User Manual

TMELand: An end-to-end pipeline for quantification and visualization of Waddington's epigenetic landscape based on gene regulatory network

Lin Zhu[1,†], Xin Kang[2,†], Chunhe Li [2,3,4,*] and Jie Zheng[1,5,*]

[1]School of Information Science and Technology, ShanghaiTech University, Shanghai, 201210, China, [2]Shanghai Center for Mathematical Sciences, Fudan University, Shanghai, 200433, China, [3]School of Mathematical Sciences, Fudan University, Shanghai, 200433, China, [4]Institute of Science and Technology for Brain-Inspired Intelligence, Fudan University, Shanghai, 200433, China and [5]Shanghai Engineering Research Center of Intelligent Vision and Imaging, Shanghai, 201210, China

[†] represents authors contributed equally to this work.
Availability: https://github.com/JieZheng-ShanghaiTech/TMELand
Contact: zhengjie@shanghaitech.edu.cn, chunheli@fudan.edu.cn

Contents

1. Introduction

1.1. Waddington's epigenetic landscape and extension

Waddington's epigenetic landscape was proposed by C. H. Waddington in 1957, which is a metaphor to describe the developmental process of cell differentiation [1]. In his theory, a marble rolling down a hill into a stable locally lowest point (i.e., attractor) represents a cell differentiating into a specific cell type, and this process is regulated by underlying gene networks. The landscape concept has been extended to study reverse differentiation processes such as stem cell reprogramming [2, 3].

1.2. Related work and motivation

There are mainly two types of approaches for landscape quantification: data-driven and model-drive. The data-driven methods include those based on Hopfield network [4] and entropy-based methods [5]. The model-driven methods take models of gene regulatory network (GRN) as input, including continuous differential equation (DE)-based models [2] and discrete Boolean network-based models [6]. The data-driven methods for landscape modeling directly infer the landscape from data, but don't model underlying GRN explicitly. Thus, compared with the model-driven methods, the data-driven methods lack interpretability to some extent. Here, we propose a pipeline that integrates data-driven GRN inference algorithms and model-driven (i.e., DE-based) methods for landscape modeling.

Recently, two software tools with GUI have been developed to model and visualize Waddington's landscape, i.e., NetLand [7] and ATLANTIS [8]. However, in addition to the static landscape, state transition paths can reflect cellular dynamics represented by the temporal evolution of gene expression levels regulated by GRNs. NetLand doesn't offer the functionality of finding transition paths. ATLANTIS extracts paths from an initial state to the terminal steady state [8], which lacks sufficient details for understanding the dynamical processes along the paths.

1.3. An overview of the TMELand

We develop an open-source, cross-platform Python application named TMELand,

based on the truncated moment equations (TME) approach for landscape construction previously developed [9]. Compared with NetLand, TMELand uses an adapted algorithm that can construct a more precise landscape. Meanwhile, compared with ATLANTIS, TMELand can provide more detailed information along the transition paths which can reveal intermediate states.

TMELand can load three main kinds of data, TSV, ODE computation model, and single-cell gene expression data. TSV text file can reflect the topology of the gene regulatory network. ODE computation model includes XPPAUT ODE and SBML two types. XPPAUT ODE refers to the .ode file, it includes a set of ode equations and some parameters. SBML is based on the XML file format, which stores computational models of biological processes. We can download SBML model files from the BioModels Database, where the BioModels Database contains rich mathematical models of biological systems. To support single-cell gene expression data, we incorporate a GRN inference algorithm to infer GRN topology automatically. After loading the model file, TMELand can visualize models, i.e., it can transform a TSV text file into a GRN topology network, parsed the ODE computation model into ODE equations, inferred topological structure of GRN from single-cell data. Then you can formulate dynamics, simulate trajectories, construct a landscape, adjust visualization-related parameters to obtain a pretty landscape, and draw paths between attractors by specify beginning and ending attractors.

TMELand is developed by python and can be used in many platforms, including Windows, Unix, and macOS. The detailed reliable packages are introduced in the 2.1 section.

2. Installation

2.1. Dependency package summary

This is a summary of the dependency package in our test environment. Our python version is 3.7.0.

Table 1. The dependency-package summary of TMELand desktop application.

Name	Version	Summary	License
Matplotlib	3.1.2	Python plotting package.	PSF
Jaal	0.1.2	Jaal is a python based interactive network visualizing tool built using Dash and Visdcc.	MIT
NumPy	1.18.1	NumPy is the fundamental package for array computing with Python.	BSD
SciPy	1.5.2	Scientific Library for Python.	BSD
python-libsbml	5.18.0	LibSBML is a library for reading, writing, and manipulating the Systems Biology Markup Language (SBML).	LGPL
SymPy	1.7.1	SymPy is an open-source Python library for symbolic computation.	BSD
pandas	1.1.5	pandas is a software library written for the Python programming language for data manipulation and analysis.	BSD
Scanpy	1.8.2	Scanpy is a scalable toolkit for analyzing single-cell gene expression data built jointly with anndata.	BSD
R	4.1.0	R is a free software environment for statistical computing and graphics.	GPL 3.0
leidenalg	0.8.8	Leiden is a general algorithm for methods of community detection in large networks.	GPL 3.0
louvain	0.7.0	louvain is a general algorithm for methods of community detection in large networks.	GPL 3.0
Pillow	7.0.0	Pillow is the friendly PIL fork by Alex Clark and Contributors. PIL is the Python Imaging Library by Fredrik Lundh and Contributors.	HPND
tkintertable	1.3.3	A pure python library for adding tables to a Tkinter application.	GPL 2.0

2.2. Installation pipeline

The pipeline is commonly used for all platforms. You can open the command line in the TMELand folder, and input the following commands. (We recommend using conda to create a new virtual environment and then install packages.)

1. Create a virtual environment and install dependency packages

```
conda create –n TMELand_env python=3.7
conda activate TMELand_env
pip install -r requirements.txt
```

2. Install R and R packages

```
conda install -c conda-forge r-base=4.1.0
conda install -c r r-mass
```

3. Install Ruby

- Linux (Ubuntu): sudo apt-get install ruby-full
- MAC OS: brew install ruby
- Windows: https://github.com/oneclick/rubyinstaller2/releases/download/RubyInstaller-3.1.0-1/rubyinstaller-3.1.0-1-x64.exe

4. Launch the TMELand

```
python ./Main.py
```

Figure 1. The pipeline of install and launch TMELand.

The following page is displayed when the TMELand is successfully installed:

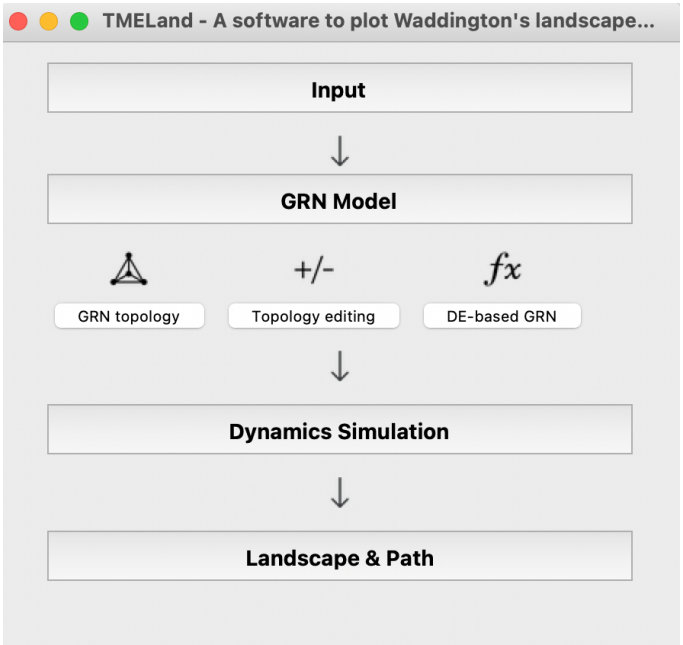

Figure 2. The main interface of the TMELand.

Users should follow this workflow to load data and do analysis, the details are described in the following sections.

3. Model definition

3.1. Loading files

TMELand can load three kinds of models, i.e., TSV and ODE computation model, and scRNA-seq. Furthermore, TMELand accepts the saved result of TMELand in self-defined TME format (JSON file). The ODE computation model contains the XPPAUT ODE format file and SBML format file. Use can choose one of four kinds of data as input.

