netplotlib Documentation

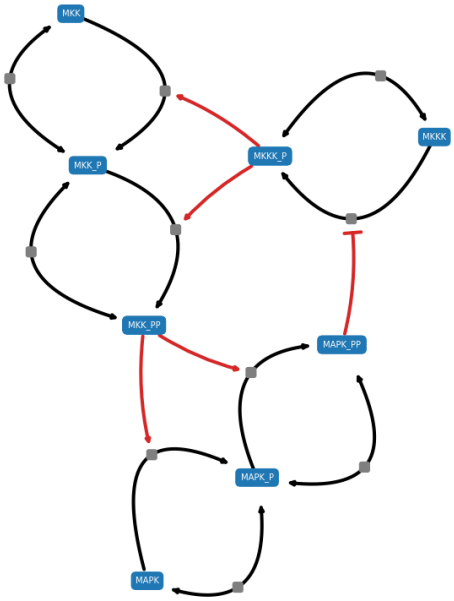
Release 1.2.1

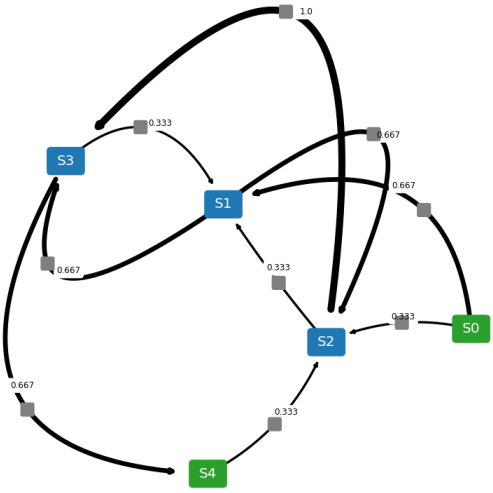
Kiri Choi

Apr 27, 2022

Contents

Netplotlib is an extension to NetworkX and matplotlib to draw and analyze reaction network diagrams in SBML or Antimony strings with ease. Netplotlib supports visualization of quantities such as flux and species rate of change. Netplotlib provides functions to visualize network model ensemble.





CHAPTER 1

Installation

Netplotlib is available through PyPI. To install, run:

```
$ pip install netplotlib
```

Netplotlib is also available by default in Tellurium.

CHAPTER 2

Quick Start

To start using netplotlib, install the package using pip or use Tellurium.

This section demonstrates how to load in a model and draw network diagrams. To start with, lets import netplotlib package and define a model. Here, we use a simple feed forward loop as an example.

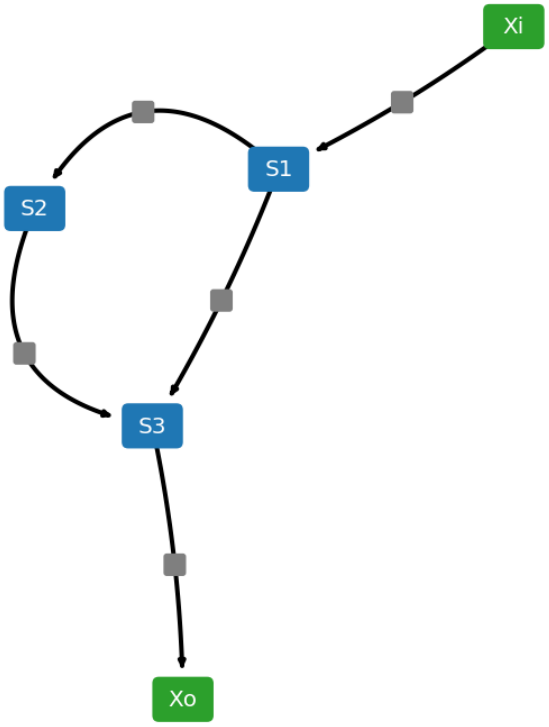
```
import netplotlib as npl

AntimonyStr = '''
$Xi -> S1; k0*Xi
S1 -> S2; k1*S1
S2 -> S3; k2*S2
S1 -> S3; k3*S1
S3 -> $Xo; k4*S3

Xi = 3; Xo = 2
k0 = 0.46; k1 = 0.73; k2 = 0.64;
k3 = 0.51; k4 = 0.22
'''
```

Next, create an Network object.

```
net = npl.Network(AntimonyStr)
```

To generate network diagrams, simply run draw().

```
net.draw()
```


Xi
S1
S2
S3
Xo

CHAPTER 3

Tutorial

3.1 Network Configuration

Network object supports following properties for configuring the network diagram.

