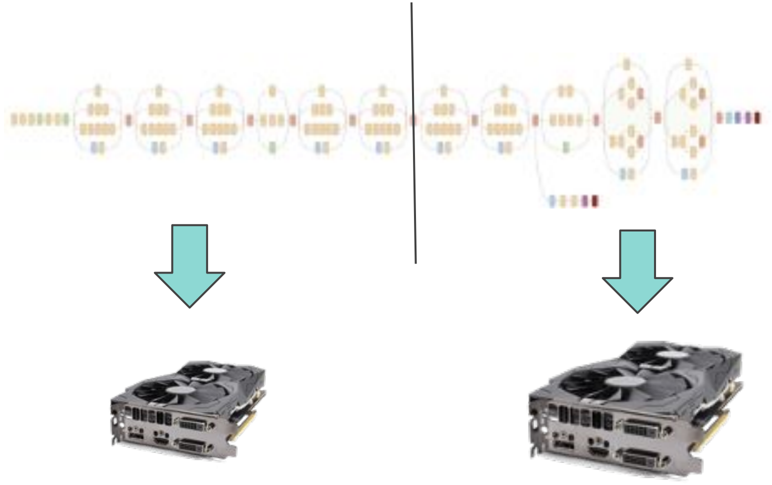
Placeto: Learning Generalizable Device Placement Algorithms for Distributed Machine Learning

R. Addanki, S. B. Venkatakrishnan, S. Gupta, H. Mao, M. Alizadeh

Presenter: Qianyi Liu

Background

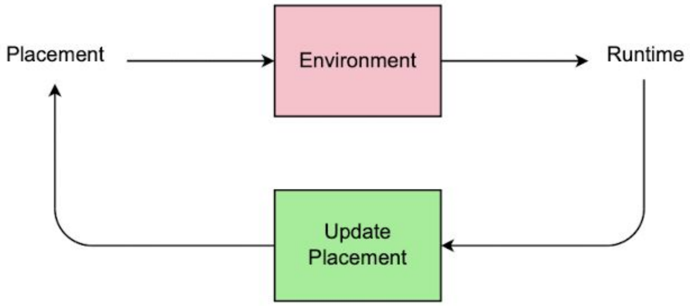
Key challenge for **distributed training**: split a large model across multiple heterogeneous devices to achieve the fastest possible training speed



- Typical approach: left to human experts
- A new solution: automated approach to device placement based on reinforcement learning

Prior works

- Mirhoseini et al (2017): train an RNN to process a computation graph and predict a placement for each operation
- Mirhoseini et al (2018): a hierarchical model, improved scalability and new optimisation techniques



Key drawbacks:

- Long time
- Don't learn generalisable device placement strategy -> requires retraining

Placeto

- Use RL to learn an efficient algorithm for device placement for **a given family of computation graphs**
- Two key ideas:

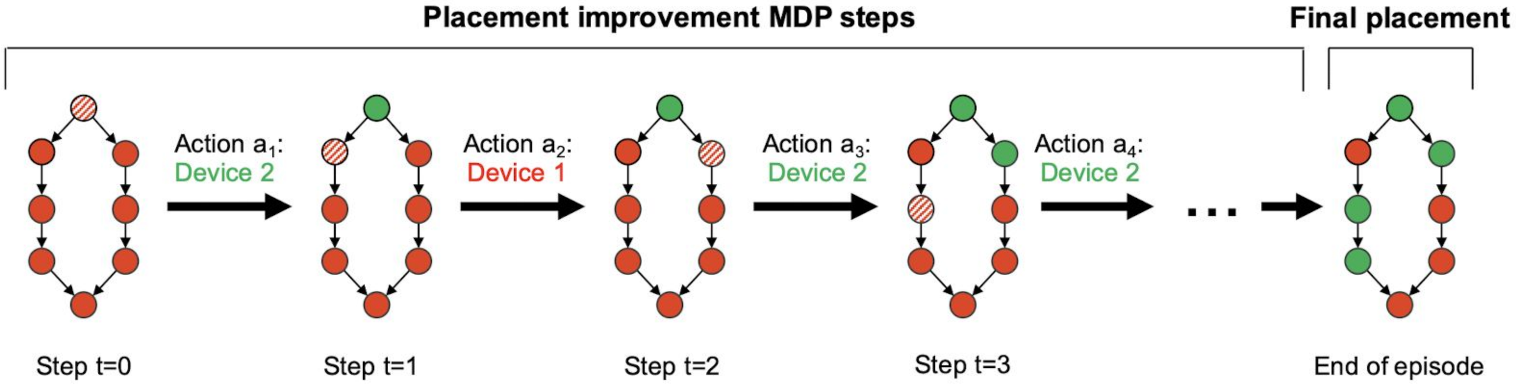
  Idea 1: Find a sequence of iterative placement improvements