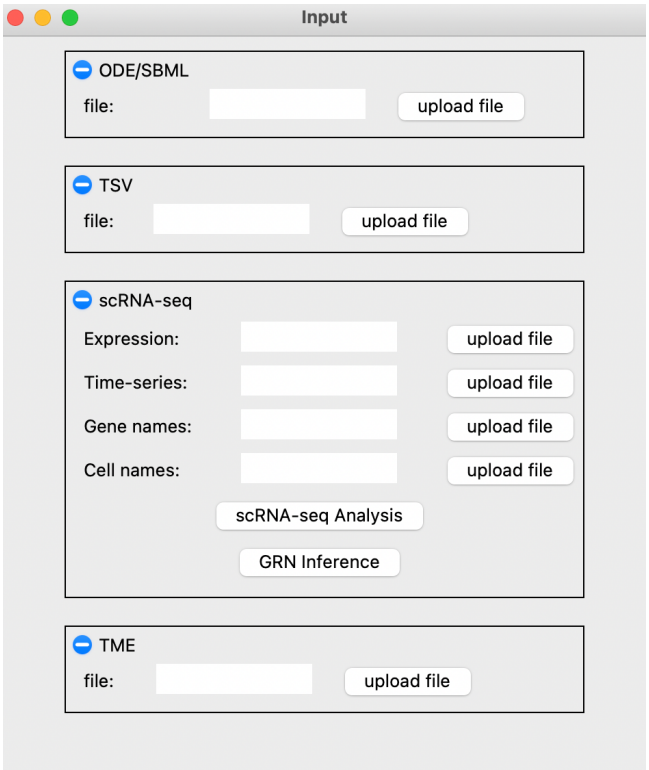

Figure 3. The 'Input' interface of the TMELand.

3.1.1. TSV

When choosing 'TSV' checkbox, you can upload the model by press "upload file" button and the file name would display in the left entry.

Figure 4. An uploaded TSV model.

TSV (Tab-separated values) format file stores a collection of interaction relationships of genes with a specific definition, which can reflect the gene regulatory network topology.

Each line of a TSV file includes two gene names and one interaction relationship between these two genes, '+' represents the first gene activates the second gene, and '-' represents the first gene inhibits the second gene. Each item in a line is separated by a tab. An example with instructions is given as follow:

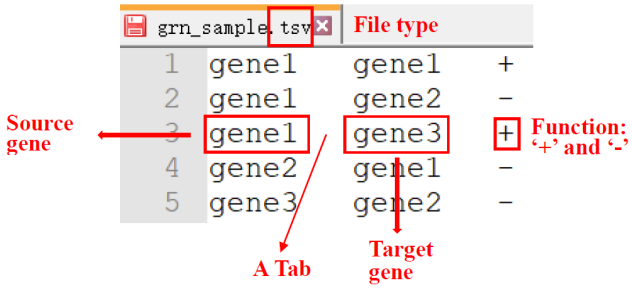

Figure 5. A TSV input file example and instructions.

The loaded topology could be visualized by "GRN topology" button, an example is shown as follow:

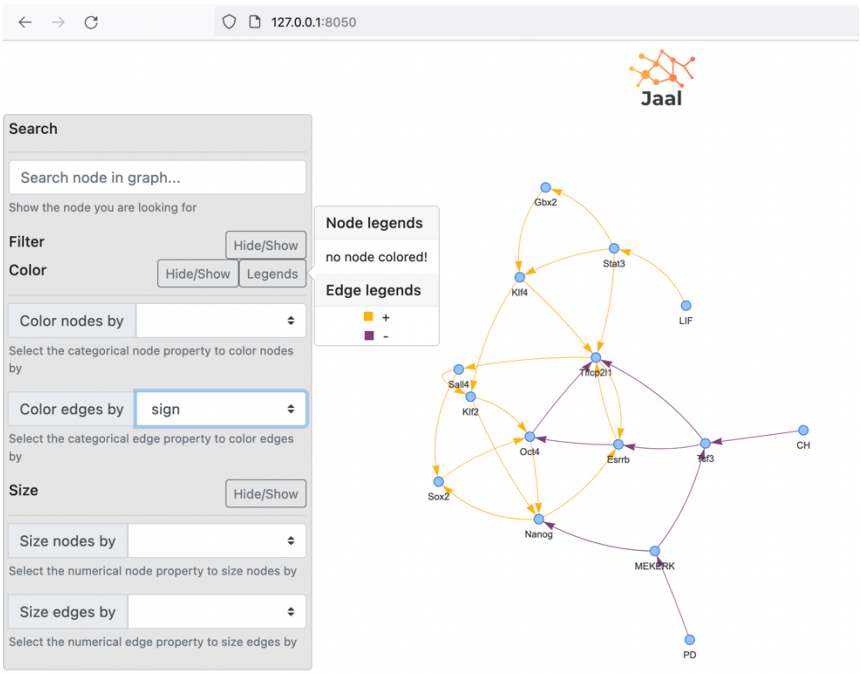

Figure 6. The visualization of a TSV file.

The graph visualization tool is Jaal (https://github.com/imohitmayank/jaal), which is launched by a child process, and then it can be accessed from a specified port of the localhost (i.e., 8050). On the panel, users can search for a node, move the graph, zoom in the display, drag a node, and highlight an edge when the cursor focuses on it, which is especially useful when loading a dense GRN structure. One problem is if the 8050 port is taken, TMELand will pop a sub window to ask user whether to release the port automatically.

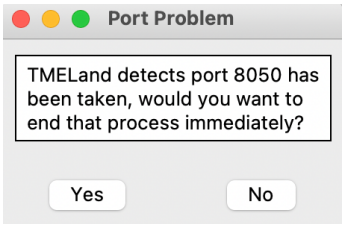

Figure 7. The popped window to ask whether to release the 8050 port.

3.1.2. XPPAUT ODE

When choosing ‘ODE/SBML’ checkbox, you can upload the model by press “upload file” button and the file name would display in the left entry. When uploading files, you can change the file type to show the ODE or SBML file.

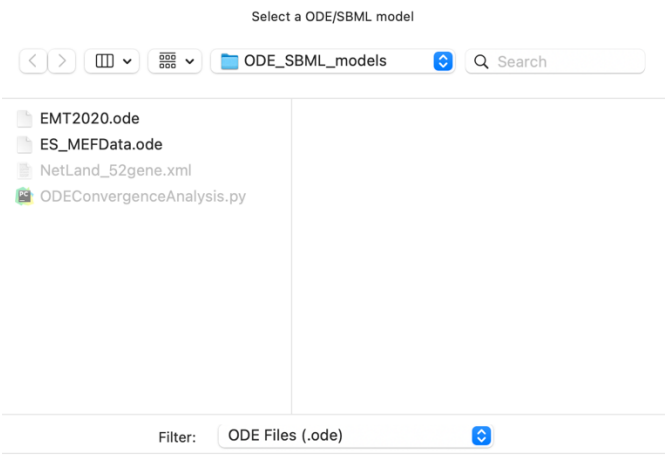

Figure 8. Choosing file type when upload ODE computational model.

XPPAUT ODE format file consists of a set of ODE equations and correlated parameters. In our support ODE models, you can follow the following rules. [10] (XPPAUT ODE format file parsing code refer to https://github.com/MCLand-NTU/MCLand)

1. (Optional) Ignore lines that begin with '**#**' or '**@**';
2. (Optional) Define parameters by using a line starting with '**par**', multiple parameters are separated by a comma;
3. (Momentary) Define differential equations by starting with '**Var'=**' or '**dVar/dt=**', **Var** refers to a variable;
4. (Optional) Define constant by starting with 'const';
5. (Optional) Define initial values by starting with 'init';

This is an example:

EMT2020.ode

```
1  mtROS'=(7*AMPK^4)/(10*(AMPK^4+104976))-mtROS/10
   +5634282409793945/(549755813888*(AMPK^4+14641))
   +7203/(2*(HIF1^4+2401))
2  noxROS'=HIF1^4/(10*(HIF1^4+6561))-noxROS/10+240
   1/(10*(AMPK^4+2401))
```

Figure 9. An example of part of an XPPAUT ODE format file.

3.1.3. SBML

SBML is an abbreviation of Systems Biology Markup Language. It is a representation format, based on XML, for communicating and storing computational models of biological processes. Generally, we can use an SBML file that describes transcriptional regulations from the BioModels Database. BioModels Database contains rich mathematical models of biological systems with a specific format. For more

information, please refer to https://www.ebi.ac.uk/biomodels/.