- scale: scaling factor for layout algorithm
- fontsize: fontsize for labels
- edgelw: linewidth of edges
- nodeColor: node color
- reactionNodeColor: reaction node color
- labelColor: label color
- labelReactionIds: boolean flag for labeling reaction ids
- reactionColor: edge color
- modifierColor: modifier edge color
- boundaryColor: boundary node color
- nodeEdgeColor: node edge color
- nodeEdgelw: linewidth of node edges
- highlight: list of species ids or reaction ids to highlight
- hlNodeColor: node color of highlighted nodes
- hlNodeEdgeColor: node edge color of highlighted nodes
- drawReactionNode: boolean flag for drawing reaction nodes
- breakBoundary: boolean flag for breaking all boundary species into separate nodes
- tightLayout: boolean flag for tighter node layout
- analyzeFlux: boolean flag for visualizing flux

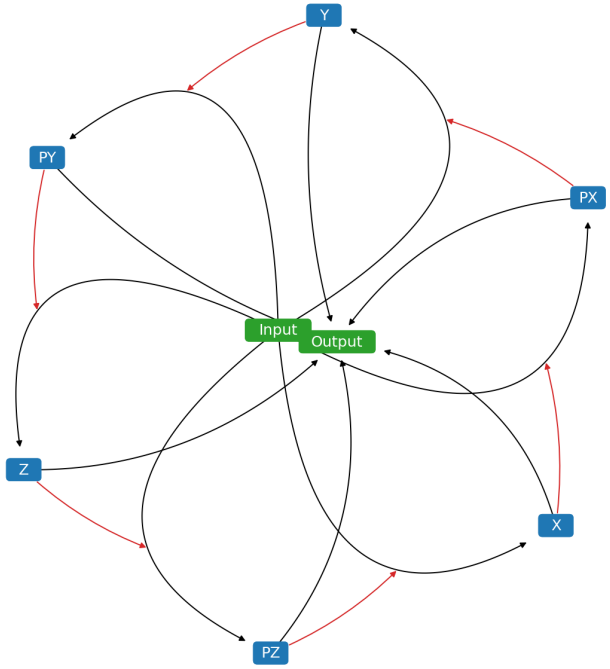
- analyzeRates: boolean flag for visualizing species rate of changes
- analyzeColorHigh: color to use for higher values during analysis
- analyzeColorLow: color to use for lower values during analysis
- analyzeColorMap: colormap to use for analysis.
- analyzeColorScale: boolean flag for using colormaps. Setting this true ignore analyzeColorHigh and analyzeColorLow
- drawInlineTimeCourse: boolean flag for plotting inline time-cource simulation
- simTime: value for simulation duration
- forceAnalysisAtSimTime: boolean flag for running analysis at the end of simTime
- plotColorbar: boolean flag for visualizing color bar
- inlineTimeCourseSelections: list of species to plot for time-course simulation
- customAxis: use custom matplotlib.pyplot.axis object for the diagram
- layoutAlgorithm: specify layout algorithm

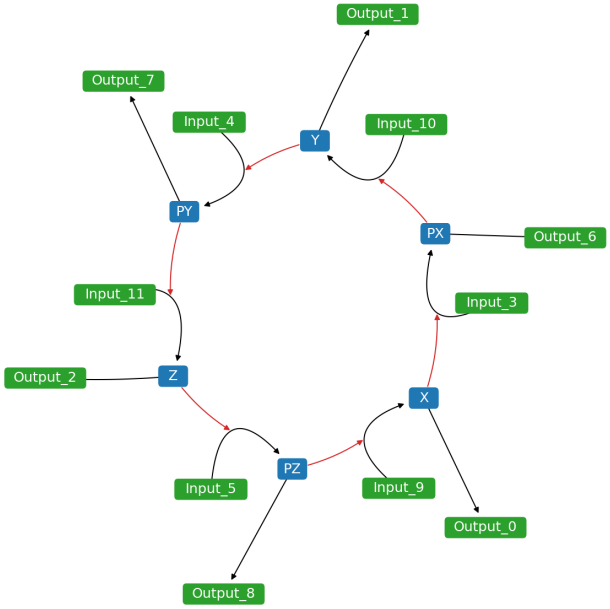
You can define these properties and run draw() to generate customized network diagrams.

3.2 Detached Boundary

Detach common boundary species by setting *breakBoundary=True*.

```
net = npl.Network(npl.testmodels.REPRESSILATOR)
net.draw()
net.breakBoundary = True
net.draw()
```

