Modeling Multi-Destination Trips with Sketch-Based Model

Michał Daniluk
Synerise
Warsaw University of Technology
Poland
michal.daniluk@synerise.com

Barbara Rychalska
Synerise
Warsaw University of Technology
Poland
barbara.rychalska@synerise.com

Konrad Gołuchowski
Synerise
Poland
konrad.goluchowski@synerise.com

Jacek Dąbrowski
Synerise
Poland
jack.dabrowski@synerise.com

ABSTRACT

The recently proposed EMDE (Efficient Manifold Density Estimator) model achieves state of-the-art results in session-based recommendation. In this work we explore its application to Booking.com Data Challenge competition. The aim of the challenge is to make the best recommendation for the next destination of a user trip, based on dataset with millions of real anonymized accommodation reservations. We achieve 2nd place in this competition. First, we use Cleora - our graph embedding method - to represent cities as a directed graph and learn their vector representation. Next, we apply EMDE to predict the next user destination based on previously visited cities and some features associated with each trip. We release the source code at: https://github.com/Synerise/booking-challenge.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Booking.com Data Challenge, neural networks, deep learning, network embeddings, recommendation systems

1 INTRODUCTION

The goal of the challenge [6] is to predict the final city of each trip using a dataset based on millions of real anonymized accommodation reservations. The released train set contains 1,166,835 unique reservations within 217,686 trips and 39,901 unique cities in 195 countries. A list of features is presented in Table 1.

The evaluation dataset is constructed similarly, however the city ID of the final reservation of each trip is concealed and requires a prediction. The test set consists of 378,667 reservations with at least 4 consecutive reservations. Predictions were made for 70,662 unique trips. The test set was drawn from the same temporal distribution as a training set.

Evaluation. The metric used for performance evaluation is precision at 4 (Precision@4). The score is understood as the average of the per-sample scores, which are either 1 if the predicted city is in top 4 predictions, or 0 otherwise. The teams were allowed to make only 2 submissions on the final test set.

Solution. We frame the problem of route prediction as a session-based recommendation task. Our contributions are as follows:

- We propose to represent cities as nodes in a directed graph, whose edges represent trips between two cities. We compute city embeddings with Cleora [18], a fast and efficient network embedding technique.
- We apply EMDE [5] to recommend the next destination based on representations of previous cities and additional numerical and categorical features such as the length of a trip or the type of user's device.
- We analyze the effectiveness and challenges of our method.

Overall challenge results. Our approach takes 2nd place out of 38 in this challenge with the final Precision@4 score of 0.5780, compared to the leading score of 0.5939 and surpasses the 3th and 4th place solution scores of 0.5741 and 0.5566, respectively.

Table 1: Dataset statistics.

Feature	Number of unique values
User ID	200153
Trip ID	217686
Check-in date	425 (31.12.2015-27.02.2017)
Check-out date	425 (1.01.2016-28.02.2017)
Affiliate ID	3254
Device Class	3
Booker Country	5
Hotel Country	195
City ID	39901

2 RELATED WORK

Within the past few years, deep learning recommendation models have had difficulties in achieving good results in recommender system competitions [9]. Instead, the winning solutions often involved gradient boosting models with substantial feature engineering efforts. Furthermore, it has been shown that conceptually simpler techniques, e.g. based on nearest neighbors [13] often outperforms neural approaches on multiple datasets.

Many deep learning session-based recommendation models consider recommendations as a sequential problem, applying recurrent networks (LSTM/GRU) [8, 11, 16, 17], which are known to have difficulties in learning long-term dependencies and scale poorly to growing item sets and increasing sequence lengths [19]. On the other hand, graph-based models (GNN) [3, 21, 22] cast recommendations as a graph traversal problem. Those methods exhibit a number of specific efficiency-related problems such as *neighborhood explosion* (the number of neighbors often grows exponentially