Due to the complication of SBML, we don't show an example here. You can refer to the 'ODE_SBML_models' folder in TMELand code for a full example.

3.1.4. scRNA-seq

The aforementioned inputs are GRN model. As a main contribution of this software tool, we support scRNA-seq data, and infer the underlying GRN using a GRN inference algorithm.

The TMELand asked user to upload four files, including a 'gene x cell' gene expression value matrix, a time list of cells, gene names list and cell names list (We use 16-cell and 32-cell data in the GUO 2010 dataset [11] as an example, the extracted data corresponds to the process of early mouse embryo development from 16-cell to ICM (inner cell mass) and TE (trophectoderm)).

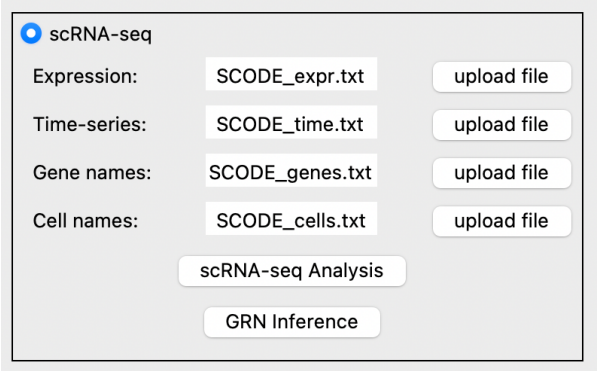

Figure 10. An example uploading single-cell data related files.

Then, to analyze data, you can use 'scRNA-seq Analysis' functionality. We provide PCA-based linear dimensionality reduction, Leiden graph clustering [12], and PAGA-based trajectory inference [13]. Users can highlight a part of the plot by specific gene, cell names or clustering result. The other visualization related parameters are by default.

Figure 11. The dimensionality reduction result of input data highlighted by different

choices.

We can see from Fig. 11A that there are three separately clusters, and the colors are distinguished by cell labels. The left bottom cluster is 16-cell, and the top and right clusters are differentiated cells. Then we highlight the plot by different genes, i.e., POU5F1 (OCT4) and CDX2 as shown in the Fig. 11B and Fig. 11C. We have known that OCT4 is the marker gene of ICM and CDX2 is the marker gene of TE, then the top cluster corresponds to the ICM and the right cluster corresponds to the TE.

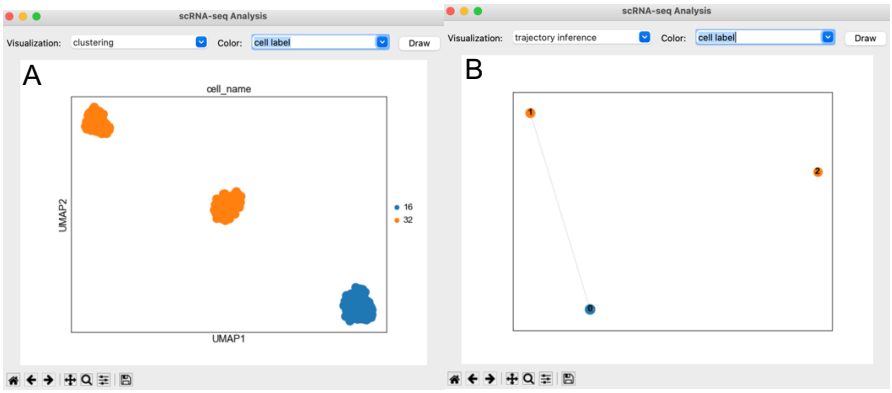

Figure 12. The clustering and trajectory inference result of input data highlighted by cell label.

The clustering result is shown in Fig. 12A and the trajectory inference result is shown in Fig. 12B. In Fig. 12A, the clusters are separately well. However, in Fig. 12B, we only see a connection between cluster 0 and cluster 1, we don't find a connection between cluster 0 and cluster 2.

Subsequently, user can infer GRN topology by "GRN generation", we use SCODE algorithm [14], and the gene number and cell number will be parsed automatically. User need to define pnum, maxite, and repeat, they represent respectively with the number of z, the number of iterations of optimization, and the running times. The inferred topological structure is stored in the local path ./Inferred_GRN/GRN-topology.tsv. However, the inferred topology reflects the strongness of regulation in all genes, if user want to keep top k edges or a sub network, user can define the k value or upload a subnet genes list shown in Fig. 14B to filter the GRN. A part of inferred GRN topology is shown in Fig. 15, the last column is absolute predicted weight. User can visualize the inferred GRN topology, and the thickness can be modified by choosing size edges by

weight on the visualization page. The GRN inference algorithm can't guarantee that the inferred GRN structure is completely correct. Thus, TMELand supports GRN topology editing. Users can add a node, or delete a node if it isn't involved in any reaction. Users can also add or delete an edge by specifying the source node, regulatory relationship and target. To speed up node or edge finding process, users can search from their collections. Fig. 16 is the GRN editing page, in the bottom frame, we can search CDX2 as source node in the regulation relationships.

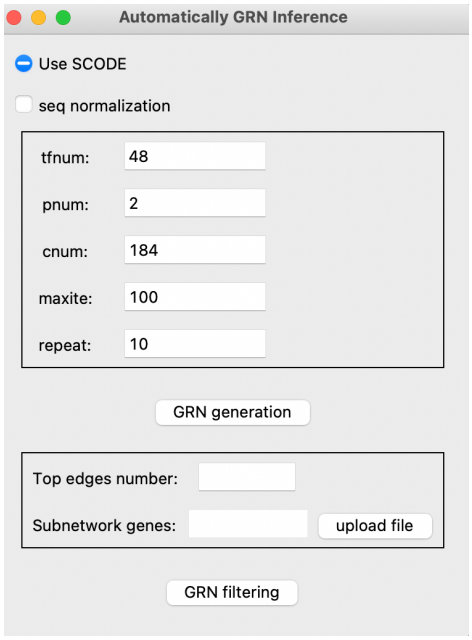

Figure 13. The GRN inference interface

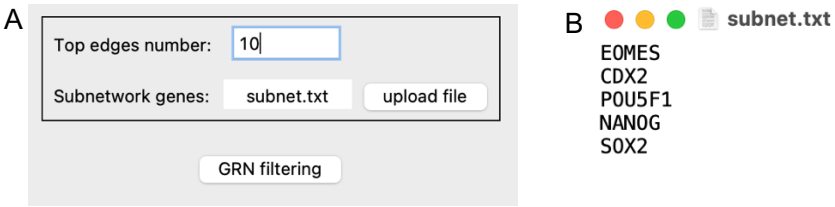

Figure 14. The filtering on the inferred GRN interface

GRN-topology

regulator	target	func	weight
CDX2	POU5F1	-	0.708331511986305
CDX2	CDX2	-	0.62520793078441
CDX2	NANOG	-	0.424890913592471

Figure 15. A part of inferred GRN topology

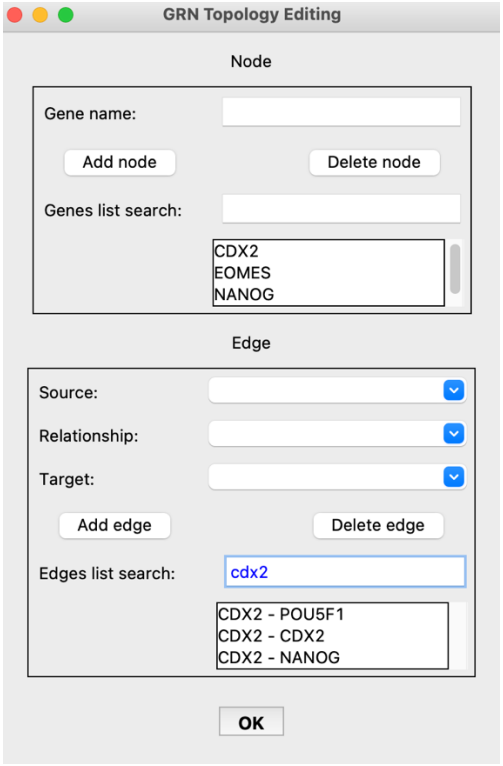

Figure 16. The GRN editing page

3.1.5. TME

TME is a self-defined model using JSON format, which is used to save computed model and next reloading. We can save landscape only or landscape with paths. This is the structure of a TME JSON file ('allPath' and 'pathAction' only appear when calculated paths; modelTopology, IM, 'Hill_fn', TM, 'Act_Inh', 'Degrade_rate' are defined for topology-based models; 'ODEModelODECode' and 'ODEModelJacCode' are defined for ODE based models mainly):

