NetLand: a software for quantitative modeling and visualization of Waddington's epigenetic landscape

Jing Guo[1,2] and Jie Zheng[1,3,4*]

[1]School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore, [2]Bioinformatics Institute, Agency for Science, Technology, and Research (A*STAR), Singapore 138671, Singapore, [3]Genome Institute of Singapore, A*STAR, Singapore 138672, Singapore and [4]Complexity Institute, Nanyang Technological University, Singapore 637723, Singapore

Email: zhengjie@ntu.edu.sg

URL: http://netland-ntu.github.io/NetLand/

Last updated: 7 Oct, 2016

Table of Contents

Introduction

Waddington’s epigenetic landscape

Waddington’s epigenetic landscape is a visual metaphor for cellular dynamics in embryonic development, proposed by Conrad Hal Waddington (1905 - 1975) in 1957 [1]. In this picture, the cells are marbles rolling down a hill along trajectories towards valleys representing stable cell states. Despite its unifying power and unfading popularity, the original concept of Waddington’s epigenetic landscape was only a qualitative metaphor that lacks rigorous interpretation of mechanisms. Nonetheless, it has inspired mathematical ideas in theoretical biology [2, 3]. Recently, quantitative computational models of Waddington’s epigenetic landscape have been proposed [4-7]. Typically, these methods use concepts and mathematical models from physics, as analogues for cellular dynamics.

Limitations

However, practical applications of these methods need address several limitations.

- There is a lack of user-friendly tools to calculate and display the landscape.
- Most of the methods are still restricted to very small networks (e.g. the 2-gene regulatory network with mutual inhibitions [7]).
- It is still difficult to visualize the 3-dimensional landscape for a regulatory network with more than two genes. Although a 52-gene regulatory network has been recently used to plot a potential landscape [8], their method either selects only two marker genes for the coordinates of state space, or requires the system to have only two attractors.

NetLand

To address these issues, we introduce the NetLand software which can fill these gaps, so that the quantitative models of Waddington’s epigenetic landscape can be widely used by researchers of diverse backgrounds.

NetLand is intended for modeling, simulation and visualization of gene regulatory networks (GRNs) and their corresponding quasi-potential landscapes. Users can import models of GRNs from a file (e.g. TSV, SBML format etc.), and manually edit the network structure. Then, NetLand will automatically encode differential equations for the kinetics of transcriptional regulations. The computational model will be used to simulate the dynamics of the input networks. Model equations and parameters can be easily modified.

To display the 3D landscape for a network of more than two genes, NetLand allows users to either choose two marker genes, or project to a latent space using a dimension reduction method called GPDM (Gaussian process dynamical model) [9, 10]. Therefore, NetLand can provide a global picture of cellular dynamics for a user-specified GRN. Although we designed the NetLand software originally for modeling stem cell fate transitions, it can be also used to study other cellular phenotypes, such as cancer cell death, cellular ageing, etc.

The source code of NetLand was written in Java, and thus it can run under Microsoft Windows, Linux/Unix and Mac OS X. This user manual contains instruction about installation and basic usage. It also includes running time assessment and case studies.

Installation

NetLand runs on Windows, UNIX/Linux and Mac OS X. We have tested NetLand under 64-bit version of Window 7/10, Linux Fedora 18 and Mac OS 10.8. The Java Platform, Standard Edition Runtime Environment (JRE) is required to be installed (Java SE is available at **http://java.com/en/download/inc/windows_upgrade_ie.jsp**). The software NetLand can be downloaded for free at **http://netland-ntu.github.io/NetLand/.** Below is a list of external dependencies, followed by installation instructions for the different operating systems.

External Libraries

A number of external open-source libraries are bundled with the NetLand program. The external library dependencies are listed in the following table.

Table 1. The list of external libraries.

Package	Description	License	Version
GeneNetWeaver (GNW) [11]	NetLand uses the network graphic interface of GNW.	MIT license	3.1
libSDE	libSDE is an Java library for numerical integration of Stochastic Differential Equations (SDEs).	MIT license	1.0.3
JMathPlot	Jmathplot is a light java library designed to allow easy plotting of 2D and 3D data in a java panel. It is used to plot the simulated time courses.	BSD License	1.0.1
JLatexMath	A Java API to render LaTeX Project logo. It is used to render mathematical equations to user-friendly format.	GNU General Public License (GPL)	1.0.3
Dizzy	Dizzy is a chemical kinetics stochastic simulation software package. It is used to perform stochastic simulation using Gillespie algorithm.	GNU Library or "Lesser" General Public License (LGPL)	1.11.4
BeautyEye	Look and Feel for swing	Apache License 2.0	3.7
Jzy3d	Jzy3d is an open source Java library that allows to easily drawing 3d scientific data.	The BSD 3-Clause License	0.9.1
JOGL 2 and GlueGen	JOGL 2 is designed to provide hardware-supported 3D graphics to applications written in Java. It is required by Jzy3d. GlueGen is a tool which automatically generates the Java and JNI code necessary to call C libraries.	The New BSD 2-clause license	2.1.5

GPDM (C++)	Gaussian process code in C++ including some implementations of GP-LVM and IVM	0.001

Dependencies for "GPDM" program

To run the "GPDM" program, libraries of BLAS, LAPACK and GFORTRAN are required.

Windows OS: The DLL (Dynamic Link Library) files are provided in the "GPDM/win" folder. **Mac OS X**: Please check if *libblas.dylib*, *liblapack.dylib* and *libgfortran.dylib* are already installed in the system. The default installation path is /usr/lib/. Otherwise users should install the libraries themselves.

Unix/Linux: Please check if *libblas.so*, *liblapack.so* and *libgfortran.so* are already installed in the system. The default installation path is /usr/lib/. Otherwise users should install the libraries themselves.

NOTE: the actual installation directory of the three libraries could be different. To check if they are installed, use a terminal, run: "*locate <library>*".

To install the dependencies, the BLAS, LAPACK and GFORTRAN can be downloaded from

- LAPACK, see http://www.netlib.org/lapack/
- BLAS, see http://www.netlib.org/blas/
- GFORTRAN, see https://gcc.gnu.org/wiki/GFortranBinaries#MacOS

Then follow the install instructions in the package. For Linux, users can use the command, e.g. *yum, apt-get*, to install these packages.

UNIX/Linux and Mac OS X

1. Download NetLand from GitHub: **http://netland-ntu.github.io/NetLand/.**
2. Unzip it to your desired location.
   The NetLand package should include the following files:
   a) **One shell script**, named "runNetLand.sh", for launching NetLand in Linux/Unix operating systems.
   b) **A jar file**, named "main.jar". It should be in the same directory as the shell script.
   c) A folder, named "**lib**". It contains the required libraries to launch NetLand.
   d) A folder namely "**GPDM**" which contains the executable files required for running "GPDM" program in NetLand.
   e) Two folders namely "toy models" and "saved results" (not necessary for launching the software) which contain toy gene network files and pre-computed results respectively.
3. Make sure you have the executive permissions to the executable files in "GPDM" folder. To check the permission of a file, typing the following command at a terminal: "*ls –l <filename>*". Use "*chmod 705 <filename>*" to gain the executive permission.
4. If everything installed correctly, you're ready to run a simulation (see next section).

Windows

1. Download NetLand from GitHub: **http://netland-ntu.github.io/NetLand/.**
2. Unzip it to your desired location.
   The NetLand package should include the following files:

a) **One shell script**, named "runNetLand.bat", for launching NetLand in Windows operating systems.
b) **A jar file**, named “main.jar”. It should be in the same directory as the shell script.
c) A folder, named "**lib**". It contains the required libraries to launch NetLand.
d) A folder namely "**GPDM**" which contains the executable files required for running "GPDM" program in NetLand.
e) Two folders namely "toy models" and "saved results" (not necessary for launching the software) which contain toy gene network files and pre-computed results respectively.

3. If everything installed correctly, you’re ready to run a simulation (see next section).

Note that the scripts must be in the same directory as the “GPDM” folder and "lib" folder.

Launching NetLand

To launch NetLand:

- **In Windows, double click the “runNetLand.bat”.**
- **In Linux, run command “./runNetLand.sh” under a terminal.**
- **In Mac OS X, run command “./runNetLand.sh” under a konsole.**

For Windows users, please check if "Java" is in your system path. If not, you can either add it to the system path or add the full path of "java.exe" to the script.

Example:

```
"C:\Program Files\Java\jre1.8.0_31\bin\java.exe" -classpath "lib\*;main.jar"
WindowGUI.NetLand
```

For Mac OS X and Linux users, the script should be run under a terminal or konsole using the command "*bash runNetLand.sh*" or "*./runNetLand.sh*". Make sure you have the executive permissions to the scripts. To check the permission, the command is “*ls –l <filename>*”. Use “*chmod 705 <filename>*” to gain the executive permission.

Memory requirement

NetLand uses the default settings of RAM in JVM. Minimal 20MB is required to launch NetLand. To manually change the RAM, users can add "*-Xms<number>m –Xmx<number>m*" to the shell scripts. This following command will allocate at least 2GB RAM and at most 4GB RAM to run NetLand.

Example:

```
java -Xms2048m -Xmx4096m -classpath "lib\*;main.jar " WindowGUI.NetLand
```

The RAM requirement for this software can vary to different computational implementations, e.g. the iteration of simulations. The details of memory consumption can be found at **Chapter 6**. **Model operation**

The analysis in NetLand is based on the computational model derived from GRNs. A dynamic model, a.k.a. a computational model, contains three key components, i.e. species, reactions and parameters. The nodes (or genes) in a GRN compose the species which are the products, reactants or modifiers in a reaction. Chemical reactions describe the regulatory relations between species. Parameters control the extent of the reactions. For example, in a transcriptional regulation, the transcriptional factors are the modifiers binding with the DNA sequence to produce mRNA. The reaction rate as a parameter reflects the speed of the process.

In NetLand, the transcriptional and translational processes of a target gene are simplified and combined into one differential equation (DE). Thus the change of the concentration of each gene is encoded in the form below:

$$\frac{d[\mathrm{Gene1}]}{dt} = K^{+}\varphi(\mathrm{Gene1}, \mathrm{Gene2}, \mathrm{Gene3} \ldots \mathrm{GeneN}) - K^{-}[\mathrm{Gene1}] + D \cdot \theta(t) \quad (1)$$

where [Gene1] denotes the concentration of Gene1. K^{+} is the maximum transcriptional rate of Gene1, K^{-} is the degradation rate, $\varphi(\cdot)$ represents the cis-regulation of Gene1 by other nodes, $\theta(t)$ is the noise term, and D controls the strength of noises. For each gene, $\varphi(\cdot)$ is composed of two parts, i.e. an active component and an inhibitive component, denoting the regulation of activators and inhibitors by Hill equations (Eq. 2).

$$\Phi(\cdot) = a * \sum_{i=1}^{ma} \frac{\mathrm{Genei}^{na}}{\mathrm{Genei}^{na} + \mathrm{ka}^{na}} + b * \sum_{i=1}^{mb} \frac{\mathrm{ki}^{ni}}{\mathrm{Genei}^{n} + \mathrm{ki}^{ni}} \quad (2)$$

Note that this formulation may not be appropriate to other types of regulations, e.g. signaling networks, metabolic networks. For the analysis of networks or models other than GRNs, we recommend that users upload an existing model saved in SBML format with reactions of each molecular. Or users can modify the reactions after loading the network.

Loading a model

NetLand can directly load a dynamic model (SBML) or a network structure file (TSV, GML, DOT) which will be automatically converted into a dynamic model. To load a SBML or a network structure file, click the "**Load Network**" function in the "**File**" list or the button at the functional bar (Figure 1).

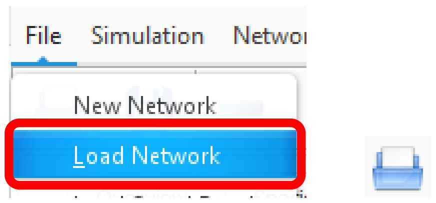


Figure 1. Functions to load a model.

