

DeepMind

Reasoning on Natural Inputs

Petar Veličković

IPAM DLC Workshop
23 February 2021



In this talk:
(Classical) Algorithms

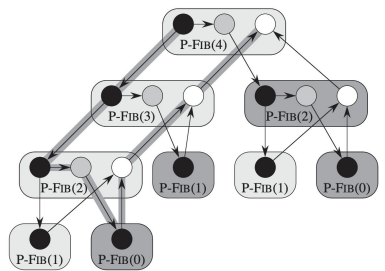


DeepMind

MERGE-SORT(A, p, r)

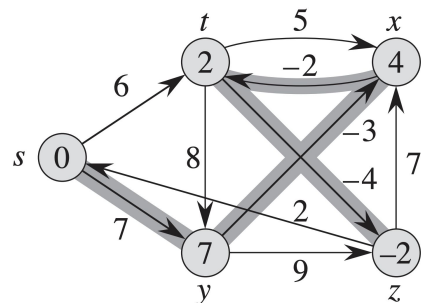
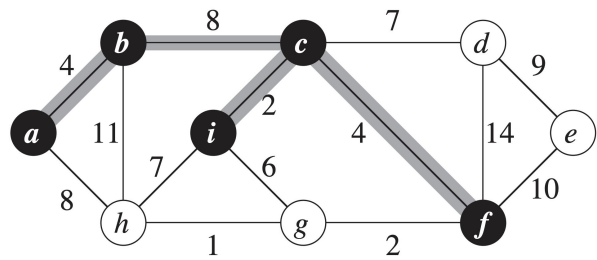
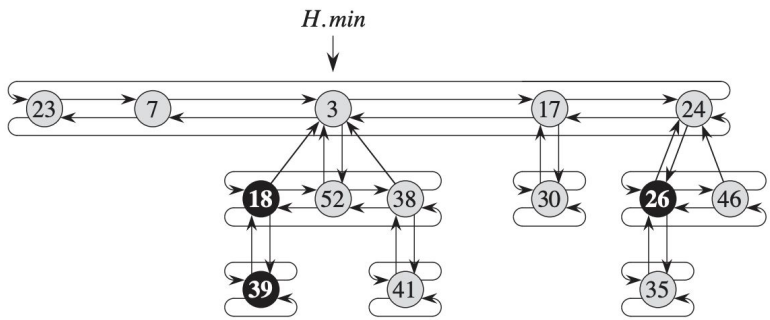
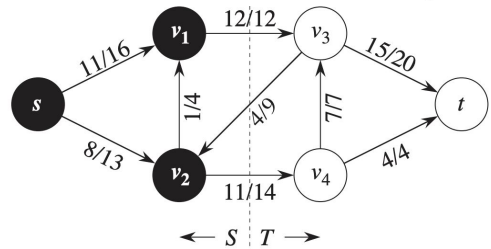
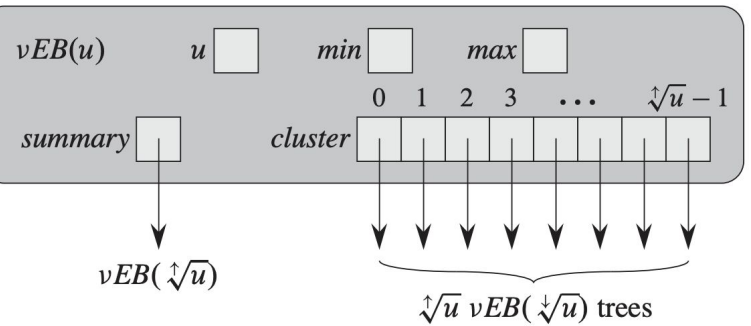
```

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( $A, p, q, r$ )
    
```

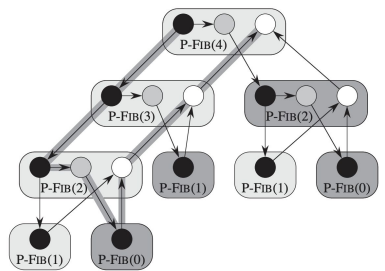


| | j | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|-----|-------|---|---|---|---|---|---|---|
| i | y_j | B | D | C | A | B | A | |
| 0 | x_i | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | A | 0 | ↑ | 0 | ↑ | 0 | ← | 1 |
| 2 | B | 0 | ↑ | 1 | ← | 1 | ↑ | 2 |
| 3 | C | 0 | ↑ | 1 | ↑ | 2 | ← | 2 |
| 4 | B | 0 | ↑ | 1 | ↑ | 2 | ↑ | 3 |
| 5 | D | 0 | ↑ | 1 | 2 | ↑ | 2 | ↑ |
| 6 | A | 0 | ↑ | 1 | 2 | ↑ | 3 | ↑ |
| 7 | B | 0 | ↑ | 1 | 2 | ↑ | 3 | ↑ |

