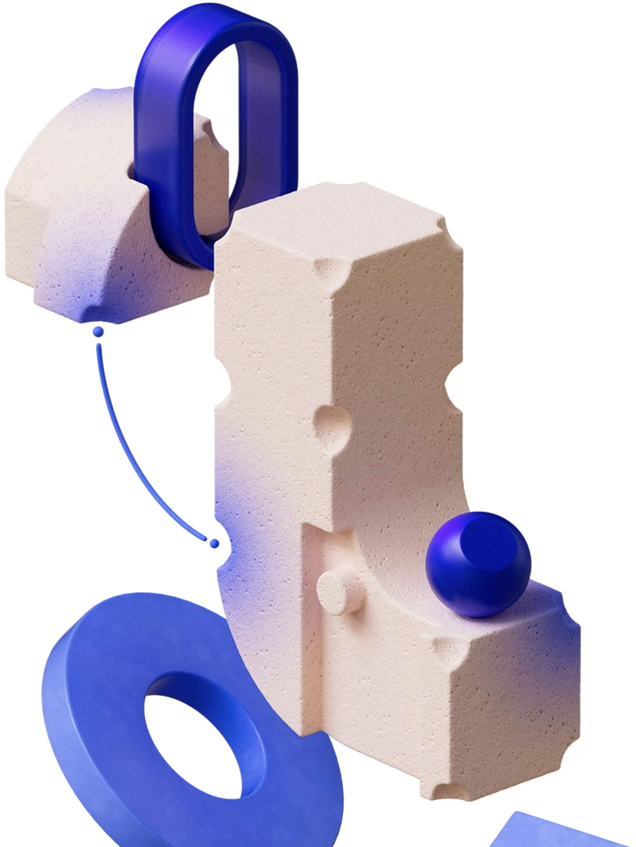
Google DeepMind

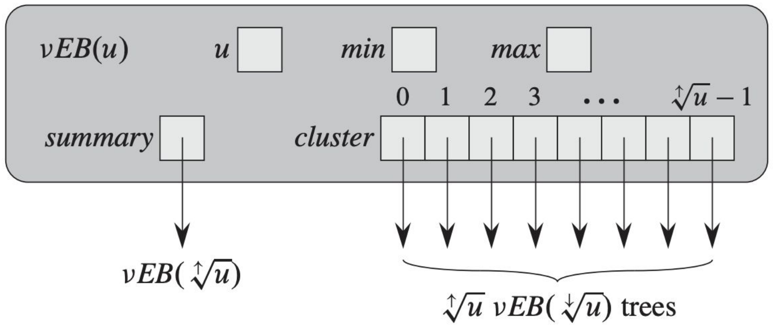


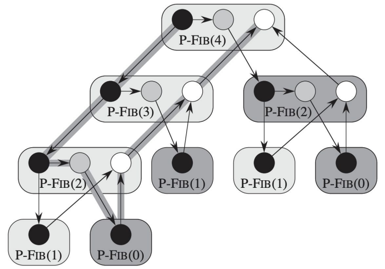
◎ *Monoids* & *Time* ⏱️
Embracing **asynchrony** in (graph) neural nets

Petar Veličković
Staff Research Scientist, Google DeepMind
Affiliated Lecturer, University of Cambridge

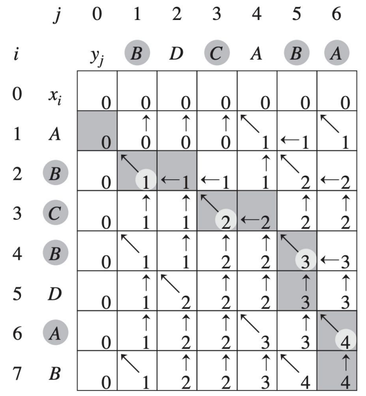
HAMLET-Physics
University of Copenhagen
20 August 2024

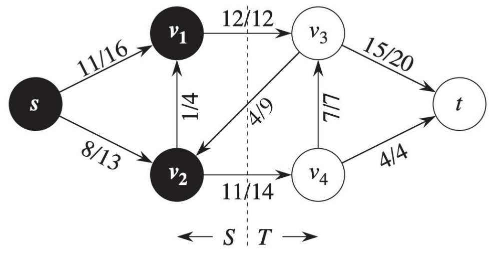
In this talk:
(Classical) **Computation**


vEB(u)
u
min
max
0 1 2 3 … $\sqrt[\uparrow]{u} - 1$
summary
cluster
vEB($\sqrt[\downarrow]{u}$)
$\sqrt[\uparrow]{u}$ vEB($\sqrt[\downarrow]{u}$) trees

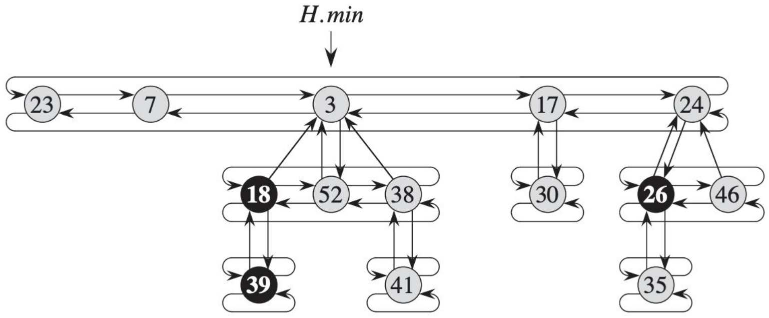


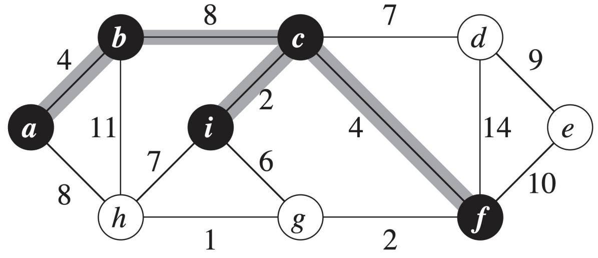
```
MERGE-SORT(A, p, r)
1  if $p < r$
2      $q = \lfloor (p + r)/2 \rfloor$
3      MERGE-SORT(A, p, q)
4      MERGE-SORT(A, q + 1, r)
5      MERGE(A, p, q, r)
```

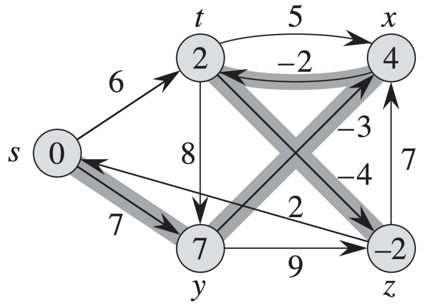




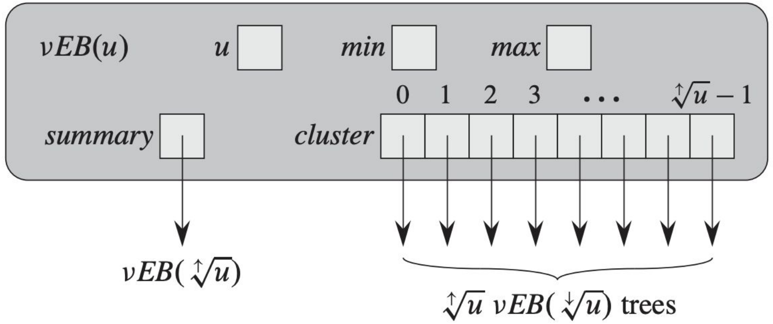
In this talk:
(Classical) Computation



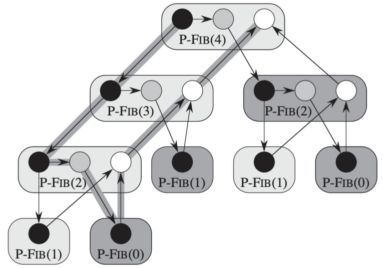




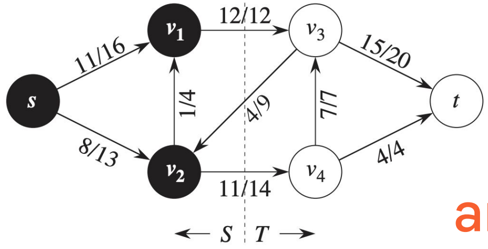
Algorithm figures: Cormen, Leiserson, Rivest and Stein. **Introduction to Algorithms.**



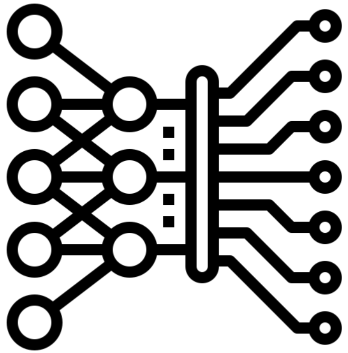
$vEB(u)$
u
min
max
$summary$
$cluster$
0 1 2 3 … $\sqrt[\uparrow]{u}-1$
$vEB(\sqrt[\uparrow]{u})$
$\sqrt[\uparrow]{u}$ $vEB(\sqrt[\downarrow]{u})$ trees

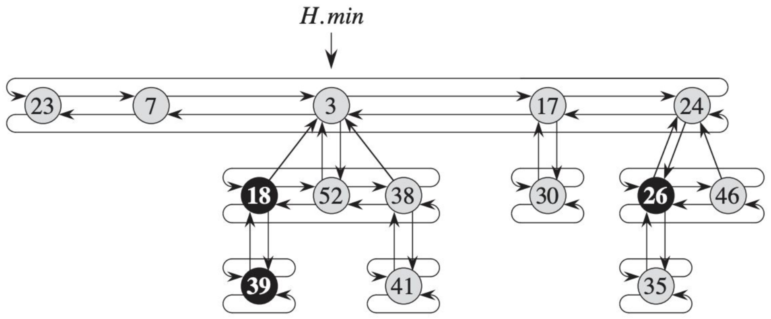


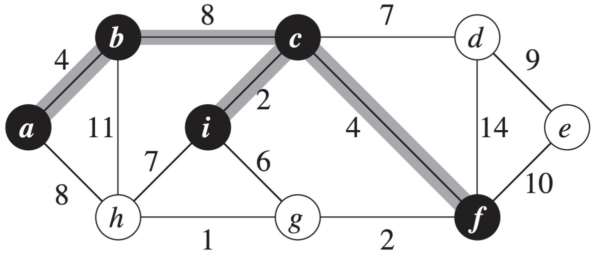
```
MERGE-SORT(A, p, r)
1  if $p < r$
2      $q = \lfloor (p+r)/2 \rfloor$
3      MERGE-SORT(A, p, q)
4      MERGE-SORT(A, q + 1, r)
5      MERGE(A, p, q, r)
```

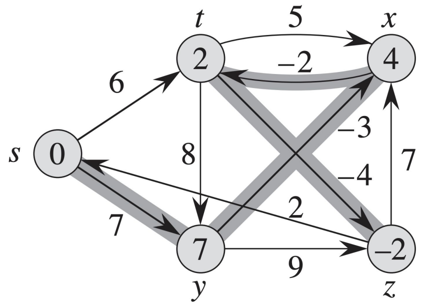


In this talk:
(Classical) Computation
and how to capture it with neural nets









Algorithm figures: Cormen, Leiserson, Rivest and Stein. **Introduction to Algorithms.**

What I **want**:
Robust **reasoning** in neural nets

The key aim is **reasoning**

What does **reasoning** mean to *our team* at GDM?