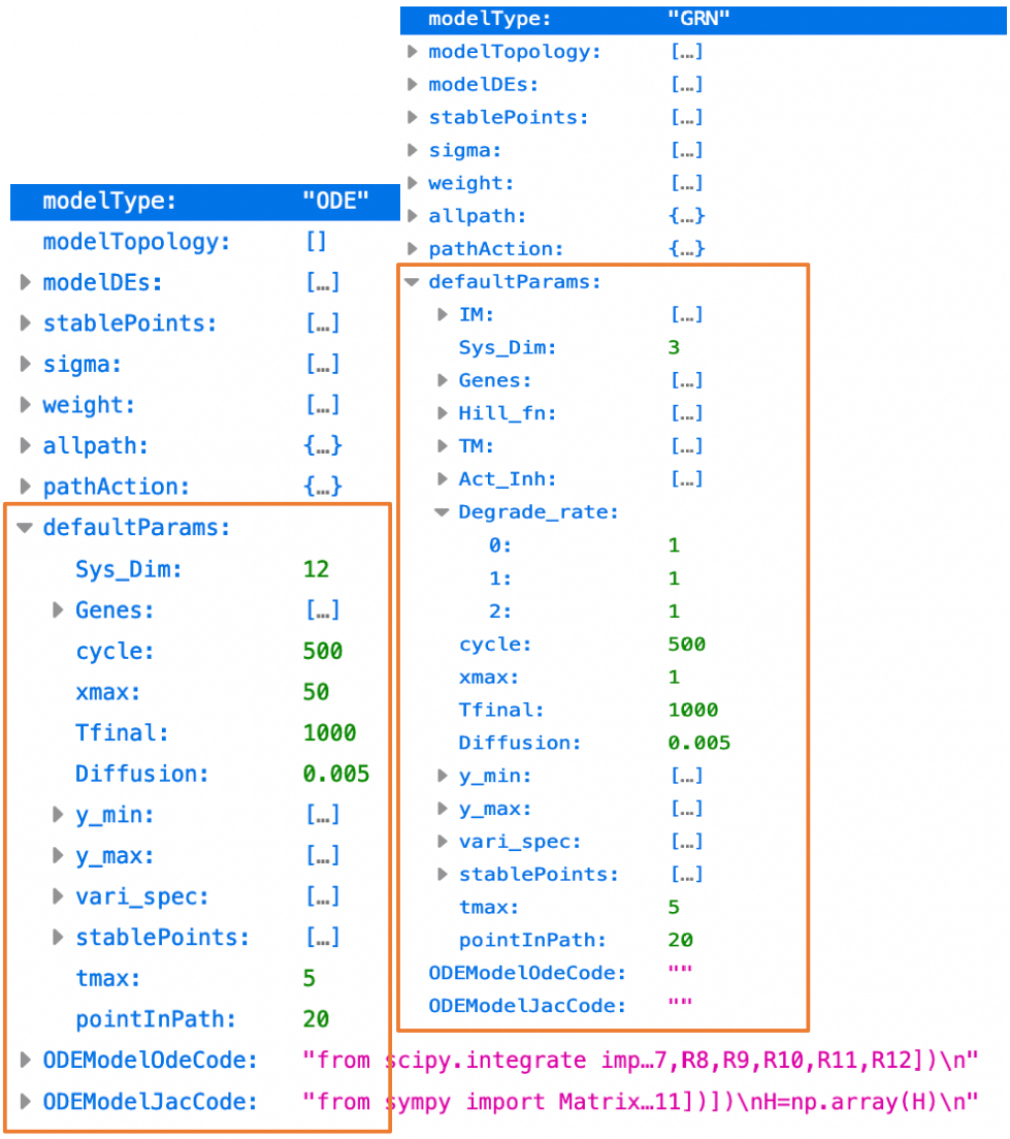

Figure 17. The structure of the TME file (JSON type).

We will introduce the 7 main items, the detailed items of default_params are given in the later sections. modelType denotes the format of the input file; stablePoints are converged locations of all trajectories; sigma is covariance; weight can reflect the width of basin (stablePoints, sigma, and weight are key elements to construct landscape); allpath stores all paths that you have drawn with dictionary format, the key is the index of beginning attractor to the index of end attractor, and the value is variation of all genes along the path; pathAction represents the difficulty level of transfer, the higher of pathAction is, the harder to transfer; default_params contains defined parameters in the TMELand and other useful information.

3.2. Model definition formulation

TMELand is based on the differential equations to assign dynamics for GRN, thus, we need to define parameters to form differential equations.

Obviously, we only need to transform the GRN topology into differential equations, because we can use ODE-based models (XPPAUT ODE and SBML) to construct landscape directly.

We use the modeling method proposed in to construct differential equations of GRN topology. Specifically, given the topological structure of a GRN, we use Equation (1) to describe the temporal evolution of the expression level of each gene. In Equation (1), x_i is the expression level of the *i*th gene, $F(x_i)$ is the driving force and takes a nonlinear ODE form as shown in Equation (2), and ζ is the Gaussian white noise, which is related to the user-defined diffusion coefficient. In Equation (2), m_{i1} is the number of activators of gene *i*, m_{i2} is the number of inhibitors of gene *i*, x_{pji} is an activation gene of gene *i*, x_{qji} is an inhibition gene of gene *i*, *A* is activation constant, *B* is the inhibition constant, *n* is the Hill coefficient, *s* is the threshold, and *k* is the degradation rate. All the aforementioned parameters will be decided by GRN topology and user input.

$$\frac{dx_i}{dt} = F(x_i) + \zeta \qquad (1)$$

$$F(x_i) = \sum_{j=1}^{m_{i1}} \frac{A x_{p_{ji}}^n}{x_{p_{ji}}^n + s^n} + \sum_{j=1}^{m_{i2}} \frac{B s^n}{x_{q_{ji}}^n + s^n} - k x_i \qquad (2)$$

The interface of the dynamics formulation is shown in Fig. 18, and it can be accessed by “DE-based GRN” button. Once user defined parameters, and the generated ODEs will display in the bottom frame.

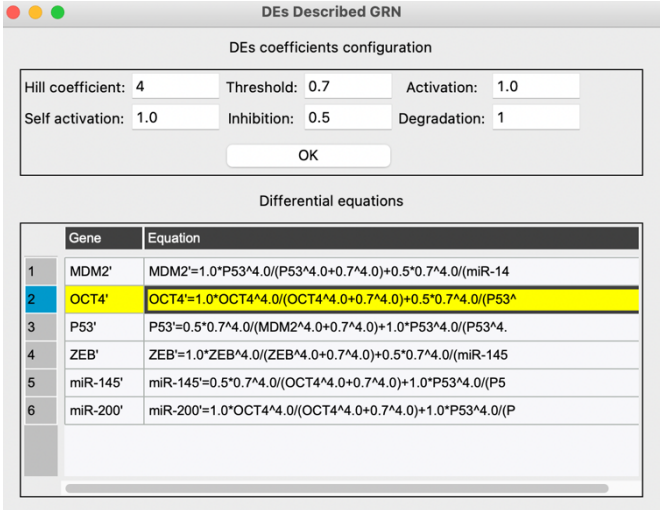

Figure 18. The interface of the 'Dynamics setting' page.

4. Model simulation

After we obtain DE-based GRN, we simulate multiple trajectories to analyze the steady states of the system. In the 'Dynamics Simulation' page, we define three parameters related to simulation, 'Number of series', 'Gene max value', and 'Time'. 'Number of series' refers to the number of trajectories. One trajectory corresponding to an initial condition, and the initial gene expression value is randomly generated under the constraint of maximum gene expression value ('Gene max value'). 'Time' is the initial evolution time points of the differential equation. Time can be set large when ODE converges slowly, and vice versa. Users can watch the convergence states in the right frame and analyze the state by our provided analysis functionality to choose appropriate parameters.