The descriptions of input formats are listed below.

a) The Systems Biology Markup Language (**SBML**) is a representation format, based on XML, for communicating and storing computational models of biological processes.

SBML can represent many different classes of biological phenomena, including metabolic networks, cell signaling pathways, regulatory networks, infectious diseases, and many others. For more information of SBML, please refer to http://sbml.org/Main_Page. NetLand can process SBML files that describe transcriptional regulations.

b) **TSV** is a file extension for a tab-delimited file used with spreadsheet software. TSV stands for Tab Separated Values. TSV files are used for raw data and can be imported into and exported from spreadsheet software. TSV files are essentially text files, and the raw data can be viewed by text editors, though they are often used when moving raw data between spreadsheets.
Example:
```
A	B	+
B	A	-
```

c) Graph Modeling Language (**GML**) is a hierarchical ASCII-based file format for describing graphs. It has been also named Graph Meta Language.
Example:
```
graph [
  comment "Two-gene network"
  directed 1
  id 1
  label "Hello, I am a graph"
        node [
      id 1
        label "Gene A"
                thisIsASampleAttribute 42
        ]
      node [
        id 2
        label "Gene B"
        thisIsASampleAttribute 44
        ]
        edge [
                source 1
                target 2
                label "Edge from Gene A to Gene B"
        ]
    ]
```

d) **DOT** is a plain text graph description language. It is a simple way of describing graphs that both humans and computer programs can use.
Example:
```
graph ethane {
    A -- B [type=s];
}
```

Creating a model

By clicking the “**New Network**” button in the “**File**” menu, a demo network with a node and a self-loop is created. Then users can modify the network, e.g. adding new nodes or edges or change regulatory equations.

Saving a model

The dynamic models can be saved in SBML format. And the structure of the network can be saved in TSV format. The saving functions can be found under the “**File**” menu (Figure 2).

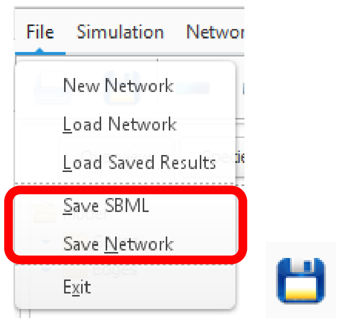


Figure 2. Functions to save the model.

Modifying a model

NetLand provides flexible ways to load and modify a transcriptional network. In this section, we describe how to use NetLand to create a network and modify the structure, e.g. deleting nodes, adding new nodes or edges.

Block nodes

The nodes in a gene regulatory network can be blocked to simulate the gene knockout experiment. By clicking the “**Block nodes**” button in “**Network**” menu or the button in the top button bar, a dialog with a gene list will be pop-up (Figure 3). After moving the blocked genes to the right list, the reactions of the remaining genes should be modified when they contain removed nodes as regulators.

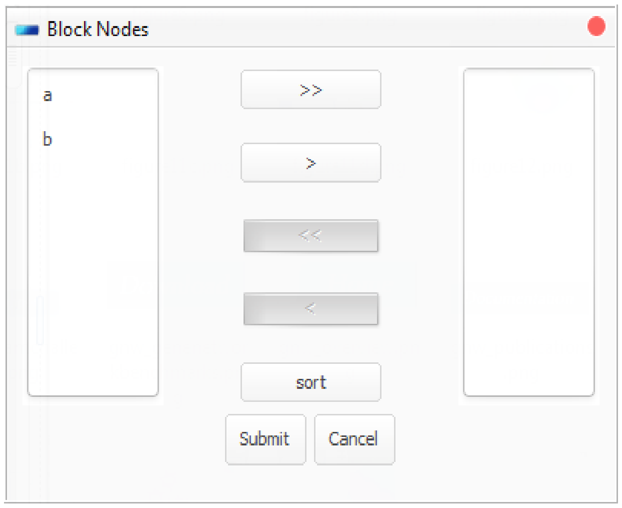
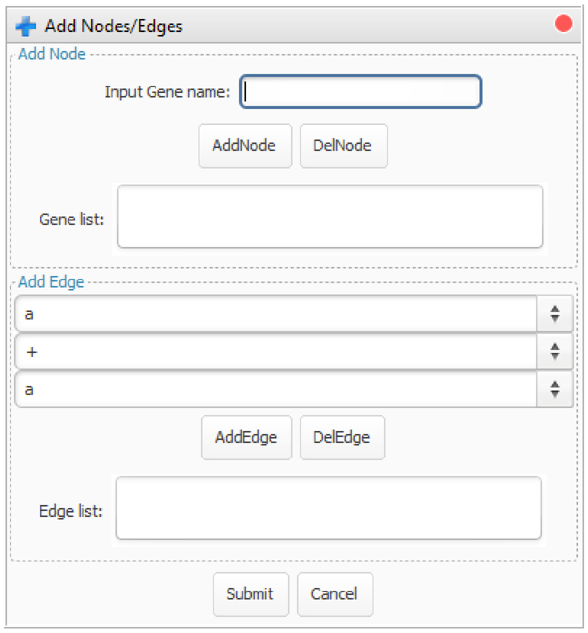


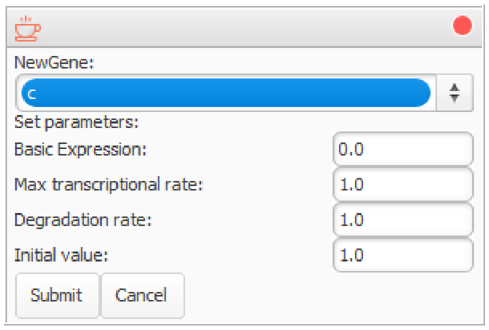
Figure 3. The panel for blocking nodes.

Add nodes/Edges

Compared to blocking functions, users can add new genes and edges to build new regulatory interactions. By clicking the "**Add nodes/edges**" item in "**Network**" menu or the button in the top button bar, a dialog will be pop-up (Figure 4a). Once you add a new gene, a set of parameters should be verified including initial values, degradation rates *etc*. (Figure 4b). The new regulatory relations (edges) are encoded as Hill equations and added to the target genes' original formulas (Figure 4c).



(a)

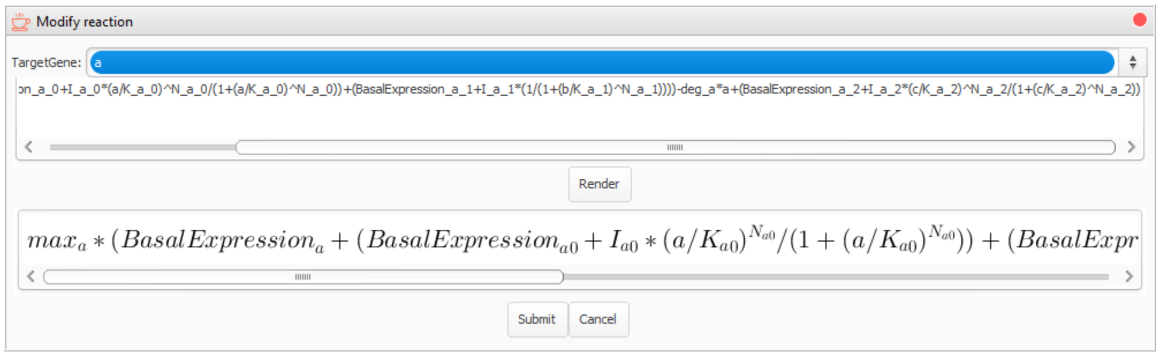


NewGene: c

Set parameters:

Basic Expression:	0.0
Max transcriptional rate:	1.0
Degradation rate:	1.0
Initial value:	1.0

(b)



```
ɔn_a_0+I_a_0*(a/K_a_0)^N_a_0/(1+(a/K_a_0)^N_a_0))+(BasalExpression_a_1+I_a_1*(1/(1+(b/K_a_1)^N_a_1))))-deg_a*a+(BasalExpression_a_2+I_a_2*(c/K_a_2)^N_a_2/(1+(c/K_a_2)^N_a_2))
```

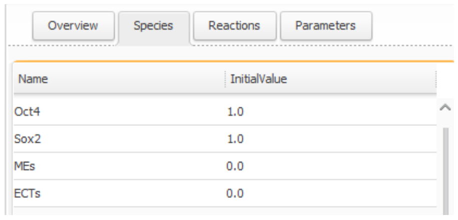
$max_a * (BasalExpression_a + (BasalExpression_{a0} + I_{a0} * (a/K_{a0})^{N_{a0}}/(1 + (a/K_{a0})^{N_{a0}})) + (BasalExpr$

(c)

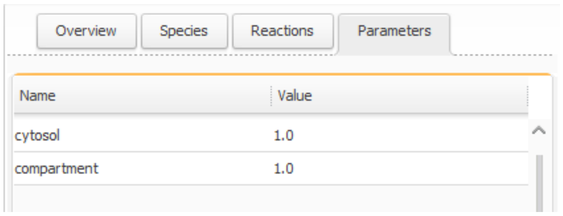
Figure 4. The panel for adding nodes and edges.

Change parameters

The initial values of genes and parameters can be directly modified in the “Species” and “**Parameters**” panel respectively (Figure 5). Double click the value column to change the number.



Name	InitialValue
Oct4	1.0
Sox2	1.0
MEs	0.0
ECTs	0.0



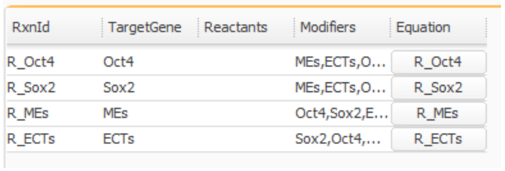
Name	Value
cytosol	1.0
compartment	1.0

(a) (b)

Figure 5. The panels of species and parameters of the dynamic model.

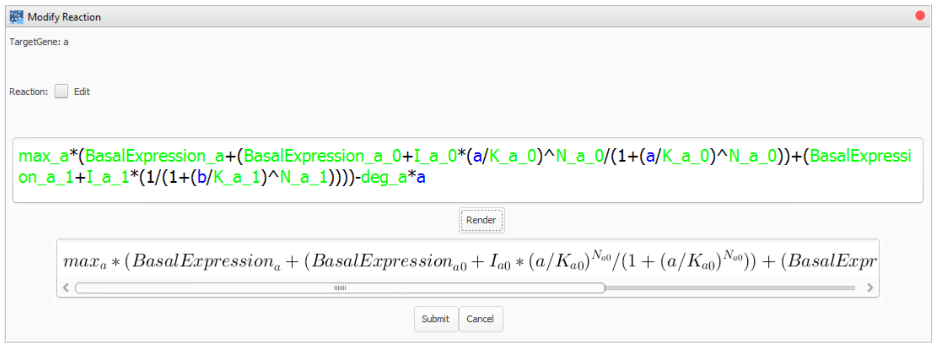
Update reactions

The reactions can be easily modified by clicking the button in the “**Reaction**” Panel (Figure 6a). Then a dialog will pop up (Figure 6b). The parameters are colored green, while species are marked blue. The “**Render**” button makes the function in the upper window displayed below in a human-readable format.



