Gene Regulation Simulation Tool

Meirav Zehavi
September 2009, under guidance of Prof. Pinter Ron

Table of Contents

1. General Information

Transcription regulation networks can be modeled by directed labeled graphs. The nodes represent proteins or mRNAs. Their states represent their activity levels. Weighted edges represent the regulations. The time is discrete. The basic idea is that in each time unit each node's state changes according to a transition function which depends on its state and the states of the sources of its incoming edges in the previous time unit.

The computational model and its link to biology (including an application) are described in [1]. In this report we shall focus on the computational model.

In this section we shall give a brief description of the basic model's input and simulations. Our tool performs simulations which are based on an extended version of this model, which is described in the following sections.

A "condition edge" is an edge from a node to a regular edge. A regular edge can't have more than one incoming condition edge. Each condition edge has a type which is "zero" or "negative".

An edge is "active" if it doesn't have an incoming condition edge or it has an incoming condition edge and one of the following is fulfilled:

- The edge's condition is "zero" and the state of its source node is zero.
- The edge's condition is "positive" and the state of its source node is positive.

1.1 The input:

- A directed weighted labeled graph that includes condition edges.

- The maximal state N a node can have. N is a positive integer.

- The nodes' initial states. These are integers between 0 and N.

- The number of simulation steps.

- The threshold values for the transition function: min_thres and max_thres. $min\_thres \leq 0 \leq max\_thres$.

- The weight of each regular edge.

- The type of each condition edge ("zero" or "positive").

1.2 The simulations:

Each combination of the initial states determines a different simulation. In each step t of each simulation the state of each node i, $s_i(t)$, is calculated. $s_i(0)$ is the initial state of the node. For $t > 0$, we use the following algorithm:

1. Compute the multiplications of i's incoming active edges with their source nodes.

2. Compute $k_i(t)$: the sum of the multiplications that were computed in the previous stage.

3. Calculate $s_i(t)$ using the following transition function:

$$s_i(t) = \begin{cases} \min(N, s_i(t-1) + 1) & , \quad k_i(t) > \text{max\_thres} \\ \max(0, s_i(t-1) - 1) & , \quad k_i(t) < \text{min\_thres} \\ s_i(t-1) & , \quad \text{otherwise} \end{cases}$$

2. Project Aims

- Extend the existing model. For example, allow the vertices' states to be computed using a variety of functions. In the basic model they can change only by +1 or -1 in each step.

- Implement a program that performs the simulations. Moreover, the program should provide statistics about all the simulations and specific information about each simulation (e.g. determine whether a steady state was reached or not).

- Implement an interface for the user by extending an existing Cytoscape's [2] plug-in written by Dor Ganor.

3. Implementation

3.1 Details of implemented extensions

- A variety of functions which are used in the calculation of $k_i(t)$ for each node i are available (in addition to the sum function).

- In the basic model each vertex's state can change only by +1 or -1 in each step. A variety of other functions is available in the extended model. A function f is chosen for each vertex i. The transition function is changed to the following function:

$$s_i(t) = \begin{cases} \min(N, s_i(t-1)+f(k_i(t))) & , f(k_i(t)) > 0 \ \wedge \ [min\_thres,max\_thres] \ni k_i(t) \\ \max(0, s_i(t-1)+f(k_i(t))) & , f(k_i(t)) < 0 \ \wedge \ [min\_thres,max\_thres] \ni k_i(t) \\ s_i(t-1) & , \text{otherwise} \end{cases}$$

This transition function is equal to the basic transition function when f is the function sign.

- Each node has its own min_thres and max_thres.

- A condition edge activates its target edge if the state of its source node is inside or outside a specified range (rather than only checking if it is in the range [0,0]). Each condition edge can have its own range.

- The user can choose information that will be provided in addition (or instead) to the simulations themselves. The information includes statistics about all the simulations and specific information about each simulation. The implemented options provide information about last states and boundaries of states that were reached by several chosen nodes, and the simulations' steady states.

- A graph can be created for a specified simulation. In order to create the graph we use jfreechart version 1.0.13. For more information: www.jfree.org.