In this talk: (Classical) Algorithms



DeepMind

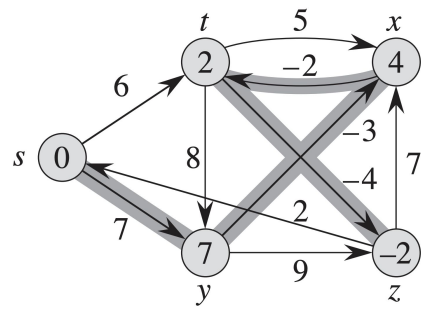
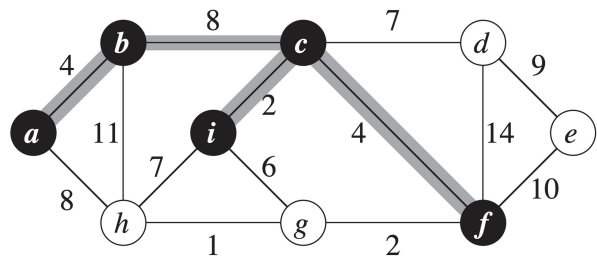
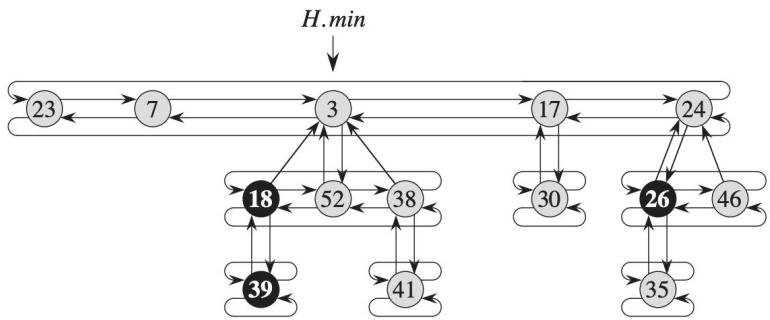
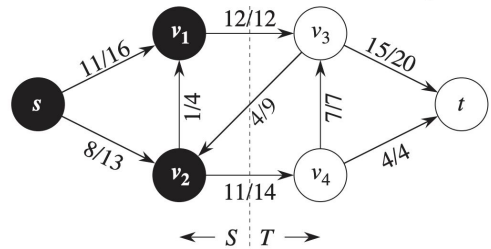
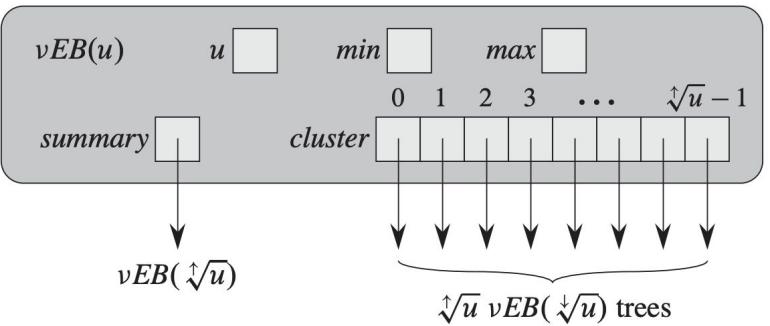


MERGE-SORT(A, p, r)

- 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)

| | j | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|-----|-------|---|---|---|---|---|---|---|
| i | y_j | B | D | C | A | B | A | |
| 0 | x_i | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | A | 0 | ↑ | 0 | ↑ | 0 | ← | 1 |
| 2 | B | 0 | ↓ | ← | ← | 1 | ↑ | ← |
| 3 | C | 0 | ↑ | ↑ | ↑ | 2 | ↑ | ↑ |
| 4 | B | 0 | ↑ | 1 | 2 | 2 | 3 | ← |
| 5 | D | 0 | ↑ | 2 | 2 | 2 | 3 | ↑ |
| 6 | A | 0 | ↑ | 1 | 2 | 2 | 3 | ↑ |
| 7 | B | 0 | ↑ | 1 | 2 | 2 | 3 | ↑ |

In this talk:
(Classical) Algorithms
(with a bit of neural spice)



1

Motivation for studying algorithms



Why algorithms?