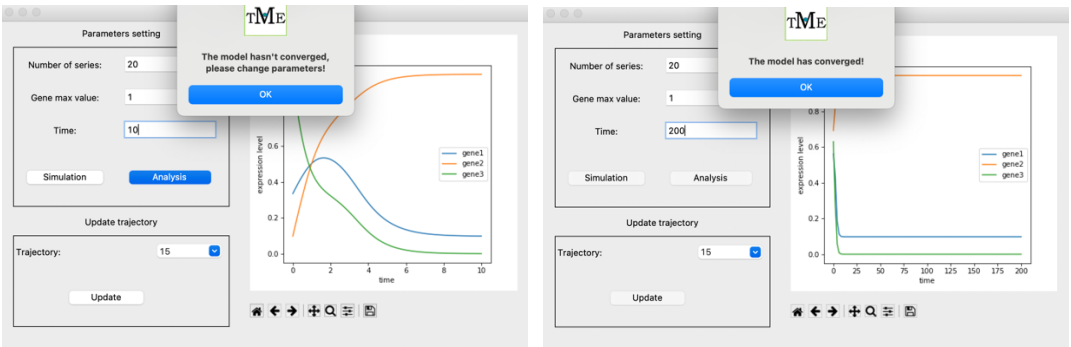

Figure 19. The interface of the 'Dynamics simulation' page.

5. Landscape construction and visualization

5.1. Probabilistic landscape

TMELand uses the probability-based method to visualize Waddington's landscape, which applies the truncated moment equation (TME) method proposed in. With the development of all simulated trajectories, we will acquire the steady-state points $\bar{\mathbf{x}}$ finally when trajectories are converged, and converged points are also known as basins of attractors. The ratio of terminal points in each basin is its weight **w**. Then, by using the calculated steady-state points $\bar{\mathbf{x}}$ and self-defined diffusion coefficient, we can calculate the covariance **σ**. We will determine the shape of the landscape using the weighted sum of multiple normal distributions P_{ss} with mean $\bar{\mathbf{x}}$, covariance **σ**, and weight **w**. Once we have the steady-state probability distribution P_{ss}, we calculate the potential U of landscape, i.e., $U=-\ln(P_{ss})$.

We define the diffusion coefficient parameter after three simulation-related parameters. Then, we click the 'Draw' button to visualize the constructed landscape with default visualization parameters (the default visualization parameters are using the first two genes as marker genes and setting each gene range is from 0 to 3). An example using default parameters is shown in Fig. 20.

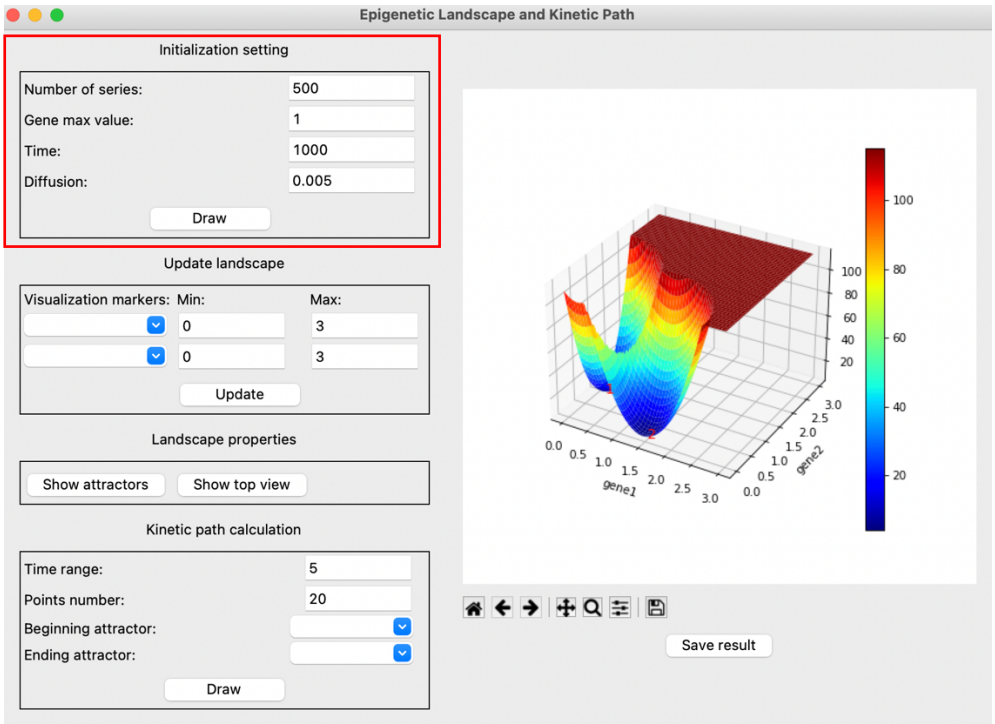

Figure 20. The screenshot of using default parameters to draw a landscape.

5.2. Visualization

To visualize the landscape, we select two marker genes as the x-axis and y-axis respectively and define the value range of axes. From the last subsection, we know that using default visualization parameters to visualize by 'Draw' button. Then, we can choose different marker genes and ranges to update landscape by the button 'Update' to obtain a more clearly landscape.

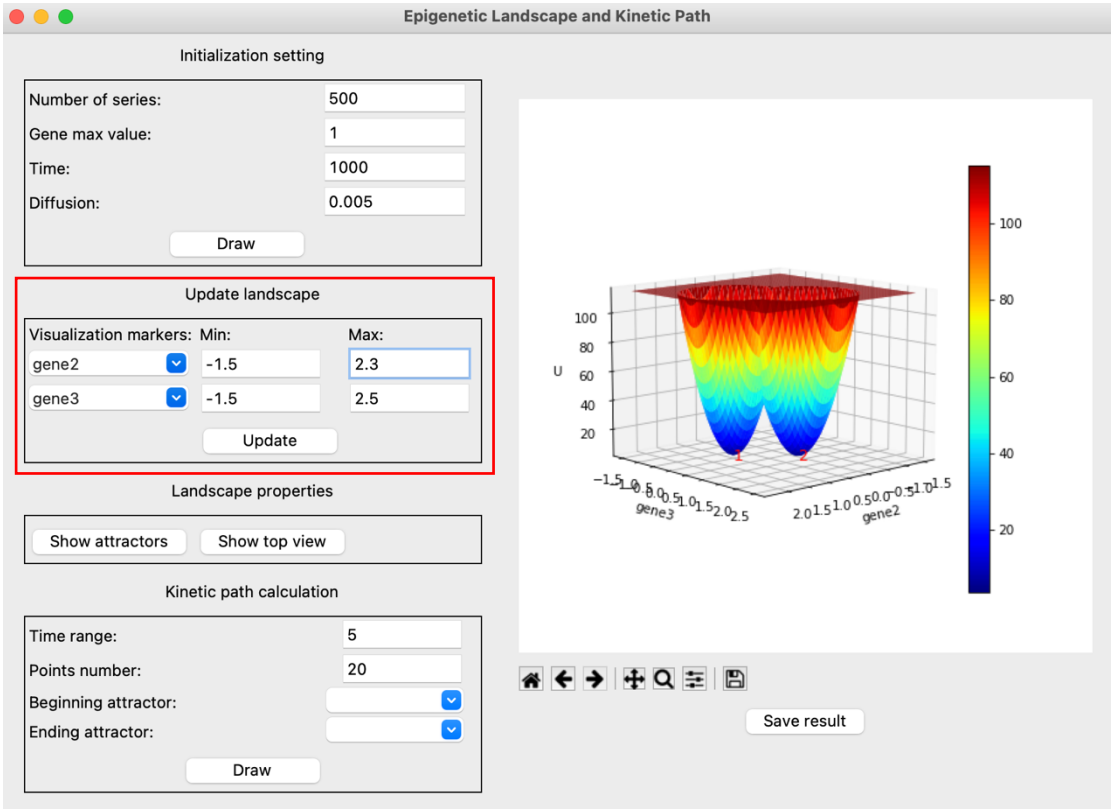

Figure 21. The screenshot of parameters to update landscape.

5.3. landscape analysis

We provide "Show attractors" to show the location and width of attractors. The width is represented by frequency, i.e., the ratio of trajectories that converged into attractors. The locations reflect the gene expression level at all genes. An example is shown in Fig. 22.

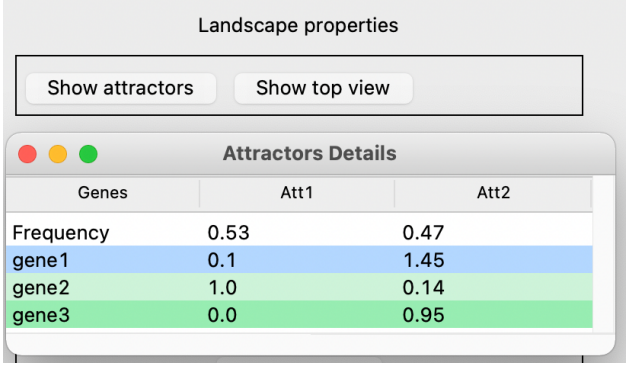

Figure 22. The interface of "Show attractors".