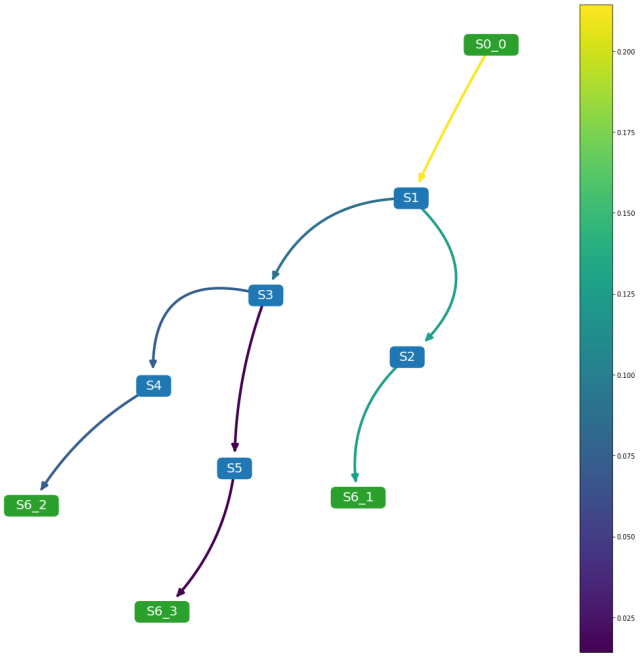




3.3 Flux Visualization

Visualize flux using colormap via setting *analyzeFlux=True*. Set *analyzeColorScale=True* to scale the colormap to minimum and maximum values. You can supply your own colormaps. To plot colorbar, set *plotColorbar=True*.

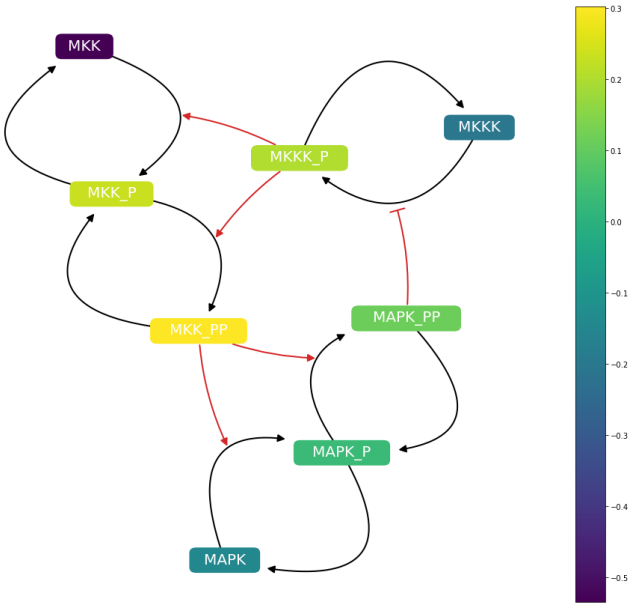
```
net.analyzeColorScale = True
net.analyzeFlux = True
net.analyzeColorMap = 'viridis'
net.plotColorbar = True
```



3.4 Rate of Changes Visualization

Visualize species rates of change using colormap via setting *analyzeRates=True*. Set *analyzeColorScale=True* to scale the colormap to minimum and maximum values. The resulting plot will show species rate of change at t=simTime. You can supply your own colormaps. To plot colorbar, set *plotColorbar=True*.

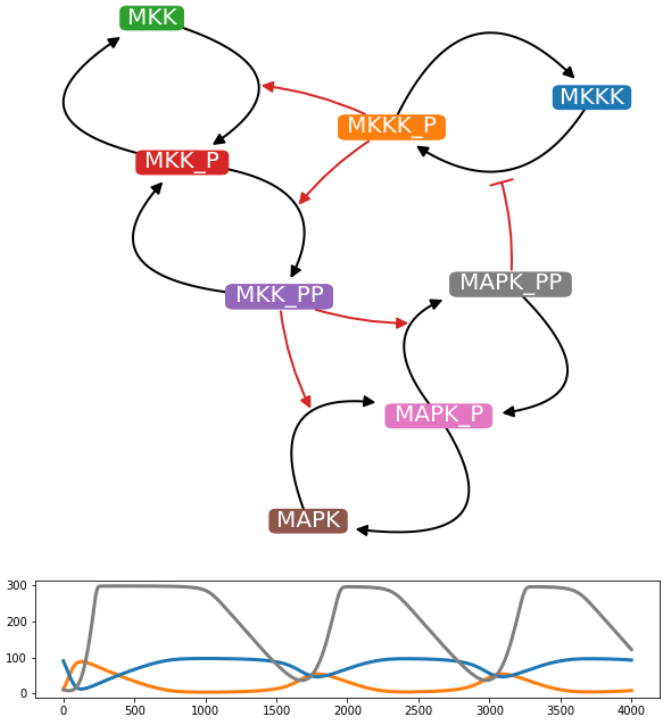
```
net.analyzeColorScale = True
net.analyzeRates = True
net.analyzeColorMap = 'viridis'
net.simTime = 3000
net.plotColorbar = True
```



3.5 Inline Time-course Plot

Plot inline time-course plot along with network diagrams with matching colors by setting *drwaInlineTimeCourse=True*. The simulation runs for t=simTime. You can selection specific species to plot by passing a list to *inlineTimeCourseSelections* property.