- Essential “**pure**” forms of combinatorial **reasoning**
 - ‘Timeless’ principles that will remain regardless of the model of computation
 - Completely decoupled from any form of **perception***

**though perception itself may also be expressed in the language of algorithms*



Why algorithms?

- Essential “**pure**” forms of combinatorial **reasoning**
 - ‘Timeless’ principles that will remain regardless of the model of computation
 - Completely decoupled from any form of **perception***
- **Favourable** properties
 - Trivial **strong** generalisation
 - **Compositionality** via *subroutines*
 - Provable **correctness** and **performance** guarantees
 - Interpretable **operations** / *pseudocode*



Why algorithms?

- Essential “**pure**” forms of combinatorial **reasoning**
 - ‘Timeless’ principles that will remain regardless of the model of computation
 - Completely decoupled from any form of **perception***
- **Favourable** properties
 - Trivial **strong** generalisation
 - **Compositionality** via *subroutines*
 - Provable **correctness** and **performance** guarantees
 - Interpretable **operations** / *pseudocode*
- Hits *close to home*
 - Algorithms and competitive programming are how I got into Computer Science



2

Maximum flow and the Ford-Fulkerson algorithm



Maximum flow problem

- **Flow network:** graph $G = (V, E)$, augmented with a *capacity function*, $c: V \times V \rightarrow \mathbb{R}^+$
 - Capacity c_{uv} denotes how much **flow** is allowed on (u, v) edge
- Two special nodes: source, **s**, and sink, **t**
 - Source unleashes “infinite” capacity, sink receives “infinite” capacity

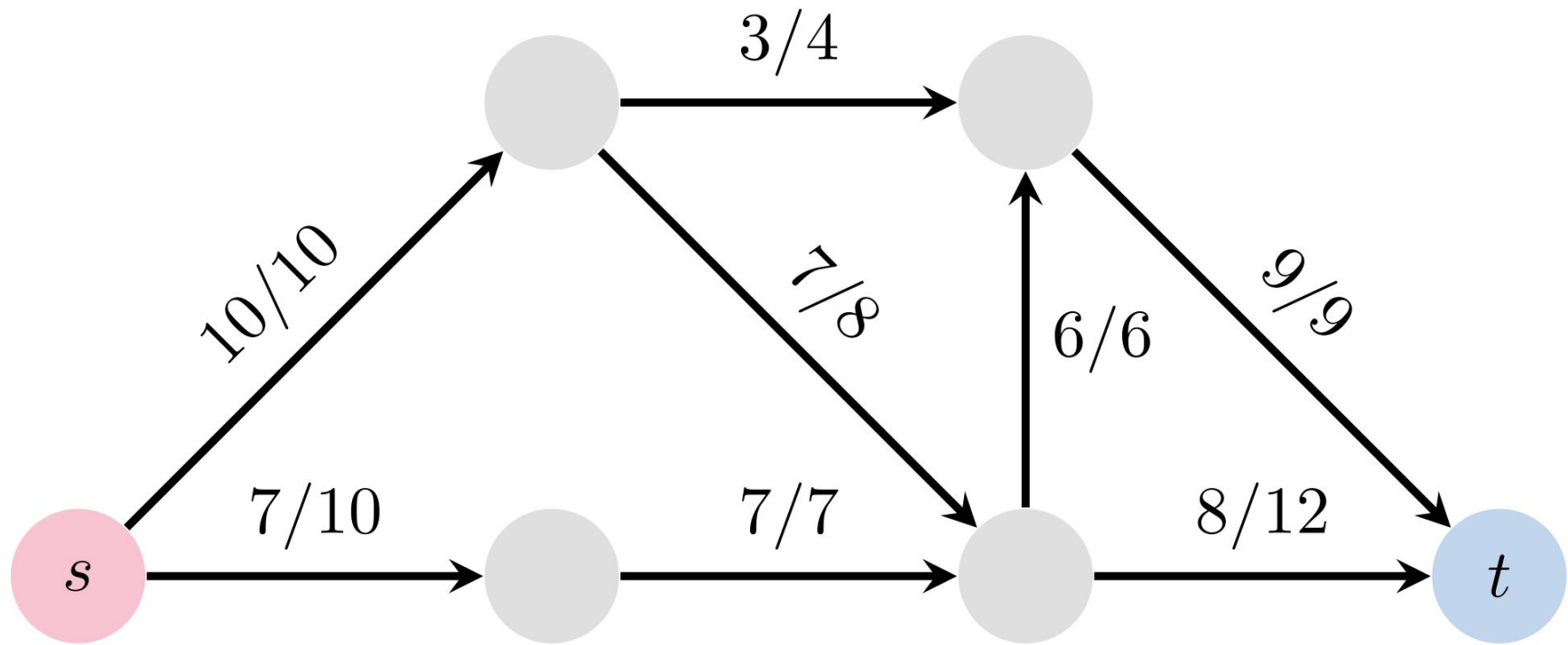
- A **flow** in G is any mapping $f: V \times V \rightarrow \mathbb{R}^+$, such that:

$$\begin{aligned} \forall u, v \in V \quad f_{u,v} &\leq c_{u,v} \\ \forall u \in V \setminus \{s, t\} \quad \sum_{v \in V} f_{v,u} &= \sum_{v \in V} f_{u,v} \end{aligned}$$

- The **value** of a flow is the total flow emanating from the source: $\sum_{v \in V} f_{s,v} - \sum_{v \in V} f_{v,s}$
 - We are interested in **maximising** it!



Max-flow example ($f=17$)



Ford-Fulkerson's Algorithm

- Such a **rigorously** defined problem often admits remarkably **elegant** and **provably correct** algorithm blueprint!

FORD-FULKERSON-METHOD(G, s, t)

1 initialize flow f to 0

2 **while** there exists an augmenting path p in the residual network G_f

3 augment flow f along p

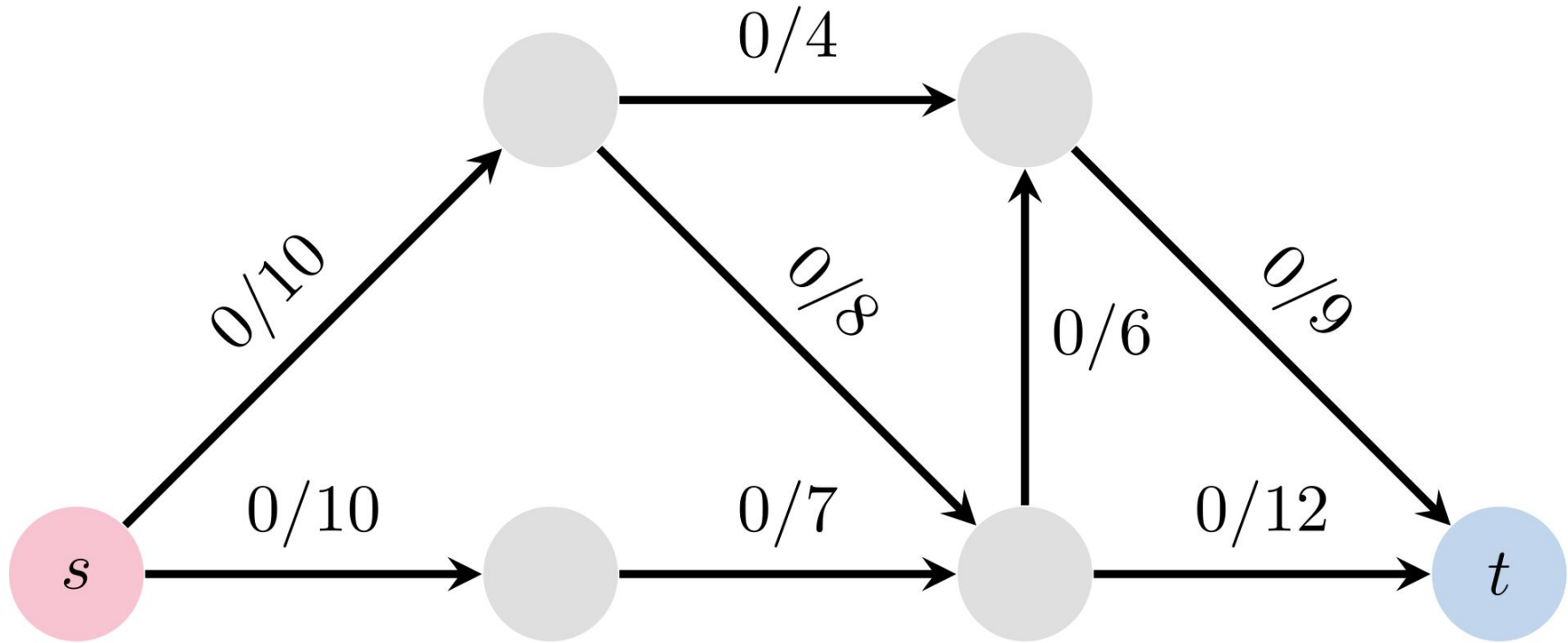
4 **return** f

**representing the capacities that remain after applying f*

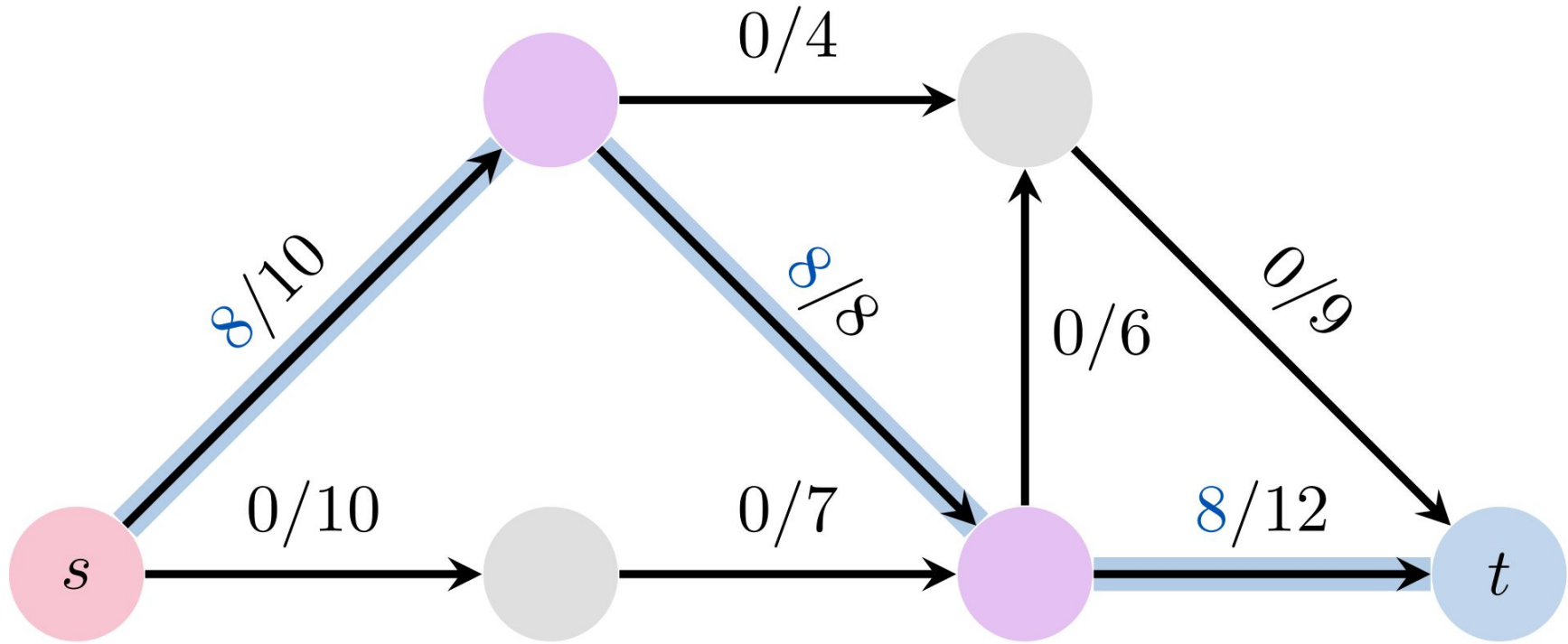
- Many specific ways to find p yield different algorithms (e.g. Edmonds–Karp, Dinitz, etc...)
 - This can be proven to terminate with correct solution