- ***Robust*** *procedure* for solving instances of a problem
- It needn't be *fully accurate* (human reasoning is often approximate!)
- It needn't be *symbolic* (may be done purely or partially in latent space)
- But it should behave ***consistently*** across all problem instances

⇒ We care about **OOD generalisation**

- *A model that truly* ***captures*** *the reasoning concept of multiplication should work equally well on* ***all*** *instances of multiplication problems…*

Even our best models do not extrapolate well

*Transformers compute **billions** of **multiplications** just to generate a **single** token; yet, they **cannot multiply 3-by-3-digit numbers** correctly.*

#digits	1	2	3	4	5
1	1.00	1.00	1.00	1.00	1.00
2	1.00	0.99	0.80	0.55	0.40
3	1.00	0.79	0.37	0.11	0.04
4	1.00	0.63	0.14	0.02	0.02
5	0.97	0.41	0.07	0.03	0.00

Table 1. Accuracy of **Gemini Ultra** on multiplications.
Prompt: "`What is $a * $b?`"
(very similar trends for GPT-4; see Shen *et al.* (2023))

The key aim is **reasoning**

What does **reasoning** mean to *our team* at GDM?

- ***Robust** procedure* for solving instances of a problem
- It needn't be *fully accurate* (human reasoning is often approximate!)
- It needn't be *symbolic* (may be done purely or partially in latent space)
- But it should behave ***consistently*** across all problem instances

⇒ We care about **OOD generalisation**
  - *A model that truly **captures** the reasoning concept of multiplication should work equally well on **all** instances of multiplication problems...*

Further, we want to do this in a way that **scales** easily to XXXB-parameters

tl;dr: *data* & *tools* are **not** enough.

We (likely) need to change the **model equations** to capture **computation** better

First, we need to cover some preliminaries.

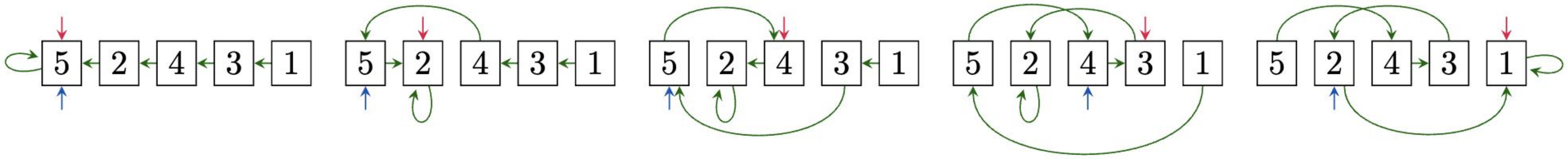
<u>Evals</u>: How can we **evaluate** OOD generalisation?

Most popular existing benchmarks *unsuitable* due to **distribution leakage**

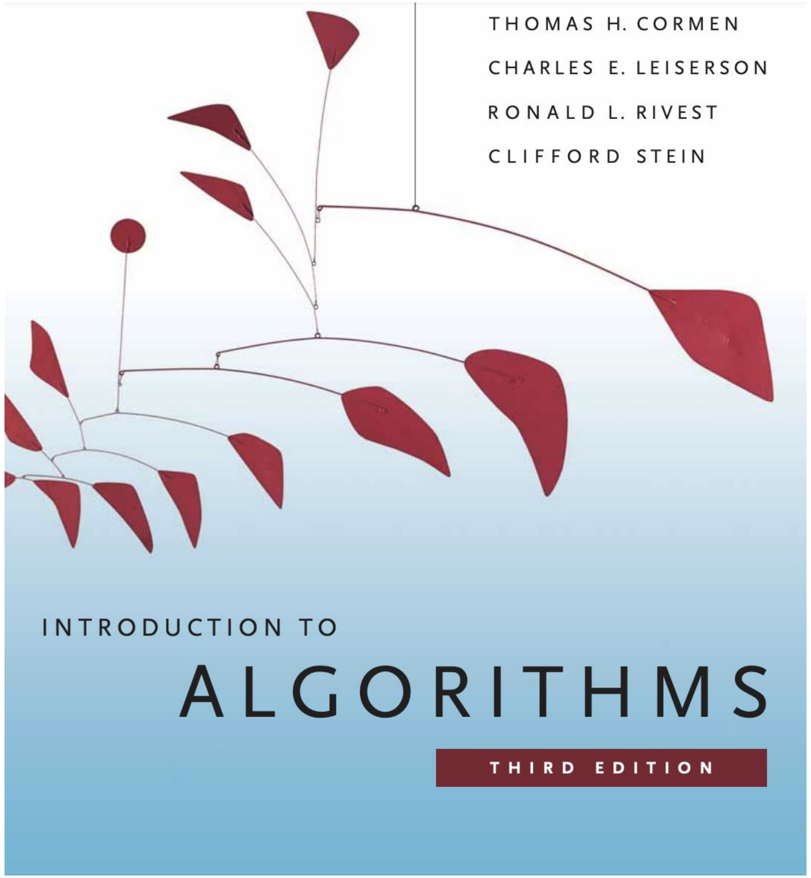
We need to take tasks for which we can generate outputs:

- Reliably
- Efficiently
- For *any* distribution of interest

This, by definition, implies we need (polynomial-time) **algorithms**



The **CLRS-30** Benchmark (ICML'22)



Sorting: Insertion sort, bubble sort, heapsort (Williams, 1964), quicksort (Hoare, 1962).

Searching: Minimum, binary search, quickselect (Hoare, 1961).

Divide and Conquer (D&C): Maximum subarray (Kadane's variant (Bentley, 1984)).

Greedy: Activity selection (Gavril, 1972), task scheduling (Lawler, 1985).

Dynamic Programming: Matrix chain multiplication, longest common subsequence, optimal binary search tree (Aho et al., 1974).

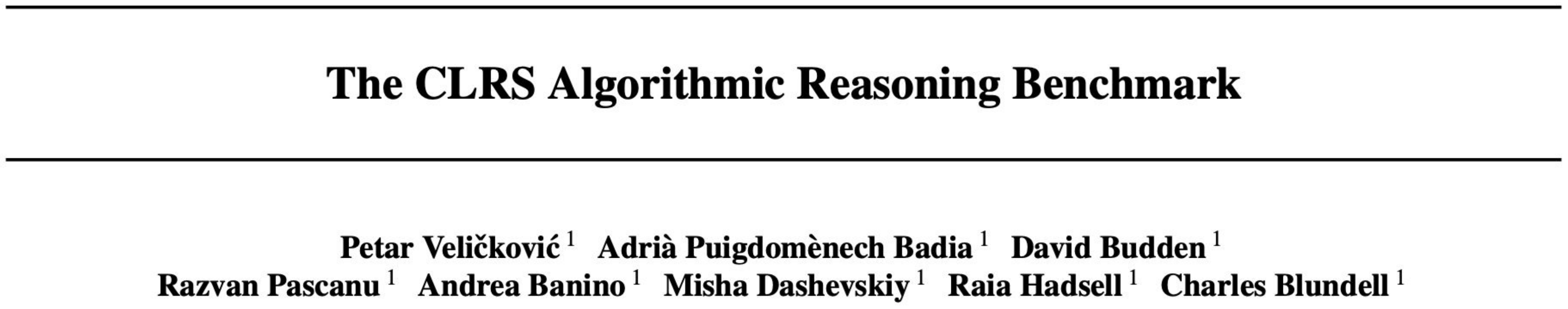
Graphs: Depth-first and breadth-first search (Moore, 1959), topological sorting (Knuth, 1973), articulation points, bridges, Kosaraju's strongly-connected components algorithm (Aho et al., 1974), Kruskal's and Prim's algorithms for minimum spanning trees (Kruskal, 1956; Prim, 1957), Bellman-Ford and Dijkstra's algorithms for single-source shortest paths (Bellman, 1958; Dijkstra et al., 1959) (+ directed acyclic graphs version), Floyd-Warshall algorithm for all-pairs shortest paths (Floyd, 1962).

Strings: Naïve string matching, Knuth-Morris-Pratt (KMP) string matcher (Knuth et al., 1977).

Geometry: Segment intersection, Convex hull algorithms: Graham scan (Graham, 1972), Jarvis' march (Jarvis, 1973).

The **CLRS-30** Benchmark (ICML'22)

CLRS-30 is *not* just a *dataset*;
it is a **dataset & baseline** ***generator***!



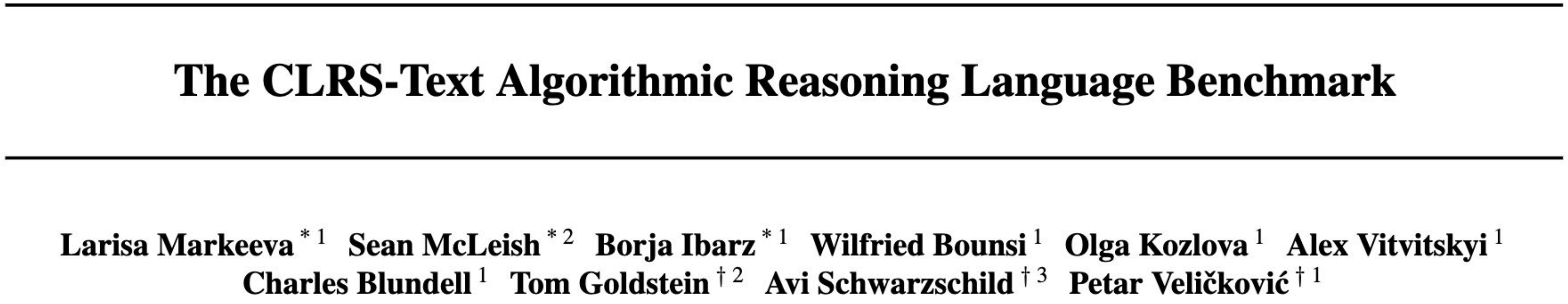
The CLRS Algorithmic Reasoning Benchmark

Petar Veličković[1] Adrià Puigdomènech Badia[1] David Budden[1]
Razvan Pascanu[1] Andrea Banino[1] Misha Dashevskiy[1] Raia Hadsell[1] Charles Blundell[1]

`https://github.com/google-deepmind/clrs`

The CLRS-Text Benchmark (ICML'24 DMLR)

Recently: bringing CLRS-30 into the **LLM** age!