5.4. save landscape

We can save the computed landscape for the next reloading by clicking the 'Save result' button. The save format is JSON, we call the TME model here. For the detailed definition of the TME model please refer to the TME section.

6. Path generation and drawing

6.1. Transition path

After drawing the landscape, a more important task is finding the transition paths between attractors. The quantification of kinetic transition paths among attractors provides guidance for us to understand cellular differentiation and reprogramming.

TMELand provides the function by solving the optimization problem to obtain the minimum action paths (MAP) as transition paths [15]. You can draw a path by clicking the 'Draw path' button after finish the construction of a landscape. Then, we can define the optimal problem of path-related parameters (i.e., time range and granularity) and specify the beginning and ending attractor to draw the transition path.

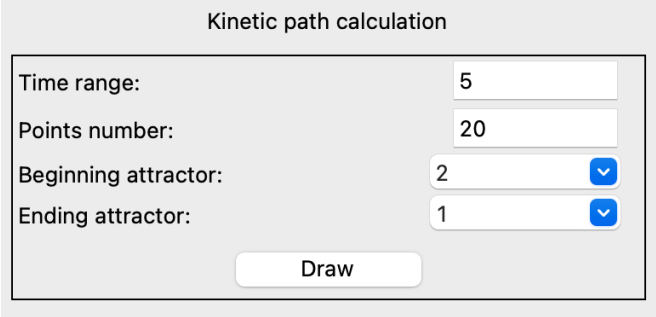

Figure 23. The screenshot of parameters setting to draw a path.

6.2. Visualization

Similar to constructed landscape, to visualize paths, we should reduce dimension from the high dimensional multiple genes to two marker genes. In this way, we obtain the 2D coordinates of paths, and then we project them onto the landscape with the same potential as the landscape. Specifically, we mesh the x-y plane into 100x100 grids, and project points of the path into grids. Every grid has a z value of the landscape projected into this grid. We can acquire z values of all points in the corresponding grid. In this way, we can draw a path in the 3D space with 3D coordinates (x, y, and z). To denote a path more clearly, we add a triangle on the path to represent the direction and the triangle is near the source point.

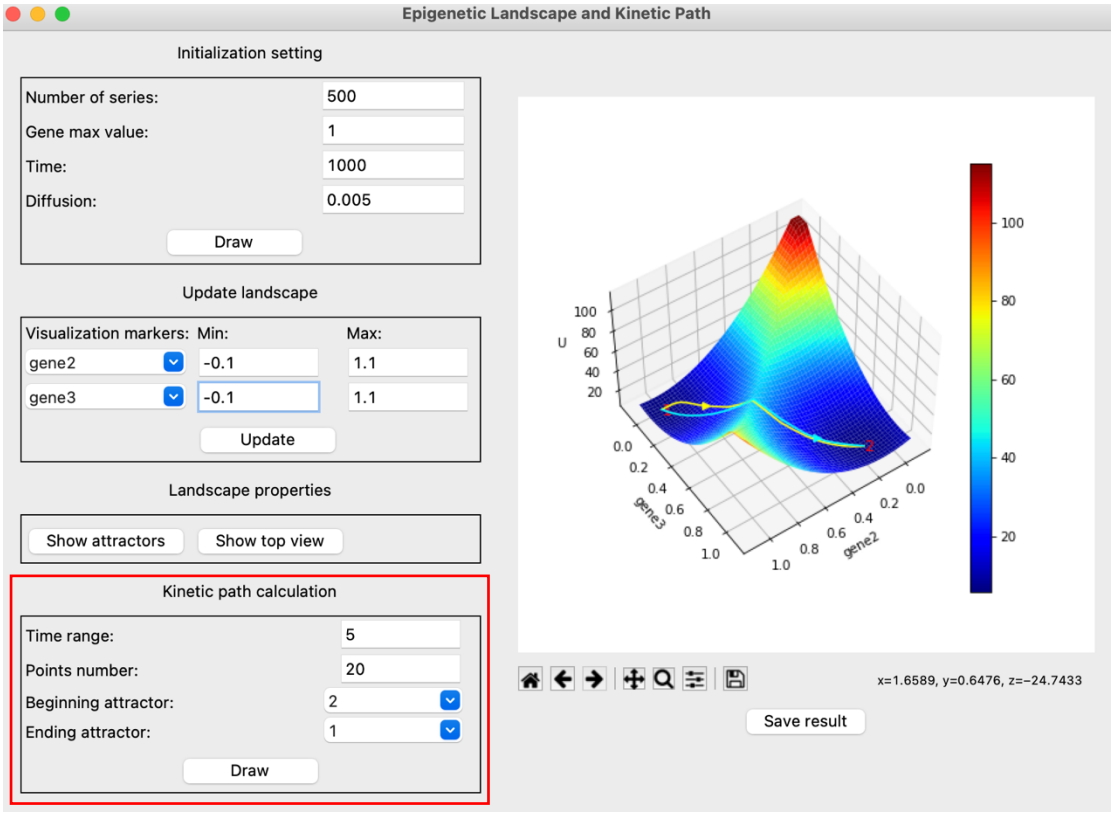

Figure 24. An example of drawing paths on a landscape.

It's worth noticing that path from attractor 1 to 2 is not consistent with the path from attractor 2 to 1, which may denote a different process. The variation of the marker gene expression value along transition paths may give us insights into the corresponding biological process.

Furthermore, if we load a TME model, we can go on drawing paths directly. This allows

us to divide the whole process into two separate sub-processes. We can also update the landscape with paths using the 'Update' button.

6.3. Save path

We can save the model after drawing transition paths by clicking the 'Save result' button. The save format is JSON, we call the TME model here. For the detailed definition of the TME model please refer to the TME section.

7. Case studies

7.1. Case study 1: Cancer attractors

For case study 1, we use the 6-gene model proposed in [16]. The model is developed to research the development of cancer stem cells (CSC). In their original work, they calculated 4 attractors, CSC, cancer, normal, and stem cell. Additionally, they also calculated transition paths among attractors (Fig. 25).

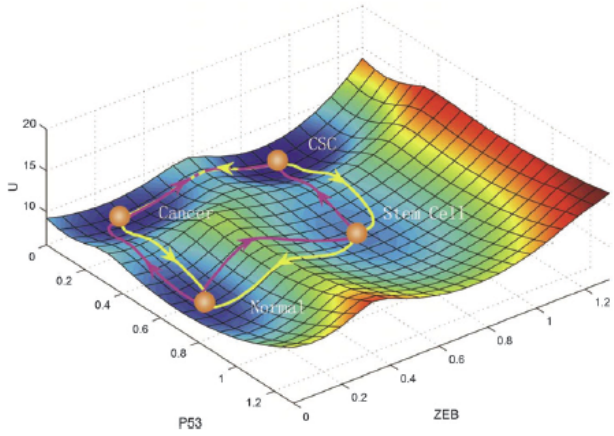

Figure 25. The calculated landscape and transition paths in the original work of Case study 1 [16].

We use TMELand to load the XPPAUT ODE format model (./CaseStudies/ CaseStudy1_Li6_ODE.ode) and draw landscape and transition paths as shown in Fig. 26. From Fig. 26, we obtain 8 attractors, and attractors 1, 2 can seem like a pair, 3 and 4, 5 and 6, 7 and 8 likewise. We draw paths among attractors 1, 3, 5, 7 and we find that paths are consistent with the original work. Accordingly, attractor 3, 4 represents CSC, attractor 1, 2 represents cancer, attractor 7, 8 represents stem cell, and attractor 5, 6

represents normal. Furthermore, we both test NetLand and MATLAB code in [17], they all give out 8 attractors, which means we may give more details than the original work. The phenomenon is especially mentioned in [17].

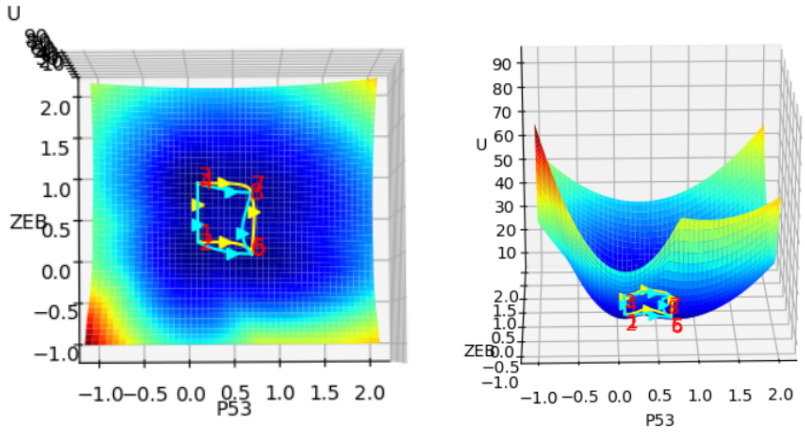

Figure 26. The result of Case study 1 using TMELand.

