GDP: Generalized Device Placement For Dataflow Graphs

Yanqi Zhou Sudip Roy Admirali Abdolrashidi Daniel Wong
Peter C. Ma Qiumin Xu Ming Zhong Hanxiao Liu
Anna Goldie Azalia Mirhoseini James Laudon

Google Brain

Jan 9, 2020

Introduction

Neural networks have demonstrated remarkable scalability – improved performance can usually be achieved by training a larger model on a larger dataset. Training such large models efficiently while meeting device constraints, like memory limitations, necessitate partitioning of the underlying dataflow graphs for the models across multiple devices.

Device Placement Using Reinforcement Learning

- **HDP** (Mirhoseini et al., 2018) uses feed forward NN to assign each op to a group and runs a seq-to-seq model to place each group to a device.
- **Spotlight** (Gao et al., 2018) heuristically groups nodes and generates placements with LSTM.
- **Placeto** (Addanki et al., 2019) uses GNN to encode the graph structure into embeddings, then uses feed forward NN to iteratively generate a placement for one node at each step.

GDP

- An end-to-end deep RL method for device placement that can generalize to arbitrary and held-out graphs.
- The placement network is 15x faster than HDP without the need for explicit grouping.
- A new batch pre-training and fine-tuning strategy based on network superposition, which leads to improved transferability, better placements especially for large graphs, and huge reduction in policy search time.
- Superior performance over a wide set of workloads including graphs with over 50k nodes.

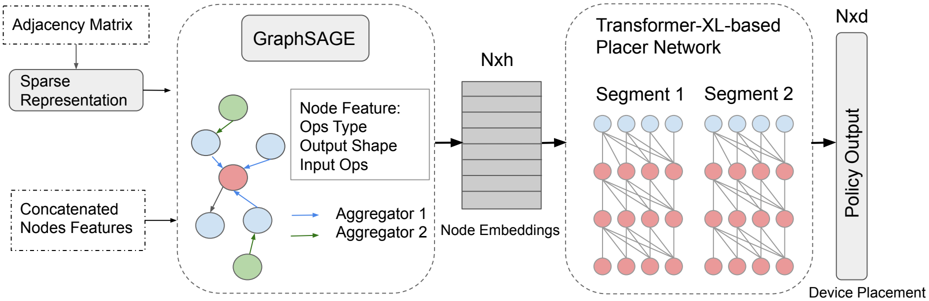
Problem Formulation

Given a dataflow graph $G(V, E)$ where V represents atomic computational operations (ops) and E represents the data dependency. The goal of GDP is to learn a policy $\pi : \mathcal{G} \mapsto \mathcal{D}$ that maximizes reward $r_{G,D}$ defined based on the run time. GDP represents policy π_θ as a nerual network parameterized by θ.

$$J(\theta) = \mathbb{E}_{G \sim \mathcal{G}, D \sim \pi_\theta(G)}[r_{G,D}] \approx \frac{1}{N} \sum_G \mathbb{E}_{D \sim \pi_\theta(G)}[r_{G,D}]$$

We refer to the case when $N = 1$ as *individual training* and the case when $N > 1$ as *batch training*.

System Overview



N: Number of nodes, h: Hidden Size, d: Number of Devices

Graph Embedding Network

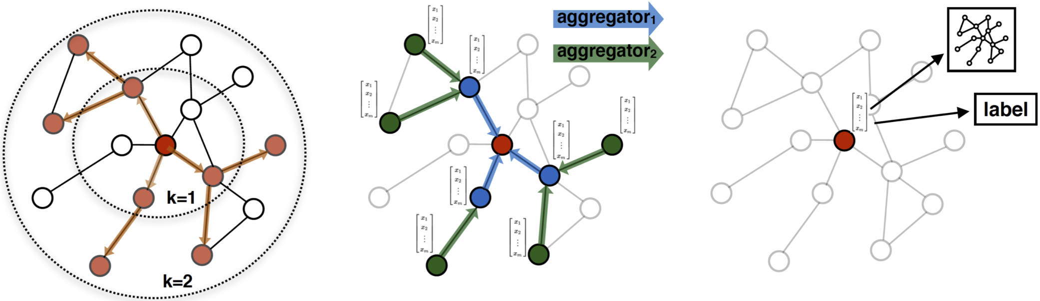
GDP adopts the feature aggregation scheme proposed in *GraphSAGE* as it shows better generalization.

$$h_{\mathcal{N}(v)}^{(l)} = \max(f_a^{(l)}(h_u^{(l)}), \forall u \in \mathcal{N}(v))$$
$$h_v^{(l+1)} = f_b^{(l+1)}(\text{concat}(h_v^{(l)}, h_{\mathcal{N}(v)}^{(l)}))$$

where h_v is the hidden feature of v, f_a and f_b are dense layers, $\mathcal{N}(v)$ represents the neighbors of v, and $h_{\mathcal{N}(v)}$ stands for the aggregated feature from the neighbors of v.