RxnId	TargetGene	Reactants	Modifiers	Equation
R_Oct4	Oct4		MEs,ECTs,O…	R_Oct4
R_Sox2	Sox2		MEs,ECTs,O…	R_Sox2
R_MEs	MEs		Oct4,Sox2,E…	R_MEs
R_ECTs	ECTs		Sox2,Oct4,…	R_ECTs

(a)



TargetGene: a

Reaction: ☐ Edit

```
max_a*(BasalExpression_a+(BasalExpression_a_0+I_a_0*(a/K_a_0)^N_a_0/(1+(a/K_a_0)^N_a_0))+(BasalExpressi
on_a_1+I_a_1*(1/(1+(b/K_a_1)^N_a_1))))-deg_a*a
```

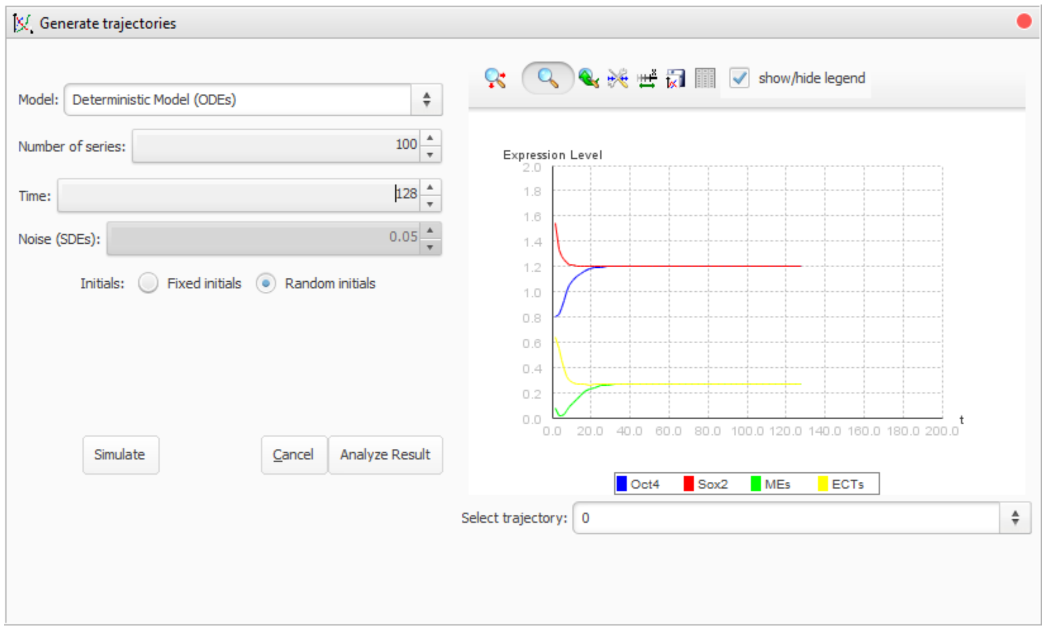
$max_a * (BasalExpression_a + (BasalExpression_{a0} + I_{a0} * (a/K_{a0})^{N_{a0}}/(1 + (a/K_{a0})^{N_{a0}})) + (BasalExpr$

(b)

Figure 6. The panel of reactions and the edit window.

Running a simulation

NetLand is designed to study GRN kinetics. Through in silico simulation, the dynamics of the GRN can be visualized in Waddington's epigenetic landscape. The transcriptional network can be encoded in deterministic models (ODEs) or stochastic models (SDEs).



Model: Deterministic Model (ODEs)

Number of series: 100

Time: 128

Noise (SDEs): 0.05

Initials: Fixed initials [✓ Random initials]

Select trajectory: 0

Figure 7. The panel of simulation analysis.

Simulators

ODE solver

A deterministic simulator, i.e. the 5(4) Dormand-Prince integrator, implemented in the package of org.apache.commons.math3.ode which provides classes to solve non-stiff Ordinary Differential

Equations problems is used to solve ODEs. This simulator is an implicit-explicit ODE solver using variable steps. The same simulator has been used in Matlab command ODE45. The minimal step size is set 0.1. The maximal step size is set 1,000. The allowed absolute and relative scalar errors are both set 0.000001. To change the settings, please contact the developer at zhengjie@ntu.edu.sg.

SDE solver

In the stochastic model, the change of species concentrations is described by Langevin equations (SDE). By control of the noise strength in SDEs, users can study the influence of noises on the transcriptional regulations. SDEs in *Itô* scheme are numerically solved by the Euler-Maruyama method implemented in libSDE [11].

Stochastic simulation algorithm (SSA)

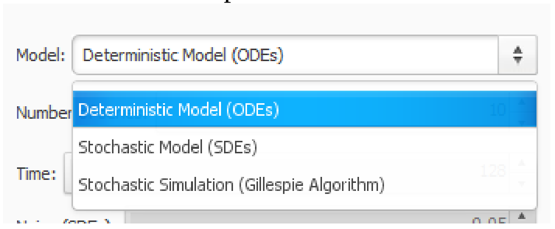
The Gillespie stochastic algorithm is used to simulate the stochastic processes [12] which is an algorithm for modeling the kinetics of a set of coupled chemical reactions, taking into account stochastic effects from low copy numbers of the chemical species.

In Gillespie's approach, chemical reaction kinetics are modelled as a markov process in which reactions occur at specific instants of time defining intervals that are Poisson-distributed, with a mean reaction time interval that is recomputed after each chemical reaction occurs. For each chemical reaction interval, a specific chemical reaction occurs, randomly selected from the set of all possible reactions with a weight given by the individual reaction rates [13].

NetLand made use of the Java implementation of the Gillespie algorithm in Dizzy [12]. This implementation uses the "direct method" variant of Gillespie's algorithm.

Usage

1. Select a simulator. The simulators are under the "**Trajectory**" panel of the "**Simulation**" menu. Users can click the "**Generate trajectories**" button at the main panel to invoke the function as well. The three simulators are provided as below.



Model: Deterministic Model (ODEs)

Number
Time:
Deterministic Model (ODEs)
Stochastic Model (SDEs)
Stochastic Simulation (Gillespie Algorithm)

2. Run a batch simulation. NetLand supports batch simulations by setting the number of time series.
3. Set the simulation time. The item "**Time**" is the elapsed time of each simulation with the same time unit as the parameters and values of species.
4. Set the initial values of species. There are two choices for setting of the initial values of genes, i.e. fixed or random. The fixed initial values are defined in the species panel. For the random initial settings, users can define the range of initial values, and then random values will be generated and assigned to genes.
5. Save the simulation results. The simulated time-course data including the dynamic model can be saved when performing the simulation if users set the saving path.

6. Start and stop a simulation. Press the "**Submit**" button to start a simulation. Press the "**Cancel**" button to stop the simulation.

Parsing the output

The simulated data are shown in right panel. To visualize the trajectories, the NetLand application provides the following features (Figure 8):

1. Move the graph with the mouse.
2. Zoom in or Zoom out with the mouse wheel.
3. Reset the view.
4. Edit the scale of the axes.
5. Save the graph to PNG.
6. Show the time-series data.
7. Show/hide the legend.
8. Select one or multiple lines in the graph by clicking the lines or the labels in the legend.
9. Select points on the trajectories. First, select a trajectory. Then hover over it. Right click will cancel the selection.

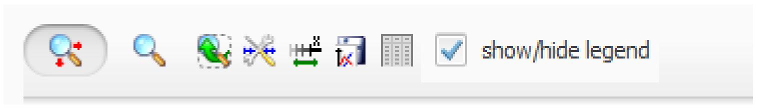
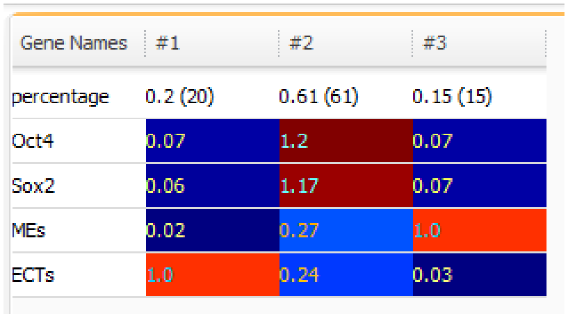


Figure 8. Functions to control the graph.

Users can also retrieve the steady states of multiple trajectories by clicking the "**Analyze result**" button. It will first analyze the convergence of each trajectory. Then the proportion and gene expression values of each steady state are shown in a table (Figure 9).



Gene Names	#1	#2	#3
percentage	0.2 (20)	0.61 (61)	0.15 (15)
Oct4	0.07	1.2	0.07
Sox2	0.06	1.17	0.07
MEs	0.02	0.27	1.0
ECTs	1.0	0.24	0.03

Figure 9. The calculated steady states.

Constructing a probabilistic landscape

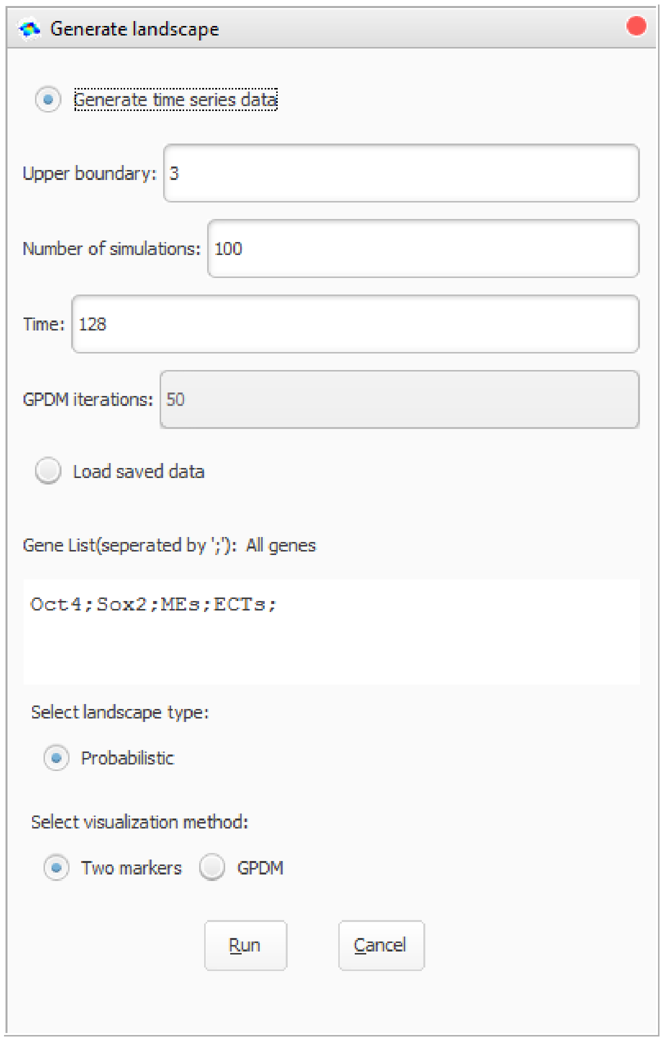
Probabilistic landscapes

For a comprehensive understanding of the determination of cellular phenotypes (e.g. cell fate decision in differentiation and reprogramming), the Waddington's epigenetic landscape has been quantified to analyze and visualize the global dynamics of gene regulatory networks [7, 14, 15]. Based on the essential idea of Waddington's epigenetic landscape, changes of phenotypes are represented by transitions overcoming energy barriers between different attractors. Thus the quantified epigenetic landscape is the stage on which the play of cell fate decisions and cell type maturation is choreographed according to physical theories [16].

The quantitative modeling and visualization of Waddington's epigenetic landscape is implemented in NetLand. We adopted the self-consistent mean field approximation method in [15] to construct the probabilistic landscape, based on the notion that the probability of gene expression states determines the stability [14, 15]. It is assumed that the probabilities of cell states (vectors of gene expression levels) are independent and follow the Gaussian distribution.

To visualize the landscape of a network with more than 2 genes, NetLand provides two options. The first option is to select two marker genes whose expression levels are the x-axis and y-axis in the landscape. However, the two marker genes may not represent the global dynamics in many cases. To avoid the loss of information, NetLand offers the second option as follows. A probabilistic dimension reduction method, Gaussian process dynamical model (GPDM) is employed to project high dimensional gene expression time-series data to a low dimensional latent space with a dynamical model [10, 17, 18]. In this case, the x-axis and y-axis are two constructed components, representing the dynamics of the whole network. For both options, the z-axis is the quasi-potential U (Figure 10).

NOTE: NetLand cannot construct the landscape of a system without any stable states. The steady state is a situation in which all state variables are constant in spite of ongoing processes that strive to change them. For example, a system with its variables exponentially increasing can never reach a steady state. If NetLand detects an unstable system, a warning message will pop up. Then users may consider modify the model in order to reach a stable system. And the current version of NetLand cannot construct the landscape of a system with stable oscillations.


