Instructions to run SSA/D-SSA for Influence Maximization

Information:

Version 1.0: Implementation of SSA/D-SSA Algorithms for Influence Maximization under Independent Cascade(IC) or Linear Threshold(LT) model. For more details about SSA as well as D-SSA, please read our paper: "H. T. Nguyen, M. T. Thai, and T. N. Dinh, Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-scale Networks , in Proc. of the annual ACM SIGMOD/PODS Conference, 2016"

Contact Authors: Hung T Nguyen (hungnt@vcu.edu)
Thang N Dinh (tndinh@vcu.edu)

Terms of use:

Requirements:

In order to compile all the tools, it requires GCC 4.7.2 and later (which also support OpenMP 3.1).

Compile:

Use `make' command to compile everything

How to use:

This package offers a set of functions to use in order to find a seed set of given size k:
0. (Optional) Computing edge weights (probabilities) as described in the experiments:
./format <input file> <output file> 1

<input file>: the path to text file in edge list format with no weights: the first line contains the number of nodes n and number of edges m, each of the next m lines describes an edge following the format: <src> <dest>. Node index starts from 1.
<output file>: the path to binary output file
The last parameter (1) means the input graph is considered as directed.

1. Conversion from a text format to binary file
./el2bin <input file> <output file>

<input file>: the path to text file in weighted edge list format: the first line contains the number of nodes n and number of edges m, each of the next m lines describes an edge following the format: <src> <dest> <weight>. Node index starts from 1.
<output file>: the path to binary output file

2. Run SSA to find the seed sets
./SSA [Options]
./DSSA [Options]

Options:

-i <binary graph file>
specify the path to the binary graph file (default: network.bin)

-o <seed output file>
specify the path to the output file containing selected seeds (default: network.seeds)

-k <number of seeds>
number of selected seed nodes in the seed set(default: 1)

-epsilon <epsilon value used>
epsilon value in (epsilon,delta)-approximation (see our paper for more details, default: 0.1)

-delta <delta value used>
delta value in (epsilon,delta)-approximation (default: 1/n)

-m <model>
diffusion model (LT or IC, default: LT)

Output format:
The outputs are printed on standard output stream in the following order

Seed Nodes: <list of selected seed nodes>

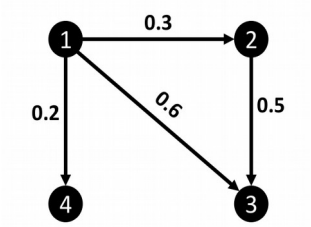
Influence: <Spread of Influence of the select seed set>
Time: <running time in seconds>
Memory: <peak memory used>

3. (Optional) Verify influence spread of a seed set - returns a (epsilon, 1/n)-estimate of the influence:
./verifyInf <binary graph file> <seed file> <epsilon> <number of threads> <model: LT or IC>

Examples on a toy network: find seed set having 2 seed nodes on the graph network.txt:

The sample network network.txt, in this case, contains only 4 nodes and 4 edges and is formated as follows:

```
4 4
1 2 0.3
2 3 0.5
1 3 0.6
1 4 0.2
```



1. Convert to binary file:
./el2bin network.txt network.bin

2. Run SSA/DSSA with k=2, epsilon=0.1,delta=0.01:

./SSA -i network.bin -k 2 -epsilon 0.1 -delta 0.01 -m LT

The output after running SSA:

Seed Nodes: 1 2

Influence: 3.01
Time: 0.01s
Memory: 20.7422 MB

Here, the selected seed set is {1;2} with the estimated influence of 3.01. SSA took 0.01 second to run and consumed 20.7422 MB of memory.

3. Verify influence spread with epsilon=0.01, assume that the seed nodes are put in network.seeds:

./verifyInf network.bin network.seeds 0.01 1 LT

The output after running SSA:

3.01

Examples on the billion-scale Twitter network: find seed set of size 100 on Twitter:

1. Convert to binary file: The .bin file and another dataset in edge list format are provided in the following link to save the converting time:

https://drive.google.com/folderview?id=0B1c5Sj253lraX2VLZkc5S05Fak0&usp=sharing

2. Run SSA/DSSA with k=100, epsilon=0.1,delta=0.01:

./SSA -i twitter.bin -k 100 -epsilon 0.1 -delta 0.01 -m LT -o twitter.seeds

The output after running SSA/DSSA:

Find Max-Coverage
Max Samples 40000
Seed Nodes: 1653 1037948 5874844 1811269 5925043 6035057 1022452 1803885 1437 3666322 7877690 3153931 1611 1022172 8308220 1025811 1039591 2581430 10510942 8 15145269 1834246 3620928 1938 22635043 971 4118241 4061596 1829999 1005817 1030336 1685 29745274 1894919 2 11509616 1042616 18831324 6021354 25237289 19874451 4642 16427353 5893972 18817674 32361295 1040848 9022532 20818739 1 19495866 1029853 8348215 5326641 1768479 4062275 1005415 4377946 1030671 4619539 31003592 5380592 22265695 23515823 339 1777545 1768428 5935586 2809 1238699 383 7634442 16016290 33835006 12308758 20329450 20788493 15745580 6259761 31748090 17301 8209508 4070836 1853148 16909 12369660 12835069 4392788 147332 15137140 5299 18455905 4206106 1075246 25135 1003381 19013935 5989290 5307431 3115653
Influence: 20743851.85
Time: 3.6s
Memory: 16963.7 MB

3. Verify influence spread with epsilon=0.01, assume that the seed nodes are put in twitter.seeds:

./verifyInf twitter.bin twitter.seeds 0.01 1 LT

The output after running SSA:

20727234.45