Different from GraphSAGE, which is unsuperised, GDP trains the embeddings jointly with the placement network.

GraphSAGE



1. Sample neighborhood

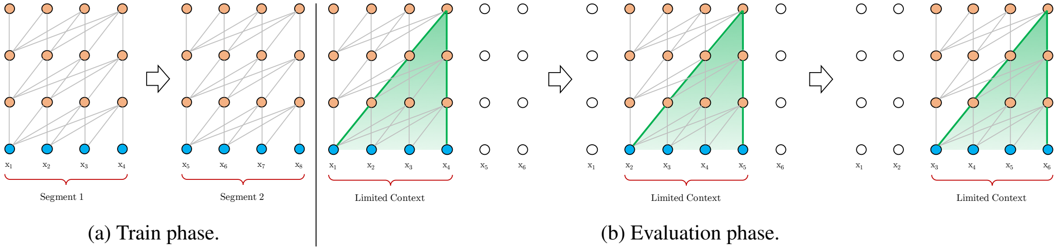
2. Aggregate feature information from neighbors

3. Predict graph context and label using aggregated information

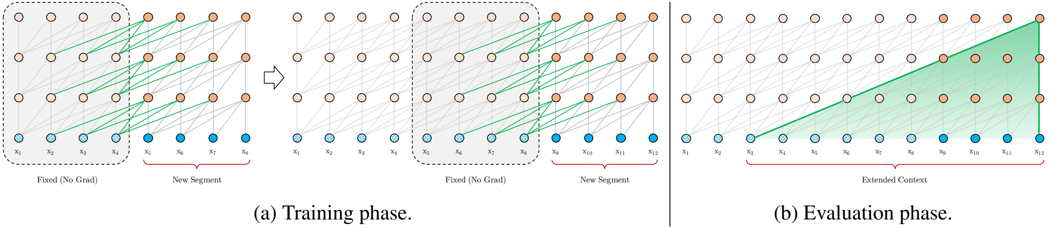
Placement Network

- Conventional seq-to-seq models usually target short sequences, which requires grouping beforehand.
- LSTM used in previous works is slower and more difficult to train than attention-based models.
- GDP adopts segment-level recurrence introduced in *Transformer-XL* to capture long-term dependencies. The key is to cache (with gradient flows disabled) and reuse the hidden states of previous segments.

Transformer XL



(a) Train phase. (b) Evaluation phase.



(a) Training phase. (b) Evaluation phase.

Batch Training

- Naive batch training is challenging because of the divergence of the dataflow graphs.
- GDP uses a feature conditioning mechanism similar to *parameter superposition*, implemented by replacing all dense layers in the placement network with:

$$x^{(l+1)} = g^{(l)}(c(x^{(0)}) \odot x^{(l)})$$

where $g^{(l)}$ stands for a dense layer in the placement network, c stands for the feature conditioning layer, and $x^{(0)}$ denotes the input feature generated by the graph-embedding network.

Experiment Setup

- We compare GDP with human expert placement (**HP**), Tensorflow **METIS** placement (a general purpose graph partitioning tool), and **HDP** (Mirhoseini et al., 2018).
- 8 Nvidia P100
- We use negative square root of the run time as the reward, and subtract the average reward of all previous trials to calculate the advantage value. Invalid placements are given a large (-10) negative reward.

Performance on Individual Graphs

Model (#devices)	**GDP-one** (s)	HP (s)	METIS (s)	HDP (s)	Run time speed up over HP / HDP	Search speed up
2-layer RNNLM (2)	0.234	0.257	0.355	0.243	9.8% / 4%	2.95x
4-layer RNNLM (4)	0.409	0.48	OOM	0.490	17.4% / 19.8%	1.76x
2-layer GNMT (2)	0.301	0.384	OOM	0.376	27.6% / 24.9%	30x
4-layer GNMT (4)	0.409	0.469	OOM	0.520	14.7% / 27.1%	58.8x
8-layer GNMT (8)	0.649	0.610	OOM	0.693	-6% / 6.8%	7.35x
2-layer Transformer-XL (2)	0.386	0.473	OOM	0.435	22.5% / 12.7%	40x
4-layer Transformer-XL (4)	0.580	0.641	OOM	0.621	11.4% / 7.1%	26.7x
8-layer Transformer-XL (8)	0.748	0.813	OOM	0.789	8.9% / 5.5%	16.7x
Inception (2)	0.405	0.418	0.423	0.417	3.2% / 3%	13.5x
AmoebaNet (4)	0.394	0.44	0.426	0.418	26.1% / 6.1%	58.8x
2-stack 18-layer WaveNet (2)	0.317	0.376	OOM	0.354	18.6% / 11.7%	6.67x
4-stack 36-layer WaveNet (4)	0.659	0.988	OOM	0.721	50% / 9.4%	20x
GEOMEAN	-	-	-	-	**16% / 9.2%**	**15x**