Generate landscape

[✓ Generate time series data]

Upper boundary: 3

Number of simulations: 100

Time: 128

GPDM iterations: 50

[Load saved data]

Gene List(seperated by ';'): All genes

```
Oct4;Sox2;MEs;ECTs;
```

Select landscape type:

[✓ Probabilistic]

Select visualization method:

[✓ Two markers] [GPDM]

Figure 10. The panel for the construction of landscape.

Usage

1. Start the simulation. Select the "**Landscape**" from "**Simulation**" menu or the function button at the front panel.
2. Generate time-series data or load pre-computed data. The parameters are sensitive to the specific network's dynamics and defined as follows:
   a) "Upper boundary". It defines the maximum gene expression value of initial states. Thus the range of the random initials is defined within [0, UpperBoundary]. This value varies in different models. It is usually set as the maximum expression values.
   b) "Number of simulations". It denotes the number of trajectories to be generated. In a multi-attractor system, the discovery of attractors depends on whether the simulated data can cover all the steady states. Besides, more simulations are required for a large scale system with a big upper boundary.

c) “Time”. It defines the simulation time for each trajectory. For networks with a large number of nodes, normally more iterations and executive time are required to get their stable states.
d) “GPDM iterations”. It is the number of iterations for the training of GPDM.

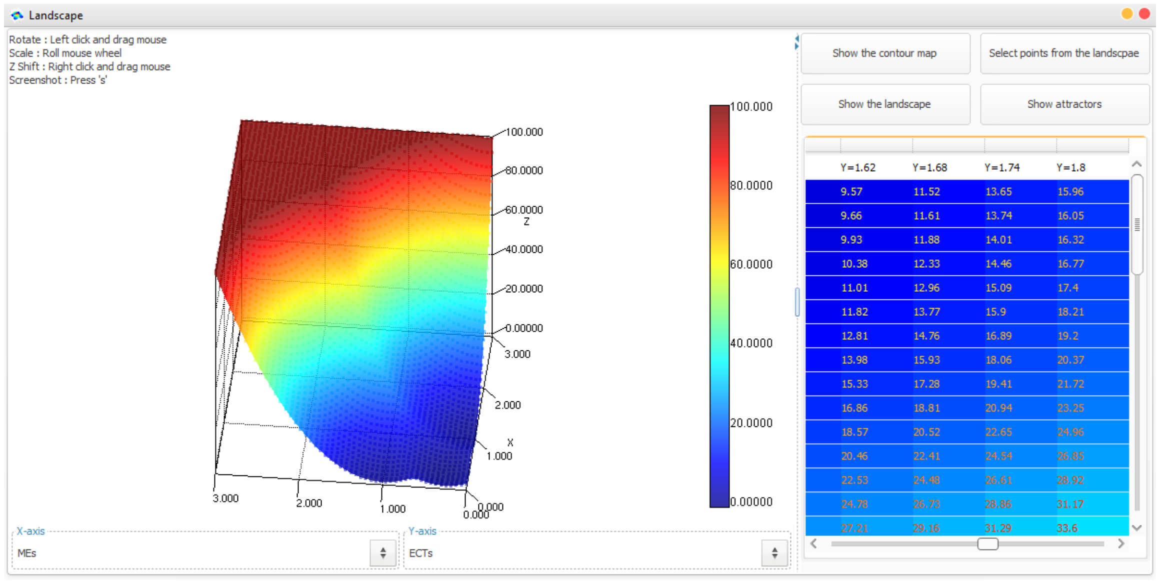
Users only need to generate the time-series data once. The pre-computed data will be stored in the memory. But if the application is closed, these data will be lost.

3. Save the constructed landscape. The landscape and the simulated time-course data including the dynamic model can be saved when performing the simulation if users set the saving path.
4. Select the visualization method. Two methods are provided, i.e. mapping to two marker genes and mapping to a 2D latent space using GPDM.
5. Start and stop a simulation. Press the “**Run**” button to start a simulation. Press the “**Cancel**” button to stop the simulation

Parsing the output

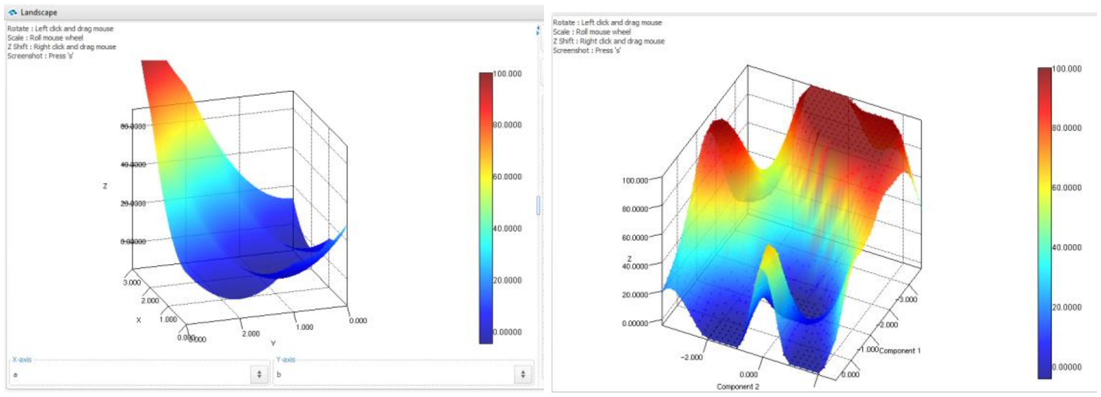
Once the simulation completes, the landscape will be shown in a pop-up window (Figure 11). Figure 12a shows an example of the constructed probabilistic landscapes using the two mapping methods. Users can project the landscape to any two genes by selecting the genes.

The detailed elevations, coordinates on the landscape and gene expression values can be accessed by double clicking the cells in the table at the bottom-right corner (Figure 12c). Users can plot the contour map or show the points on the landscape using the function buttons at the top-right panel (Figure 12b). The attractors can be retrieved by “**Show attractors**” function.

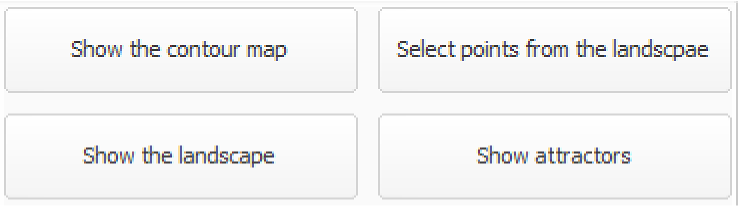


Y=1.62	Y=1.68	Y=1.74	Y=1.8
9.57	11.52	13.65	15.96
9.66	11.61	13.74	16.05
9.93	11.88	14.01	16.32
10.38	12.33	14.46	16.77
11.01	12.96	15.09	17.4
11.82	13.77	15.9	18.21
12.81	14.76	16.89	19.2
13.98	15.93	18.06	20.37
15.33	17.28	19.41	21.72
16.86	18.81	20.94	23.25
18.57	20.52	22.65	24.96
20.46	22.41	24.54	26.85
22.53	24.48	26.61	28.92
24.78	26.73	28.86	31.17
27.21	29.16	31.29	33.6

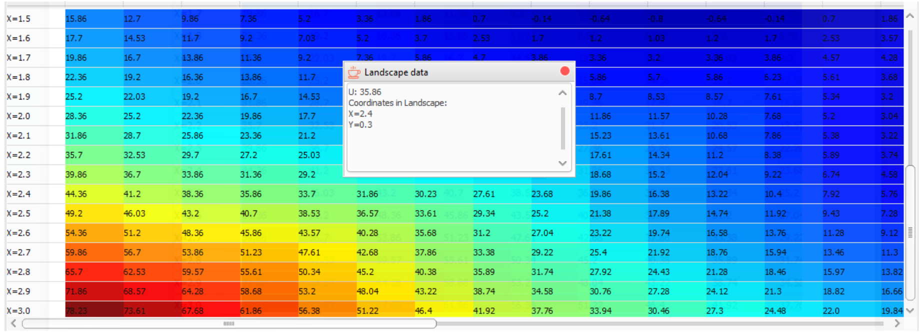
Figure 11. The result panel of the constructed landscape.



(a)



(b)



X=1.5	15.86	12.7	9.86	7.36	5.2	3.36	1.86	0.7	-0.14	-0.64	-0.8	-0.64	-0.14	0.7	1.86
X=1.6	17.7	14.53	11.7	9.2	7.03	5.2	3.7	2.53	1.7	1.2	1.03	1.2	1.7	2.53	3.57
X=1.7	19.86	16.7	13.86	11.36	9.2	7.36	5.86	4.7	3.86	3.36	3.2	3.36	3.86	4.57	4.28
X=1.8	22.36	19.2	16.36	13.86	11.7	[illegible]	[illegible]	[illegible]	[illegible]	5.86	5.7	5.86	6.23	5.61	3.68
X=1.9	25.2	22.03	19.2	16.7	14.53	[illegible]	[illegible]	[illegible]	[illegible]	8.7	8.53	8.57	7.61	5.34	3.2
X=2.0	28.36	25.2	22.36	19.86	17.7	[illegible]	[illegible]	[illegible]	[illegible]	11.86	11.57	10.28	7.68	5.2	3.04
X=2.1	31.86	28.7	25.86	23.36	21.2	[illegible]	[illegible]	[illegible]	[illegible]	15.23	13.61	10.68	7.86	5.38	3.22
X=2.2	35.7	32.53	29.7	27.2	25.03	[illegible]	[illegible]	[illegible]	[illegible]	17.61	14.34	11.2	8.38	5.89	3.74
X=2.3	39.86	36.7	33.86	31.36	29.2	[illegible]	[illegible]	[illegible]	[illegible]	18.68	15.2	12.04	9.22	6.74	4.58
X=2.4	44.36	41.2	38.36	35.86	33.7	31.86	30.23	27.61	23.68	19.86	16.38	13.22	10.4	7.92	5.76
X=2.5	49.2	46.03	43.2	40.7	38.53	36.57	33.61	29.34	25.2	21.38	17.89	14.74	11.92	9.43	7.28
X=2.6	54.36	51.2	48.36	45.86	43.57	40.28	35.68	31.2	27.04	23.22	19.74	16.58	13.76	11.28	9.12
X=2.7	59.86	56.7	53.86	51.23	47.61	42.68	37.86	33.38	29.22	25.4	21.92	18.76	15.94	13.46	11.3
X=2.8	65.7	62.53	59.57	55.61	50.34	45.2	40.38	35.89	31.74	27.92	24.43	21.28	18.46	15.97	13.82
X=2.9	71.86	68.57	64.28	58.68	53.2	48.04	43.22	38.74	34.58	30.76	27.28	24.12	21.3	18.82	16.66
X=3.0	78.23	73.61	67.68	61.86	56.38	51.22	46.4	41.92	37.76	33.94	30.46	27.3	24.48	22.0	19.84

Landscape data

U: 35.86
Coordinates in Landscape:
X=2.4
Y=0.3

(c)

Figure 12. (a) The constructed landscapes. (b) Function buttons (c) The table of the grid data in the landscape colored by the value of quasi-potentials.

Running time assessment and memory test

Running time assessment

To test the performance of NetLand, we generated a set of random networks with 2 to 200 genes. The experiments were implemented on a PC of 8-core CPU 3.60 GHz Intel Xeon Win 7 Systems using the default memory settings of JVM.

First, we compared the running time and memory usage of different simulators. The simulation time was set 128. For each network, one trajectory was generated with random initial values. Figure 13 shows the time consumed.