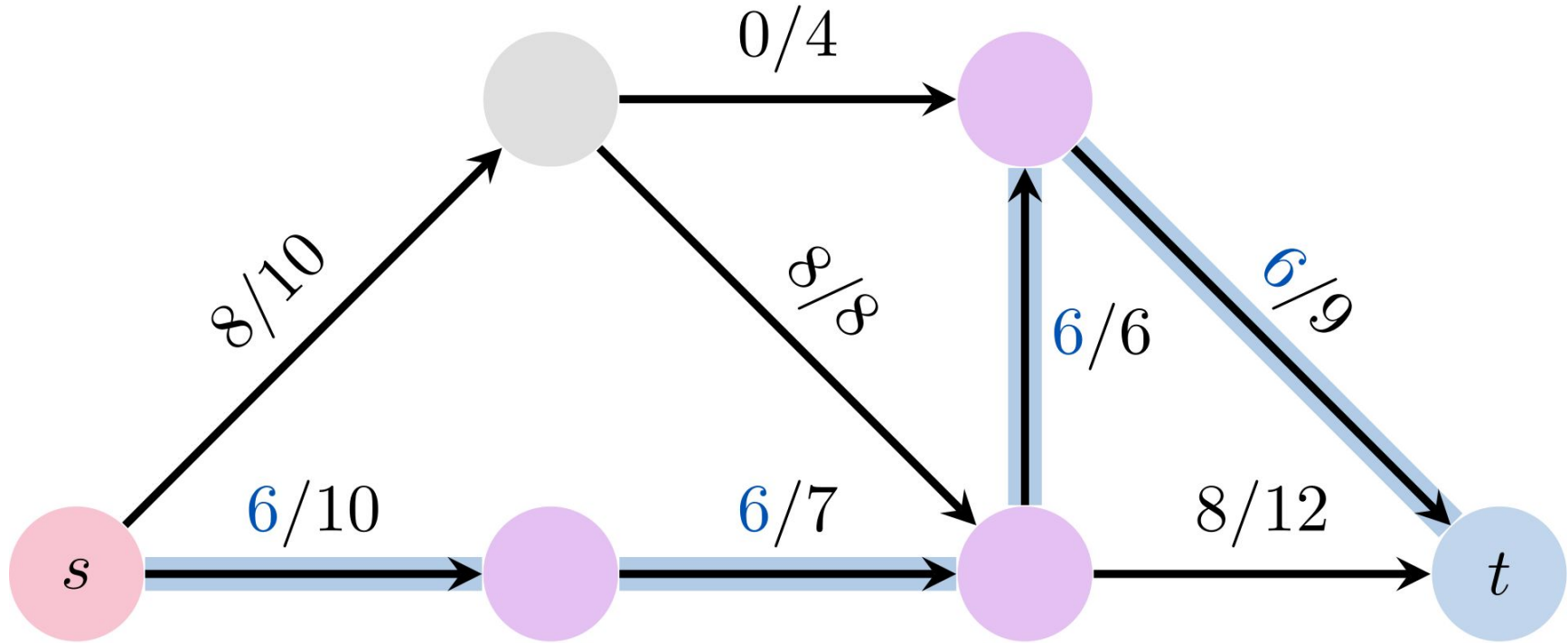
Ford-Fulkerson in action



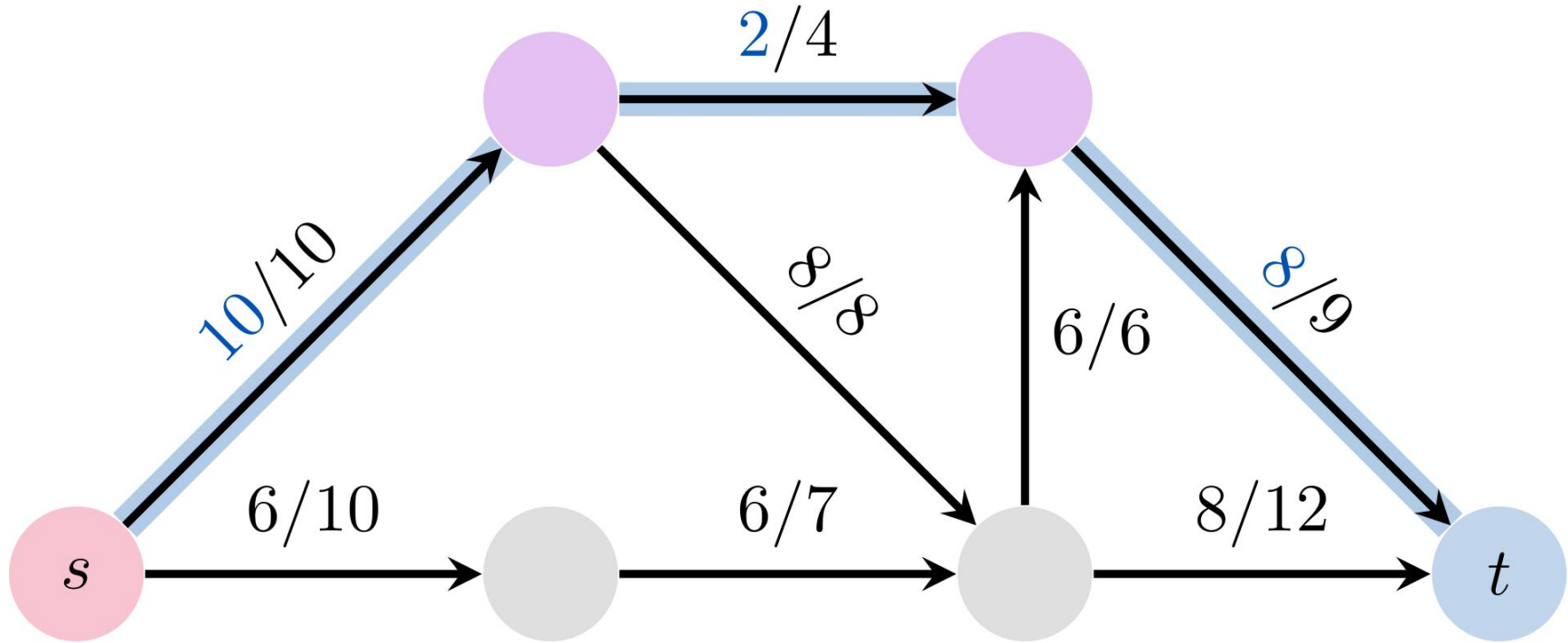
