

A Grouping Algorithm for Training Tree-Shaped Models on Multiple GPUs with High Efficiency

Cai-Feng Lin¹, Ding-Yong Hong², Pangfeng Liu¹,
Jan-Jan Wu², Tzu-Hsien Tsai¹

¹Department of Computer Science and Information Engineering
National Taiwan University

²Institute of Information Science
Academia Sinica, Taipei, Taiwan

October 20, 2025

- We study how to distribute tree-shaped data across batches and devices to reduce the training time of graph neural networks.
- We introduce a cost model to formulate the relation between the training time and the data distribution.
- We prove that, under this cost model, finding the optimal distribution of tree-shaped data is NP-complete.
- We propose a heuristic algorithm to derive a data distribution that achieves a speedup over the distribution from PyTorch method.

Importance

- Graph Neural Networks (GNNs) have many important applications.
 - Transportation Systems¹
 - Mechanics²
 - Combinatorial optimization³
- Given these important applications, reducing the training time of GNNs is essential.

¹Amit Roy et al. "SST-GNN: simplified spatio-temporal traffic forecasting model using graph neural network". In: *Pacific-asia conference on knowledge discovery and data mining*. Springer. 2021, pp. 90–102.

²Nolan Black and Ahmad R Najafi. "Learning finite element convergence with the multi-fidelity graph neural network". In: *Computer Methods in Applied Mechanics and Engineering* 397 (2022), p. 115120.

³Morris Yau et al. "Are graph neural networks optimal approximation algorithms?" In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 73124–73181.

The Batch Training Procedure with Data Parallelism

- The dataset is divided into batches.
- A batch is processed by multiple devices.
- The blue slots are the time for updating the model parameters.

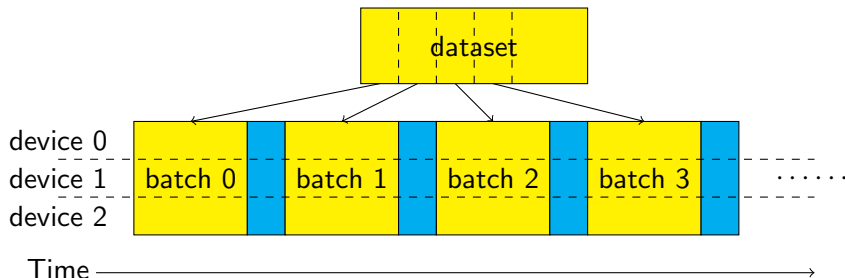


Figure 1: Batch training with data parallelism

The Main Problem

- How to distribute the tree-shaped data across batches and devices so that the training time of GNNs is reduced?

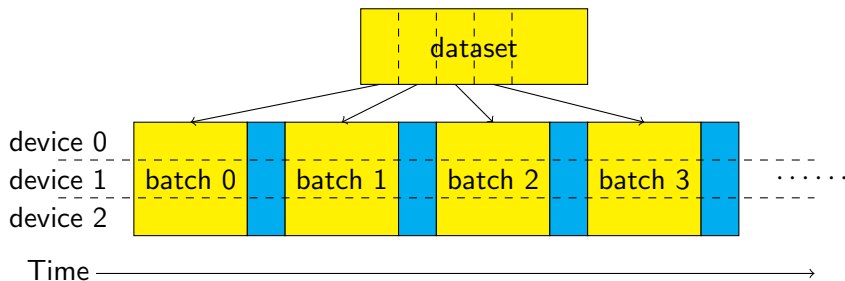


Figure 2: Batch training with data parallelism

The Data Distribution from PyTorch

- Let the number of devices be m , and the batch size be s .
- For simplicity, we refer to the tree-shaped data as trees.
- PyTorch distributes the i -th tree to the $(i \bmod m)$ -th device.
- PyTorch distributes the i -th tree in each device to the $\lfloor \frac{i}{s} \rfloor$ -th batch.
- Therefore, PyTorch does not consider the properties of data.
- However, the training time of GNNs is related to the properties of trees.

- The Deep Graph Library (DGL) offers an efficient schedule to process a forest on a single device, as follows.
- The leaves of every tree are initially labeled as ready nodes, and the others are labeled as hidden nodes.
- The following two steps are run iteratively until all nodes in the forest are processed.
 - ① Process a limited number of ready nodes.
 - ② If all children of a hidden node are processed, then this hidden node is labeled as a ready node.

An Illustration to the Schedule of DGL

- Figure 3 illustrates the schedule of DGL.

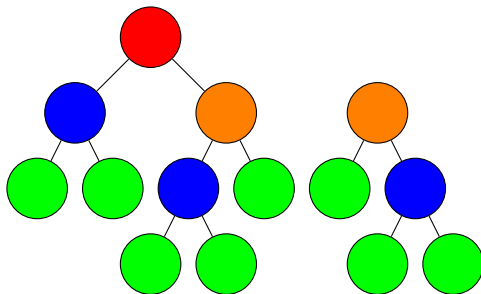


Figure 3: An illustration to the schedule of DGL

An Improvement over PyTorch Distribution

- For simplicity, we refer to the training time as cost.
- According to the schedule of DGL, the following two factors affect the cost of a forest on a single device.
 - the maximum depth among the trees in the forest
 - the number of nodes in the forest
- Therefore, if we consider the properties of trees when distributing trees to batches and devices, the cost of GNN training may be reduced.