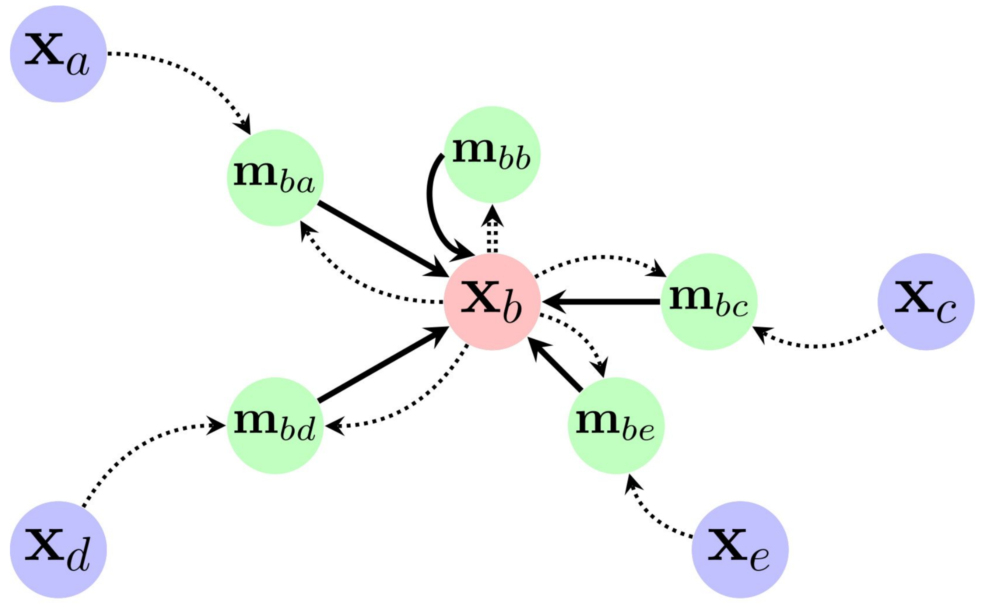


The CLRS-Text Algorithmic Reasoning Language Benchmark

Larisa Markeeva[*1] **Sean McLeish**[*2] **Borja Ibarz**[*1] **Wilfried Bounsi**[1] **Olga Kozlova**[1] **Alex Vitvitskyi**[1]
Charles Blundell[1] **Tom Goldstein**[†2] **Avi Schwarzschild**[†3] **Petar Veličković**[†1]

`https://github.com/google-deepmind/clrs/tree/master/clrs/_src/clrs_text`

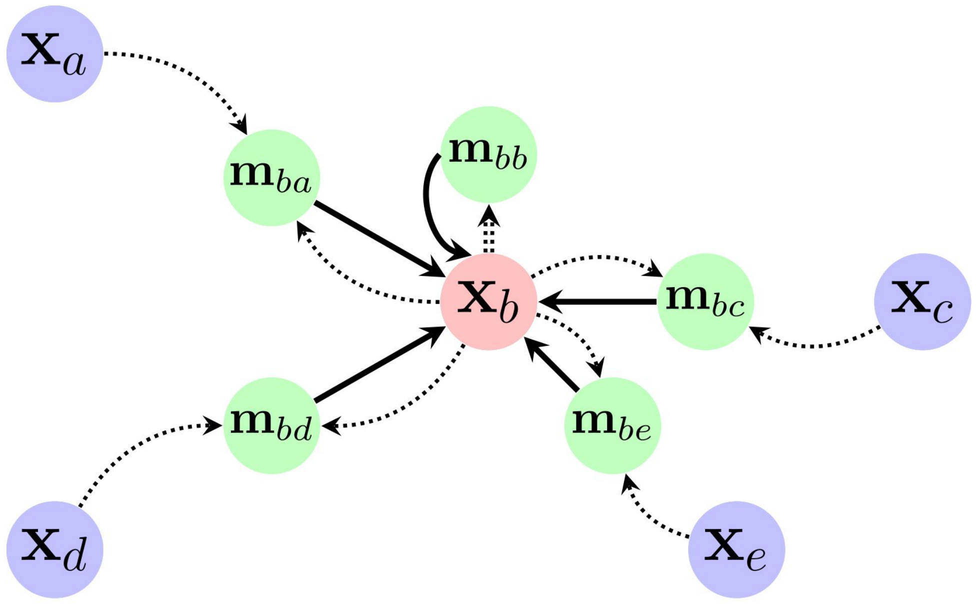
Models: Preliminaries on graph neural networks



Message-passing

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\right)$$

Models: Preliminaries on graph neural networks



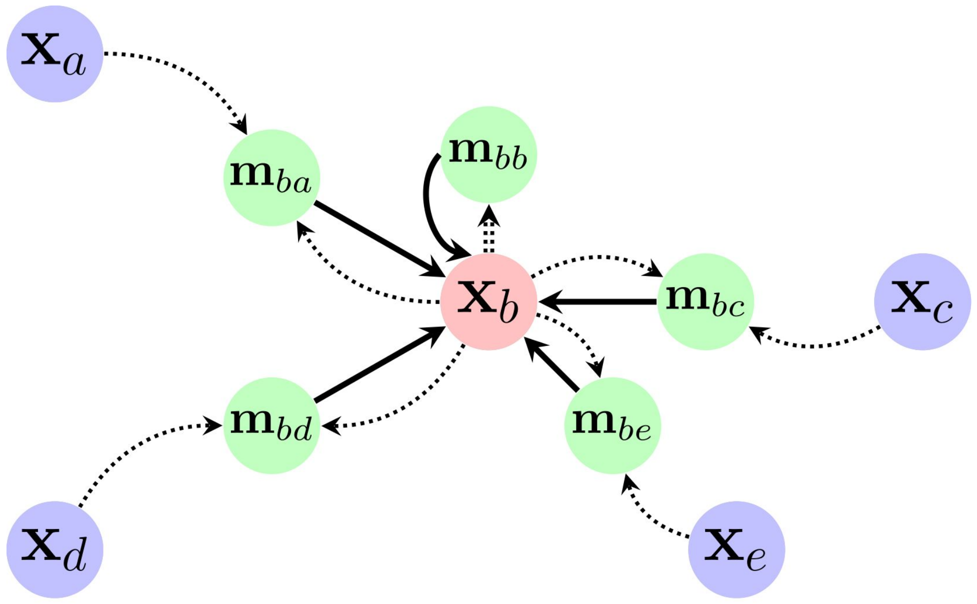
Message-passing

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u\,,\ \bigoplus_{v\in\mathcal{N}_u} \psi\left(\mathbf{x}_u\,,\,\mathbf{x}_v\right)\right)$$

Message function

$$\psi : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}^m$$

Models: Preliminaries on graph neural networks



Message-passing

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\right)$$

Message function

$$\psi : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}^m$$

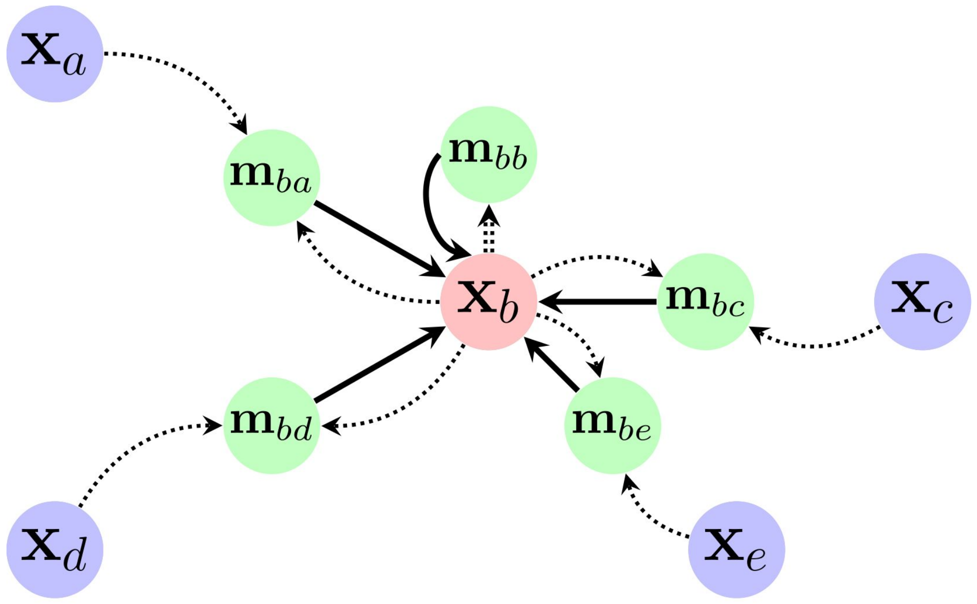
Aggregation function

$$\bigoplus : \mathrm{bag}(\mathbb{R}^m) \to \mathbb{R}^m$$

(a choice of *monoid* structure on $\mathbb{R}^m$)

The aggregation function, $\oplus$, needs to be **permutation invariant** (e.g. sum, max, average)

Models: Preliminaries on graph neural networks



Message-passing

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\right)$$

Message function

$$\psi : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}^m$$

Aggregation function

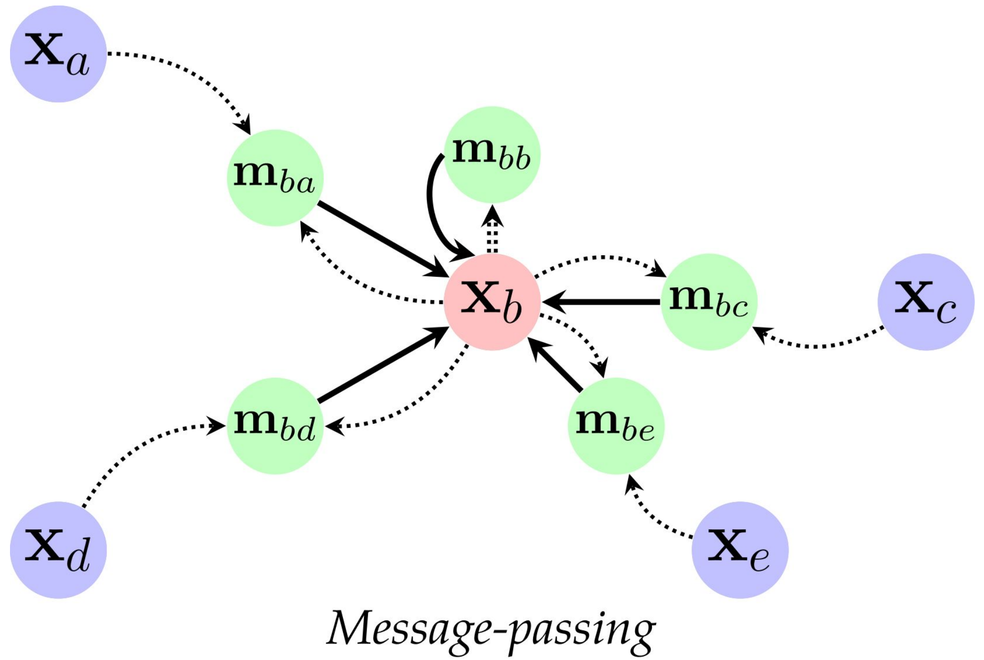
$$\bigoplus : \mathrm{bag}(\mathbb{R}^m) \to \mathbb{R}^m$$