The details about the mentioned extensions are described in the "User Guide" section.

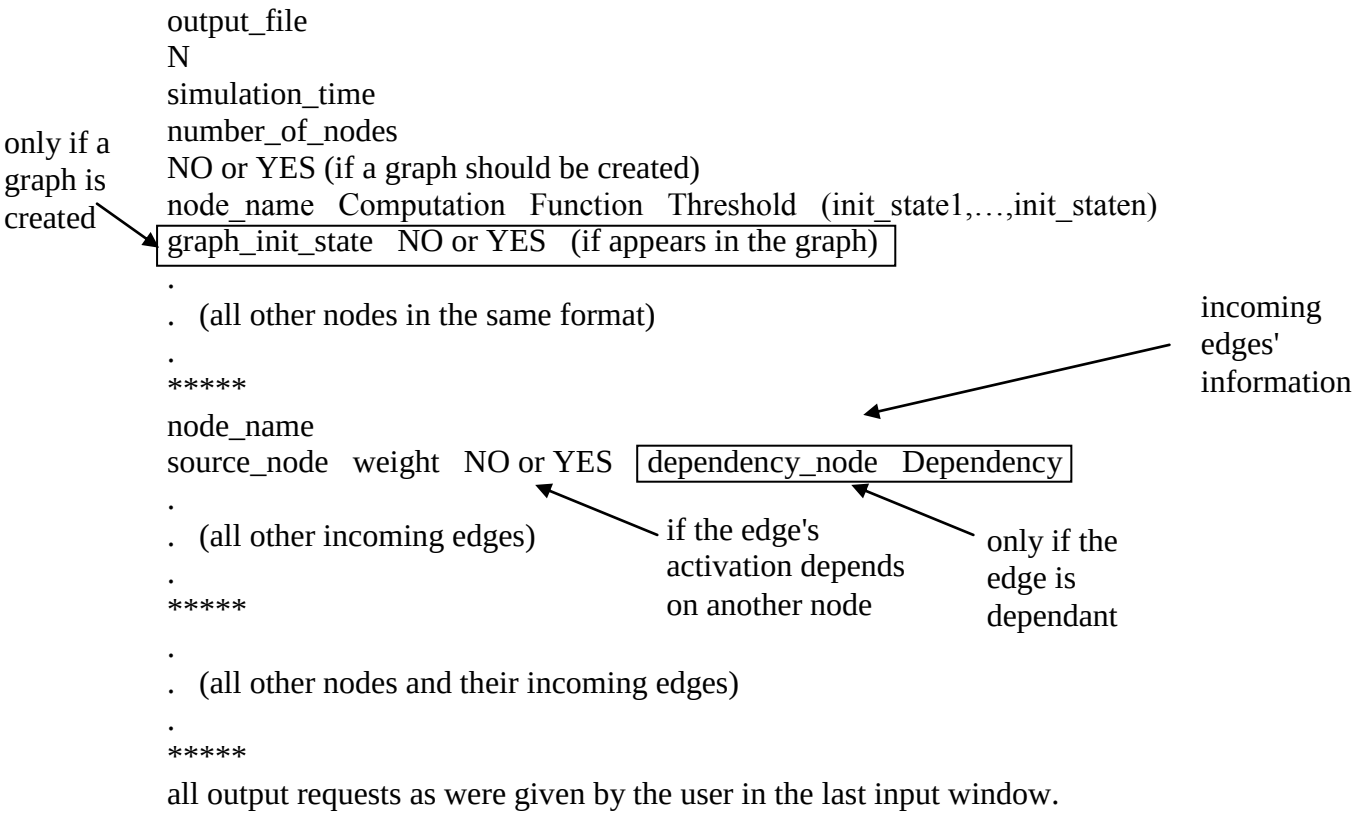
3.2. "net2text" and "model"

The implementation is divided into two different programs: "net2text" and "model". "net2text" implements the interface. It is based on an

existing Cytoscape's plug-in written by Dor Ganor. The user works in Cytoscape in order to draw the input graph and insert the information regarding the simulations and the graph's attributes. A validation of the input is performed and it is translated into a text file named "network.txt". "net2text" activates the second program, "model". "model" uses "network.txt." as its input. It runs the simulations and outputs the requested information.