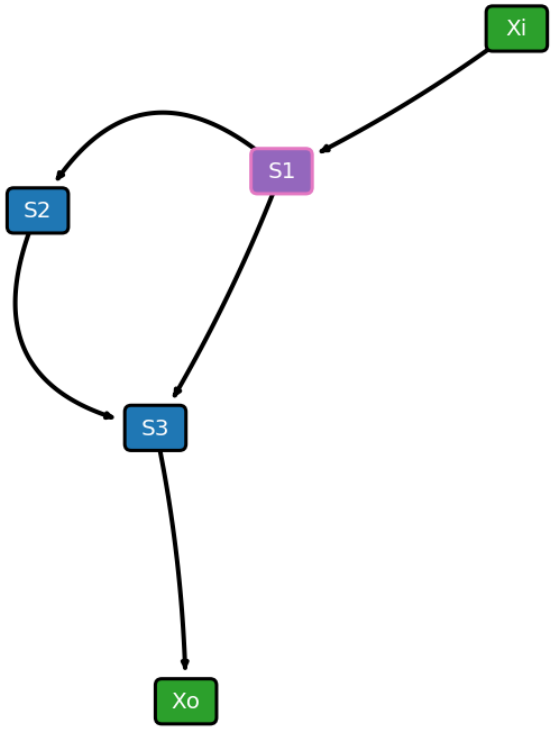
```
net.drawInlineTimeCourse = True
net.inlineTimeCourseSelections = ['MKKK_P', 'MKKK', 'MAPK_PP']
net.simTime = 4000
```



3.6 Highlighting

Highlight specific nodes by passing a list to highlight property. Choose the highlight colors using hlNodeColor and hlNodeEdgeColor properties.

```
net.drawReactionNode = False
net.nodeEdgelw = 3
net.highlight = ['S1']
net.draw()
```



3.7 Selecting Layout Algorithm

Currently, netplotlib supports following layout algorithms:

'kamada-kawai'

'spring'

'dot'

'neato'

'twopi'

By default, netplotlib uses Kamada-Kawai layout algorithm. Certain layout algorithms require external graphviz to be configured and pygraphviz package installed.

3.8 NetworkEnsemble Configuration

NetworkEnsemble object supports following properties for configuring the network diagram.

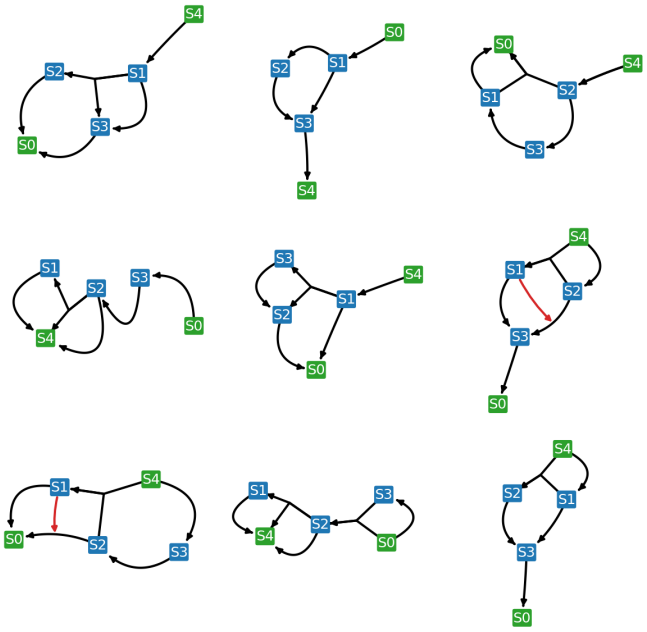
- scale: scaling factor for layout algorithm
- fontsize: fontsize for labels
- edgelw: linewidth of edges
- nodeColor: node color

- reactionNodeColor: reaction node color
- labelColor: label color
- labelReactionIds: boolean flag for labeling reaction ids
- reactionColor: edge color
- modifierColor: modifier edge color
- boundaryColor: boundary node color
- nodeEdgeColor: node edge color
- nodeEdgelw: linewidth of node edges
- highlight: list of species ids or reaction ids to highlight
- hlNodeColor: node color of highlighted nodes
- hlNodeEdgeColor: node edge color of highlighted nodes
- edgeLabel: boolean flag for displaying edge weights
- edgeLabelFontSize: fontsize of edge weight labels
- drawReactionNode: flag for drawing reaction nodes
- breakBoundary: boolean flag for breaking all boundary species into separate nodes
- weights: list of custom weights to override
- edgeTransparency: boolean flag for changing the transparency of the edges accordin to edge weights
- plottingThreshold: value of threshold to prevent from displaying weighted edges
- removeBelowThreshold: boolean flag for preventing weighted edges below plottingThreshold from displaying
- analyzeFlux: boolean flag for visualizing flux
- customAxis: use custom matplotlib.pyplot.axis object for the diagram
- layoutAlgorithm: specify layout algorithm

3.9 Grid Plot

Plot a grid plot of network diagrams of individual models in the list by running *drawNetworkGrid()* function.