Ford-Fulkerson in action



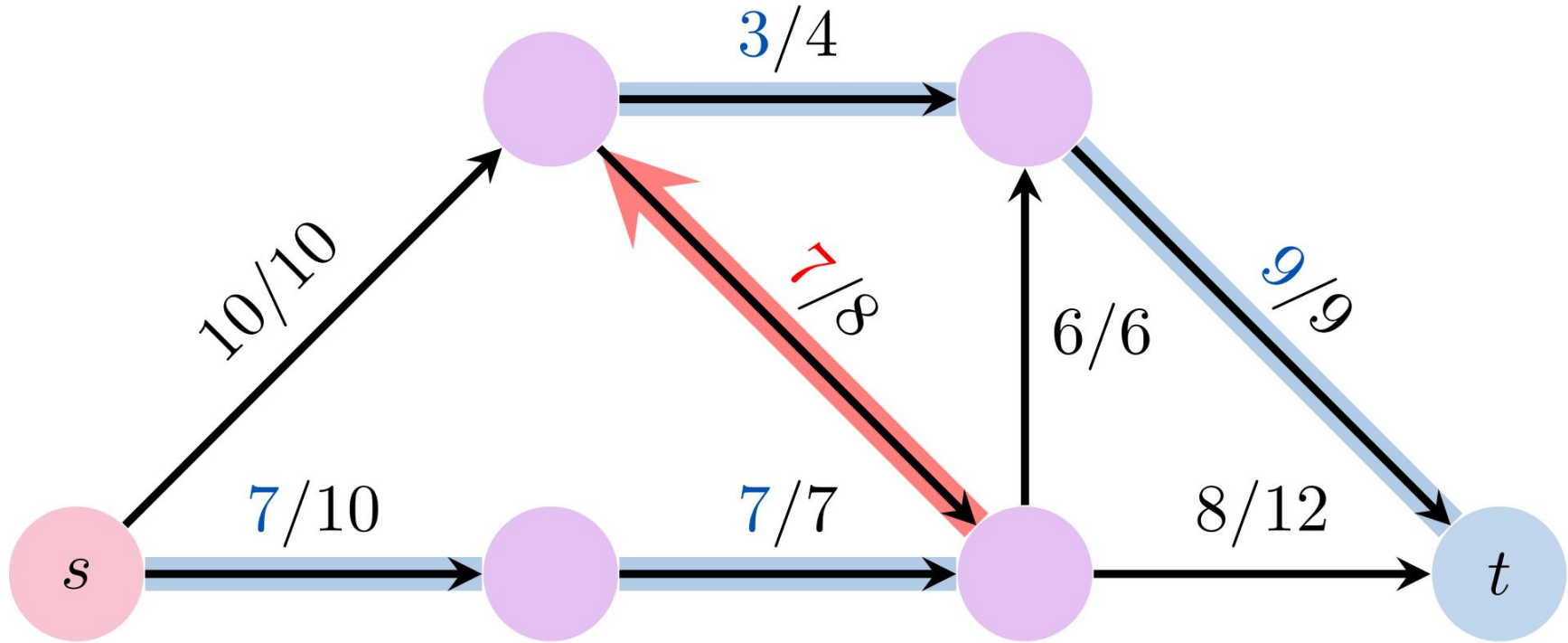
Ford-Fulkerson in action



Ford-Fulkerson in action