Update function

$$\phi : \mathbb{R}^k \times \mathbb{R}^m \to \mathbb{R}^l$$

Message and update functions are the only **parametric** components of a GNN

Models: Preliminaries on graph neural networks



Message-passing

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi\left(\mathbf{x}_u, \mathbf{x}_v\right)\right)$$

Message function

$$\psi : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}^m$$

Aggregation function

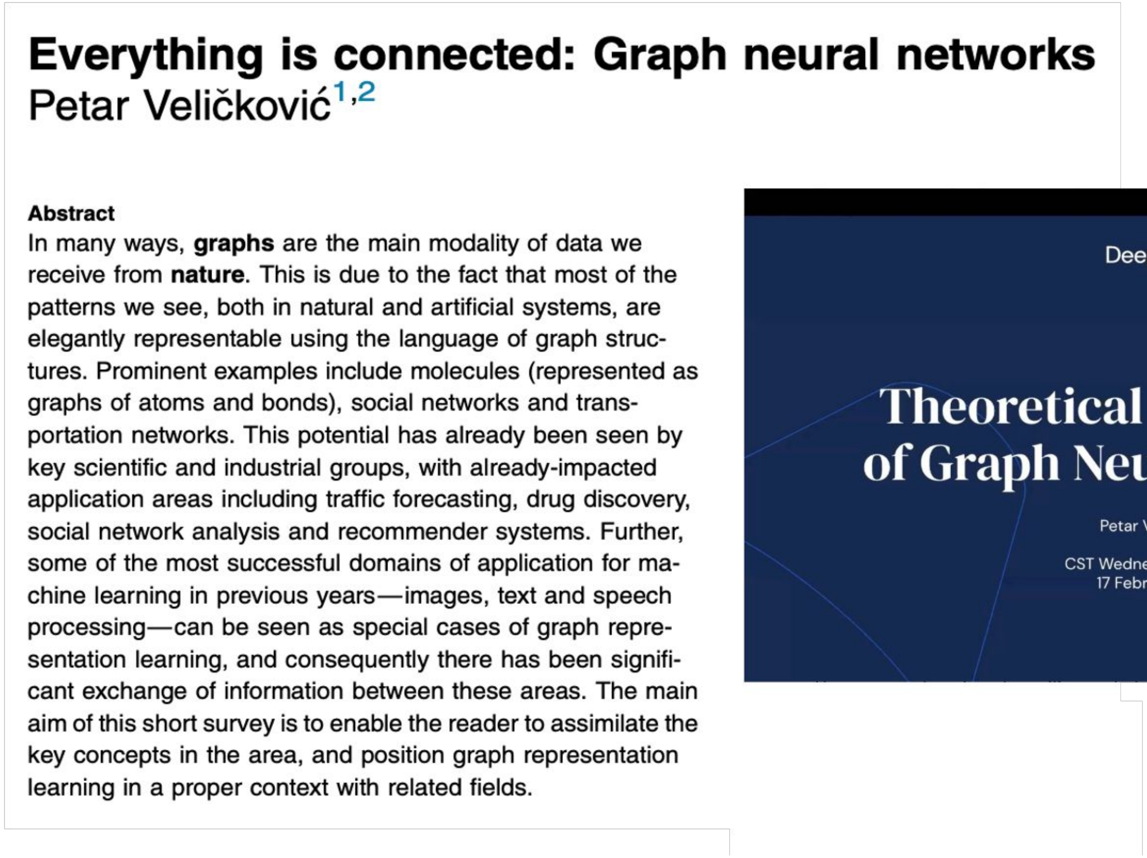
$$\bigoplus : \mathrm{bag}(\mathbb{R}^m) \to \mathbb{R}^m$$

Update function

$$\phi : \mathbb{R}^k \times \mathbb{R}^m \to \mathbb{R}^l$$

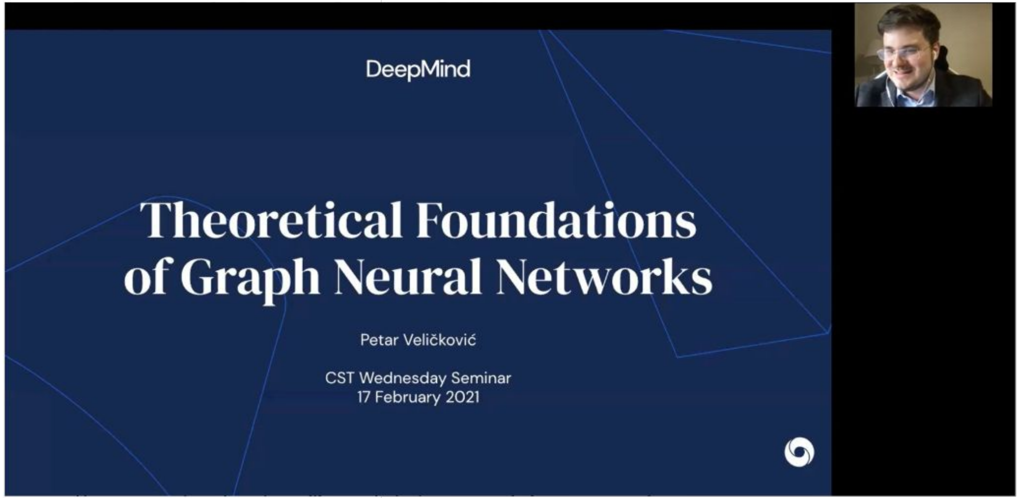
e.g. can be a single-layer NNs: $\psi(\mathbf{x}_u, \mathbf{x}_v) = \mathrm{ReLU}(\mathbf{W}_1\mathbf{x}_u + \mathbf{W}_2\mathbf{x}_v + \mathbf{b})$
with parameters $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{m \times k}$, $\mathbf{b} \in \mathbb{R}^m$ optimised using gradient descent.

For more information on GNNs...



Everything is connected: Graph neural networks
Petar Veličković[1,2]

Abstract
In many ways, **graphs** are the main modality of data we receive from **nature**. This is due to the fact that most of the patterns we see, both in natural and artificial systems, are elegantly representable using the language of graph structures. Prominent examples include molecules (represented as graphs of atoms and bonds), social networks and transportation networks. This potential has already been seen by key scientific and industrial groups, with already-impacted application areas including traffic forecasting, drug discovery, social network analysis and recommender systems. Further, some of the most successful domains of application for machine learning in previous years—images, text and speech processing—can be seen as special cases of graph representation learning, and consequently there has been significant exchange of information between these areas. The main aim of this short survey is to enable the reader to assimilate the key concepts in the area, and position graph representation learning in a proper context with related fields.



DeepMind

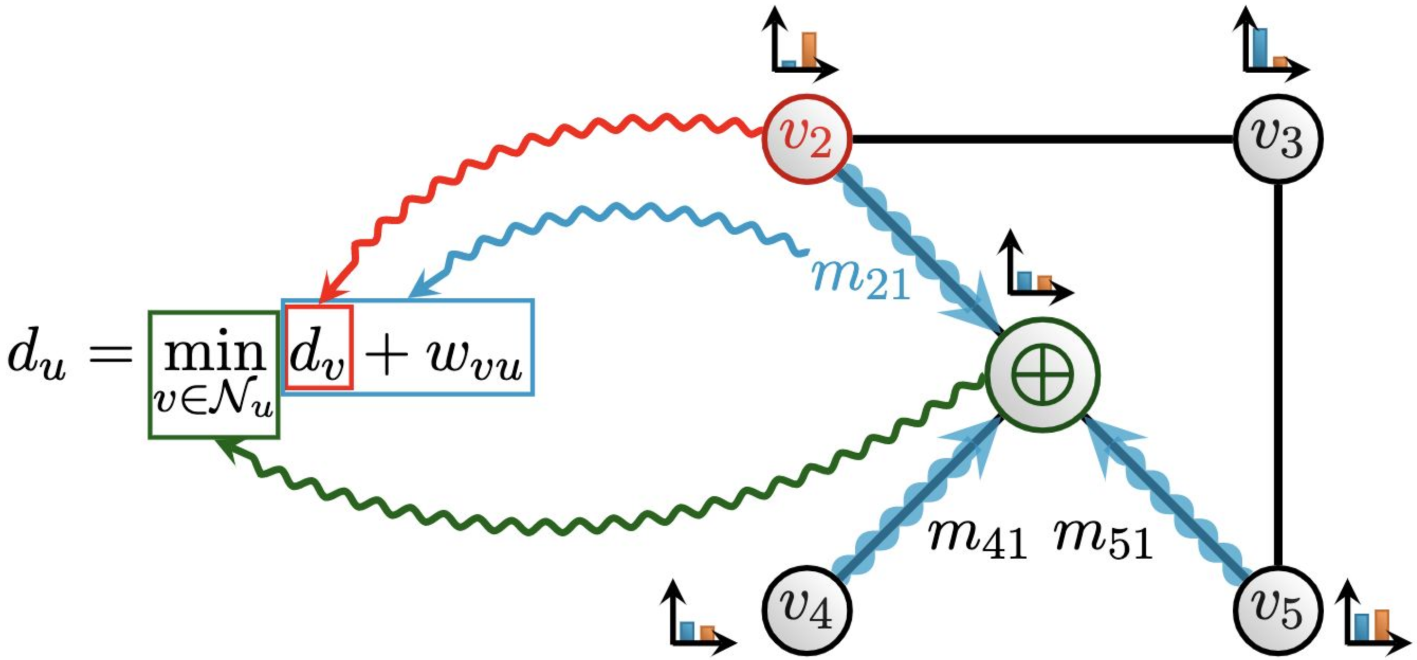
Theoretical Foundations of Graph Neural Networks

Petar Veličković

CST Wednesday Seminar
17 February 2021

What I **do**:
Playing the **alignment** game

Key concept: *algorithmic alignment*



To align GNNs with computation...

So far, our approaches have modified:

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi\left(\mathbf{x}_u, \mathbf{x}_v\right)\right)$$

To align GNNs with computation...