Our Contributions

- We propose a cost model which suggests the consideration of tree properties when we distribute trees across batches and devices.
- Under this cost model, we prove that finding the optimal distribution is NP-complete.
- We propose a heuristic algorithm to distribute trees to batches.
- We derive a 4-approximation algorithm to distribute trees in a batch to multiple devices.
- The distribution derived from our algorithms achieves a speedup over the distribution from PyTorch.

The Cost Model

- Let B be the number of batches.
- Since batches are processed sequentially, the total cost C of the training process is the sum of the costs of all batches.

$$C = \sum_{1 \leq i \leq B} (\text{the cost of the } i\text{-th batch}) \quad (1)$$

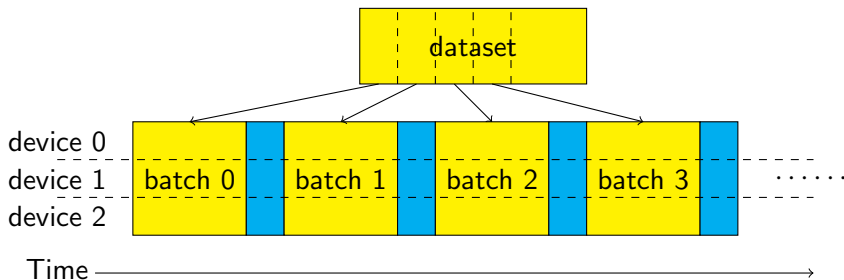


Figure 4: Batch training with data parallelism

The Cost of a Single Batch

- Let $F_{i,j}$ be the set of trees assigned to the j -th device in the i -th batch.
- Let m be the number of devices.
- Let $c(F)$ be the cost of processing a forest F on a single device.
- Since devices can run in parallel, the cost of a batch is the maximum processing time across all devices.

$$\text{the cost of the } i\text{-th batch} = \max_{1 \leq j \leq m} (c(F_{i,j})) \quad (2)$$

$$C = \sum_{1 \leq i \leq B} \max_{1 \leq j \leq m} (c(F_{i,j})) \quad (3)$$

The Cost of Processing a Forest on a Single Device

- We observe that $c(F)$ is linear, but not bilinear, to the number of nodes and the maximum depth among the trees in the forest F .

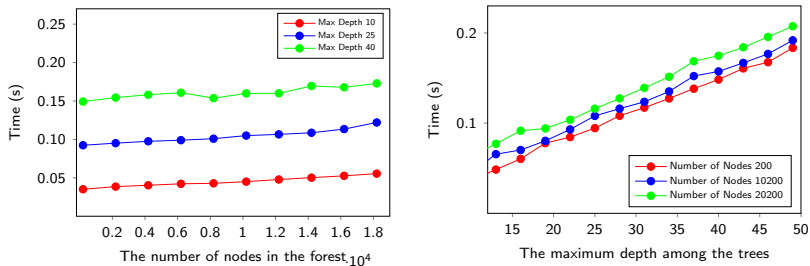


Figure 5: Two properties that affects the training time

The Tree Grouping Problem

- Let $s(t)$ be the number of nodes in the tree t .
- Let $d(t)$ be the depth of the tree t .
- The cost $c(F)$ can be expressed as shown in Equation 4, where α , β , and γ are constants.

$$c(F) = \alpha \sum_{t \in F} s(t) + \beta \max_{t \in F} (d(t)) + \gamma \quad (4)$$

- The *Tree Grouping Problem* is to decide the batch and device to which each tree is distributed so that the total cost C is minimized.

$$C = \sum_{1 \leq i \leq B} \max_{1 \leq j \leq m} (c(F_{i,j})) \quad (5)$$

- We reduce the tree grouping problem to the optimization version of the 2-partitioning problem.
- The optimization version of 2-partitioning problem divides a set of integers into two subsets so that the larger subset sum is minimized.
- If we set the number of batches to one and the number of devices to two, then the tree grouping problem divides a forest into two forests such that the larger cost of the forest is minimized.

$$C = \sum_{1 \leq i \leq B} \max_{1 \leq j \leq m} (c(F_{i,j})) = \max(c(F_{1,1}), c(F_{1,2})) \quad (6)$$

- Let $\alpha = 1$, $\beta = 0$, and $\gamma = 0$. The cost of processing a forest on a single device becomes the number of nodes in the forest.

$$c(F) = \alpha \sum_{t \in F} s(t) + \beta \max_{t \in F} (d(t)) + \gamma = \sum_{t \in F} s(t) \quad (7)$$

- Under this parameter setting, a tree can be represented by an integer equal to its number of nodes
- The cost of processing a forest on a single device becomes the sum of the integers representing the trees in the forest.