7.2. Case study 2: Stem cell differentiation and reprogramming

For case study 2, we use the 52-gene model proposed in [3]. The model is developed to research the process of differentiation and reprogramming. In their original work, they calculated 2 attractors, one represents stem cell attractor and another one represents differentiation attractor. They also calculated transition paths between these two attractors to find out how the differentiation and reprogramming happened (Fig. 27).

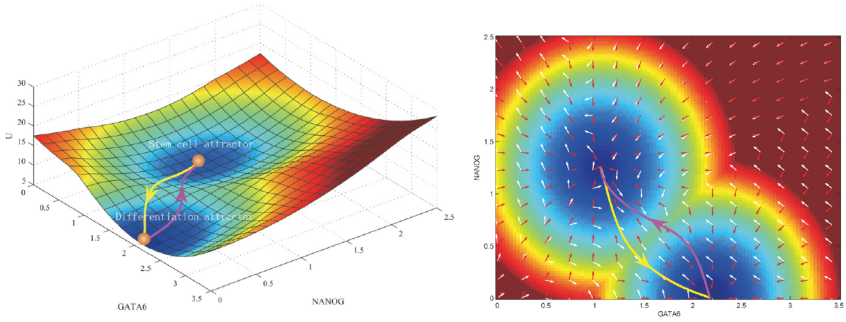

Figure 27. The calculated landscape and transition paths in the original work of Case study 2 [3].

We use TMELand to load the XPPAUT ODE format model (./CaseStudies/ CaseStudy2_Li52_ODE.ode) and draw landscape and transition paths as shown in Fig 28. From Fig. 28, we obtain 2 attractors. By comparing the results generated by TMELand and results in their original work, we find that they are consistent.

Accordingly, attractor 1 represents stem cell attractor, and attractor 2 represents differentiation attractor.

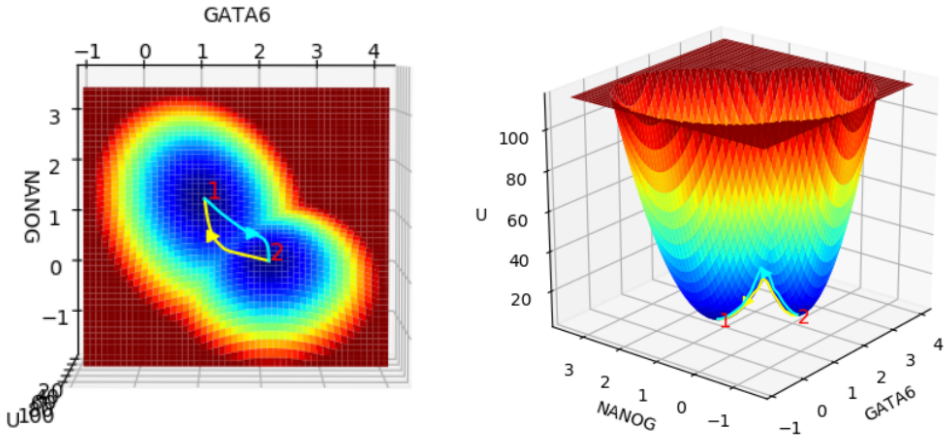

Figure 28. The result of Case study 2 using TMELand.

We test NetLand on Case study 2 and using the same parameters to compare the results of TMELand and NetLand. We set the 'Number of series' as 100, 'Gene max value' is 1, 'Time' is 128, and the visualization range is from 0 to 3. The results of TMELand are (a), (b) in Fig. 29 and the results of NetLand are (c), (d) in Fig. 29. The test file is ./ CaseStudies/NetLand_52gene.xml.

In the paper of the TME method, they have pointed out that they consider correlations between any two genes, which means genes are not independent in GRN. Take all correlations of genes into account is also more meaningful. As we can see in Fig. 29, the axis of attractor in (a) is oblique, which means GATA6 has a relationship with NANOG. However, we can't find a strong correlation of marker genes in (c). Thus, the comparison implies that TMELand is much better.

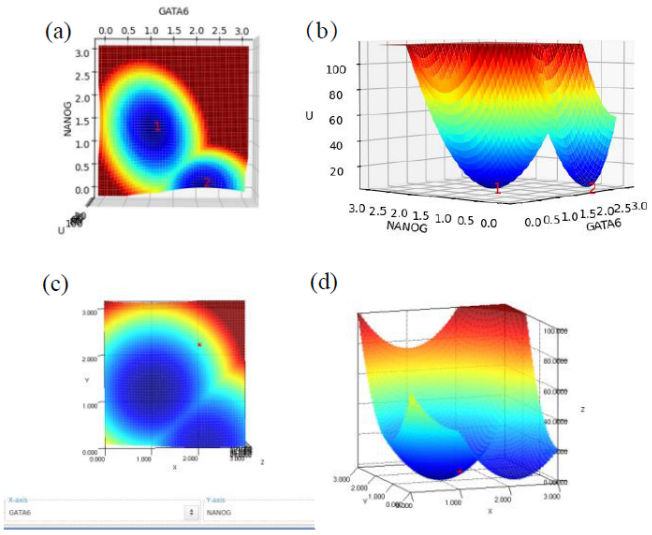

Figure 29. The comparison of TMELand and NetLand for Case study 2 using the same parameters. (a) and (b) are results of TMELand; (c) and (d) are results of NetLand.

7.3. Case study 3: EMT-metabolism network

The EMT-metabolism network has the same topology as the EMT-metabolism subnetwork of the GRN in [9]. In [9], they construct a GRN composed of metabolism, EMT, and metastasis to characterize cancer. However, the EMT-metabolism network here has different models and parameters, which obtain four stable points. We use TMELand to load the XPPAUT ODE model (./CaseStudies/CaseStudy3_EMT2020_ODE.ode) and draw landscape and transition paths as shown in Fig. 30.

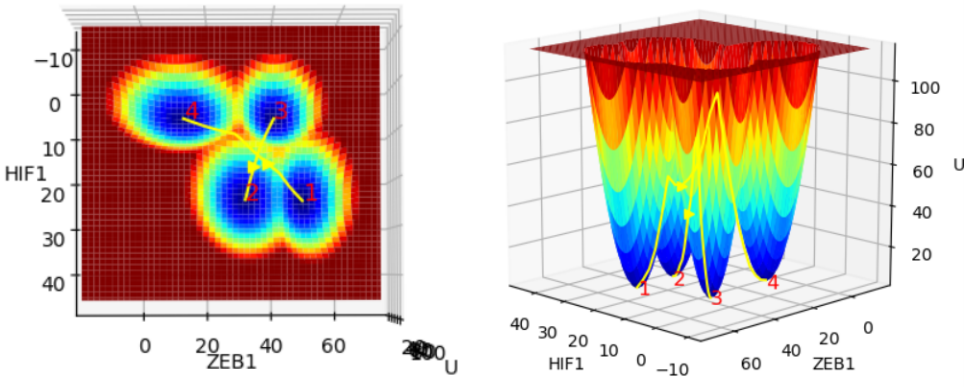

Figure 30. The result of Case study 3 using TMELand.

8. Time and memory test

We test the performance of TMELand on all case studies (XPPAUT ODE models), and the number of genes is 6, 12, 52. The test environment is a 12-core CPU 3.70 GHz Intel(R) Core(TM) i7-8700K Ubuntu 18.04.2 LTS 64-bit system and 32G memory. The version of Python is 3.7.0. We use the time and tracemalloc package to measure the consuming time and memory separately.

We compare time-consuming and memory costs for both computing landscape and transition path. The consuming time for landscape and path is separate, and the memory is all peak value when the process is running. The computing parameters are described in Table 2.

Table 2. The parameters setting for performance test.