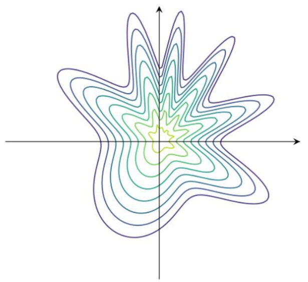
So far, our approaches have modified:

- The **parametric functions** of the GNN

IterGNN, Tang *et al.*, (NeurIPS'20) forces the parametric functions to be **homogeneous**

$$\psi(\mathbf{x}_u, \mathbf{x}_v) = \mathrm{ReLU}(\mathbf{W}_1\mathbf{x}_u + \mathbf{W}_2\mathbf{x}_v)$$

$$f(\lambda x) = \lambda f(x)$$



(a) $f(\lambda x) = \lambda f(x),\ \forall \lambda > 0, x \in R^2$

$$\mathbf{h}_u = \boxed{\phi\left(\mathbf{x}_u, \bigoplus_{v\in\mathcal{N}_u} \boxed{\psi\left(\mathbf{x}_u, \mathbf{x}_v\right)}\right)}$$

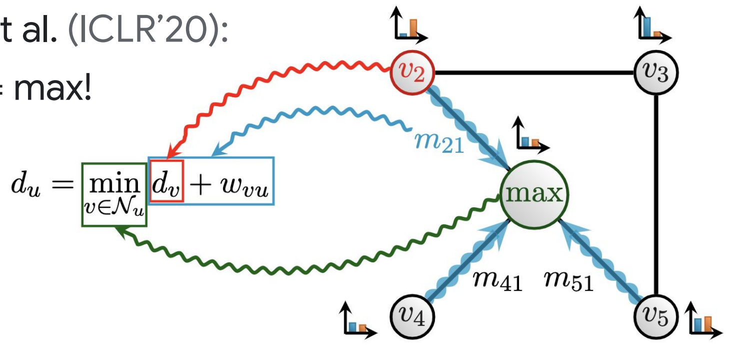
To align GNNs with computation...

So far, our approaches have modified:

- The parametric functions of the GNN
- The aggregation function of the GNN

Veličković et al. (ICLR'20):

Just set ⊕ = max!


$d_u = \boxed{\min_{v\in\mathcal{N}_u}} \boxed{\boxed{d_v} + w_{vu}}$
v_2
v_3
m_{21}
max
m_{41}
m_{51}
v_4
v_5

$$\mathbf{h}_u = \boxed{\phi\left(\mathbf{x}_u, \boxed{\bigoplus_{v\in\mathcal{N}_u}} \boxed{\psi\left(\mathbf{x}_u, \mathbf{x}_v\right)}\right)}$$

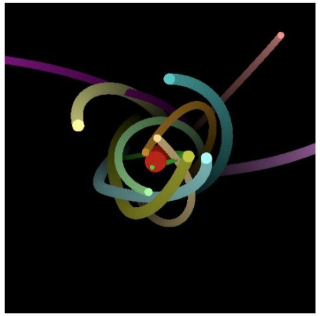
To align GNNs with computation...

So far, our approaches have modified:

- The **parametric functions** of the GNN
- The **aggregation function** of the GNN
- The **features** going into the GNN

n-body physics (Battaglia et al., NeurIPS'16)

Forces follow an **inverse square law** – transform the distance features!



$$\boldsymbol{w}_{(u,v)}^{(t)} = m_v \cdot \left(\boldsymbol{x}_v^{(t)} - \boldsymbol{x}_u^{(t)}\right) / \|\boldsymbol{x}_u^{(t)} - \boldsymbol{x}_v^{(t)}\|_2^3$$

$$\mathbf{h}_u = \boxed{\phi\left(\boxed{\mathbf{x}_u}, \boxed{\bigoplus}_{v\in\mathcal{N}_u} \boxed{\psi\left(\boxed{\mathbf{x}_u}, \boxed{\mathbf{x}_v}\right)}\right)}$$

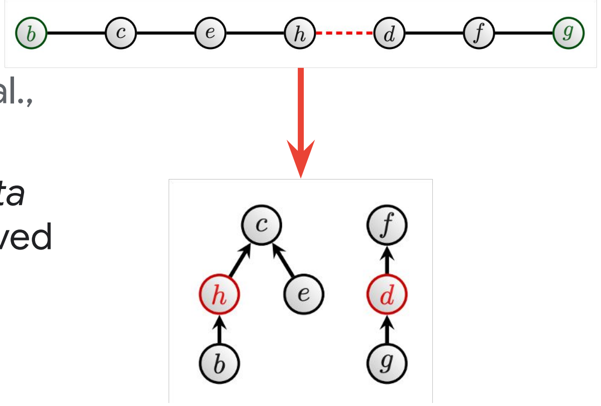
To align GNNs with computation...

So far, our approaches have modified:

- The **parametric functions** of the GNN
- The **aggregation function** of the GNN
- The **features** going into the GNN
- The **computation graph** over which the GNN operates

PGN (Veličković et al., NeurIPS'20)

Align edges to a *data structure* for improved time complexity!



$$\mathbf{h}_u = \boxed{\phi\left(\boxed{\mathbf{x}_u}, \boxed{\bigoplus}_{v\in\boxed{\mathcal{N}_u}} \boxed{\psi\left(\boxed{\mathbf{x}_u}, \boxed{\mathbf{x}_v}\right)}\right)}$$

To align GNNs with computation...

So far, our approaches have modified:

- The **parametric functions** of the GNN
- The **aggregation function** of the GNN
- The **features** going into the GNN
- The **computation graph** over which the GNN operates

$$\mathbf{h}_u = \boxed{\phi\left(\boxed{\mathbf{x}_u}, \boxed{\bigoplus}_{v\in\boxed{\mathcal{N}_u}} \boxed{\psi\left(\boxed{\mathbf{x}_u}, \boxed{\mathbf{x}_v}\right)}\right)}$$

... what else even is there to do???

Recap

So far, our approaches have modified:

- The **parametric functions** of the GNN
- The **aggregation function** of the GNN
- The **features** going into the GNN
- The **computation graph** over which the GNN operates

$$\mathbf{h}_u^{\boxed{(t+1)}} = \boxed{\phi\left(\boxed{\mathbf{x}_u}^{\boxed{(t)}}, \underset{v\in\boxed{\mathcal{N}_u}}{\boxed{\bigoplus}} \boxed{\psi\left(\boxed{\mathbf{x}_u}^{\boxed{(t)}}, \boxed{\mathbf{x}_v}^{\boxed{(t)}}\right)}\right)}$$

An **important** point is missing, and it was not even in the equation…
the ***clock*** ⏱️ which **synchronises** the message passing mechanism

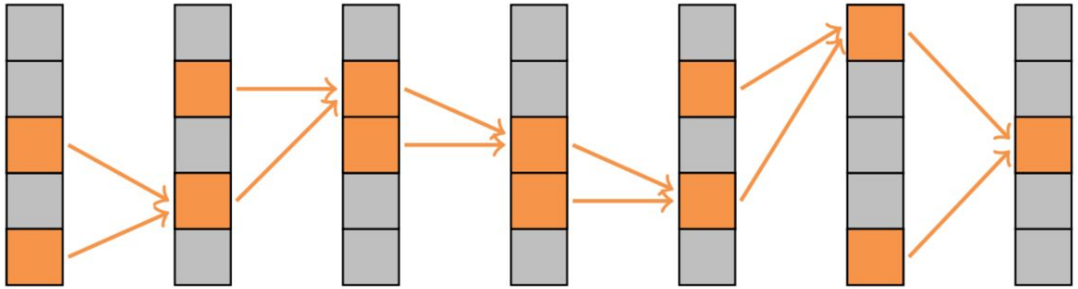
What I am **betting on:** Reconciling (G)NNs with ⏳ **Asynchrony** ⌛

Asynchrony ⏲️, the next frontier!

- As we discussed at the beginning, GNNs/Transformers tend to struggle when asked to reason, **especially** over very long trajectories, **especially** out-of-distribution

- Reasoning: executing a robust **procedure** for solving instances of a **problem**

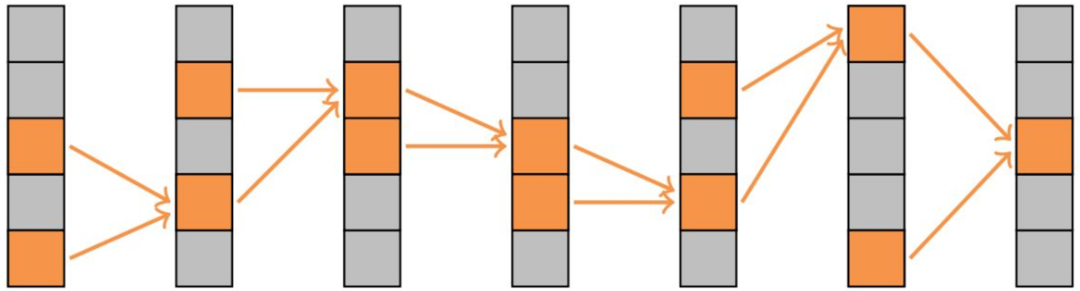
- Here we hit a *roadblock* to long-range generalisation: most problem-solving techniques are **asynchronous**—only a **handful of variables** can be meaningfully updated at each step.



- This is a topological constraint of the **problem**—*not* a limitation of aligning to algorithms!

Asynchrony ⏱️, the next frontier!

- Here we hit a *roadblock* to long-range generalisation: most problem-solving techniques are **asynchronous**—only a **handful of variables** can be meaningfully updated at each step.

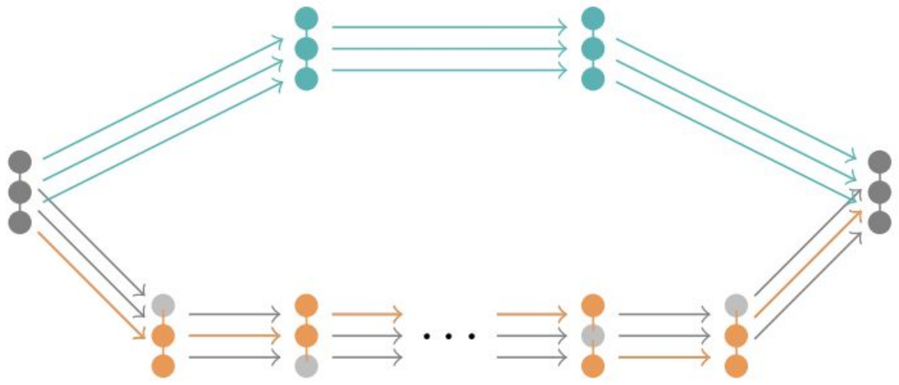


- This is a topological constraint of the **problem**—*not* a limitation of aligning to algorithms!
    - If you update a node when you shouldn't, it is **opportunistic** & prone to failures

- Yet, our models are **synchronous**: update ***all*** node/token states, ***everywhere***, ***all the time***

- Think about the *message / update functions* such a model needs to learn to generalise!
    - Identity-like for most nodes… but highly complex in others.

How to reconcile (G)NNs with asynchrony?



