subsequently simulated using the build option. If a new AG model is created, the network model is not automatically nor is the lineage tree. Therefore if a new AG model is created, a new network has to be created as well in addition to the lineage tree model.

The application is currently only a basic prototype showing most of the allowing the user to perform simple simulations. The application can be expanded to allow for a graphical user interface to be built on top of the existing code. Please refer to the final report on the architecture used in structuring the framework.

3. Credits

This software was developed by Chooi Guan Lim at the University of Queensland.