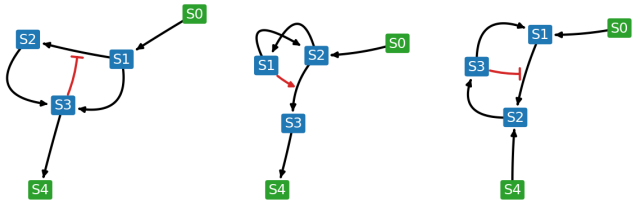
```
net.drawNetworkGrid(3, 3)
```

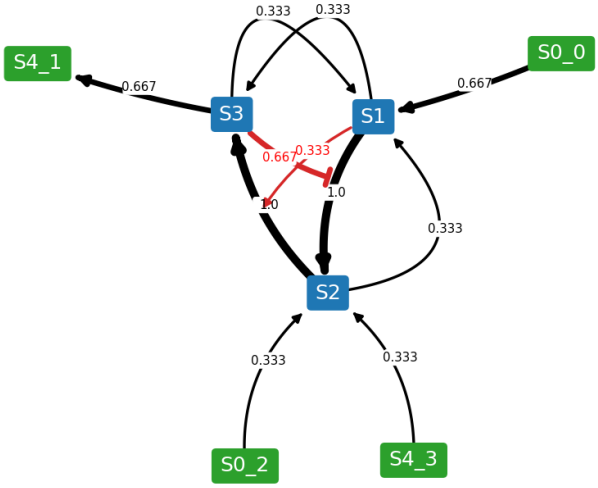


3.10 Weighted Network Diagram

Combine models in the list and generate a one network diagram where the edges are weighted according to the frequency. To generate a weighted network diagram, run *drawWeightedDiagram()* function. Below are images of an example where an ensemble of three models are combined into a weighted network diagram.

