GRegNetSim: A Tool for the Discrete Simulation and Analysis of Genetic Regulatory Networks

Meirav Zehavi\textsuperscript{1,2,*}, Dor Ganor\textsuperscript{2} and Ron Y Pinter\textsuperscript{2}

Abstract

Background: Discrete simulations of genetic regulatory networks have been used to study subsystems of yeast successfully. However, implementations of the two models underlying these simulations do not support a graphic interface, are not freely available, and require computations necessary to analyze their results to be done manually. Furthermore, differences between these two models suggest that an enriched model, encompassing both existing models, is needed.

Results: We developed a software tool, called GRegNetSim, that allows the end-user (a biologist) to describe genetic regulatory networks graphically. The input is graphic (as an input to Cytoscape, an open-source platform for the representation and analysis of biological networks). The user can specify various transition functions at different nodes of the network, supporting, for example, threshold and gradient effects, and then apply the network to a variety of inputs. GRegNetSim displays the relationship between the inputs and the mode of behavior of the network in a graphic form that is easy to interpret. Furthermore, it can automatically extract statistical data necessary to analyze the simulations.

Conclusions: The discrete simulations performed by GRegNetSim can be used to elucidate and predict the behavior, structure and properties of genetic regulatory networks in a unified manner. GRegNetSim is implemented as a Cytoscape App. Installation files, examples and source code, along with a detailed user guide, are freely available at https://sites.google.com/site/gregnetsim/.

Keywords: Genetic regulatory network; Cytoscape; Discrete simulation; Network analysis

Background

To elucidate and predict the behavior, structure and properties of genetic regulatory networks on a large scale, it is essential to devise efficient computational models that are expressive enough to capture various qualitative properties such as transience, robustness and stability. Indeed, numerous paradigms and modeling techniques have been suggested for the elucidation of metabolic, signaling, and regulatory pathways. On the one hand, continuous models, either stochastic or employing differential equations ([Karlebach et al., 2008]; [Mogliner et al., 2012]; [Morelli et al., 2012]; [Munsky et al., 2012]; [Smolen et al., 2000]), are detailed and therefore powerful. However, these models often require data that is unavailable (such as kinetic constants, concentration levels and timings). Moreover, a discrete method might be more appropriate when focusing on behavioral properties ([Bolouri et al., 2002]; [Bernot et al., 2004]; [Rubinstein et al., 2007]). On the other hand, boolean models are simple and computationally efficient ([Li et al., 2004]),
but their expressive power is rather limited and thus they fail to capture important features of genetic regulatory networks such as the transient and cascade characteristics of developmental pathways ([Rubinstein et al., 2007]; [Schaub et al., 2007]).

The discrete model introduced by [Rubinstein et al., 2007] extends the boolean model; it is simple and intuitive, yet richer and therefore more expressive than the boolean model. [Rubinstein et al., 2007] successfully employed this model to analyze the transcription of early meiosis-specific genes in budding yeast, predicting qualitative behaviors and specific interactions between components that were validated experimentally. In a follow-up work, [Rubinstein et al., 2013] revisited this model. They introduced several modifications, such as a log-related transition function, which were necessary to obtain a faithful representation of the network at hand. [Rubinstein et al., 2013] again exhibited the strong predicting capability of the discrete model, this time by studying the cell-cycle in budding yeast, elucidating its intrinsic structure and qualitative behavior.

Implementations of the computational models used by [Rubinstein et al., 2007] and [Rubinstein et al., 2013] do not support a graphic interface and are not freely available. In fact, some of the computations that were necessary to analyze the results were done manually. Furthermore, the differences between these models suggest that an enriched model that encompasses both of them, as well as other natural extensions, is necessary to provide a tool that is capable of handling various networks in a unified manner. We remark that recently, independently of our work, [Rubinstein et al., 2016] also developed a graphic interface for the execution of simulations.

Results
We developed a software tool, GREGNetSIM, that implements the model of [Rubinstein et al., 2007] along with a wide variety of optional extensions, including the one considered by [Rubinstein et al., 2013]. Apart from a textual user interface, GREGNetSIM provides an easy-to-use graphic user interface. It also implements computations, including statistical calculations, that were previously performed manually to analyze subsystems of yeast. A schematic diagram illustrating the behavior of GREGNetSIM is given in Figure 1 – the depicted components are described below. A detailed user guide, as well as examples for outputs of GREGNetSIM, are available with the installation files. The examples concern the developmental pathway of meiosis in the budding yeast *Saccharomyces cerevisiae*, and match the results described by [Rubinstein et al., 2007].

The Model
We first describe the discrete model of [Rubinstein et al., 2007], and then explain the extensions integrated in GREGNetSIM.

Basic Model
The basic model uses the convention of representing genetic regulatory by directed graphs, where nodes represent proteins/mRNAs and edges represent regulations. We use discrete-time simulations like aGPSS ([Greenberg, 1972]).
**Network State Representation.** At each time unit, every node has a *state* between 0 and *N* that reflects its activity level. Correspondingly, the *network state* is a vector of length *n*, the number of nodes in the network, with entries between 0 and *N*. The network state can also be viewed as an *n*-digit number of base *N*. The next state of a node *v* is determined by a transition function that depends on the states of *v* and its incoming neighbors (see below).

**Conditional Edges.** A *conditional edge* is an edge that goes from a node to another edge. Each conditional edge is associated with a type: “zero” or “positive”. It is *satisfied* if both its type and the state of its source node are zero/positive. We say that an edge is *active* if it has no incoming unsatisfied conditional edges. Intuitively, a conditional edge indicates that its target edge exists – that is, the source node of the edge indeed affects the target node – only in the presence or absence of another node (the source node of the conditional edge).