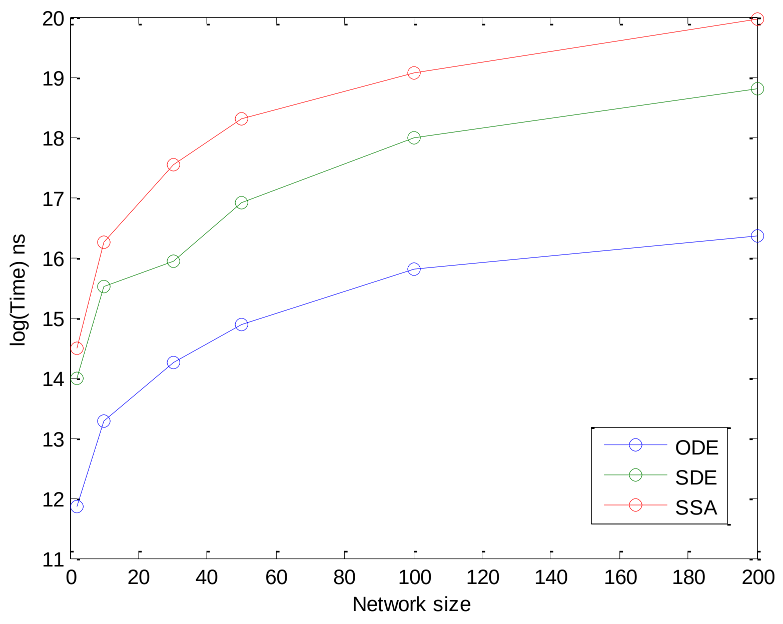


Figure 13. Running time with different simulators.

Then, we constructed the landscape of each test network. The iteration value and time duration were set 100 and 200 respectively. The iterations for training GPDM was 50. The result is shown in Figure 14.

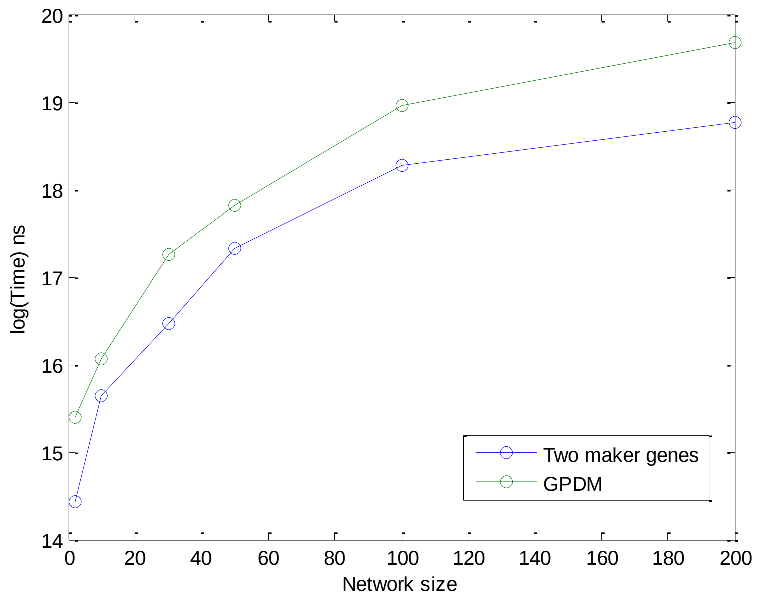
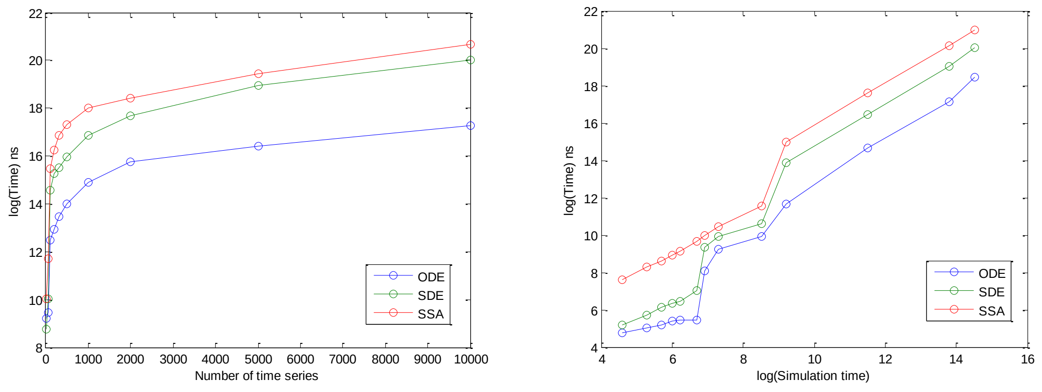


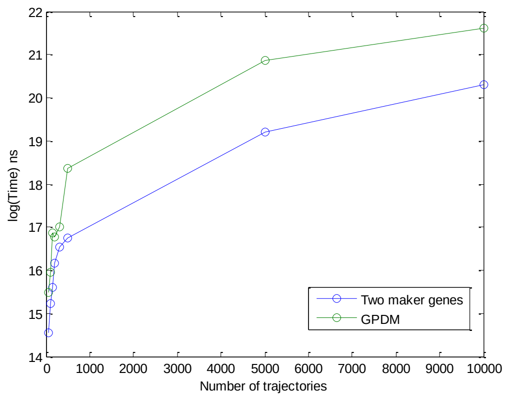
Figure 14. Running time for the construction of a landscape.

In addition, we tested the execution time for generating multiple trajectories with different simulation time. The 'seesaw' model was chosen as the template. Similarly, we compared the execution time for generating landscapes using different iterations and simulation time. The 'seesaw' model was chosen as the template. The results are shown in Figure 15-16.

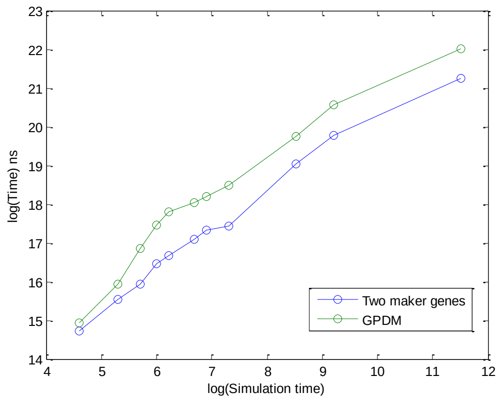


(a) (b)

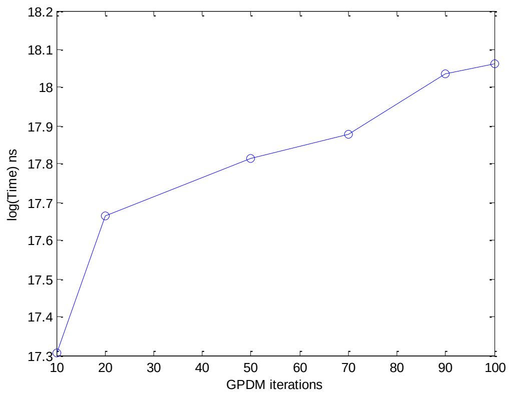
Figure 15. (a) Running time for generating different numbers of trajectories. The simulation time was set 128. (b) Running time for generating different lengths of trajectories.



(a)



(b)



(c)

Figure 16. (a) Running time for constructing landscapes using different numbers of trajectories. The simulation time was set 128. (b) Running time for the construction landscapes using different lengths of trajectories. The number of trajectories was set 100. GPDM was trained for 50 iterations. (c) Running time for the construction landscapes using different iterations of GPDM.

Memory consumption test

We tested the memory consumption of NetLand on the same set of random networks using JProfiler. The experiments were implemented on a PC of 8-core CPU 3.60 GHz Intel Xeon Win 7 Systems with 2GB MaxHeapSize of JVM.

First, we compared the memory usage of different simulators. The simulation time was set 128. For each network, one trajectory was generated with random initial values. Figure 17 shows the memory consumed.

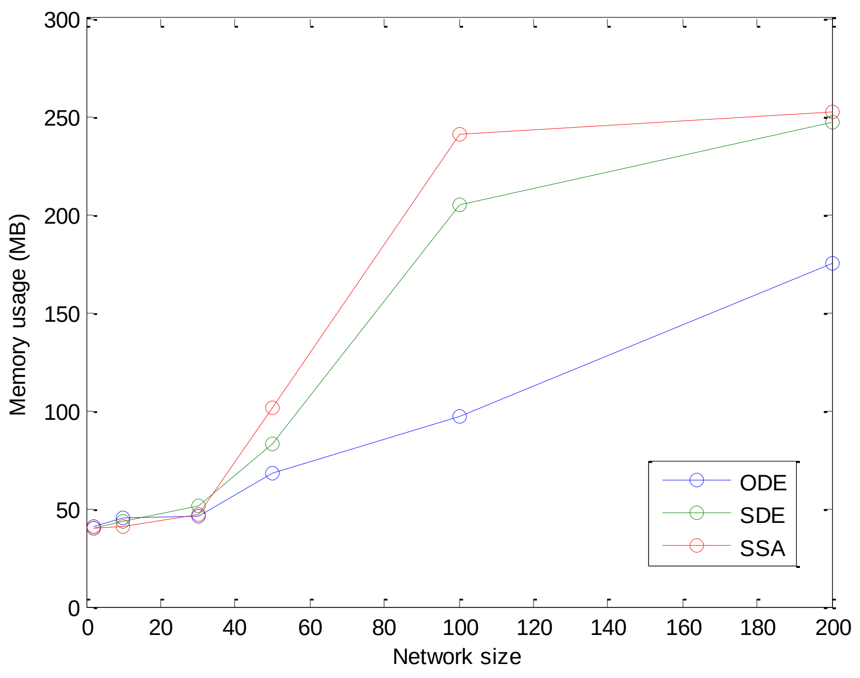


Figure 17. Memory usage with different simulators.

Then, we constructed the landscape of each test network. The iteration value and time duration were set 100 and 200 respectively. The iterations for training GPDM was 50. The result is shown in Figure 18.

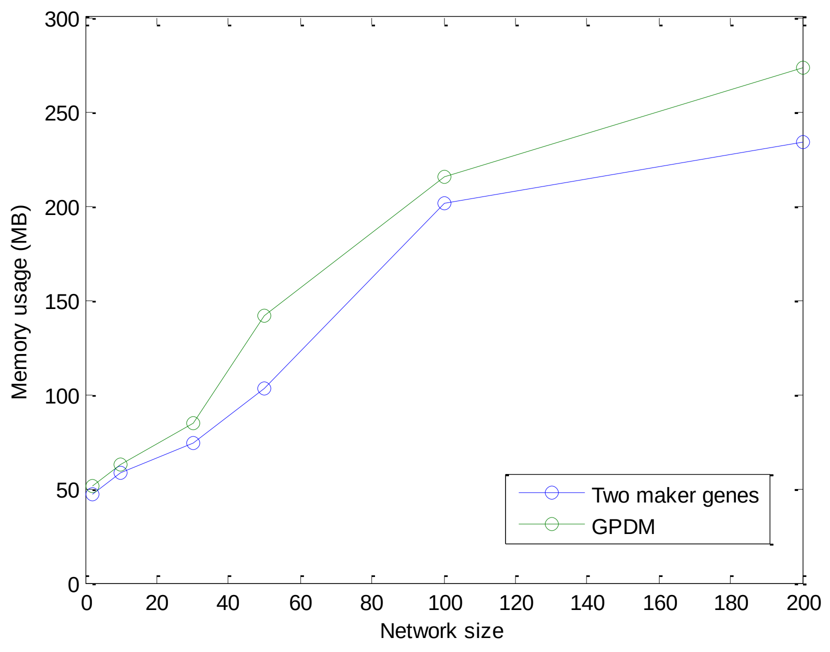
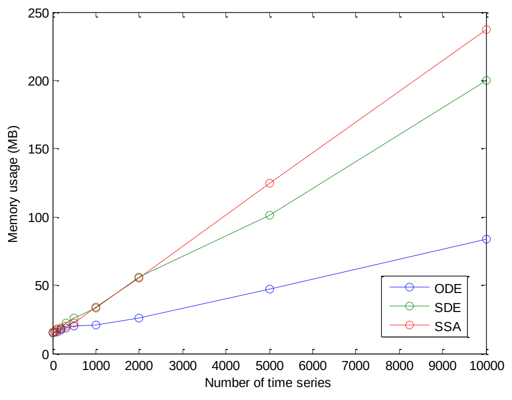
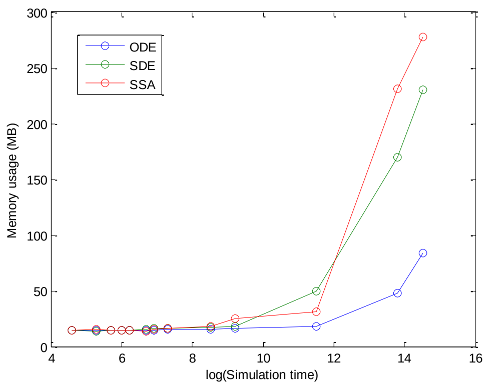


Figure 18. Memory usage for the construction of a landscape.