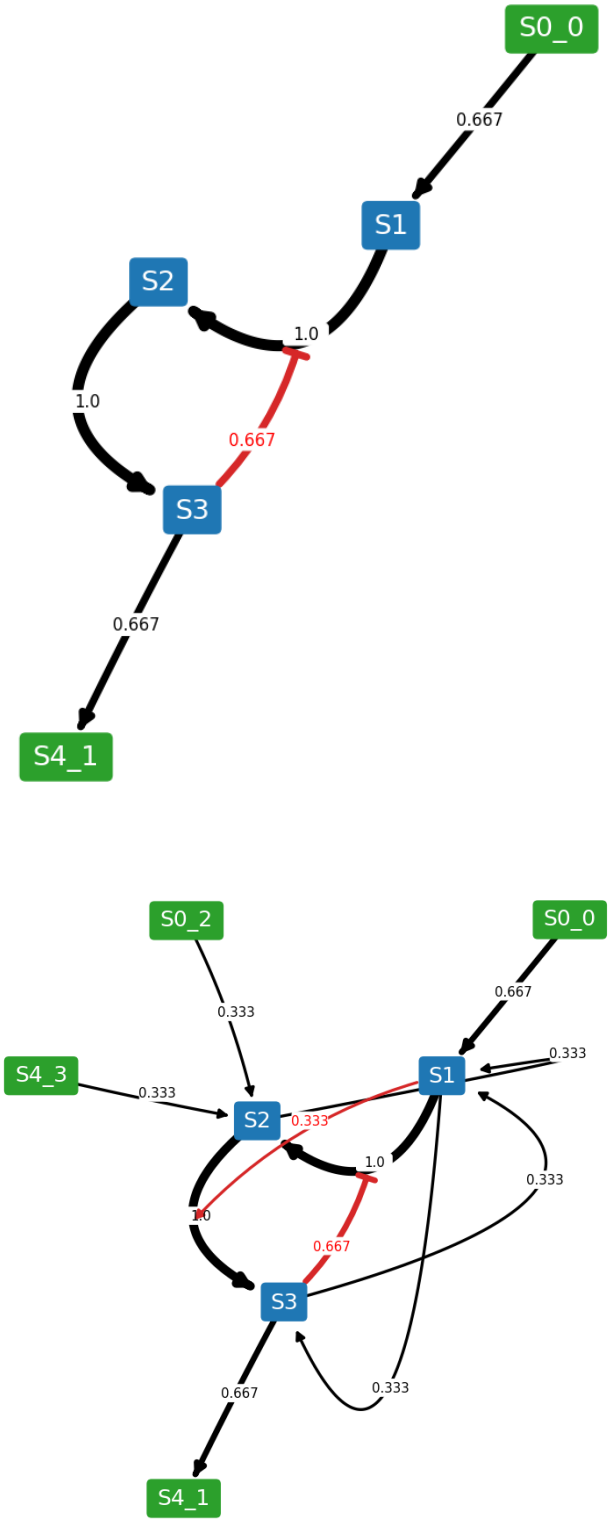
```
net.drawWeightedDiagram()
```





It is possible to set a threshold where edges below the threshold are removed from the resulting network diagram. To set a threshold, use *plottingThreshold* and *removeBelowThreshold* properties. For example, if *plottingThreshold=0.5* and *removeBelowThreshold=True*, any edges that appear in less than half of the model ensemble will be ignored. However, sometimes you might want to put the ignored reactions back while keeping the layout. To do so, set *removeBelowThreshold=False* while keeping the *plottingThreshold*.

```
net.removeBelowThreshold = True
net.plottingThreshold = 0.5
net.drawWeightedDiagram()
net.removeBelowThreshold = False
net.drawWeightedDiagram()
```


S0_0
0.667
S1
S2
1.0
1.0
0.667
S3
0.667
S4_1

S0_2
S0_0
0.333
0.667
0.333
S4_3
S1
0.333
S2
0.333
1.0
0.333
0.667
S3
0.667
0.333
S4_1

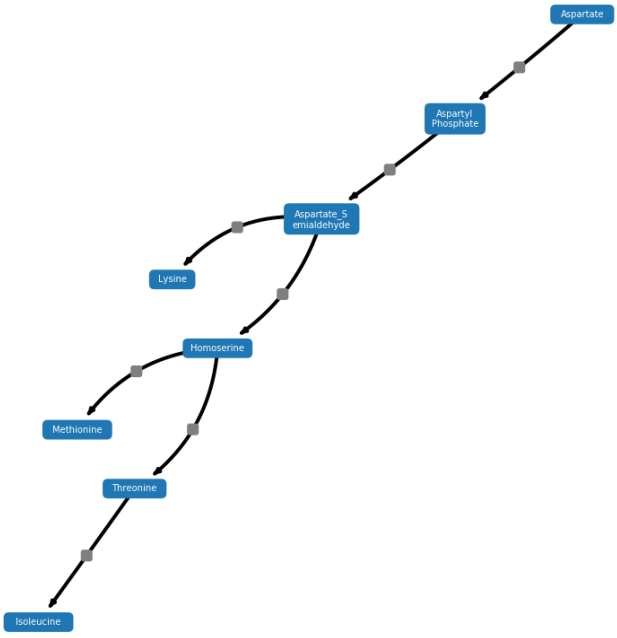
3.11 Test Cases

Netplotlib comes with set of test cases. All test cases are available under netplotlib.testmodels submodule.