  - Simpler to learn → training efficiency ↑

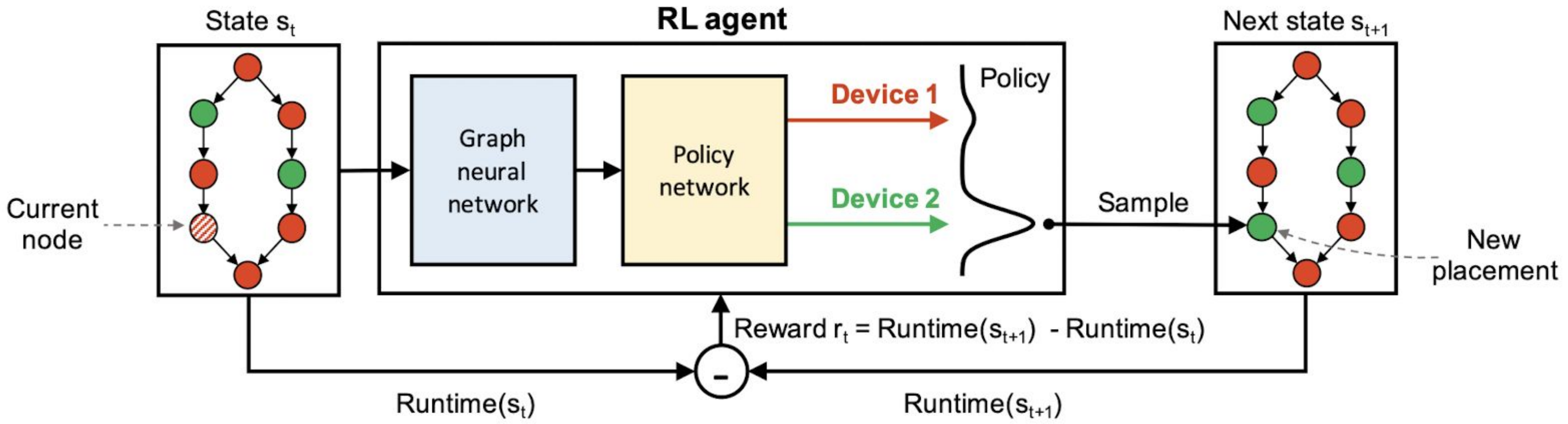
  Idea 2: use graph embeddings to encode the computation graph structure

  - Doesn’t depend on sequential order of nodes
  - GNN + message passing
  - Generalisability ↑

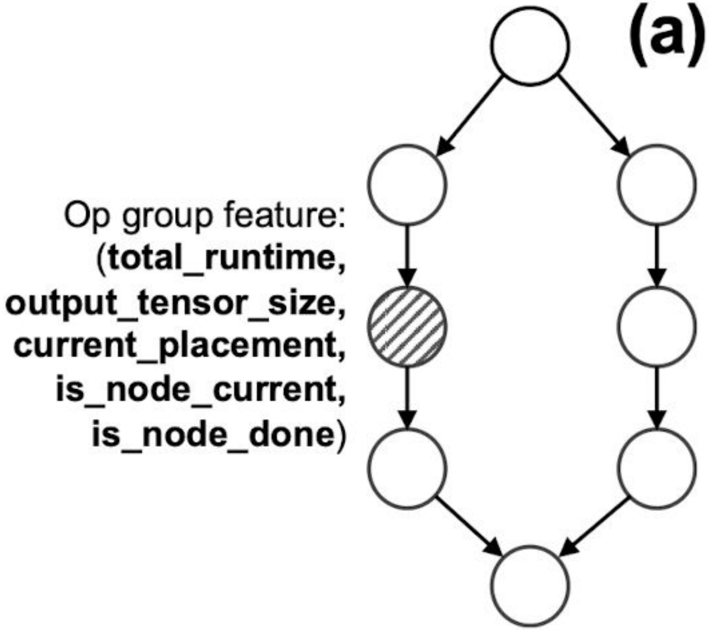
Learning procedure: a Markov decision process