The two programs can be used separately since "model", the default program that "net2text" activates, can be replaced by the user. The user can choose a different program that will use "network.txt", or he can create or use an already created text file and activate only "model".

3.3 The format of network.txt



4. User Guide

4.1. Adding the plug-in to Cytoscape

Adding the plug-in to Cytoscape does not require any installation. All that is necessary to do is to add the plugin's jar file, "net2text", to Cytoscape's "plugins" folder, located in Cytoscape's installation folder.

Adding the jar file to that folder adds the plug-in to Cytoscape. Next time the application is opened, the plugin's actions will appear in the "Plugins" menu.

The model executable, "model", should be added to Cytoscape’s folder (since it uses a text file, "network.txt", which will be created by the plug-in and is located in this folder).

4.2. Adding Hyper-edges (Cytoscape V2.6.0)

Hyper-edges are a recently added functionality in Cytoscape, which is required in our graphs. Hyper-edges come as a plug-in called "HyperEdgeEditor" in V2.6.0 and should be integrated into future versions of Cytoscape. Older versions of Cytoscape do not have the hyper-edge functionality and therefore they will not support our plug-in.

In V2.6.0 the HyperEdgeEditor is already added to Cytoscape's plug-in library and only needs to be activated. Go to the "Plugins" menu in the menus bar and select "Manage Plugins". Under "Available to install" open "Functional Enrichment" and select the "HyperEdgeEditor" plug-in. Press "install" to complete the process. Next time Cytoscape is opened, the HyperEdgeEditor will appear in the "Editor" tab of Cytoscape's Control Panel, and hyper-edges can be used in the graphs.

4.3. Creating the graph

The elements in the Editor are used to create the nodes and the edges of the graph.

4.3.1. Regular edges

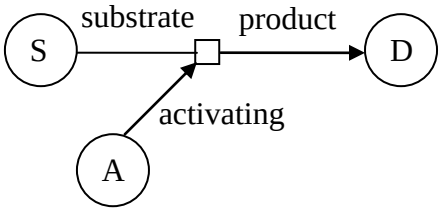
Activating edges are used to create regular edges with default weight of 1.

Inhabiting edges are used to create regular edges with default weight of -1.

4.3.2. Using connector-nodes

Connector nodes are unique nodes used in the reactions to enable the creation of condition edges.

A regular source node is connected to a connector node by an edge of type substrate. The connector node is connected to the destination node by an edge of type product. This is an edge unit. The weight of the substrate edge is determined according to the weight of the product edge. Its default weight is 1. A regular node is connected to the connector node by an edge of type activating. This is the condition edge. It holds the dependency condition and its weight is 0.



4.3.3. Rules in creating the graph

The following validation rules are checked on the graphs as part of the plugin's action:

- Each connector node has exactly one outgoing edge of type product, one incoming edge of type substrate and one incoming edge of type activating.
- Regular nodes are connected only by activating or inhibiting edges.
- The graph isn’t empty (it has at least one node).

4.4. Preliminary action and attributes

The plug-in contains three actions. The first action is called "Net2TextInitializer". It is used to add colors to the graph's edges and attributes to its nodes and edges (with default values explained in 4.4.5 and 4.4.6). The activation of this action is done by selecting "Net2TextInitializer" from the "Plugins" menu of the menu bar.

4.4.1. The edge's colors

The colors of the edges are set as follows:

- Green for edges with a positive weight.
- Blue for edges with the weight 0.
- Red for edges with a negative weight.

Activating the preliminary action will color all the edges of the graph according to their default weights. Any edge added to the graph afterwards will not have a weight value and therefore will be colored with the default color, which is black. In order to color a new edge, its weight attribute should be set. Setting the weight will automatically add the compatible color to the graph (this means that re-activation of the preliminary action is not required in order to update colors).