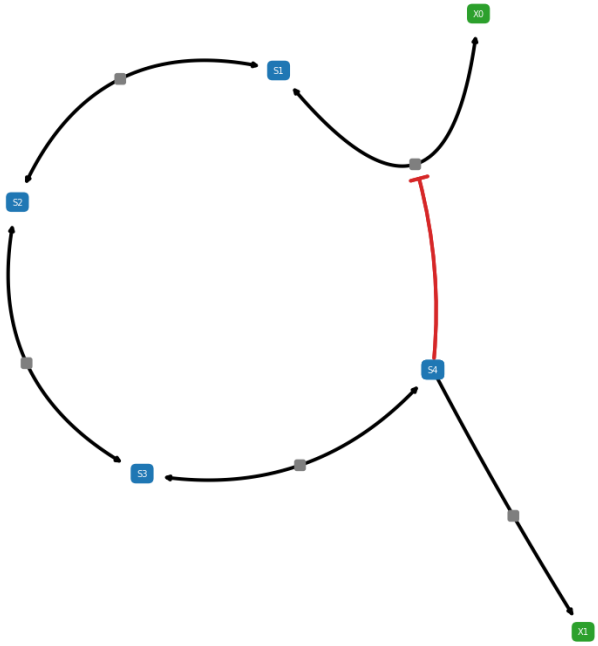
CHAPTER 4

Examples

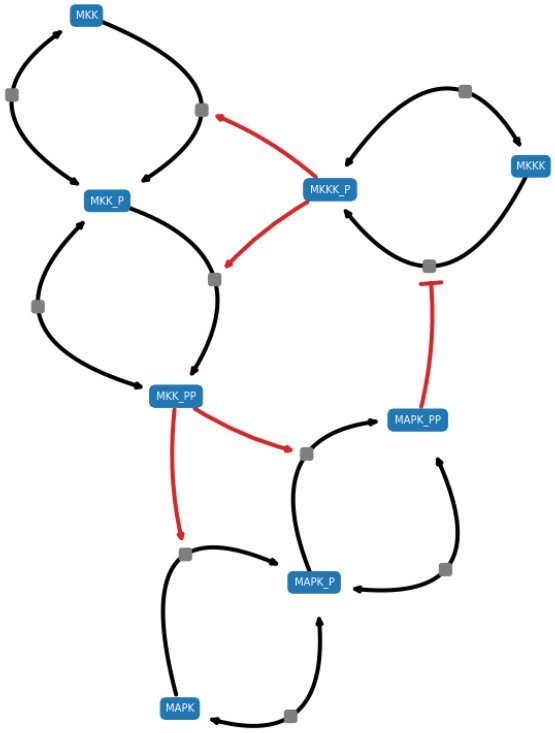
4.1 Aspartate Metabolism



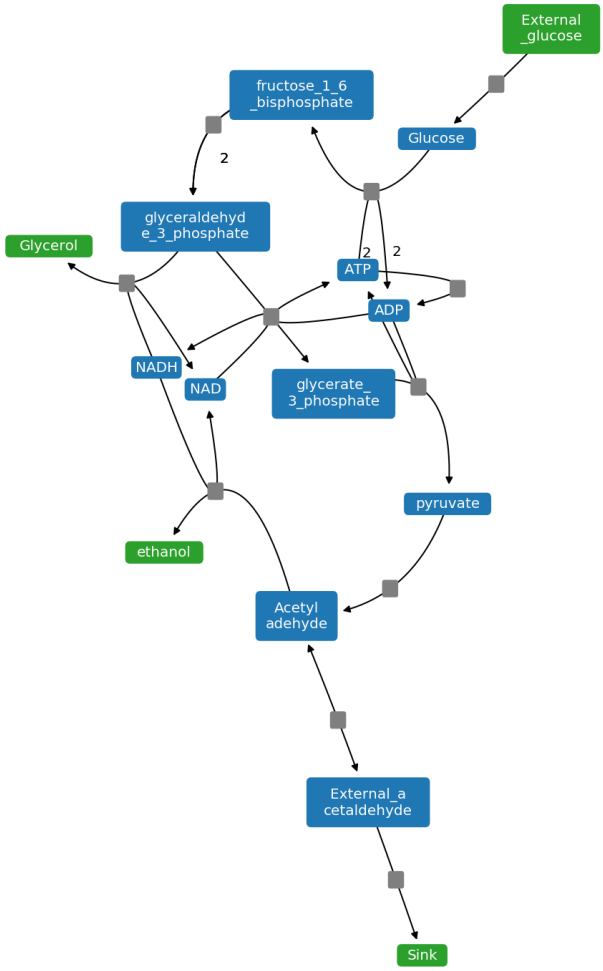
4.2 Feedback Loop



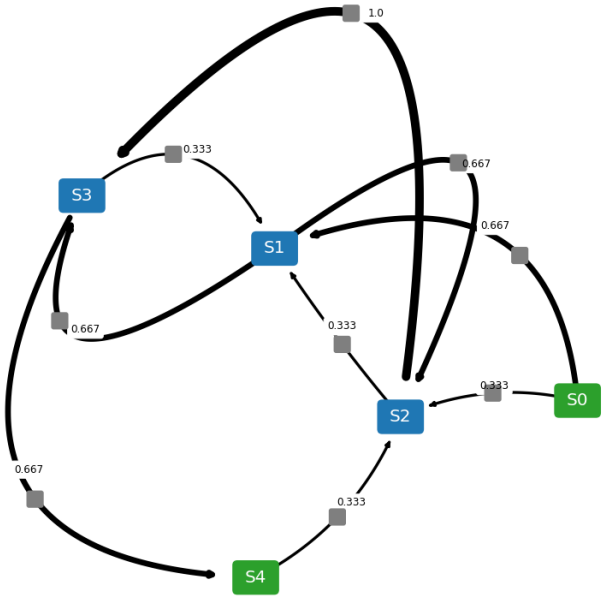
4.3 MAPK Cascade



4.4 Glycolysis



4.5 Network Ensemble



CHAPTER 5

netplotlib API

5.1 Single Network

netplotlib.**Network**(*model*)

Creates a new Network object.