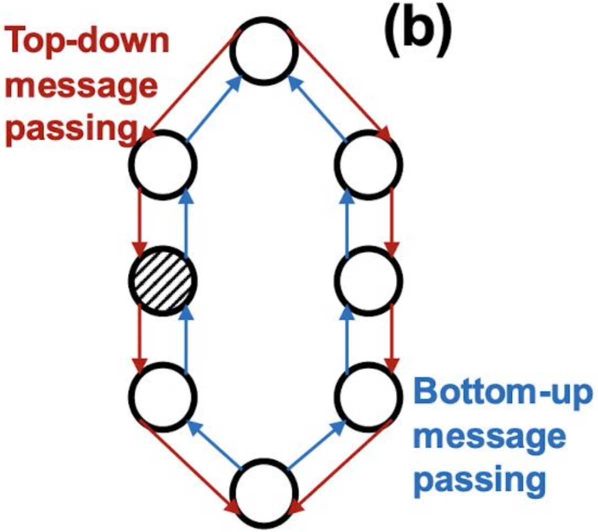
RL to learn the MDP policy – a neural network



Graph embedding step 1: Compute per-group attributes



Graph embedding step 2: Local neighbourhood summarisation

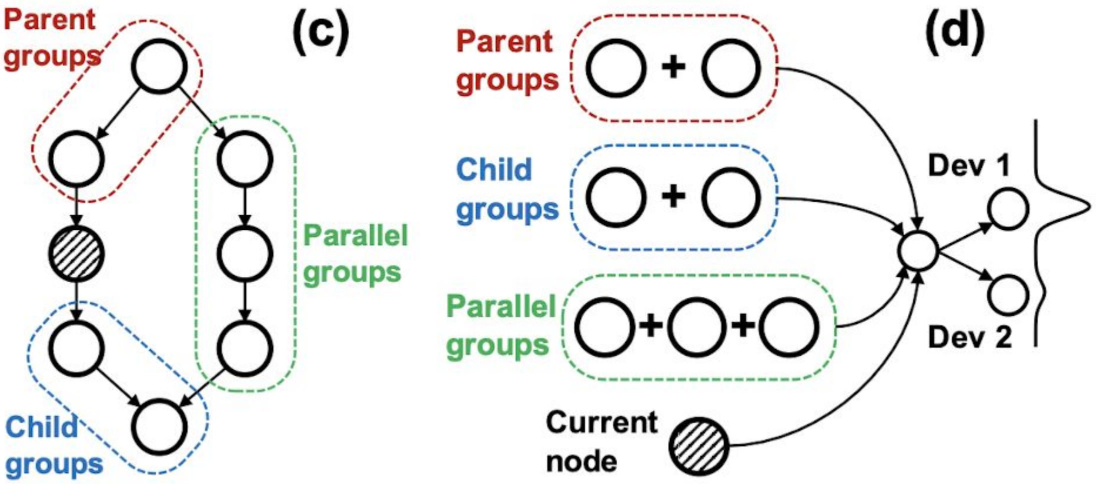


- A sequence of message passing steps to aggregate neighbourhood information for each node

$$\mathbf{x}_v \leftarrow g\left(\sum_{u \in \xi(v)} f(\mathbf{x}_u)\right),$$

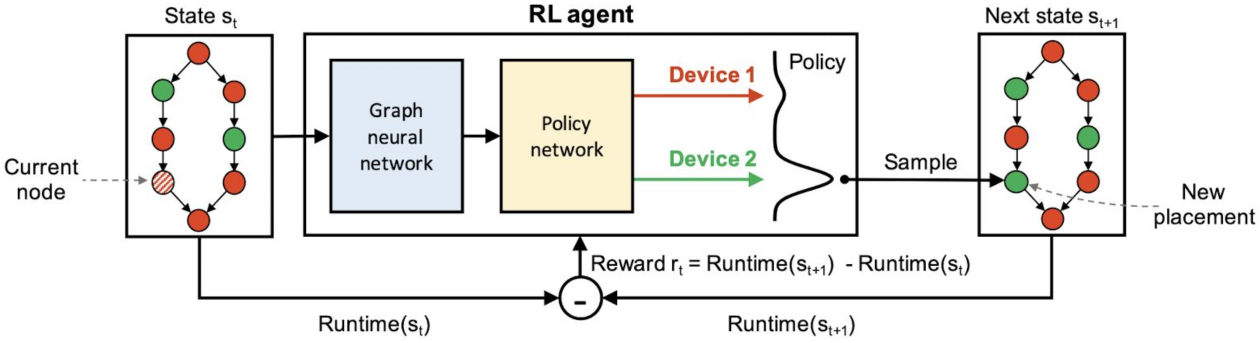
- Two directions: top-down + bottom-up

Graph embedding step 3: Pooling summaries



- Create a global summary of the entire graph, from the point of view of node v

Full picture



- Rewards are generated from a simulator rather than actual hardware measurement during training