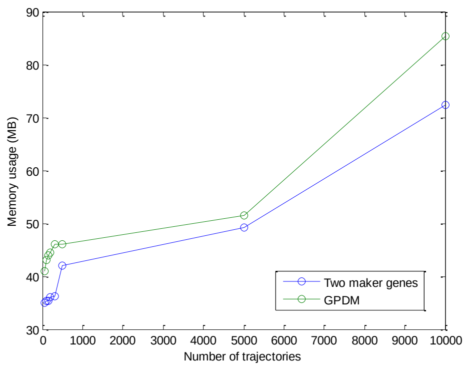
In addition, we tested the memory consumption for generating multiple trajectories with different simulation time. The 'seesaw' model was chosen as the template. Similarly, we compared the memory consumption for generating landscapes using different iterations and simulation time. The 'seesaw' model was chosen as the template. The results are shown in Figure 19-20.

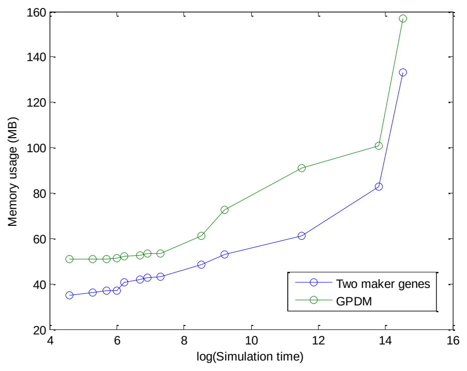




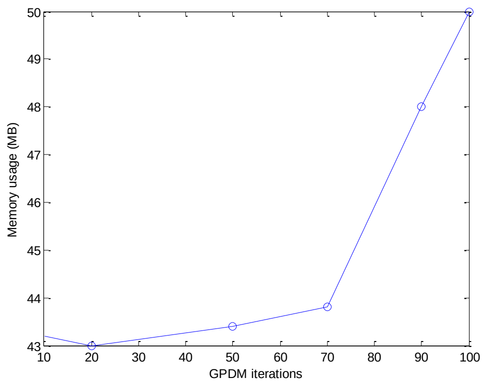
(a) (b)

Figure 19. (a) Memory usage for generating different numbers of trajectories. The simulation time was set 128. (b) Memory usage for generating different lengths of trajectories.





(a) (b)



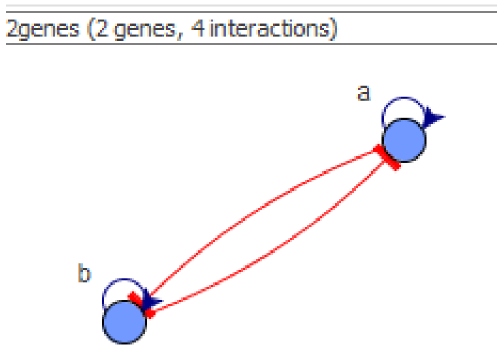
(c)

Figure 20. (a) Memory usage for constructing landscapes using different numbers of trajectories. The simulation time was set 128. (b) Memory usage for the construction landscapes using different lengths of trajectories. The number of trajectories was set 100. GPDM was trained for 50 iterations. (c) Memory usage for the construction landscapes using different iterations of GPDM.

Case studies

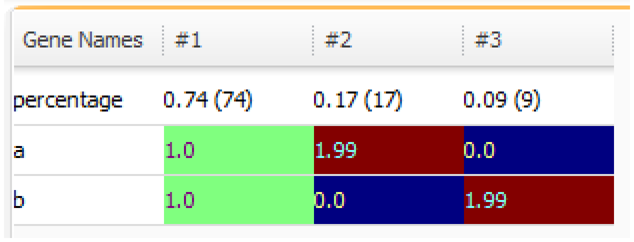
Example 1: A two-gene network with mutual inhibitions and self-loops

The two-gene regulatory network with mutual inhibitions and positive autoregulations is published in [7]. The probabilistic landscapes visualized by mapping to two marker genes and a 2D latent space are shown below.



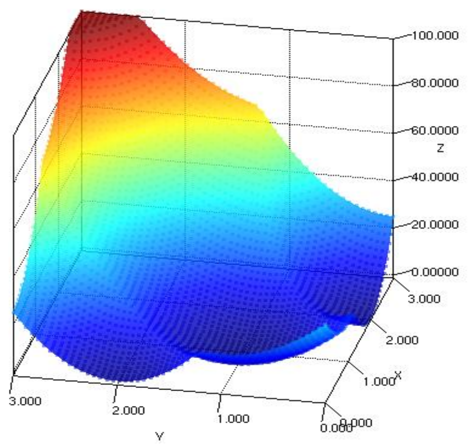
Parameters: UpperBoundary=3, number of simulations=100, time=128, GPDM iterations=50

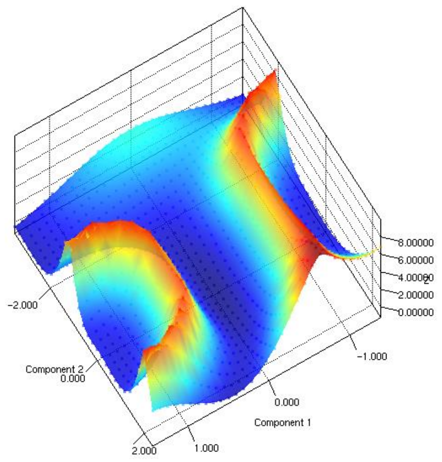
Three Attractors:



Gene Names	#1	#2	#3
percentage	0.74 (74)	0.17 (17)	0.09 (9)
a	1.0	1.99	0.0
b	1.0	0.0	1.99

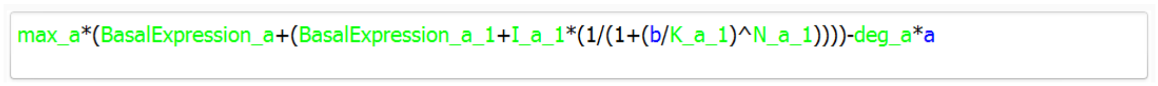
Landscapes:



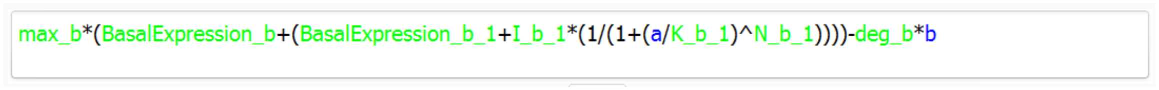


Then we removed the self-loops from the network. Two attractors are identified.

The modified reactions:

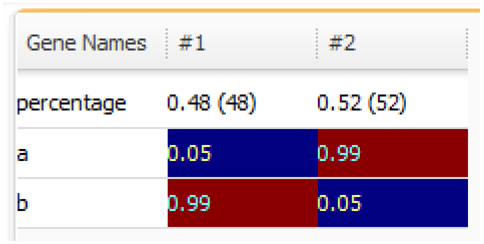


```
max_a*(BasalExpression_a+(BasalExpression_a_1+I_a_1*(1/(1+(b/K_a_1)^N_a_1))))-deg_a*a
```



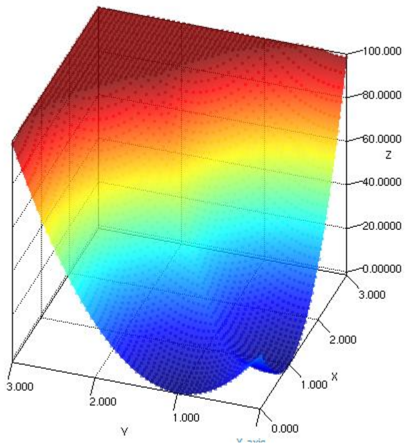
```
max_b*(BasalExpression_b+(BasalExpression_b_1+I_b_1*(1/(1+(a/K_b_1)^N_b_1))))-deg_b*b
```

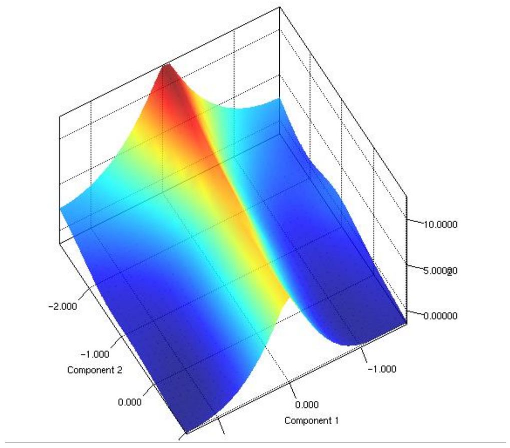
Attractors:



Gene Names	#1	#2
percentage	0.48 (48)	0.52 (52)
a	0.05	0.99
b	0.99	0.05

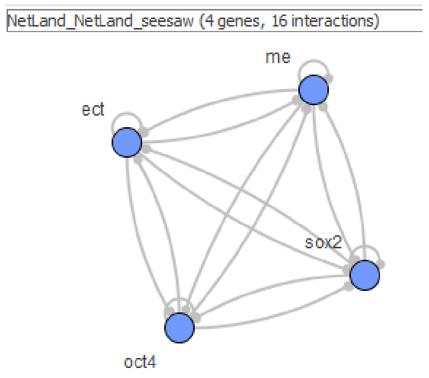
Landscapes:

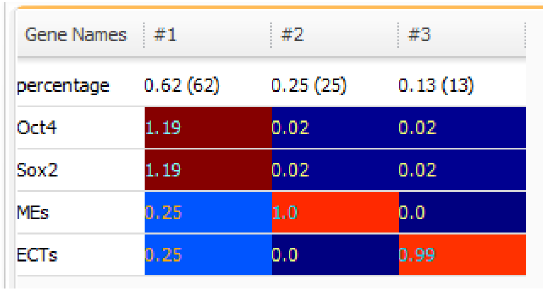




Example 2: The seesaw model with 3 attractors

To study the stem cell differentiation and reprogramming process, a computational model, ‘seesaw model’ with 4 genes (Figure 21) was constructed in [19]. NetLand was applied to simulate and analyze the model. First, the regulatory relations between genes were extracted into a TSV format file. After loading it to NetLand, the differential equations including the parameters were modified according to the paper. Once the dynamic model was settled, further analysis and simulation can be performed. Figure 22 shows the simulation result and landscapes constructed by NetLand.





Gene Names	#1	#2	#3
percentage	0.62 (62)	0.25 (25)	0.13 (13)
Oct4	1.19	0.02	0.02
Sox2	1.19	0.02	0.02
MEs	0.25	1.0	0.0
ECTs	0.25	0.0	0.99

Figure 21. A seesaw model used to simulate stem cell reprogramming. It contains three attractors.