Parameters **model** – SBML or Antimony string of a model

An instance of Network object accepts following properties:

- scale: scaling factor for layout algorithm
- fontsize: fontsize for labels
- edgelw: linewidth of edges
- nodeColor: node color
- reactionNodeColor: reaction node color
- labelColor: label color
- labelReactionIds: boolean flag for labeling reaction ids
- reactionColor: edge color
- modifierColor: modifier edge color
- boundaryColor: boundary node color
- nodeEdgeColor: node edge color
- nodeEdgelw: linewidth of node edges
- highlight: list of species ids or reaction ids to highlight
- hlNodeColor: node color of highlighted nodes
- hlNodeEdgeColor: node edge color of highlighted nodes
- drawReactionNode: boolean flag for drawing reaction nodes

- breakBoundary: boolean flag for breaking all boundary species into separate nodes
- tightLayout: boolean flag for tighter node layout
- analyzeFlux: boolean flag for visualizing flux
- analyzeRates: boolean flag for visualizing species rate of changes
- analyzeColorHigh: color to use for higher values during analysis
- analyzeColorLow: color to use for lower values during analysis
- analyzeColorMap: colormap to use for analysis.
- analyzeColorScale: boolean flag for using colormaps. Setting this true ignore analyzeColorHigh and analyzeColorLow
- drawInlineTimeCourse: boolean flag for plotting inline time-cource simulation
- simTime: value for simulation duration
- forceAnalysisAtSimTime: boolean flag for running analysis at the end of simTime
- plotColorbar: boolean flag for visualizing color bar
- inlineTimeCourseSelections: list of species to plot for time-course simulation
- customAxis: use custom matplotlib.pyplot.axis object for the diagram
- layoutAlgorithm: specify layout algorithm

Network.**draw**(*show=True*, *savePath=None*, *dpi=150*)
Draw network diagram

Parameters

- **show** – flag to show the diagram
- **savePath** – path to save the diagram
- **dpi** – dpi settings for the diagram

Network.**getLayout**(*returnState=False*)
Return the layout of the model

Parameters **returnState** – boolean flag for returning the networkx.Graph object

Returns pos Dictionary of all nodes and corresponding coordinates

Network.**reset**()
Resets all properties

Network.**savefig**(*path*, *dpi=150*)
Save network diagram to specified location

Parameters

- **path** – path to save the diagram
- **dpi** – dpi settings for the diagram

Network.**setLayout**(*pos*)
Set custom layout and bypass whe is generated by the layout algorothm

Parameters **pos** – Dictionary of all nodes and corresponding coordinates

5.2 Network Ensemble

netplotlib.**NetworkEnsemble**(*models*)
Creates a new NetworkEnsemble object.

Parameters **models** – list of SBML or Antimony strings of models

An instance of NetworkEnsemble object accepts following properties:

- scale: scaling factor for layout algorithm
- fontsize: fontsize for labels
- edgelw: linewidth of edges
- nodeColor: node color
- reactionNodeColor: reaction node color
- labelColor: label color
- labelReactionIds: boolean flag for labeling reaction ids
- reactionColor: edge color
- modifierColor: modifier edge color
- boundaryColor: boundary node color
- nodeEdgeColor: node edge color
- nodeEdgelw: linewidth of node edges
- highlight: list of species ids or reaction ids to highlight
- hlNodeColor: node color of highlighted nodes
- hlNodeEdgeColor: node edge color of highlighted nodes
- edgeLabel: boolean flag for displaying edge weights
- edgeLabelFontSize: fontsize of edge weight labels
- drawReactionNode: flag for drawing reaction nodes
- breakBoundary: boolean flag for breaking all boundary species into separate nodes
- weights: list of custom weights to override
- edgeTransparency: boolean flag for changing the transparency of the edges accordin to edge weights
- plottingThreshold: value of threshold to prevent from displaying weighted edges
- removeBelowThreshold: boolean flag for preventing weighted edges below plottingThreshold from displaying
- analyzeFlux: boolean flag for visualizing flux
- customAxis: use custom matplotlib.pyplot.axis object for the diagram
- layoutAlgorithm: specify layout algorithm

Network.**drawWeightedDiagram**(*show=True*, *savePath=None*, *dpi=150*)
Draw weighted reaction network based on frequency of reactions

Parameters

- **show** – flag to show the diagram

- **savePath** – path to save the diagram
- **dpi** – dpi settings for the diagram

Returns allRxn list of all reactions in the list of models presented as a pair of reactants and products

Returns count normalized count of reactions in allRxn throughout the list of models

Network.**drawNetworkGrid**(*nrows*, *ncols*, *auto=False*, *show=True*, *savePath=None*, *dpi=150*)
Plot a grid of network diagrams

Parameters

- **nrows** – number of rows
- **ncols** – number of columns
- **auto** – Automatically configure nrows and ncols based on the number of models. Overrides nrows and ncols.
- **show** – flag to show the diagram
- **savePath** – path to save the diagram
- **dpi** – dpi settings for the diagram

Network.**getLayout**(*returnState=False*)
Return the layout of the model

Network.**reset**()
Resets all properties

Network.**savefig**(*path*, *dpi=150*)
Save weighted network diagram to specified location

Parameters

- **path** – path to save the diagram
- **dpi** – dpi settings for the diagram

Network.**setLayout**(*pos*)
Set custom layout and bypass whe is generated by the layout algorothm

Parameters **pos** – Dictionary of all nodes and corresponding coordinates

5.3 Miscellaneous

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Index