TMgen Documentation

Release 0.1

Victor Heorhiadi

September 21, 2016

Contents

TMgen is a python library designed to easily generate network traffic matrices. The library supports a number of TM models, export to a number of cross-language formats, and very basic visualization tools.

CHAPTER 1

Quickstart

This document oulitnes how to easily start using the TMgen tool.

1.1 Download and installation

Since tmgen in still under developmnet please install from the github repository

1. Clone

```
git clone https://github.com/progwriter/tmgen
```

2. Install using pip

```
cd tmgen
pip install .
```

1.2 Example usage

TMgen defines a `TrafficMatrix` object that is returned by all generator functions. Internally it contains a 3-d numpy array which contains volume of traffic between origin-destination pairs for different time epochs. For TM models that do not have a time component, the third dimension is of size 1.

Quick overview on how to use them is given here, for full details please see the *TMgen API*

1.2.1 Generating a traffc matrix

Lets generate a uniform traffic matrix for a network with 3 nodes:

```
>>> from tmgen import uniform_tm
>>> tm = uniform_tm(3, 100, 300)
>>> print tm
array([[[ 219.73016387],
        [ 161.41332385],
        [ 120.68272977]],
       [[ 246.9339771 ],
        [ 189.98348848],
        [ 290.19555679]],
       [[ 119.98074672],
```

```
       [ 150.05173824],
       [ 255.83845338]]])
```

This gives us a 3x3x1 array with values between 100 and 300 — volume for each node pair, with only one time epoch.

1.2.2 Accessing tm entries

TMgen gives you a number of ways to access the tm values. Lets generate an exponentiam TM with the mean volume of 500 and 2 time epochs.

```
>>> from tmgen import exp_tm
>>> tm = exp_tm(3, 500, 2)
```

Accessing the *matrix* attrbute gives us the underlying Numpy array:

```
>>> tm.matrix
array([[[  6.37997965e+02,   1.09182535e+02],
        [  3.14477723e+02,   8.80934257e+02],
        [  3.48359849e+02,   1.00303448e+03]],
       [[  6.51216211e+02,   1.16041768e+03],
        [  3.16695016e+02,   6.97480254e-01],
        [  3.00624933e+02,   1.26349570e+02]],
       [[  1.43754204e+03,   6.61064394e+00],
        [  3.31300472e+02,   1.12039376e+02],
        [  5.79562994e+02,   4.57798655e+01]]])
```

Also we can request a tm at a specific epoch (0-indexed):

```
    >>> tm.at_time(1)
    array([[  1.09182535e+02,   8.80934257e+02,   1.00303448e+03],
           [  1.16041768e+03,   6.97480254e-01,   1.26349570e+02],
           [  6.61064394e+00,   1.12039376e+02,   4.57798655e+01]])
```

Or the values between any node pair:

```
>>> tm.between(0,2,'all')
array([  348.35984873,  1003.03447668])
>>> tm.between(0,2,'max')
1003.0344766776753
>>> tm.between(0,2,'mean')
675.69716270379729
```

See *TMgen API* for functions.

1.2.3 Saving/Loading a traffic matrix

TMs can be easily loaded to pickle and from pickle:

```
>>> tm.to_pickle('mytm')
>>> tm = TrafficMatrix.from_pickle('saved_tm')
```

CHAPTER 2

TMgen API

2.1 Generators

tmgen.models.**modulated_gravity_tm**(*int num_pops*, *int num_tms*, *double mean_traffic*, *double pm_ratio=1.5*, *double t_ratio=0.75*, *double diurnal_freq=0.041666666666666664*, *double spatial_variance=100*, *double temporal_variance=0.01*) → TrafficMatrix

Generate a modulated gravity traffic matrix with the given parameters

Parameters

- **num_pops** – number of Points-of-Presence (i.e., origin-destination pairs)
- **num_tms** – total number of traffic matrices to generate (i.e., time epochs)
- **mean_traffic** – the average total volume of traffic
- **pm_ratio** – peak-to-mean ratio. Peak traffic will be larger by this much (must be bigger than 1). Default is 1.5
- **t_ratio** – trough-to-mean ratio. Default is 0.75
- **diurnal_freq** – Frequency of modulation. Default is 1/24 (i.e., hourly) if you are generating multi-day TMs
- **spatial_variance** – Variance on the volume of traffic between origin-destination pairs. Pick someting reasonable with respect to your mean_traffic. Default is 100
- **temporal_variance** – Variance on the volume in time

Returns

tmgen.models.**random_gravity_tm**(*int num_pops*, *double mean_traffic*, *double spatial_variance*) → TrafficMatrix

tmgen.models.**gravity_tm**(*int num_pops*, *ndarray populations*, *double total_traffic*) → TrafficMatrix

Compute the gravity traffic matrix

Parameters

- **num_pops** – number of poins of presence
- **populations** – array with populations (weights) for each PoP
- **total_traffic** – total amount of traffic in the network

Returns TraffixMatrix object

tmgen.models.**uniform_tm**(*int num_pops*, *double low*, *double high*, *int num_epochs=1*) → TrafficMatrix

Return a uniform traffic matrix. Entries are chosen independently from each other, uniformly at random, between given values of low and high.

Parameters

- **num_pops** – number of points-of-presence
- **low** – lowest tm value
- **high** – highest tm value
- **num_epochs** – number of

Returns TrafficMatrix object

tmgen.models.**exp_tm**(*int num_pops*, *double mean_traffic*, *int num_epochs=1*) → TrafficMatrix

tmgen.models.**spike_tm**(*int num_pops*, *int num_spikes*, *double mean_spike*, *int num_epochs=1*) → TrafficMatrix

Generate a traffic matrix using the spike model.

Parameters

- **num_pops** – number of nodes in the network
- **num_spikes** – number of ingress-egress spikes. Must be fewer than $numpops^2$
- **mean_spike** – average volume of a single spike
- **num_epochs** – number of time epochs

Returns

2.2 TrafficMatrix instance

class tmgen.tm.**TrafficMatrix**(*ndarray tm*)

Represents a traffic matrix.

at_time(*self*, *int t*) → ndarray

Return a numpy array reprenseting a single TM at epoch t.

Parameters **t** – the number of the epoch (0-indexed).

Return type numpy array

between(*self*, *int o*, *int d*, *modestr='all'*)

Return the traffic matrix between the given ingress and egress nodes. This method supports multiple temporal modes: 'all', 'min', 'max', and 'mean'

Parameters

- **o** – source node (origin)
- **d** – destination node
- **modestr** – temporal mode

Returns Numpy array if modestr=='all', double otherwise

static from_pickle(*fname*)

Load a TrafficMatrix object from a file

Parameters **fname** – the file name on disk

Returns new TraffixMatrix

matrix
matrix: numpy.ndarray

mean(*self*) → TrafficMatrix
Returns a new traffic matrix that is the average across all epochs.

num_epochs(*self*) → int

Returns The number of epochs in this traffic matrix

num_pops(*self*) → int

Returns The number of PoPs in this traffic matrix

to_pickle(*self*, *fname*)
Save the matrix to a python pickle file

Parameters **fname** – the file name

worst_case(*self*) → TrafficMatrix
Return a new, single-epoch traffic matrix that chooses maximum volume per OD pair (in time)

Returns a new TrafficMatrix

2.3 Plotting

t