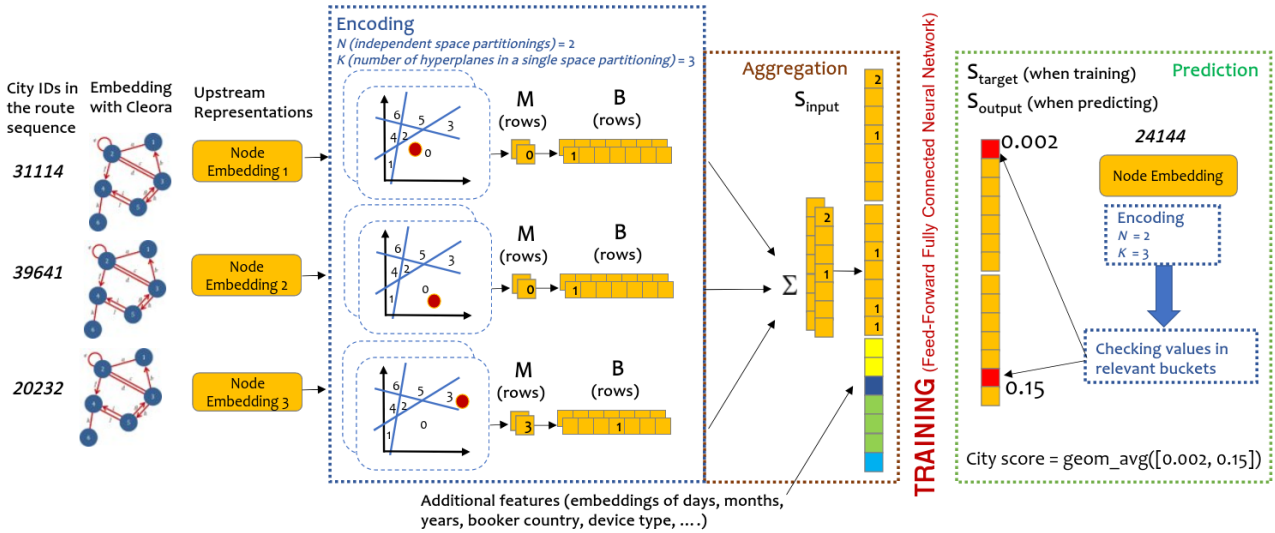


Figure 1: EMDE architecture overview.

when increasing node distances are considered). Such problems demand additional remedial measures which often hurt performance [1, 2, 23]. Yet, as the sequential aspect of recommendation is considered vital, most efforts are focused on researching even more complex neural network architectures in order to represent the ordered relations accurately.

Instead of focusing on sequential aspect, our Efficient Manifold Density Estimator (EMDE) [5] learns users' behaviors as weighted item sets: an aggregate vector (*sketch*), which is a histogram-like vector representation of densities on manifolds spanned by embedding vectors. EMDE allows to compress a variable-length sequence of cities into a simple 1-D vector of constant size, which can be fed into a simple shallow feed-forward neural network. It achieves competitive results on many datasets, and has the ability to use a multi-modal vector representation of each destination.

3 ALGORITHM

The first step of our algorithm is obtaining information-rich city embeddings. We observe that the route prediction dataset can be represented as a directed graph, where each city is a node, and each directed edge represents a trip from one city to another. The dataset has a graph structure with a lot of revisited cities in a single trip, 62% of them have at least one cycle. In addition, representing data as a graph gives the ability to have a connection between different trips. In order to compute node embeddings we apply Cleora [18], which relies on multiple iterations of normalized, weighted averaging of each node's neighbor embeddings, followed by normalization across dimensions. The procedure is fast and we empirically find that it produces high quality embeddings (§4.3).

3.1 EMDE

Efficient Manifold Density Estimator (EMDE) introduced in [5] is a probability density estimator for high dimensional spaces, which is inspired by Count-Min Sketch algorithm (CMS) and locality sensitive hashing (LSH). It works by dividing the item embedding space into regions and assigning items to specific regions (buckets) based on similarity of their embedding vectors. EMDE operates on structures analogous to multidimensional histograms, called *sketches*.

An example of the full EMDE algorithm run is depicted in Figure 1. Full explanation and the intuitions behind the algorithm are presented in [5]. Below we give a brief summary of the algorithm steps:

(1) **Encoding.** EMDE operates on manifolds spanned by embedding vectors. In our case, these are embeddings of cities understood as graph nodes. First, the embedding manifolds are partitioned with a data-dependent LSH method called DLSH. As a result, LSH regions are created primarily where the data are present. Each partitioning is done with K hyperplanes and is repeated N independent times (see `Encoding` section in Fig. 1). We denote K as the *sketch width* and N as *sketch depth*. The regions of the partitioning are analogous to hash buckets in CMS. While a single region is usually large (typically 64-256 regions are created per single partitioning), multiple independent partitionings allow to obtain a high resolution map of the manifold via intersection or ensembling.

(2) **Region Assignment.** The region IDs of each input cities are stored in matrices M, which are then binarized to obtain matrices B (notations are consistent with `Encoding` section in Fig. 1), which form the *sketches* of individual cities.