	Case study 1 (6)	Case study 2 (52)	Case study 3 (12)
Number of series	500	500	500
Gene max value	1	1	50
Time	1000	1000	1000
Diffusion	0.005	0.005	0.05
Markers	p53, Zeb	NANOG, GATA6	HIF1, ZEB1
Visualization range	[-2,3], [-2,3]	[-1.5,3], [-0.5,4]	[-10,45], [-15,70]
Time range	5	5	5
Points number	20	20	20
Beginning attractor	1	1	1
Ending attractor	2	2	2

The comparison result as follows:

Table 3. The comparison of time and memory cost of TMELand for different N (N represents gene number).

		N=6	**N=12**	**N=52**
Time	**Landscape**	11.2	36.9	224.1

(s)	Path	19.9	114.6	1815.6
Memory (MB)	Landscape	34.7	36.0	146.7
	Path	5.8	4.9	16.9

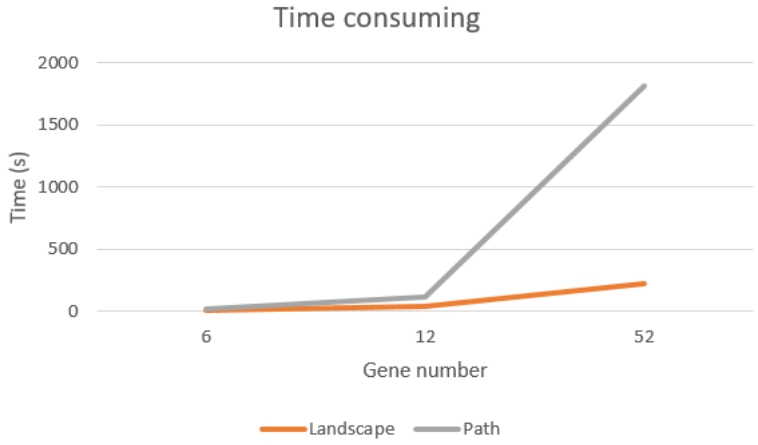

Figure 31. The time-consuming changes with an increase in the number of genes for test models.

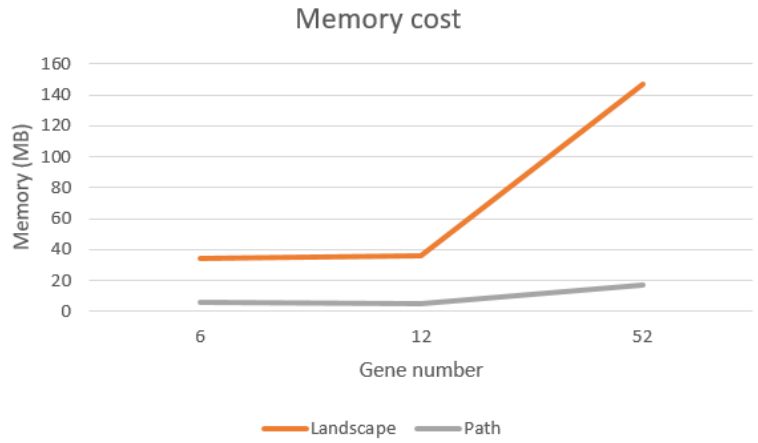

Figure 32. The memory cost changes with the increase in the number of genes for test models.

From Fig. 31 and Fig. 32, we can watch that with the increase of the number of genes, both time-consuming and memory cost increase gradually. However, as for time-consuming, the time of computing a path grows faster than computing the landscape. The memory cost of the landscape is much higher and grows faster than a path.

Furthermore, we also noticed that different model types have different performances. For example, we both compare the performance of XPPAUT ODE format and TSV format of N=52 model, and the latter is much slower than the former. That may because

of many for-loops in computing for a TSV model, such as the process of forming ODEs for a TSV model. So, it's better to provide an ODE-based model, rather than a TSV model when you pursue computing efficiency.

9. License

TMELand followed the GNU General Public License (GPL) v3.0 license and as found in the LICENSE file in the main directory. If you have any problem, please contact zhengjie@shanghaitech.edu.cn or chunheli@fudan.edu.cn.

10. Contact information

TMELand has been thoroughly tested, you can use it freely with instructions of this User Manual. However, we can't cover all models during our test. If there is any bug, error, or improvement when you use it, please post your issue on the GitHub website (https://github.com/JieZheng-ShanghaiTech/TMELand) or send your suggestions or bug reports to zhengjie@shanghaitech.edu.cn or chunheli@fudan.edu.cn.

References

[1]. C. Waddington, "The strategy of the genes," 1957.
[2]. J.Wang, K. Zhang, L. Xu, and E.Wang, "Quantifying the waddington landscape and biological paths for development and differentiation," Proceedings of the National Academy of Sciences, vol. 108, no. 20, pp. 8257–8262, 2011.
[3]. C. Li and J. Wang, "Quantifying cell fate decisions for differentiation and reprogramming of a human stem cell network: landscape and biological paths," PLoS Comput Biol, vol. 9, no. 8, p. e1003165, 2013.
[4]. A. T. Fard, S. Srihari, J. C. Mar, and M. A. Ragan, "Not just a colourful metaphor: modelling the landscape of cellular development using hopfield networks," NPJ systems biology and applications, vol. 2, no. 1, pp. 1–9, 2016.
[5]. C. R. Banerji, D. Miranda-Saavedra, S. Severini, M. Widschwendter, T. Enver, J. X. Zhou, and A. E. Teschendorff, "Cellular network entropy as the energy potential in waddington's differentiation landscape," Scientific reports, vol. 3, no. 1, pp. 1–7, 2013.
[6]. H. Chu, D. Lee, and K.-H. Cho, "Precritical state transition dynamics in the attractor landscape of a molecular interaction network underlying colorectal tumorigenesis," PloS one, vol. 10, no. 10, p. e0140172, 2015.
[7]. J. Guo, F. Lin, X. Zhang, V. Tanavde, and J. Zheng, "Netland: quantitative modeling and visualization of waddington's epigenetic landscape using probabilistic potential," Bioinformatics, vol. 33, no. 10, pp.

1583–1585, 2017.
[8]. O. S. Shah, M. F. A. Chaudhary, H. A. Awan, F. Fatima, Z. Arshad, B. Amina, M. Ahmed, H. Hameed, M. Furqan, S. Khalid et al., “Atlantis-attractor landscape analysis toolbox for cell fate discovery and reprogramming,” Scientific Reports, vol. 8, no. 1, pp. 1–11, 2018.
[9]. X. Kang, J.Wang, and C. Li, “Exposing the underlying relationship of cancer metastasis to metabolism and epithelial-mesenchymal transitions,” iScience, vol. 21, pp. 754–772, 2019.
[10]. Ket Hing Chong, X.Z. and J.Z., MCLand: A Python program for drawing spontaneously emerging paths in Waddington's epigenetic landscape by Monte Carlo estimation Submitted, 2021.
[11]. G. Guo, M. Huss, G. Q. Tong, C.Wang, L. L. Sun, N. D. Clarke, and P. Robson, “Resolution of cell fate decisions revealed by single-cell gene expression analysis from zygote to blastocyst,” Developmental cell, vol. 18, no. 4, pp. 675–685, 2010.
[12]. V. A. Traag, L. Waltman, and N. J. Van Eck, “From louvain to leiden: guaranteeing well-connected communities,” Scientific reports, vol. 9, no. 1, pp. 1–12, 2019.
[13]. F. A. Wolf, F. K. Hamey, M. Plass, J. Solana, J. S. Dahlin, B. G ottgens, N. Rajewsky, L. Simon, and F. J. Theis, “Paga: graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells,” Genome biology, vol. 20, no. 1, pp. 1–9, 2019.
[14]. H. Matsumoto, H. Kiryu, C. Furusawa, M. S. Ko, S. B. Ko, N. Gouda, T. Hayashi, and I. Nikaido, “Scode: an efficient regulatory network inference algorithm from single-cell rna-seq during differentiation,” Bioinformatics, vol. 33, no. 15, pp. 2314–2321, 2017.
[15]. E. Weinan, W. Ren, and E. Vanden-Eijnden, “Minimum action method for the study of rare events,” Communications on Pure and Applied Mathematics, vol. 57, no. 5, pp. 637–656, 2004.
[16]. C. Li and J. Wang, “Quantifying the landscape for development and cancer from a core cancer stem cell circuit,” Cancer Research, vol. 75, no. 13, pp. 2607–2618, 2015.
[17]. X. Zhang, K. H. Chong, L. Zhu, and J. Zheng, “A monte carlo method for in silico modeling and visualization of waddington’s epigenetic landscape with intermediate details,” BioSystems, vol. 198, p. 104275, 2020.