Shooting Large-scale Traffic Engineering by Combining Deep Learning and Optimization Approach

This is a Pytorch implementation of LO-TE presented on CoNEXT 2025. LO-TE provides a two-step approach to efficiently resolve large-scale traffic engineering problems: obtaining an initial solution and refining it to achieve a near-optimal TE solution.

Getting started

Hardware requirements

- **CPU**: 16+ cores (more cores recommended for parallel training with multiple samples).
- **Memory**: 64+ GB RAM (512+ GB recommended for larger topologies).
- **GPU**: 8+ GB memory (reduce `--max-flow-num` to lower GPU memory usage, such as setting it to 1/n of the total number of traffic demands).
- **Operating System**: Linux (tested on Ubuntu 20.04 and 22.04).

Dependencies

To set up the environment, ensure the following dependencies are installed:

1. **Install PyTorch**
   Follow the official installation guide for your system: PyTorch Installation.

2. **Install PyTorch Geometric**
   Refer to the official documentation for installation instructions: PyTorch Geometric Installation.

3. **Install Gurobi and obtain a license**

   - Install the `gurobipy` package.
   - Acquire a Gurobi license for academic use here: Gurobi Academic License.

4. **Python environment setup**

   - The required Python dependencies for training and evaluation are listed in `requirements.txt`.
   - However, we recommend installing the dependencies manually using `pip` to ensure compatibility with your operating system and version.

Download traning and testing data

We have uploaded the compact dataset for validating the functionality of the artifact. We have also uploaded the full dataset used in LO-TE paper, including topology information, traffic demands, candidate paths，and labeled/unlabeled data for training and testing.

We recommend downloading the necessary data and copying it into the `./data` directory before running the program.

Training and Testing LO-TE

We have listed the training and testing commands in `run_minmlu.sh`, `run_maxthrpt.sh`, and `run_minweight.sh`. We show an example of traning and testing LO-TE on Cogentco topology and traffic burst model below:

- Pretrain a general model with supervised learning

```
python3 train.py --objective min_mlu --init-te-solution his --log-dir
general-minmlu-his --train-data-dir ./data/train/minmlu/general --training-
epochs 10 --training-mode SL  --use-cuda --T 1 --max-flow-num 1000000
```

- Unsupervised learning for a specific topology and traffic model

```
python3 train.py --objective min_mlu --init-te-solution his --log-dir
minmlu_his_Cogentco_traffic_burst_USL --model-load-dir general-minmlu-his -
-train-data-dir ./data/train/minmlu/Cogentco_traffic_burst_unlabel100 --
training-epochs 50 --training-mode USL  --use-cuda --T 1  --num-sample-
process 5 --K 5 --alpha 1 --max-flow-num 1000000
```

- Testing LO-TE

```
python3 test.py --objective min_mlu --init-te-solution his --use-cuda    --
model-load-dir minmlu_his_Cogentco_traffic_burst_USL --test-data-dir
./data/test/minmlu/Cogentco_traffic_burst   --T 1 --max-flow-num 10000000
--alpha 10
```

The summary of performance metrics (e.g., maximum link utilization, throughput, average transmission weight) of the finetuned solution and the optimal solution will be listed at the end of output log of "test.py".

Generating training and testing data by yourself

We have also provided the major data generation codes for generating your own training and testing data.

- `./data_generation/MCF_data` are used for generate training data and testing data for LO-TE.
- `./data_generation/path_data` are used for generate candidate paths for LO-TE. Refer to `./data_generation` for more details.

If you have any questions, please post an issue or send an email to chenyiliu9@gmail.com.

Citation

```
@inproceedings{liu2025lote,
  title={Shooting Large-scale Traffic Engineering by Combining Deep
```

```
Learning and Optimization Approach},
  author={Liu, Chenyi and Deng, Haotian and Aggarwal, Vaneet and Yang, Yuan
and Xu, Mingwei },
  booktitle=={Proceedings of the ACM CoNEXT 2025 Conference},
  year={2025},
  publisher={ACM}
}
```