Supplementary Documents For Lower Bounds on Rate of Convergence of Matrix Products in All-Pairs Shortest Path of Social Network

1. Dataset:

A dataset for Chinese Movie Actors Sina Weibo Following Relationship 2011-2015.

- `dataset` folder.
  - actorId.txt
    This file contains profiles of movie actors who has an account in Sina Weibo. Following information are provided for each actor: the number of fans(10k., Sina Weibo verified name, Sina Weibo Account Id, Sina Weibo Nickname, actor self description. All actors have at least 10,000 of fans, it shows that all users in this dataset are famous to some point.
- `data` folder.
  - train.npz
    This file is a compressed adjacent NumPy matrix of 8508 actors. The adjacent matrix `m` stores connection for each node pairs. For all node `u`, set `m[u,u]=0`, and for any node `u` follows node `v`, then set `m[u, v]=1`, other values in matrix `m` are set to `m[u, v]=3.4028235e+38` which is the maximum value of floating-point in 32bit.
  - apsp.npy This file is extract from train.npz.

2. Library Requirement

2.1 Programming Environment

Installation of python 3.6+, NVIDIA CUDA10.2, JDK 1.8.0, Maven 3.6.1 is required.

2.2 Python Dependant

See requirements.txt

2.3 Java jar build.

Use maven to build executing jar. Using command: `mvn package -f pom[-gpu].xml`, then get the result jar file `apsp-cpu.jar` or `apsp-gpu.jar`.

3. Experimental Guideline

3.1 Floyd-Warshal

Command parameters explained as follows:
1. input matrix in numpy format
2. diameter of the network
3. output matrix
4. algorithm name

- floydwarshall for Floyd-Warshal all pairs shortest path algorithm
- powerlawbound for this paper's algorithm.

Full command as follows:

```
java -jar apsp-cpu.jar matrix.npy 9 floydwarshall
```

3.2 Alon N

- CPU implementation

```
docker run --rm --gpus=all -it powerlawapsp:1.0 python3 train.py
-c config/dpmm_config.json
```

- GPU implementation

```
docker run --rm --gpus=all -it powerlawapsp:1.0 python3 train.py
-c config/dpmm_gpu_config.json
```

3.3 PowerLawBound

Build the dockers

- pytorch:1.1 docker file: See `Dockerfile-pytorch_1.1`
- pwoerlawapsp:1.0 docker file: See `Dockerfile`

An example of execute command:

```
docker run --rm --gpus=all -it powerlawapsp:1.0 python3 train.py
-c config/{config}.json
```

3.3.1 PowerLawBound-CPU-NumPy

Use config file `naive_apsp_config.json`

3.3.2 PowerLawBound-GPU-CuPy

Use config file `naive_apsp_gpu_config.json`

3.3.3 PowerLawBound-CPU-SciPy-sparse-Numpy

Use config file `apsp_config.json`

3.3.4 PowerLawBound-GPU-CuPy-cuSparse-Cupy

Use config file `apsp_gpu_config.json`

3.3.5 PowerLawBound-GPU-CUBLAS

```
java -jar apsp-gpu.jar matrix.npy 9
```

3.3.6 PowerLawBound-CPU-OPENBLAS

```
java -jar apsp-cpu.jar matrix.npy 9
```

4. Dockers for Reproduction

4.1 Docker for Powerlaw

Use docker file `powerlawapsp\Dockerfile`

4.2 Docker file for PowerLawBound-GPU-CUBLAS

Use docker file `powerlawapspjava\Dockerfile-gpu`

4.3 Docker file for PowerLawBound-CPU-OPENBLAS

Use docker file `powerlawapspjava\Dockerfile`