(3) **Aggregation.** The sketches of individual cities are aggregated with simple summation to obtain a sketch of the whole trip, represented by vector S_{input} (see `Aggregation` section in Fig. 1)

(4) **Training.** The aggregate sketches have constant size and thus can serve as input to a simple fully-connected feed-forward neural network, which is trained to output a sketch S_{target} of the hidden part of the trip.
(5) **Prediction.** Score prediction procedure reuses the sketches of all cities, which have been encoded in the first algorithm step. For each city, its relevant region IDs are stored in the matrices M. When an output sketch S_{output} is produced by the network, the values contained in relevant region IDs are retrieved for each city and averaged with the geometric mean to produce a final per-city score.

The advantages of using EMDE are numerous, especially for large datasets. The sketches are additive, and can accommodate any number of cities within a fixed-size representation. Sketch size is independent of the number of samples and original embedding dimensions. EMDE can easily incorporate multiple modalities of input data, as well as continuous and categorical features, by simple vector concatenation. Note that the aggregate sketches produced by EMDE inherently lose information on city ordering.

4 EXPERIMENTS

In this section, we first describe data preprocessing (§4.1), then we describe our model in details, present the results, and analyze the effectiveness of our method (§4.2).

4.1 Data Preparation

The goal of the competitions is to predict only the **final destination** of a each trip. However, in order to give the model more information, we augment the dataset to include prediction of all previous cities based on user history. For example, we split a four-city trip into three training examples, predicting the second, third and fourth city during the trip.

Train/Valid Split. We create our own validation set that imitates the hidden test set, by sampling out 70,662 trips from the train set. The validation set, same as the test set, has only final destinations as target. In addition, trips in validation and test sets consist of cities that appear only in the training set. Our training set includes also non-final targets from the validation set. All datasets are from the same temporal distribution.

Features. For each data point we compute the following categorical and continuous features: type of user's device, country from which the reservation was made, an ID of affiliate channels, country of the hotel, the length of stay in predicted city, the number of days since beginning of a trip, the number of days till the end of a trip, number of days since last booking, number of cities in a trip, week days of check-in and check-out, month, year.

4.2 Model

City embeddings. In order to obtain city embeddings, we represent the dataset as a directed graph of city trips. Each node in the graph denotes a city, and each directed weighted edge represents the journey between them. Weight of the edge denotes the number of trips from one city to another. We construct the graph from both training and testing examples (excluding the final missing cities). In order to embeded cities we use Cleora [18] with iteration number $i = 1$ and $i = 3$ and embedding length of 1024. As a result, each city is represented by two embeddings which are the input to the EMDE model.

EMDE Configuration. We encode all embeddings with sketches of depth $N = 40$ and width $K = 128$. Each embedding configuration (Cleora embeddings computed with $i = 1$ and $i = 3$) is encoded separately, representing two complementary modalities of data. We observe that adding random sketch codes (not based on LSH) for each item improves the model performance, allowing the model to separate very similar cities to differentiate their popularity.

Model. We train a three-layer residual feed forward neural network with 3000 neurons in each hidden layer, with leaky ReLU activations and batch normalization. The input of the network consists of:

(1) Three width-wise L2-normalized, concatenated sketches: *first city sketch* representing the first city in a trip, *prev city sketch* representing the previous city in a trip, and *all cities sketch* containing all other cities. We use the *first city sketch* to facilitate training, because in about 15% of training examples, the final city is the same as the first city. In order to create a representation of a user's behaviour in the *all cities sketch*, we aggregate the sketches of cities with a simple summation (as is normally done in EMDE), multiplying them with constant decay which reduces the influence of cities which are older in time.
(2) Normalized numerical features such as number of days since beginning of a trip or number of unique cities visited so far.
(3) Categorical features that are represented by the PyTorch `nn.Embedding` layer such as the day of the week of check-in, month, year or country from which the reservation was made. The size of the embedding depends on the number of unique feature values. We set it to 120 for previous hotel country and affiliate channel features. For other features, the size of the embedding is 20.
(4) A binary flag indicating if the target is the final destination (always true in case of the test set).

The output of the model is a sketch that represents our target city. The procedure of producing the *all cities sketch* is shown in Figure 1, also showing the addition of numerical and categorical features by simple concatenation.

Training. We train our model on a single nVidia GeForce RTX 2080 Ti 11GB RAM GPU card. Training takes circa 45 minutes on this configuration. We use AdamW optimizer [12] with first momentum coefficient of 0.9 and second momentum coefficient of 0.999[1] with an initial learning rate of 0.0005, weight decay of 0.01 and a mini-batch size of 128 for optimization.

Since the distribution of final destinations is different than distribution of non-final cities, we train the model in two stages: 1) using non-final target destinations, and 2) fine-tuning the model on the examples with final destinations. The model was trained for 2 epochs, and then fine-tuned again with smaller learning rate only on final cities for 1 epoch.