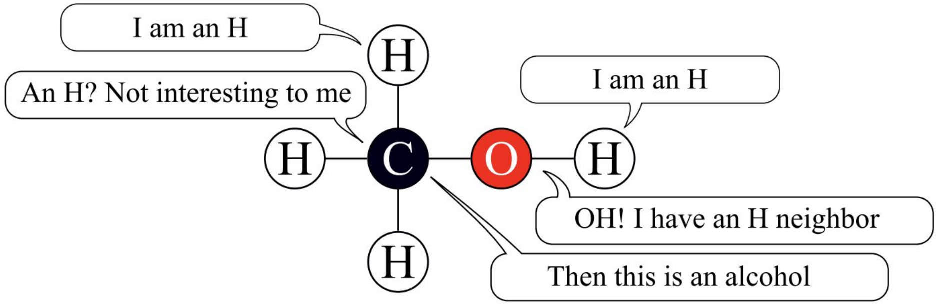
Parallel Algorithmic Alignment

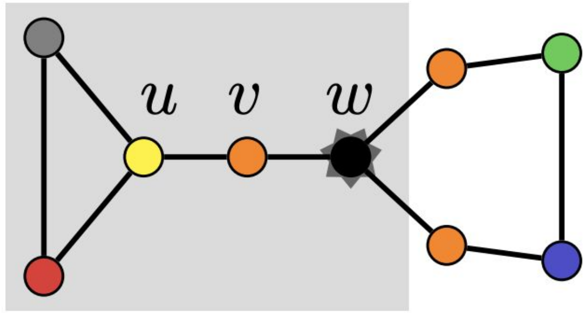
Learn to execute ***parallel*** *algorithms whenever possible!*

Engelmayer, Georgiev, Veličković (LoG'23)

Elegant, scalable and theoretically sound! But ***not always possible*** *:(*

How to reconcile (G)NNs with asynchrony?





Asynchronous GNNs

*Exchange and react to **individual** messages*

Faber and Wattenhofer (LoG'23)

Cooperative GNNs

*Nodes decide whether to **listen** or **broadcast***

Finkelshtein et al. (ICML'24)

*Directly solves the problem! But **hard to scale** on modern hardware, with many **discrete decisions***

How to reconcile (G)NNs with asynchrony?

$$\begin{tikzcd}
\mathbf{m}_{ba} \arrow[r] & \mathbf{x}'_{b\{a\}} \arrow[dr, "\phi"] & & \mathbf{x}_b \arrow[ll, "\phi"'] & & & & \\
\mathbf{m}_{bb} \arrow[rr] & & \mathbf{x}'_{b\{a,b\}} \arrow[rr, "\psi"] \arrow[dr, "\phi"] & & \mathbf{m}'_{db} \arrow[rr, dotted] & & d & \\
\mathbf{m}_{bc} \arrow[rrr] & & & \mathbf{x}'_b \arrow[rr, "\psi"] & & \mathbf{m}''_{db} \arrow[rr, dotted] & & d
\end{tikzcd}$$

Asynchronous Algorithmic Alignment

*Design **synchronous** GNNs that are provably **invariant** under various forms of **asynchrony***

Dudzik, von Glehn, Pascanu, Veličković (LoG'23)

*Hits a sweet spot between **soundness**, **scalability**, and **feasibility**!*

What we **developed**: **Asynchronous** Algorithmic Alignment

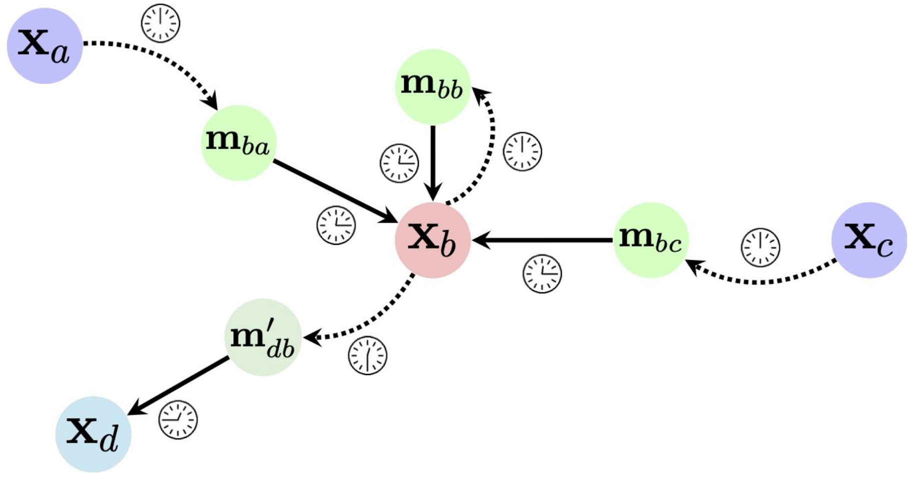
The general strategy towards asynchronous alignment

- Assume that our target algorithm supports certain kinds of asynchrony
    - e.g. variable updates can be processed in arbitrary order, without affecting the result
- Our strategy is to describe several *levels* of asynchronous alignment
- At each level, we will remove certain synchronisation points from the GNN
- Their removal, while maintaining invariance, will invoke a **constraint** on the GNN's modules
- Satisfying these constraints will induce ***asynchronous algorithmic alignment*** to algorithms supporting the same level of asynchrony

The synchronisation points of GNNs

Need to block on **all messages arriving** before invoking the **aggregator** and **update**

Need to block on the **update function** before invoking the **message function**



$$\begin{tikzcd}
\mathbf{m}_{ba} \arrow[dr, "\oplus"] & \mathbf{x}_b \arrow[dr, "\phi"] & & & \\
\mathbf{m}_{bb} \arrow[r] & \mathbf{m}_b \arrow[r] & \mathbf{x}'_b \arrow[r, "\psi"] & \mathbf{m}'_{db} \arrow[r, dashed] & d \\
\mathbf{m}_{bc} \arrow[ur] & & & &
\end{tikzcd}$$

What we need to describe

- There are three types of data at play here:
  - The set of **messages**, M
  - The set of node **states**, S *[which change as a result of receiving messages]*
  - The set of node **arguments**, A *[inputs to the message function]*

[in practice, often $M = S = A = \mathbb{R}^k$ *]*

What we need to describe

- There are three types of data at play here:
  - The set of **messages**, M
  - The set of node **states**, S *[which change as a result of receiving messages]*
  - The set of node **arguments**, A *[inputs to the message function]*

- Further, we will assume that both M and A admit a ***monoid*** $\circledcirc$ structure
  - Monoids are "*groups without inverses*": a great abstraction to study **computation**
  - There exist exact correspondences between *monoid actions* and *state machines*
  - In essence, we assume both messages and arguments can be ***iteratively assembled***

Petar Veličković

@PetarV_93

Staff Research Scientist @GoogleDeepMind | Affiliated Lecturer @Cambridge_Uni | Associate @clarehall_cam | GDL Scholar @ELLISforEurope. Monoids. 🇷🇸🇲🇪🇧🇦

What we need to describe

- There are three types of data at play here:
  - The set of **messages**, M
  - The set of node **states**, S *[which change as a result of receiving messages]*
  - The set of node **arguments**, A *[inputs to the message function]*

- Further, we will assume that both M and A admit a ***monoid*** ◎ structure

- We will write these two monoids as $(M, \cdot, 1)$ and $(A, +, 0)$

[The operators and neutral elements are chosen just for distinguishability; they may be identical.]

A single (potentially aggregated) message then *transforms* the **state** and *emits* an **argument**

$$(m, s) \mapsto (m \bullet s, \delta_m(s))$$

$$\bullet : M \times S \to S$$

$$\delta : M \times S \to A$$

Level 1 of asynchronous alignment

- First, note that our **message monoid** in GNNs is $(\mathbb{R}^k, \oplus, \mathbf{0}_\oplus)$

- To make the aggregation order irrelevant, we only need this monoid to be **commutative**:

$$\mathbf{m} \oplus \mathbf{n} = \mathbf{n} \oplus \mathbf{m}$$

- Then, messages can be aggregated as soon as they arrive, *without synchronisation*

- Most (if not all) GNNs use a commutative monoid for their aggregator!
    - See LCM (Ong and Veličković, LoG’22) for more details.

Level 1 of asynchronous alignment

$$\mathbf{x}'_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\right)$$

$$\begin{tikzcd}
\mathbf{m}_{ba} \arrow[dr, "\bigoplus"] & & \mathbf{x}_b \arrow[dr, "\phi"] & & & \\
\mathbf{m}_{bb} \arrow[r] & \mathbf{m}_{b\{a,b\}} \arrow[r, "\bigoplus"] & \mathbf{m}_b \arrow[r] & \mathbf{x}'_b \arrow[r, "\psi"] & \mathbf{m}'_{db} \arrow[r, dotted] & d \\
\mathbf{m}_{bc} \arrow[urr] & & & & &
\end{tikzcd}$$

Message monoid actions

$$(m, s) \mapsto (m \bullet s, \delta_m(s))$$

- We are interested in the effect of **multiple** messages arriving
    - Asynchrony would imply we can let those messages either act *separately*, or *simultaneously*, and still achieve the *same* result in both

- Forcing our transformations to be *monoid actions* induces a form of ***asynchrony invariance***

Message monoid actions

$$(m, s) \mapsto (m \bullet s, \delta_m(s))$$

- We are interested in the effect of **multiple** messages arriving
  - Asynchrony would imply we can let those messages either act *separately*, or *simultaneously*, and still achieve the *same* result in both

- Forcing our transformations to be *monoid actions* induces a form of ***asynchrony invariance***

- For **state transformations**, we would require:

$$1 \bullet s = s$$

$$(n \cdot m) \bullet s = n \bullet (m \bullet s)$$

1. neutral message does not change state, and
2. we can either transform state by two messages separately, or first compose them and transform once – the resulting state is the same

In GNNs

$$(m, s) \mapsto (m \bullet s, \delta_m(s))$$

$$1 \bullet s = s$$

$$(n \cdot m) \bullet s = n \bullet (m \bullet s)$$

- These conditions transfer to the following GNN update function conditions:

$$\phi(\mathbf{x}, \mathbf{0}_{\oplus}) = \mathbf{x}$$

$$\phi(\mathbf{x}, \mathbf{m} \oplus \mathbf{n}) = \phi(\phi(\mathbf{x}, \mathbf{m}), \mathbf{n})$$

- Then, messages can trigger updates as soon as they arrive, *without synchronisation*
- One simple way to satisfy this is to set $\phi = \bigoplus$ and rely on the associativity of $\bigoplus$

Message monoid actions

$$(m, s) \mapsto (m \bullet s, \delta_m(s))$$

- For **argument emission**, we would require:

$$\delta_1(s) = 0$$
$$\delta_{n \cdot m}(s) = \delta_m(s) + \delta_n(m \bullet s)$$

1. neutral message emits neutral argument, and
2. we can either compose two messages and emit an argument, or emit one argument from the first message, then emit a second argument from the transformed state by the second message, and combine the two emitted arguments – the resulting argument is the same

This condition is known under many names in mathematics:

1-cocycles, *derivations*, and *crossed homomorphisms*

Message monoid actions

$$(m, s) \mapsto (m \bullet s, \delta_m(s))$$

$$\delta_1(s) = 0$$
$$\delta_{n \cdot m}(s) = \delta_m(s) + \delta_n(m \bullet s)$$

- Assuming an argument monoid of $(\mathbb{R}^m, \otimes, \mathbf{1}_\otimes)$ this translates into following constraints:

$$\phi(\mathbf{x}, \mathbf{0}_\oplus) = \mathbf{1}_\otimes$$
$$\phi(\mathbf{x}, \mathbf{m} \oplus \mathbf{n}) = \phi(\mathbf{x}, \mathbf{m}) \otimes \phi(\phi(\mathbf{x}, \mathbf{m}), \mathbf{n})$$

- Then, messages can emit partial arguments as soon as they arrive, *without synchronisation*

- If $M = S = A$ and $\oplus = \otimes$, it is sufficient to make the update **idempotent** to satisfy this.
  - Combining with the previous rule, we can set, e.g., $\phi = \oplus = \max$

Level 2 of asynchronous alignment

$$\mathbf{x}'_u = \max\left(\mathbf{x}_u, \max_{v\in\mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\right)$$

$$\begin{tikzcd}
\mathbf{m}_{ba} \arrow[r] & \mathbf{x}'_{b\{a\}} \arrow[dr, "\phi"] & & \mathbf{x}_b \arrow[ll, "\phi"'] & & \\
\mathbf{m}_{bb} \arrow[rr] & & \mathbf{x}'_{b\{a,b\}} \arrow[dr, "\phi"] & & & \\
\mathbf{m}_{bc} \arrow[rrr] & & & \mathbf{x}'_b \arrow[r, "\psi"] & \mathbf{m}'_{db} \arrow[r, dotted] & d
\end{tikzcd}$$

Message computations

- Once **argument(s)** are computed, they can be used to produce new **messages**
  - This is a ***stateless*** *monoid transformation*, as edges don't keep persistent state
  - *(If they did, our framework would treat them as nodes anyway!)*

- Assume the specific case of *single-argument* messages ($\psi : A \to M$).