**Input.** The input consists of a directed edge-weighted graph that models a genetic regulatory network, along with a set of conditional edges (that do not have common targets). Moreover, to perform simulations, the input is also defined by the maximum state, *N*, that can be reached by each node; an initial state, *s_i(0)*, for each node *i*; the length of the simulation; threshold values, *min_i ≤ 0 ≤ max_i*, for each node *i*.

**Simulations.** At least one initial state should be assigned to each node. Each combination of initial states (one per node) defines an input for which a simulation.
is performed. For example, if each node has two initial states, $2^n$ distinct simulations are performed.

**A Single Simulation.** Each simulation consists of the number of time units specified by the user. At each time unit, $t \geq 1$, the following procedure determines the state $s_i(t)$ of a node $i$:

1. For each incoming active edge of $i$, calculate the multiplication of its weight and the state of its source node.
2. Calculate $k_i(t)$, the sum of the multiplications from Step 1.
3. Calculate $s_i(t)$ according to the following transition function:

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

**Enriched Model**

Our enriched model implements three extensions.

**The Type of a Conditional Edge.** First, each conditional edge can now have a type that is more general than “zero” or “positive”. In the enriched model, each type is a triple, $(x, a, b)$, where $x \in \{\text{in, out}\}$, and $a$ and $b$ are integers satisfying $a \leq b$. The conditional edge is satisfied if $x = \text{in}$ and the state of its source node belongs to $[a, b]$, or if $x = \text{out}$ and the state of its source node does not belong to $[a, b]$. Such a condition is essential to capture the situation where an interaction is present only when the activity level of a different node belongs to a range that is not simply $[0]$ or $[1, N]$ (e.g., when the activity level surpasses a threshold strictly larger than 1).

**The Calculation of $k_i(t)$.** In the enriched model, each node $i$ can have its own method of calculating $k_i(t)$. The calculation of $k_i(t)$ can be chosen from a collection that includes, for example, the sum, multiplication and average of a number of values. This option can be used to model situations where the overall affect of nodes is not additive.

**A Collection of Functions.** Our third extension enables each node $i$ to replace the addition/subtraction of 1 in the transition function by a different computation, given by a function $f_i(t)$, which is chosen from a collection of functions that includes, for example, extended sign functions and polynomials.[1] Thus, the transition function is now defined by the following formula:

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

where $K_i(t)$ is a predicate that is true if and only if $k_i(t) \notin [\min_i, \max_i]$.

[1] The extension examined by [Rubinstein et al., 2013] replaces the basic transition function by a log-related function.
Implementation

Data Entry

GRegNetSim is suitable for Cytoscape 2.6–2.8 ([Smoot et al., 2011]), and relies on the HyperEdgeEditor App; thus, graphs that contain conditional edges can be easily drawn by dragging elements from the “Editor”. The app provides an action that adds attributes to the elements of the graph (including those that define the transition function associated with each node), along with default values, and colors the edges according to their weights. When rerunning a set of simulations (e.g., to examine a different set of initial states), it is not necessary to reenter attributes. A series of windows gathers information about each specific set of simulations, such as the value $N$, the length of the simulations, the initial states of the nodes, and the optional special calculations GRegNetSim should perform to analyze the simulations (see Figure 2).

Analysis of Results

Apart from printing the simulations (i.e., the network state at each time unit of every simulation), GRegNetSim is equipped with optional computations, including statistical calculations (see Figure 3), that allow analyzing the simulations in a simple manner. The importance of these computations, previously performed manually, was exhibited by [Rubinstein et al., 2007] and [Rubinstein et al., 2013].

- GRegNetSim can print the time when a steady state was reached (or “not reached”) at each simulation. It also prints the percentage of simulations that reached a steady state at chosen time units.
- GRegNetSim can draw graphs that visualize requested simulations in a manner that is easy to interpret (see Figure 4). To this end, it uses [JFreeChart].

Figure 2 GRegNetSim provides a graphic user interface (screenshot).
Figure 3. GRegNetSim implements a variety of computations, including statistical calculations, that were previously performed manually.

- For each simulation, GRegNetSim can print only the final states of chosen nodes. Moreover, it prints the percentage of simulations that reached each combination of final states of these nodes.
- GRegNetSim can print the largest/smallest state reached by chosen nodes (per simulation), and the first time unit when they reached specific states.

Conclusions
We presented GRegNetSim, a software tool for the discrete simulation and analysis of genetic regulatory networks. Special cases of the model implemented by GRegNetSim have been used on two subsystems of yeast successfully. To make the model accessible, GRegNetSim provides a graphic user interface, extensions that allow analyzing different networks in a unified manner, and computations necessary to interpret the simulations.

Declaration

Ethics
Not applicable.

Consent to publish
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Authors contributions
RYP planned and supervised the project. MZ and DG designed the user interface and wrote the user guide. MZ implemented the software tool and created the website. MZ and RYP wrote the manuscript. All authors read and approved the final manuscript.

Availability and Requirements
- Project name: GRegNetSim
- Project home page: https://sites.google.com/site/gregnetsim/
- Operating system(s): Platform independent
- Programming language: Java
- Other requirements: Cytoscape 2.8
- License: Free
- Any restrictions to use by non-academics: None

Author details
1 Simons Institute for the Theory of Computing, UC Berkeley, 121 #2190, Melvin Calvin Laboratory, Berkeley, United States. 2 Department of Computer Science, Technion City, Haifa, 3200003, Israel.
Figure 4 GRegNetSim displays the relationship between the inputs and the mode of behavior of the network in a graphic form.

References