Decoding. The retrieval of items from the encoded sketch representation is done at the prediction stage. We retrieve scores for all items using the EMDE prediction procedure described in §3.1. Additionally, we post-process the output scores by multiplying by

[1] Standard configuration recommended by [10]

city popularity weights (calculated as the logarithm of the number of city occurrences as the final destination in a trip), thus boosting the scores of popular cities. Finally, we pick 4 cities with the highest scores as the top predictions.

Results. Our final score of Precision@4 metric on the validation set is 0.601%.

Table 2: Ablation study results.

Metric	Basic	+Data	+Features	+Popularity	+Ensembling
Precision@4	0.552	0.573	0.595	0.598	0.601
Difference	-	+0.021	+0.022	+0.003	+0.003

4.3 Ablation studies

In order to understand the effect of crucial parts of the training process, we conduct additional experiments. To ensure a comparable number of parameters of all models, we adjusted the hidden size to have roughly the same total number of model parameters.

The ablation results are summarized in Table 2. In the `Basic` setting we train a pure EMDE model, which takes as input only the three concatenated city sketches (*first city sketch*, *prev city sketch*, *all cities sketch*) and trains on final destinations only. This baseline model achieved Precision@4 score of 0.552. By including examples that are not final destination, and adding flag to the input that indicates if the examples is the last destination (`Data`), we observed a 0.021 increase of the precision score. Concatenating continuous and categorical features to the input of neural network (`Features`) improves the precision score by 0.022 compared to the model without features. In addition to pure EMDE decoding we verify the impact of popularity boosting of final scores (`Popularity`). It increases Precision@4 score from 0.595 to 0.598. Furthermore, ensembling of 5 models (`Ensembling`) improves the precision by 0.003%

Table 3: Performance of EMDE against of sequential models (without ensembling).

Metric	EMDE	GRU + Cleora	GRU
Precision@4	0.598	0.588	0.5786

In the next set of experiments, we compare EMDE against a sequential baseline model. The results are presented in Table 3. We train a GRU model [4] with hidden size of 1024 (selected empirically for best results), which takes as input either trainable city embeddings, or city embeddings learned by Cleora. All other input features are the same as in EMDE model. For fair comparison, we feed GRU outputs to the same three-layer feed-forward residual architecture as in EMDE. The application of sequential model decreases Precision@4 score from 0.598 to 0.588. In addition, training GRU with randomly initialized trainable embeddings decreases Precision@4 score to 0.578.

We also verify the impact of a graph embedding technique used to embed cities. We contrast Cleora with Word2Vec [15] from the area of natural language processing, Node2Vec [7] which learns a low-dimensional representations for nodes in a graph by optimizing a neighborhood preserving objective, and LINE [20], which preserves the first-order node proximity and second-order node proximity separately, and then concatenates the two representations. In addition to network embedding techniques, we also compare to embeddings produced by a sequential model. We train a GRU model with trainable city embeddings to predict the final destination of a trip, and then use these learned embeddings to represent the cities as EMDE input. The results are summarized in Table 4. Cleora is a clear winner in terms of performance while also being significantly faster to train [18].

Table 4: Performance of our system (without ensembling) using various graph representation methods for computing input embeddings.

Metric	Cleora	Node2Vec	LINE	Word2Vec	GRU
Precision@4	0.5984	0.5956	0.5949	0.5907	0.5865

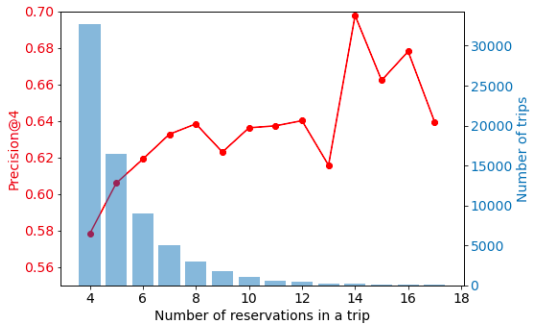
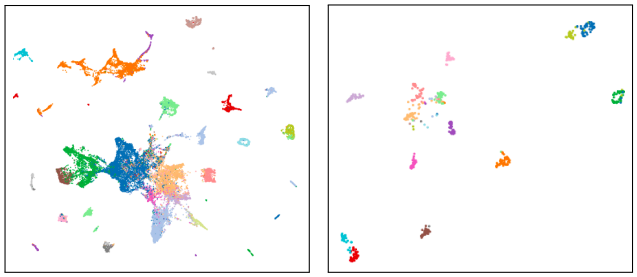


Figure 2: Red plot presents the performance of the model in terms of number of reservations in a trip. The distribution of trip lengths is shown in the blue chart.