- The conditions for making ψ a 1-cocycle amount to making it a *monoid homomorphism*:

$$\psi(0) = 1$$
$$\psi(a + b) = \psi(a) \cdot \psi(b)$$

1. neutral argument produces neutral message, and
2. we can either produce a single message from combining two arguments, or produce one message from each argument, and then compose them – the resulting message is the same

Message computations

- Once **argument(s)** are computed, they can be used to produce new **messages**
  - This is a ***stateless*** *monoid transformation*, as edges don't keep persistent state
  - *(If they did, our framework would treat them as nodes anyway!)*

- Assume the specific case of *single-argument* messages ($\psi : A \to M$).

- The conditions for making ψ a 1-cocycle amount to making it a *monoid homomorphism*:

$$\psi(0) = 1$$
$$\psi(a + b) = \psi(a) \cdot \psi(b)$$

This generalises to messages produced from ***multiple*** arguments, $\psi : A_1 \times \cdots \times A_k \to M$
we'd need ψ to be a *monoid multimorphism* for it to be a 1-cocycle
(if we keep *k - 1* arguments *fixed*, it needs to be a monoid homomorphism in the *remaining* one)

Message computations

$$\psi(0) = 1$$
$$\psi(a + b) = \psi(a) \cdot \psi(b)$$

- If we follow all our prior assumptions, this translates into the following constraints:

$$\psi(-\infty) = -\infty$$
$$\psi(\max(\mathbf{a}, \mathbf{b})) = \max(\psi(\mathbf{a}), \psi(\mathbf{b}))$$

- Then, messages can be sent whenever arguments are updated, *without synchronisation*

- To satisfy this, we can make the message function a ***tropical linear*** layer in the max-semiring
  - Matrix multiplication with a weight matrix, but changing **times** to **plus**, and **plus** to **max**
  - Can obtain a *multimorphism* via a tropical *multilinear* layer

Level 3 of asynchronous alignment

$$\mathbf{x}'_u = \max\left(\mathbf{x}_u, \max_{v\in\mathcal{N}_u}\left(\mathbf{W}\otimes_{(\max,+)}\mathbf{x}_u + \mathbf{U}\otimes_{(\max,+)}\mathbf{x}_v\right)\right)$$

$$\begin{tikzcd}
\mathbf{m}_{ba} \arrow[r] & \mathbf{x}'_{b\{a\}} \arrow[dr, "\phi"] & & \mathbf{x}_b \arrow[ll, "\phi"'] & & \\
\mathbf{m}_{bb} \arrow[rr] & & \mathbf{x}'_{b\{a,b\}} \arrow[dr, "\phi"] & & & \\
\mathbf{m}_{bc} \arrow[rrr] & & & \mathbf{x}'_b \arrow[r, "\psi"] & \mathbf{m}'_{db} \arrow[r, dotted] & d
\end{tikzcd}$$

Level 3 of asynchronous alignment

- It’s worthy to take a step back and see what the “tropical linear” layer does!

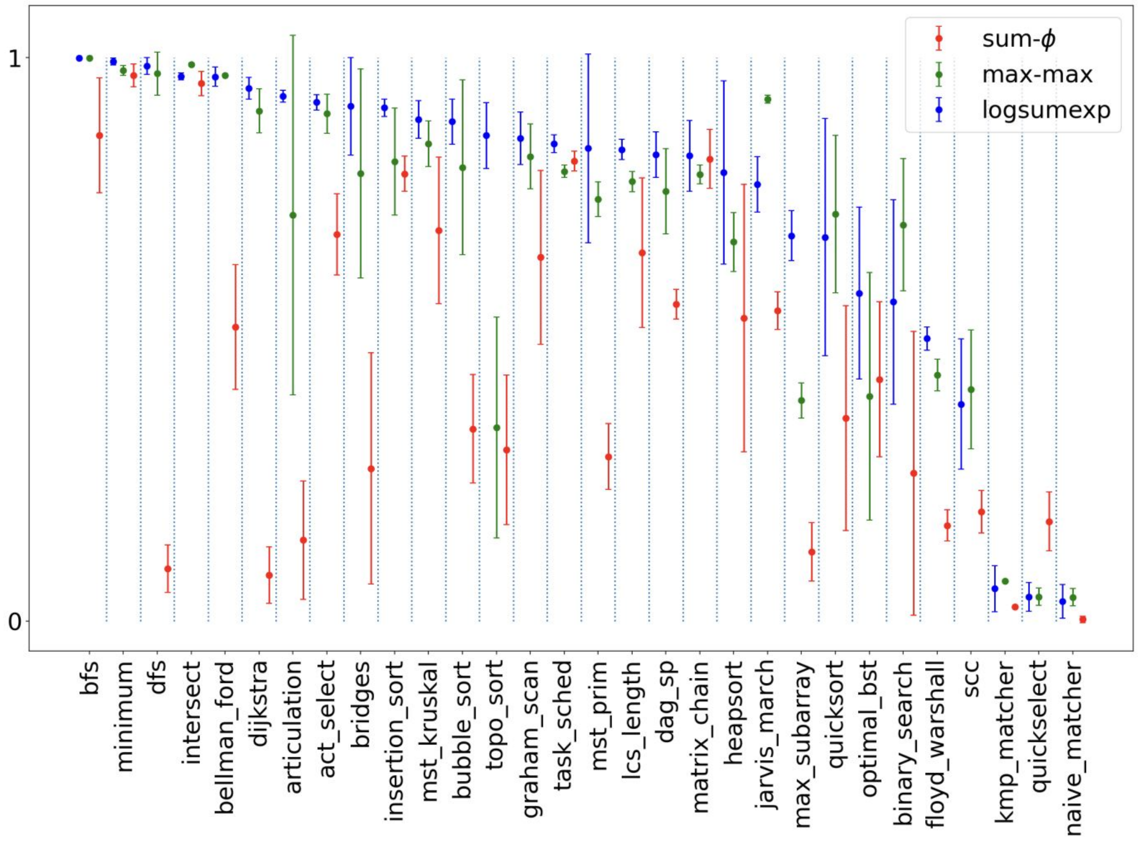
- It is well-known we can express *shortest path-finding* algorithms as repeated matrix-vector multiplication of the distance matrix with node distances, in the max-plus semiring.

- Here, we are multiplying a **weight matrix** with *high-dimensional* ***vectors*** in every node

- Hence, we can imagine such a layer as solving several path-finding problems in parallel
    - Each starting from different *initial distance conditions*

- Issue: we have ***quadratically*** many parameters, but only ***linearly*** many will receive *gradients*!
    - As a quick fix: use the logsumexp semiring instead, as a smooth approximation.

Ascending this ladder improves OOD execution capability



Want to know more?

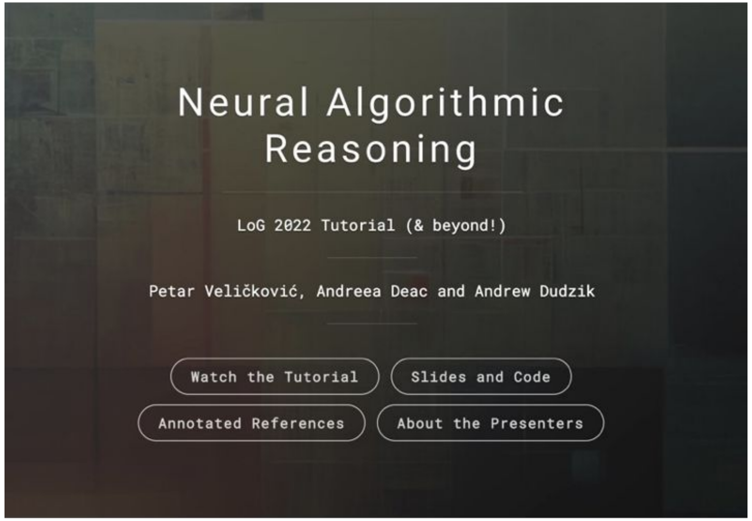
Want to know more? (NAR side)

There's **a lot** of stuff I didn't have time for today.

Want to learn more? Detailed list of references?

Check our *LoG'22 Tutorial*!