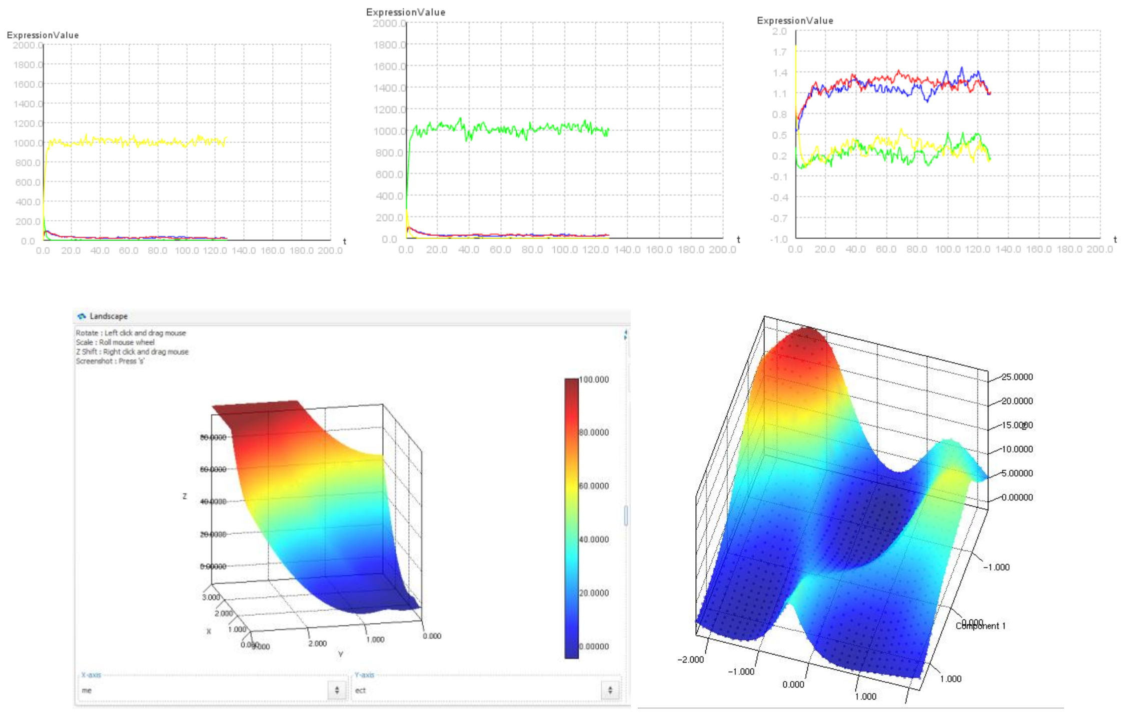
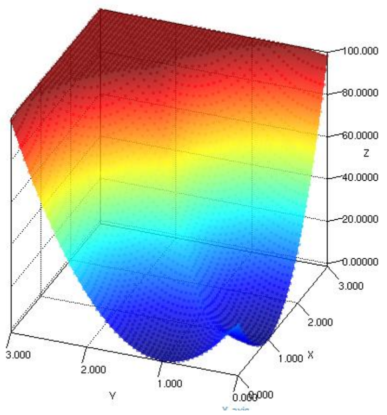


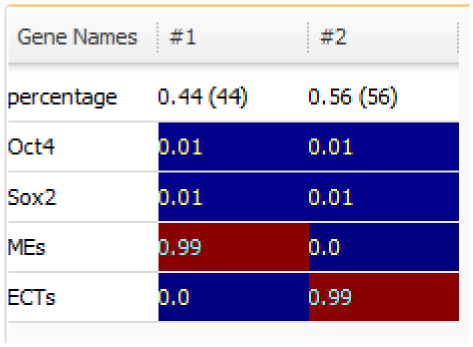
Figure 22. Simulation results of the seesaw model.

A series of parameter sensitivity analysis was performed.

- Set parameter KM=0

When the value of Klf4 and MYC (KM) is set to be 0, the cells cannot maintain their pluripotency to start differentiation. There will be two attractors left in the landscape representing ME and ECT states.





Gene Names	#1	#2
percentage	0.44 (44)	0.56 (56)
Oct4	0.01	0.01
Sox2	0.01	0.01
MEs	0.99	0.0
ECTs	0.0	0.99

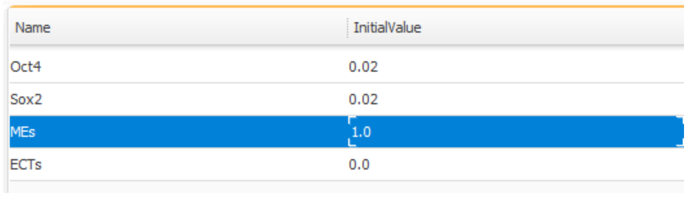
- Set parameter KM=0.4

Besides the above two attractors, there will be an additional steady state representing stem cell state (Figure 22).

- Reprogramming experiment with different cocktails
  - +Yamanaka's factors (+Sox2+Oct4)

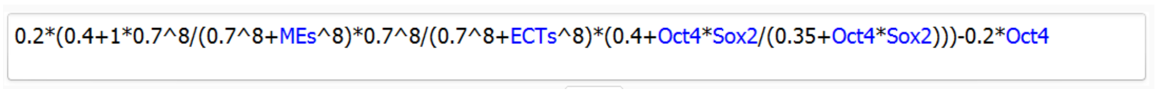
With the replenishment of Yamanaka's factors (Oct4, Sox2, Klf4, Myc), the somatic cells will be reprogrammed into iPSCs (similar to stem cells). There is one attractor left, i.e. stem cell state. To check whether all somatic cells goes into stem cells, we set the initial state as ME state and made use of the analyze function in simulation. It shows that all ME cells turned into stem cell state.

a) Set the initial state as ME state.



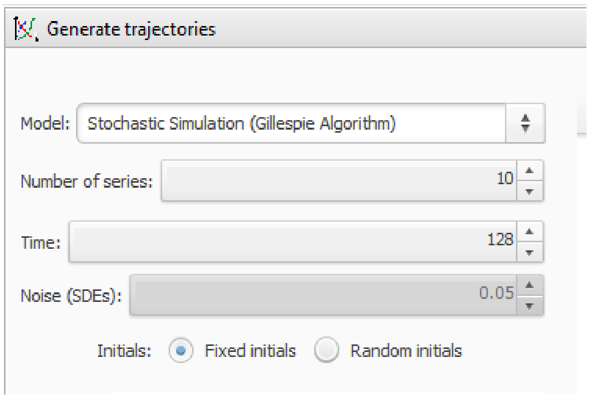
Name	InitialValue
Oct4	0.02
Sox2	0.02
MEs	1.0
ECTs	0.0

b) Increase the base production rate of Sox2 and Oct4.



```
0.2*(0.4+1*0.7^8/(0.7^8+MEs^8)*0.7^8/(0.7^8+ECTs^8)*(0.4+Oct4*Sox2/(0.35+Oct4*Sox2)))-0.2*Oct4
```

c) Simulate 10 trajectories using Gillespie algorithm.


Generate trajectories

Model: Stochastic Simulation (Gillespie Algorithm)

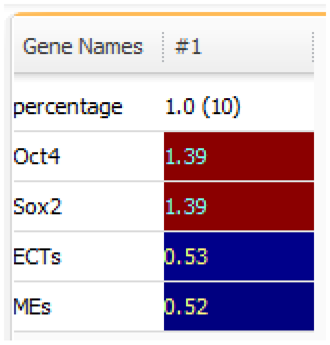
Number of series: 10

Time: 128

Noise (SDEs): 0.05

Initials: [✓ Fixed initials] Random initials

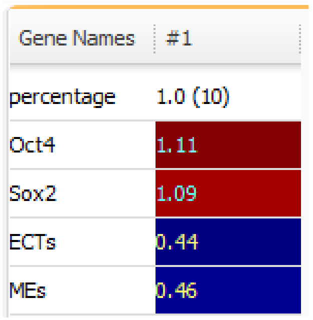
d) Get attractors.



Gene Names	#1
percentage	1.0 (10)
Oct4	1.39
Sox2	1.39
ECTs	0.53
MEs	0.52

- +Lineage specifiers (+ME+ECT)

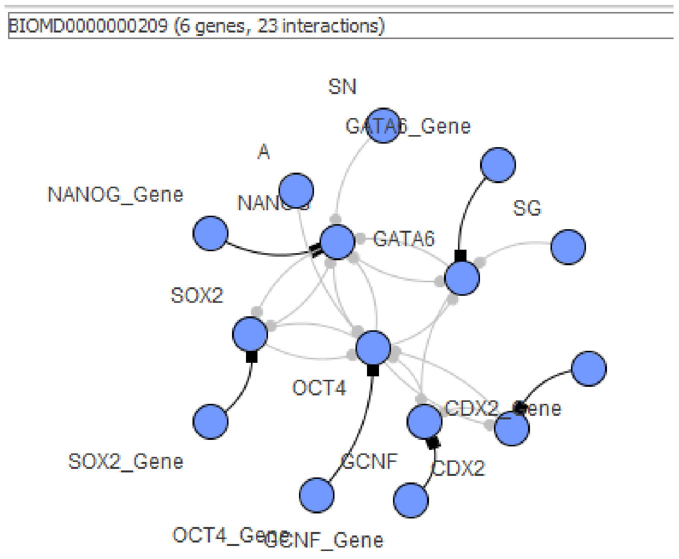
In this experiment, ME and ECT work as the substitutions of Oct4 and Sox2. Then follow the steps in the above experiment. The somatic cells were all reprogrammed into iPSCs (similar to stem cells).



Gene Names	#1
percentage	1.0 (10)
Oct4	1.11
Sox2	1.09
ECTs	0.44
MEs	0.46

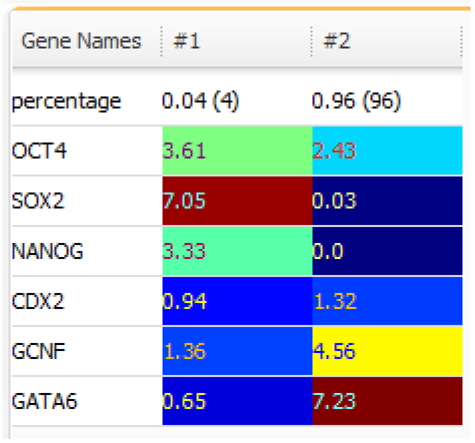
Example 3: A dynamic model in embryonic stem cell

A dynamical model of lineage determination based upon a minimal circuit which contains the Oct4/Sox2/Nanog core as well its interaction with a few other key genes is discussed in [20] (BIOMD0000000209.xml from BioModels).