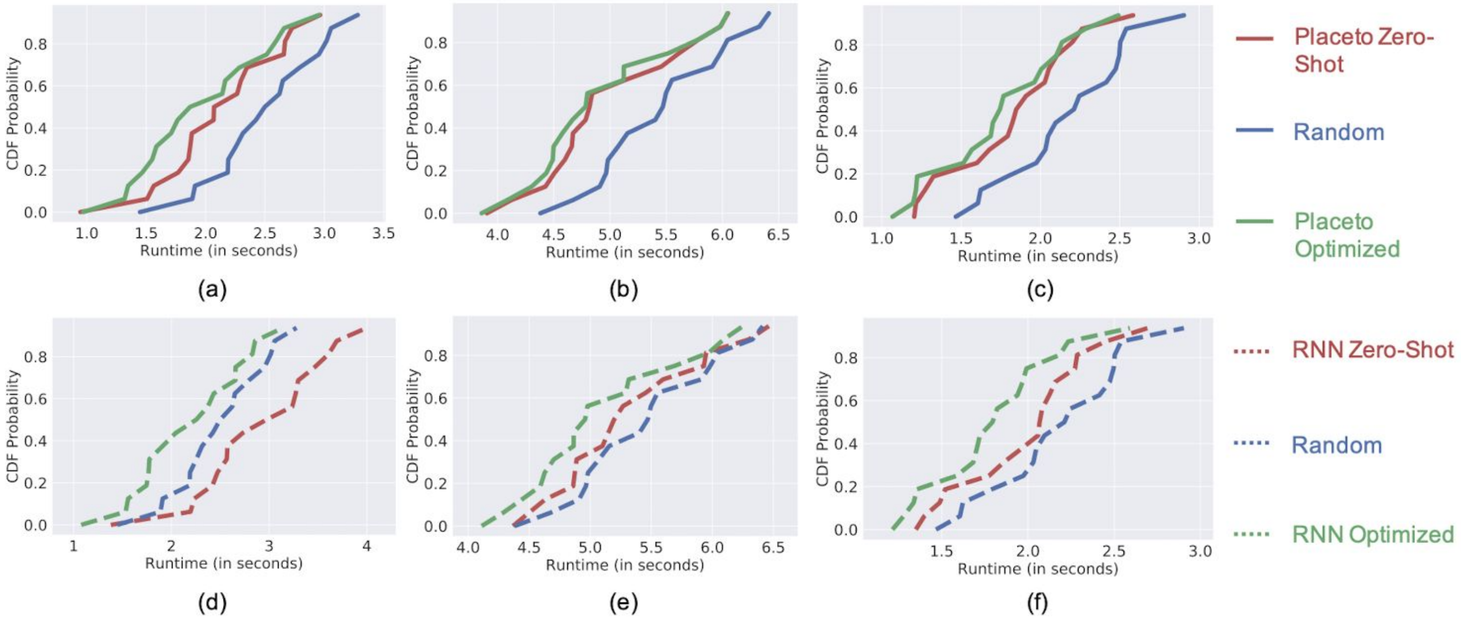
Evaluation: performance

Metric:

1) Runtime of the best placement found
2) Time taken to find the best placement (# of placement evaluations)

	Placement runtime (sec)							Training time (# placements sampled)		Improvement	
Model	CPU only	Single GPU	#GPUs	Expert	Scotch	Placeto	RNN-based	Placeto	RNN-based	Runtime Reduction	Speedup factor
Inception-V3	12.54	1.56	2	1.28	1.54	1.18	1.17	1.6 K	7.8 K	- 0.85%	**4.8** ×
			4	1.15	1.74	1.13	1.19	5.8 K	35.8 K	5%	**6.1** ×
NMT	33.5	OOM	2	OOM	OOM	2.32	2.35	20.4 K	73 K	1.3 %	**3.5** ×
			4	OOM	OOM	2.63	3.15	94 K	51.7 K	**16.5 %**	0.55 ×
NASNet	37.5	1.28	2	0.86	1.28	0.86	0.89	3.5 K	16.3 K	3.4%	**4.7** ×
			4	0.84	1.22	0.74	0.76	29 K	37 K	2.6%	**1.3** ×

Evaluation: generalisability



Takeaways

Pros:

- Novelty: first attempt to use GNN to encode graph structure in device placement optimisation — learns generalisable placement policy
- Impressive performance: find better placements faster than RNN-based approach

Cons:

- Operator needs to be manually grouped based on heuristics — not an end-to-end solution
- Generalisability is limited to graphs from the same family

Discussion