4.4.2. Viewing the attributes

An attribute's value can be seen and selected by adding it in the Data Panel. For example, the attribute "Weight" can be selected in the Data Panel as follows:

1. Go to the "Edge Attribute Browser" in the Data Panel.

2. Push the button "Select Attribute" (in V2.6.0 the button is located on the top left corner of the Data Panel and looks like a small gray table). Pushing the button opens the list of the edges' attributes.

3. Select "Weight". Selection of other unneeded attributes can be cleared. Now, after selecting an edge in the graph its weight can be seen in the "Edge Attribute browser" tab of the Data Panel.

4.4.3. Setting attributes

An attribute of an edge or a node can be edited by selecting it on the graph, double clicking on the wanted attribute in the Data Panel, and changing its value.

Changing the sign of the weight attribute of an edge (positive, negative or zero) will automatically change the edge's color accordingly.

4.4.4. Re-activating the preliminary action

Re-activation of the preliminary action will restore the default values of the attributes and the corresponding colors of the edges. Any manual changes to these values will be lost.

"Net2TextPartialInitializer" is the same as the preliminary action (adds colors and attributes), except that it applies default values only to attributes that weren’t already initialized. "Net2TextPartialInitializer" saves manual changes. It is located in the "Plugins" menu of the menu bar.

4.4.5. The nodes' attributes

- **Computation**: A variety of mathematical and statistical functions is available. Their argument is the value of the multiplications of the node's incoming active edges with their sources' states.

  - SUM: Sum.
  - MUL: Multiplication.
  - MIN: Minimum.
  - MAX: Maximum.
  - AVG: Average.
  - MED: Median.

Notes: - SUM is the default option.
- If the node isn't a target of any active edges, the computation's result is 0.
- The computation's result is rounded.

- **Function**: A variety of mathematical functions is available. Their argument is the result of the computation function. Their result is the value that will be added to the node's state in the calculation of its next state (if it is not in the thresholds' range).

  - SIG(n): Extended sign. Returns n if the argument is positive, -n if the argument is negative and 0 otherwise. n is an integer.
  - POL($(c_1,d_1),(c_2,d_2),\ldots,(c_n,d_n)$): Extended polynomial. Each c_i and each d_i is an integer. If the argument's value x is 0 then x^d, when d is negative, is defined as 0.
  - LOG(base): Logarithm. base is the logarithm's base, which must be positive and not equal to 1. base can be an integer or e (syntax: LOG(e)). If the argument isn't positive, the result is defined as 0.
  - POW(base): Power. base is the power's base, which is an integer or e.
  - ABS: Absolute value.
  - SQT: Square root. If the argument is negative, the result is defined as 0.

Notes: - SIG(1) is the default option.
- The function's result is rounded.

- **Threshold**: Each node has its own thresholds.
  Syntax: (min_thres,max_thres).

min_thres $\leq 0 \leq$ max_thres.
The default is (0,0).

4.4.6. The edges' attributes

- **Dependency**: Apply this attribute to edges that activate other edges.
  Syntax: IN(min,max) or OUT(min,max). min and max are integers and min $\leq$ max.
  The target edge is activated if its source's state is in or equal to the range, or out of the range, corresponding to IN or OUT.
  The default is OUT(0,0).

- **Weight**: Change this attribute's default value to edges that don't activate other edges and are not of type substrate. An edge's weight is an integer.
  The default value is determined in the following way:
  - For edges outgoing from regular nodes:
    * Condition edges – "0".
    * Edges of type "inhabiting" – "-1".
    * Regular edges of type "activating" – "1".
    * Edges of type "product"– "1".
  - For edges outgoing from connector nodes: The value is "1" or "-1" according to the connector node's "product" outgoing edge.

  Note: After the preliminary edges of type inhibiting can be given a positive weight and edges of type activating can be given a negative weight. The colors are changed according to the weights.

4.5. Main action

The main action of the plug-in is called "Net2TextPlugin". It is used to gather the simulations' information from the user. It creates the output file using "model" as its default.

Selecting this action opens a series of input windows.

4.5.1. First input window (general information)

The first window gathers the following information:

- N, the maximal state a node can have. N must be a positive integer.