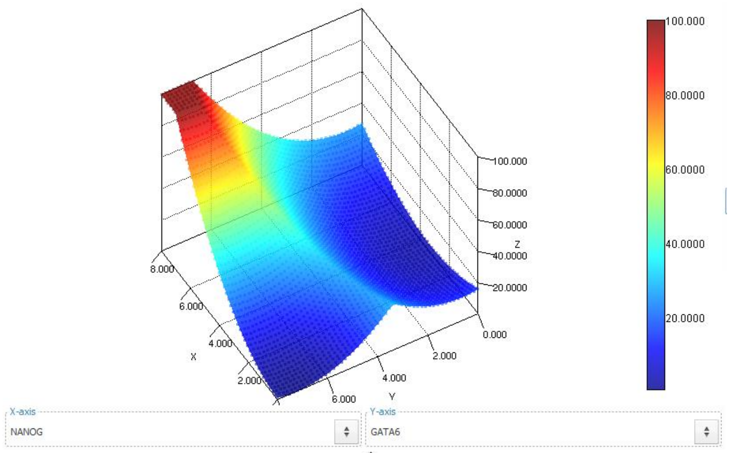
Parameters: UpperBoundary=8, number of simulations=100, time=400, GPDM iterations=50

Attractors:



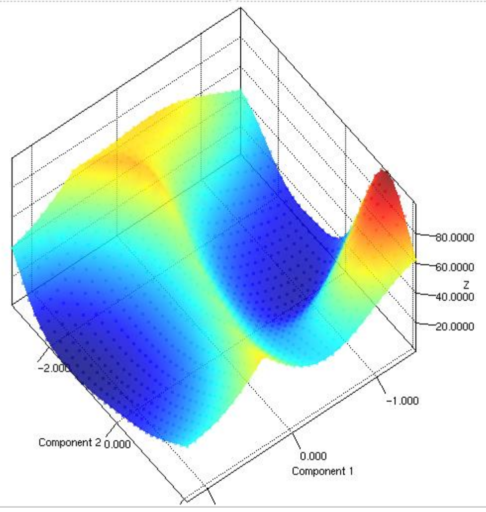
Gene Names	#1	#2
percentage	0.04 (4)	0.96 (96)
OCT4	3.61	2.43
SOX2	7.05	0.03
NANOG	3.33	0.0
CDX2	0.94	1.32
GCNF	1.36	4.56
GATA6	0.65	7.23

Landscapes:



X-axis
NANOG

Y-axis
GATA6

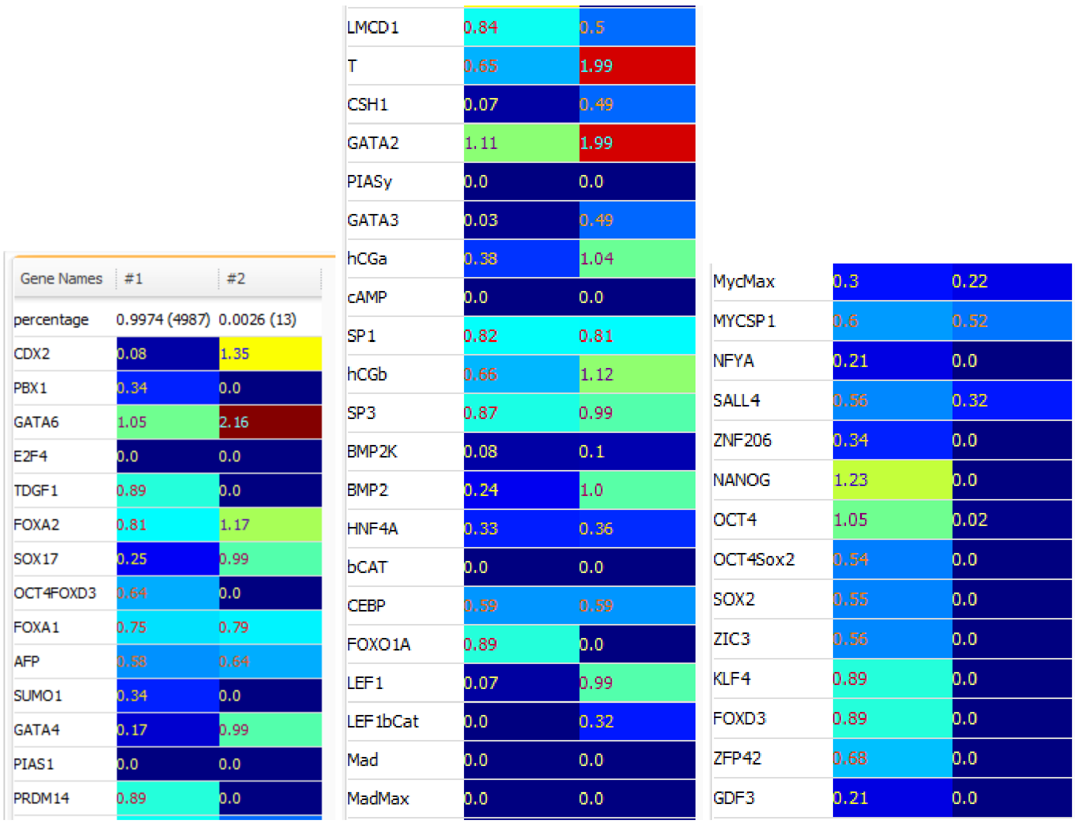


Example 4: A 52-gene network

A stem cell developmental network with 52 genes was constructed to study cell fate decision [8]. It contains marker genes for the pluripotent stem cell state and the differentiation state.

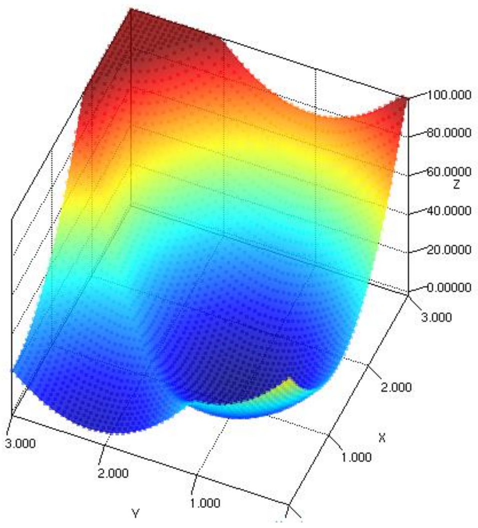
Parameters: UpperBoundary=3, number of simulations=5000, time=128, GPDM iterations=50

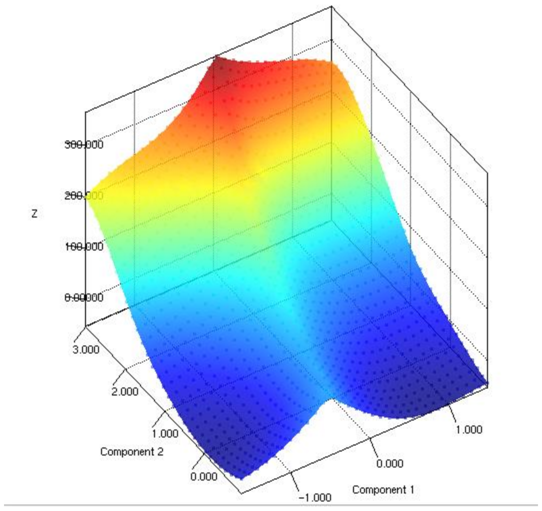
Attractors:



Gene Names	#1	#2
percentage	0.9974 (4987)	0.0026 (13)
CDX2	0.08	1.35
PBX1	0.34	0.0
GATA6	1.05	2.16
E2F4	0.0	0.0
TDGF1	0.89	0.0
FOXA2	0.81	1.17
SOX17	0.25	0.99
OCT4FOXD3	0.64	0.0
FOXA1	0.75	0.79
AFP	0.58	0.64
SUMO1	0.34	0.0
GATA4	0.17	0.99
PIAS1	0.0	0.0
PRDM14	0.89	0.0
LMCD1	0.84	0.5
T	0.65	1.99
CSH1	0.07	0.49
GATA2	1.11	1.99
PIASy	0.0	0.0
GATA3	0.03	0.49
hCGa	0.38	1.04
cAMP	0.0	0.0
SP1	0.82	0.81
hCGb	0.66	1.12
SP3	0.87	0.99
BMP2K	0.08	0.1
BMP2	0.24	1.0
HNF4A	0.33	0.36
bCAT	0.0	0.0
CEBP	0.59	0.59
FOXO1A	0.89	0.0
LEF1	0.07	0.99
LEF1bCat	0.0	0.32
Mad	0.0	0.0
MadMax	0.0	0.0
MycMax	0.3	0.22
MYCSP1	0.6	0.52
NFYA	0.21	0.0
SALL4	0.56	0.32
ZNF206	0.34	0.0
NANOG	1.23	0.0
OCT4	1.05	0.02
OCT4Sox2	0.54	0.0
SOX2	0.55	0.0
ZIC3	0.56	0.0
KLF4	0.89	0.0
FOXD3	0.89	0.0
ZFP42	0.68	0.0
GDF3	0.21	0.0

Landscapes:


100.000
80.0000
60.0000
Z
40.0000
20.0000
0.00000
3.000
2.000
X
1.000
3.000
2.000
1.000
Y


Z
300.000
200.000
100.000
0.00000
3.000
2.000
1.000
Component 2
0.000
-1.000
0.000
1.000
Component 1

License

Contact information and bug reporting

The authors have carried out extensive testing and debugging of the program, which should be generally stable and functional if the instructions in this manual are followed. It is our hope that, with the GUI, users without much computational sophistication can also apply our algorithm to analyze their network smoothly. However, since the authors are not professional programmers, NetLand does not behave or look like a commercial software. Please send your suggestions, comments, reports of bugs and errors to *Jie Zheng* at zhengjie@ntu.edu.sg. Thank you!

References

1. Waddington, C.H., *The strategy of the genes; a discussion of some aspects of theoretical biology* 1957, London,: Allen & Unwin.

2. Kauffman, S.A., *Metabolic stability and epigenesis in randomly constructed genetic nets.* Journal of theoretical biology, 1969. **22**(3): p. 437-67.

3. Thom, R., *An inventory of Waddingtonian concepts.* Theorical Biology: Epigenetic and Evolutionary Order from Complex Systems, 1989: p. 1-7.

4. Sasai, M. and P.G. Wolynes, *Stochastic gene expression as a many-body problem.* Proc Natl Acad Sci U S A, 2003. **100**(5): p. 2374-9.

5. Huang, S., *The molecular and mathematical basis of Waddington's epigenetic landscape: a framework for post-Darwinian biology?* BioEssays : news and reviews in molecular, cellular and developmental biology, 2012. **34**(2): p. 149-57.

6. Wang, J., L. Xu, and E. Wang, *Potential landscape and flux framework of nonequilibrium networks: robustness, dissipation, and coherence of biochemical oscillations.* Proc Natl Acad Sci U S A, 2008. **105**(34): p. 12271-6.

7. Bhattacharya, S., Q. Zhang, and M. Andersen, *A deterministic map of Waddington's epigenetic landscape for cell fate specification.* BMC systems biology, 2011. **5**: p. 85.

8. Li, C. and J. Wang, *Quantifying cell fate decisions for differentiation and reprogramming of a human stem cell network: landscape and biological paths.* PLoS Comput Biol, 2013. **9**(8): p. e1003165.

9. Lawrence, N., *Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models.* J. Mach. Learn. Res., 2005. **6**: p. 1783-1816.

10. Wang, J.M., D.J. Fleet, and A. Hertzmann, *Gaussian process dynamical models for human motion.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2008. **30**(2): p. 283-298.

11. Schaffter, T., D. Marbach, and D. Floreano, *GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods.* Bioinformatics, 2011. **27**(16): p. 2263-2270.

12. Ramsey, S., D. Orrell, and H. Bolouri, *Dizzy: stochastic simulation of large-scale genetic regulatory networks.* Journal of bioinformatics and computational biology, 2005. **3**(02): p. 415-436.

13. Gillespie, D.T., *A general method for numerically simulating the stochastic time evolution of coupled chemical reactions.* Journal of computational physics, 1976. **22**(4): p. 403-434.

14. Zhou, J.X., et al., *Quasi-potential landscape in complex multi-stable systems.* Journal of The Royal Society Interface, 2012. **9**(77): p. 3539-3553.

15. Li, C. and J. Wang, *Quantifying cell fate decisions for differentiation and reprogramming of a human stem cell network: landscape and biological paths.* PLoS computational biology, 2013. **9**(8): p. e1003165.

16. Huang, S., I. Ernberg, and S. Kauffman. *Cancer attractors: a systems view of tumors from a gene network dynamics and developmental perspective*. in *Seminars in cell & developmental biology*. 2009. Elsevier.

17. Lawrence, N., *Probabilistic non-linear principal component analysis with Gaussian process latent variable models.* The Journal of Machine Learning Research, 2005. **6**: p. 1783-1816.

18. Gopalan, N. and T. IAS, *Gaussian Process Latent Variable Models for Dimensionality Reduction and Time Series Modeling.*

19. Shu, J., et al., *Induction of pluripotency in mouse somatic cells with lineage specifiers.* Cell, 2013. **153**(5): p. 963-975.

20. Chickarmane, V. and C. Peterson, *A computational model for understanding stem cell, trophectoderm and endoderm lineage determination.* PLoS One, 2008. **3**(10): p. e3478.