Performance on Batch Training

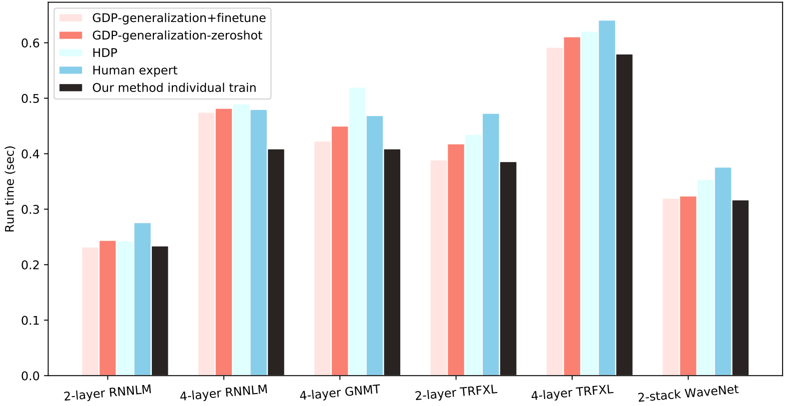
Run time comparing on GDP-batch vs. GDP-one

Model	Speed up	Model	Speed up
2-layer RNNLM	0	Inception	0
4-layer RNNLM	5%	AmoebaNet	-5%
2-layer GNMT	0	4-stack 36-layer WaveNet	3.3 %
4-layer GNMT	0	2-stack 18-layer WaveNet	15%
2-layer Transformer-XL	7.6%	8-layer Transformer-XL	1.5%
4-layer Transformer-XL	3%		

A possible explanation for the performance gain is the additional feature conditioning layer in the batch training effectively enlarged the model.

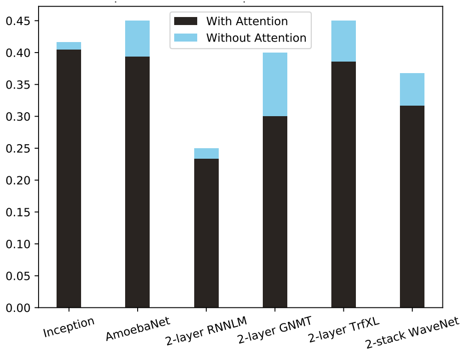
Performance on Hold-out Graphs

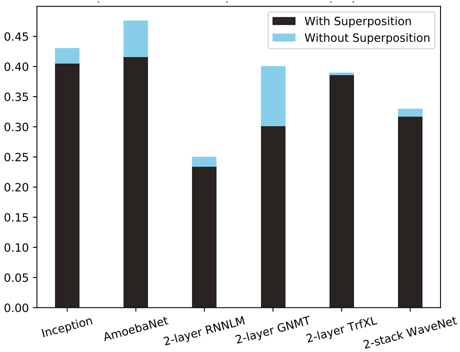
We run GDP on unseen graphs with and without finetuning, called **GDP-gereralization-finetune** and **GDP-gereralization-zeroshot** respectively.



Ablation Studies

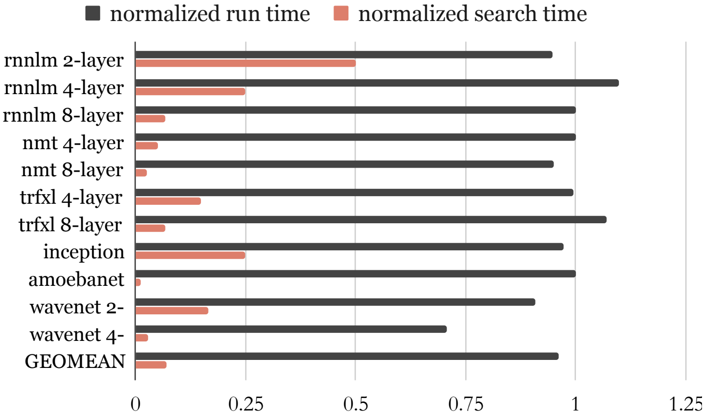
We did ablation studies on the attention and the superposition layer in the placement network. They improved the average run time by 18% and 6.5% respectively.





Pre-training Graph Embeddings

We train GDP-batch like before, but then fine-tuning on each specific graphs. The run time and search time are reduced by 5% and 86% respectively, compared with GDP-one.



Thank you!