- The simulations' time. This is the number of steps to be done in each simulation. It doesn't include step 0, which represents the initial states of each simulation. The simulations' time must be a positive integer.
- The model that is activated by the plug-in in order to perform the simulations. This line should be left empty in order to activate the default model (see the "Implementation" section for more information).
- The output file. The button "Choose Output File" opens a file chooser in which a text file should be chosen. The output will be written in this file. Notice that the file must have a ".txt" extension.

Moving to the second input window is done by clicking on the "Next" button in the last line.

4.5.2. Second input window (initial states)

The second window gathers information regarding the initial states of the nodes in the simulations. This window contains a table in which each column represents a node from the graph and each row represents a possible initial state. The titles of the columns state the names of the nodes represented by them. The initial states appear in the cells.

Selecting an initial state for a node is done by checking its box. The last cell of each column contains a "Select All" check box. Checking this box selects all of the possible initial states for its corresponding node. At least one initial state must be entered for each node, thus at one box must be checked in each column.

Moving to the third input window is done by clicking on the "Next" button on the bottom of the window.

4.5.3. Third input window (graph)

The third input window gathers the necessary information for creating a graph for a single simulation. Its structure is the same as the second window. Choosing "Select" for a node means that this node will appear in the graph. If the following conditions are not fulfilled, the graph won’t be created (you can click "Next" if you aren't interested in creating a graph):

- Exactly one initial state is chosen for each node.
- At least one node is chosen to appear in the graph. If more then ten nodes are chosen, only the first ten will appear in the graph.
- The chosen initial states were chosen in the previous window.

Moving to the fourth input window is done by clicking on the "Next" button on the bottom of the window.

4.5.4. Fourth input window (output requests)

In the fourth input window the information that will be printed in the output file is chosen. There are ten text boxes that can be filled (at least one of them should be filled).

The available options:

- AllSim: Prints all the simulations (the nodes' states in each step of each simulation). This option can be chosen only once.

- LastStates($node_1$,…,$node_n$): Prints the last states of the specified nodes after each simulation. The specified nodes must appear in the graph. After all the simulations it prints the percentage of simulations that reached each possible combination of the last states. This option can be chosen only once.

- SteadyStateStatistics($step_1$,…,$step_n$): Prints the step in which the simulation reached a steady state (or NOT REACHED) after each simulation. After the simulations it prints for each specified step the percentage of simulations that reached a steady state before (and including) this step.
  Notes: - $0 \leq step_1 < step_2 < \ldots < step_n$
  - The last step in a simulation doesn't detect a steady state since the simulation stops exactly after it.
  - This option can be chosen only once.

- Limits(SE,node,state) or Limits(BE,node,state): Prints the minimal and maximal states of the specified node that were reached in each simulation and the minimal step in which it was reached. The specified node must appear in the graph. It also prints the first step in which the node's state was smaller or equal, or bigger or equal (corresponding to SE and BE) than the specified state, or NOT REACHED.

The finish button performs a graph correctness check (according to section 4.3.3) and a verification of the syntax of all the attributes. The text file ("network.txt") which represents the graph and all the other necessary information is created, and the chosen model is activated, creating the requested output.

4.4.5. Re-activating the main action

Re-activation of the main action allows the changing of the simulations' values and run the model again on your graph.

4.6. Save and load your network

Saving the network is done by selecting the "save as" or "save" options from the "File" menu.

Loading the network is done by selecting the "open" option from the "File" menu. If the network's appearance or the Editor is different from how it should be, select "BioChemicalReaction" in "Current Visual Style" which is located in the "VisMapper" tab of the Control Panel. Also, if the edges' colors don’t appear, select the plug-in’s “net2textPartialInitializer”.

5. References

[1] Amir Rubinstein, Vyacheslav Gurevich, Zohar Kasulin-Boneh, Lilach Pnueli, Yona Kassir and Ron Y. Pinter. Faithful modeling of transient expression and its application to elucidating negative feedback regulation. Proceedings of the National Academy of Sciences (PNAS), Vol. 104, No. 15, pp. 6241-6246, April 2007.

[2] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Research, 13(11):2498–2504, November 2003.