Distribute Trees to Batches

- Distribute the i -th deepest tree into the $\lfloor \frac{i}{k} \rfloor$ -th batches, where k is the maximum number of trees in a batch.
- We only consider the depth because, through experiments, we find that the value of β is larger than the value of α .


$$c(F) = \alpha \sum_{t_i \in F} s(t_i) + \beta \max_{t_i \in F} (d(t_i)) + \gamma \quad (8)$$

Distribute Trees within a Batch to Devices

- Let m be the number of devices.
- Distribute each of the m deepest trees to different devices.
- Distribute the remaining trees sequentially to the devices with the lowest current cost.

4-approximation

- Let D be the set consisting of all distributions from trees to devices.
- Let D^* , a subset of D , consists of the distributions that map each of the m deepest trees to m different devices.
- We prove that our distribution is a 4-approximation to the best distribution in D through the following two steps.
 - ① We prove that our distribution is a 2-approximation of the optimal distribution in D^* by showing that the list scheduling algorithm⁴ produces the same distribution as ours.
 - ② We prove that the best distribution in D^* is a 2-approximation to the best distribution in D .

⁴Ronald L. Graham. “Bounds on multiprocessing timing anomalies”. In: *SIAM journal on Applied Mathematics* 17.2 (1969), pp. 416–429. 

Experiment

- The devices are eight Tesla V100-32GB GPUs, Intel Eeon Gold 6154 CPUs, and the 720GB system memory. The GPUs are connected by NVLink.
- The model is Tree-LSTM⁵.
- The dataset is Stanford Sentiment TreeBank.
- The experiment is implemented using the PyTorch and DGL⁶ framework.

⁵Kai Sheng Tai, Richard Socher, and Christopher D. Manning. “Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. Beijing, China, July 2015, pp. 1556–1566.

⁶Minjie Yu Wang. “Deep graph library: Towards efficient and scalable deep learning on graphs”. In: *ICLR workshop on representation learning on graphs and manifolds*. 2019.

Two GPUs

- Our algorithm achieves the speedup between 1.61 and 1.86 when the number of GPUs is two.

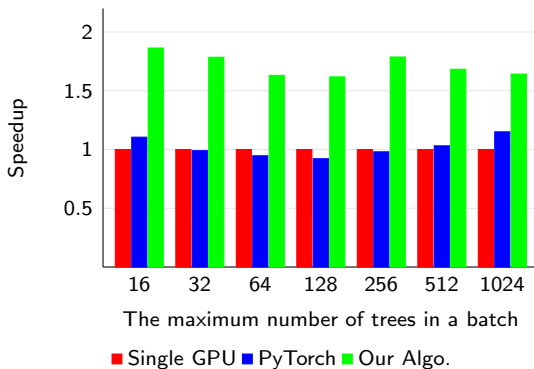


Figure 6: The speedup of our distribution

Four GPUs

- Our algorithm achieves the speedup between 2.53 and 3.43 when the number of GPUs is four.

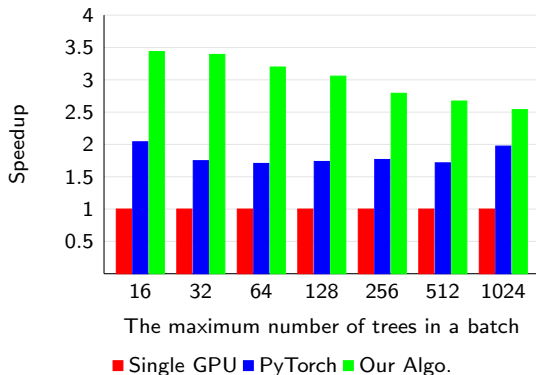


Figure 7: The speedup of our distribution

Eight GPUs

- Our algorithm achieves the speedup between 3.68 and 7.25 when the number of GPUs is eight.

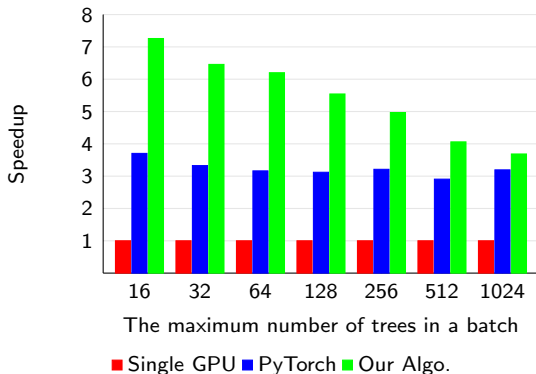


Figure 8: The speedup of our distribution

Conclusions

- We study how to distribute tree-shaped data across batches and devices to reduce the training time of GNNs.
- We propose a cost model to formulate the relation between training time and data distribution.
- We prove that finding the optimal distribution is NP-complete.
- We propose a heuristic algorithm to distribute trees into batches.
- We derive a 4-approximation algorithm to distribute trees within a batch into devices.
- We show that our data distribution achieves a speedup up over the distribution from PyTorch.