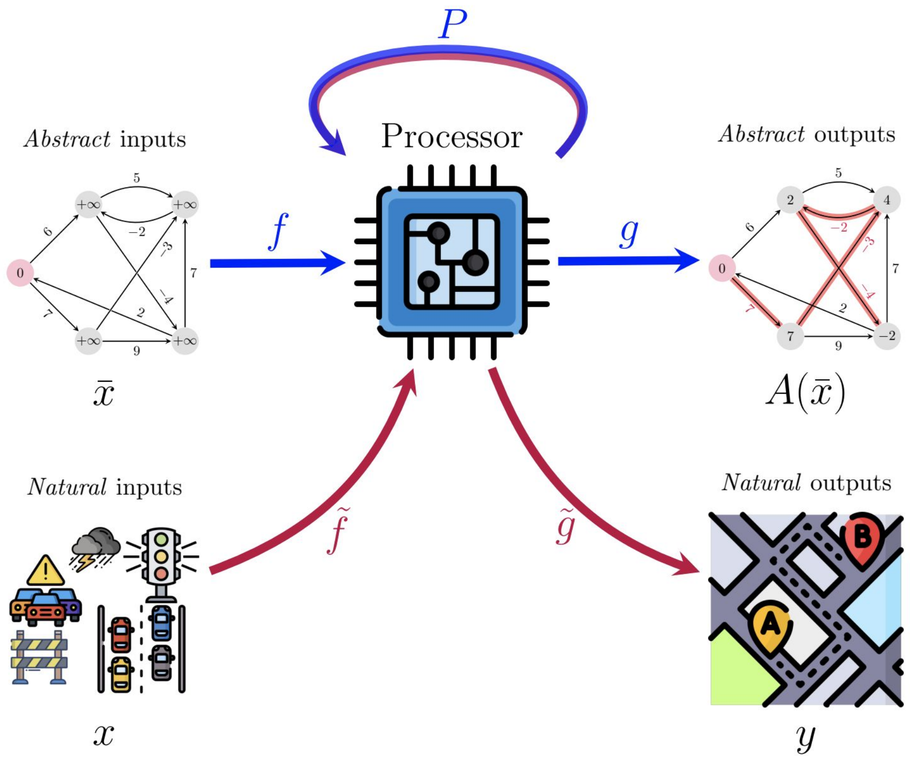
`algo-reasoning.github.io`



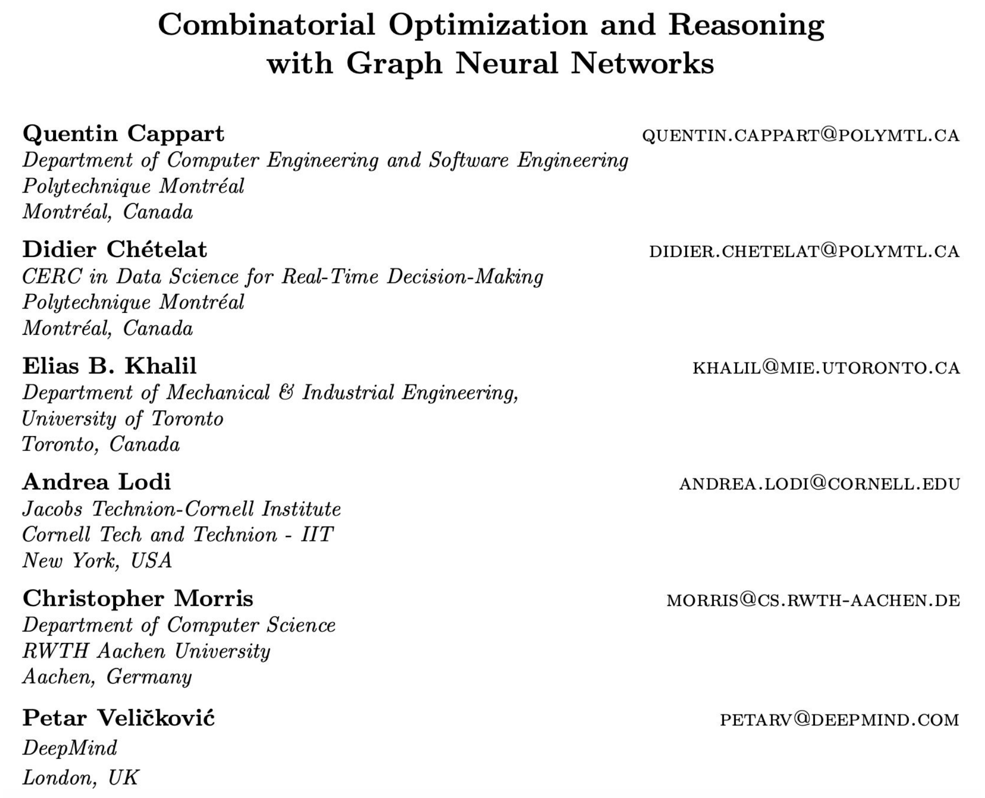
Neural Algorithmic Reasoning

LoG 2022 Tutorial (& beyond!)

Petar Veličković, Andreea Deac and Andrew Dudzik



Want to know more? (CO side)



Combinatorial Optimization and Reasoning with Graph Neural Networks

Quentin Cappart QUENTIN.CAPPART@POLYMTL.CA
Department of Computer Engineering and Software Engineering
Polytechnique Montréal
Montréal, Canada

Didier Chételat DIDIER.CHETELAT@POLYMTL.CA
CERC in Data Science for Real-Time Decision-Making
Polytechnique Montréal
Montréal, Canada

Elias B. Khalil KHALIL@MIE.UTORONTO.CA
Department of Mechanical & Industrial Engineering,
University of Toronto
Toronto, Canada

Andrea Lodi ANDREA.LODI@CORNELL.EDU
Jacobs Technion-Cornell Institute
Cornell Tech and Technion - IIT
New York, USA

Christopher Morris MORRIS@CS.RWTH-AACHEN.DE
Department of Computer Science
RWTH Aachen University
Aachen, Germany

Petar Veličković PETARV@DEEPMIND.COM
DeepMind
London, UK

For the CO practitioners in the audience:

Our ***61-page* survey** on GNNs for CO!

https://arxiv.org/abs/2102.09544

(JMLR'23)

Section 3.3. details algorithmic reasoning, with comprehensive references.

Want to know more? (CDL side)

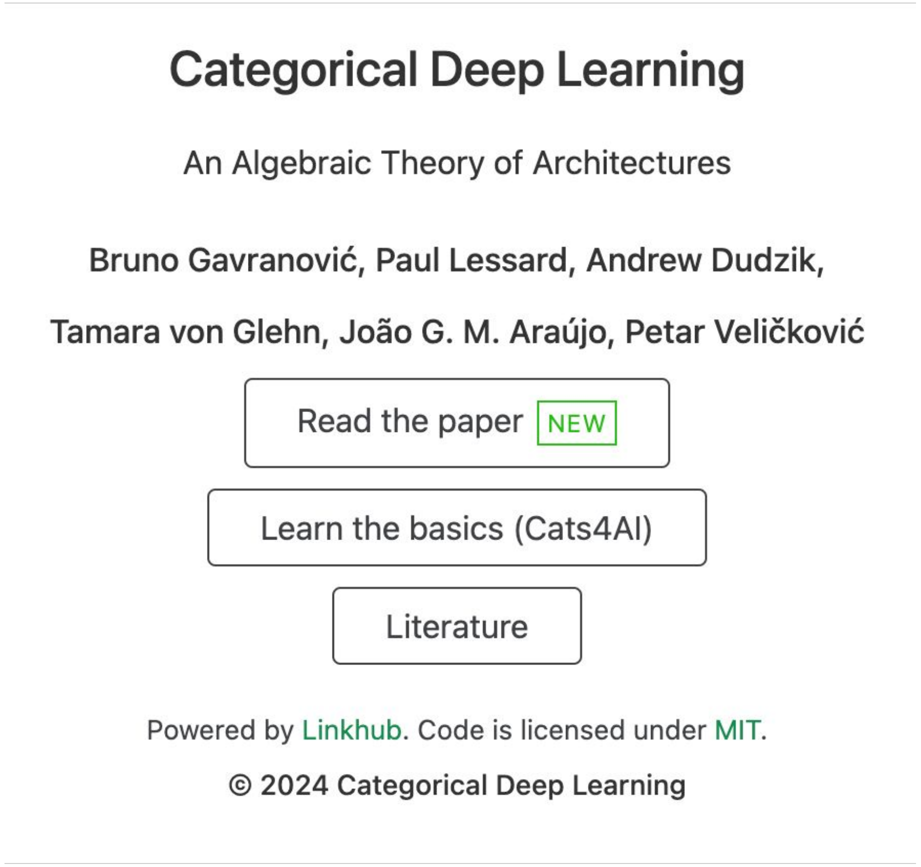
The idea of removing constraints from *groups* to analyse neural networks is a hint at a much richer picture, based on *category theory*

We recently wrote a paper outlining this picture:

`categoricaldeeplearning.com`

and feedback is very much welcome!

As a bonus: we can rediscover 1-cocycle conditions using the gadgets proposed in CDL (see Appendix).



Categorical Deep Learning

An Algebraic Theory of Architectures

Bruno Gavranović, Paul Lessard, Andrew Dudzik,

Tamara von Glehn, João G. M. Araújo, Petar Veličković

Powered by Linkhub. Code is licensed under MIT.

© 2024 Categorical Deep Learning

Want to know more? (CDL side)

The idea of removing constraints from *groups* to analyse neural networks is a hint at a much richer picture, based on *category theory*

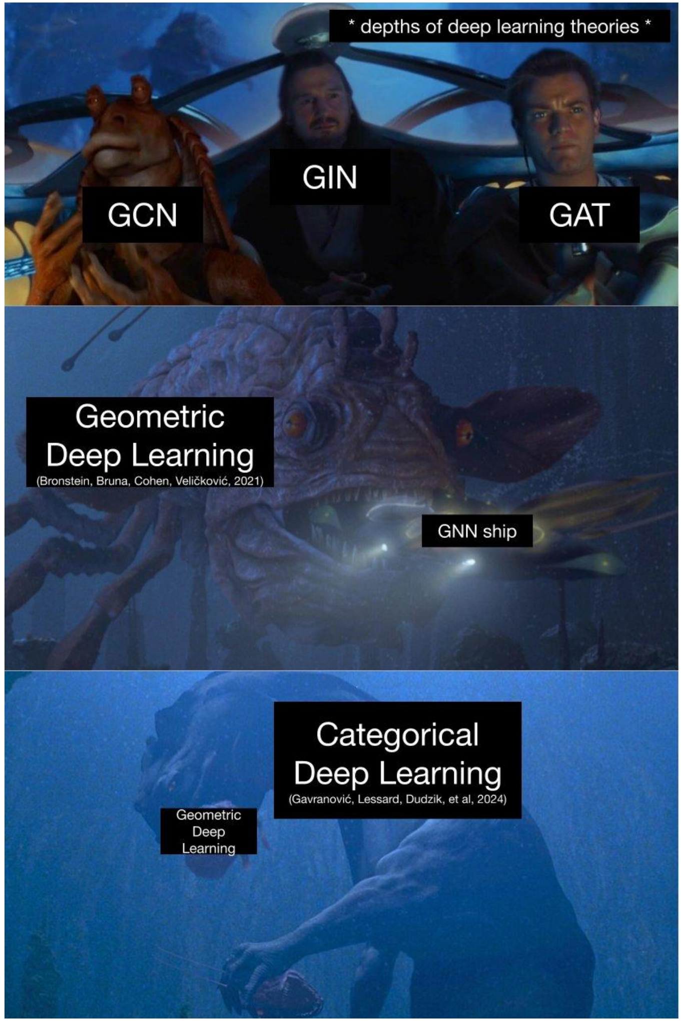
We recently wrote a paper outlining this picture:

`categoricaldeeplearning.com`

and feedback is very much welcome!

As a bonus: we can rediscover 1-cocycle conditions using the gadgets proposed in CDL (see Appendix).

Meme credits go to Michael Galkin, as always :)


* depths of deep learning theories *
GCN
GIN
GAT
Geometric Deep Learning (Bronstein, Bruna, Cohen, Veličković, 2021)
GNN ship
Categorical Deep Learning (Gavranović, Lessard, Dudzik, et al, 2024)
Geometric Deep Learning

Thank you.

Petar Veličković

petarv@google.com

https://petar-v.com

Google DeepMind

With many thanks to Andrew Dudzik, Tamara von Glehn and Razvan Pascanu