Error analysis. To discover possible sources of errors, we analyze predictions on different length of trips (Figure 2). We notice that almost 50% of reservations have only 4 bookings, and 97.6% trips have less than 10 bookings. That taken into account, we observe that **the performance of our model increases with the number of reservations in a trip**. We hypothesize that it is easier for EMDE to capture dependencies in long sequences because in long trips the last part of the trip is often located in the same country (thus the cities are relatively close by), while in short sequences final locations can change abruptly.

In about 15% validation examples, the last destination is the same as the first one. Our model achieves 0.902 Precision@4 score on these examples, which leads to the conclusion that it learned to capture this particular phenomenon.

City embeddings visualization. Figure 3 shows a visualization of high dimensional city embeddings mapped to 2-D space with UMAP [14]. It can be seen that Cleora embeddings exhibit the awareness of geographic closeness of cities, which is well-represented irrespective of city popularity. As such, we show that the embeddings hold semantic knowledge which was not contained directly in the training graph (which was comprised of sequences of city IDs only, without any extra features).



(a) Top 20 most popular countries (b) Top 60 to 80 most popular countries

Figure 3: 2-D visualization of city embeddings learned by Cleora. Each color denotes a different hotel country.

5 SUMMARY

In this paper we present our model which achieves 2nd place in Booking.com data challenge competition. We show that predicting the final city can be seen as a recommendation task. The system utilizes a graph embedding method to create multi-modal city vector representations, which are then encoded by EMDE into fixed-size *sketch* structures. We show that accurate density estimation of sequences mapped to item sets can outperform inherently sequential methods.

REFERENCES

[1] Jiyang Bai, Yuxiang Ren, and Jiawei Zhang. 2020. Ripple Walk Training: A Subgraph-based training framework for Large and Deep Graph Neural Network. *arXiv preprint arXiv:2002.07206* (2020).

[2] Jianfei Chen, Jun Zhu, and Le Song. 2017. Stochastic training of graph convolutional networks with variance reduction. *arXiv preprint arXiv:1710.10568* (2017).

[3] Tianwen Chen and Raymond Chi-Wing Wong. 2020. Handling Information Loss of Graph Neural Networks for Session-based Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1172–1180.

[4] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

[5] Jacek Dabrowski, Barbara Rychalska, Michał Daniluk, Dominika Basaj, Piotr Babel, and Andrzej Michałowski. 2020. An efficient manifold density estimator for all recommendation systems. *arXiv preprint arXiv:2006.01894* (2020).

[6] Dmitri Goldenberg, Kostia Kofman, Pavel Levin, Sarai Mizrachi, Maayan Kafry, and Guy Nadav. 2021. Booking.com WSDM WebTour 2021 Challenge. https://www.bookingchallenge.com/. In *ACM WSDM Workshop on Web Tourism (WSDM WebTour'21)*.

[7] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[8] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 843–852.

[9] Dietmar Jannach, G Moreira, and Even Oldridge. 2020. Why are deep learning models not consistently winning recommender systems competitions yet. *RecSys Challenge'20, September 26, 2020, Virtual Event, Brazil* (2020).

[10] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[11] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[12] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[13] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2019. Performance comparison of neural and non-neural approaches to session-based recommendation. In *Proceedings of the 13th ACM conference on recommender systems*. 462–466.

[14] Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[16] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. An Intent-guided Collaborative Machine for Session-based Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1833–1836.

[17] Massimiliano Ruocco, Ole Steinar Lillestøl Skrede, and Helge Langseth. 2017. Inter-session modeling for session-based recommendation. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*. 24–31.

[18] Barbara Rychalska, Piotr Bąbel, Konrad Gołuchowski, Andrzej Michałowski, and Jacek Dąbrowski. 2021. Cleora: A Simple, Strong and Scalable Graph Embedding Scheme. arXiv:2102.02302 [cs.LG]

[19] Corentin Tallec and Yann Ollivier. 2017. Unbiasing truncated backpropagation through time. *arXiv preprint arXiv:1705.08209* (2017).

[20] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. In *Proceedings of the 24th international conference on world wide web*. 1067–1077.

[21] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 346–353.

[22] Feng Yu, Yanqiao Zhu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2020. TAGNN: Target attentive graph neural networks for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1921–1924.

[23] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2021. Accurate, efficient and scalable training of Graph Neural Networks. *J. Parallel and Distrib. Comput.* 147 (2021), 166–183.