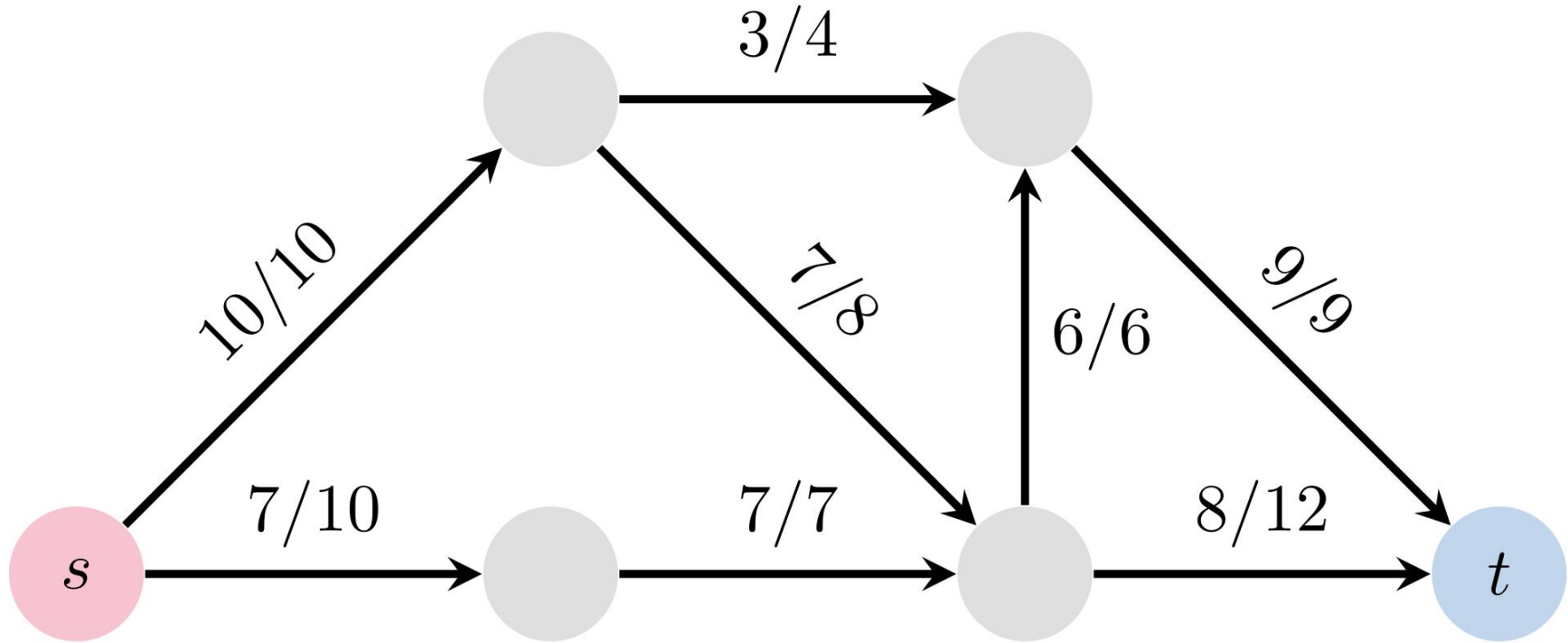
Ford-Fulkerson in action



(the flow may also be **returned**!)



Final solution!



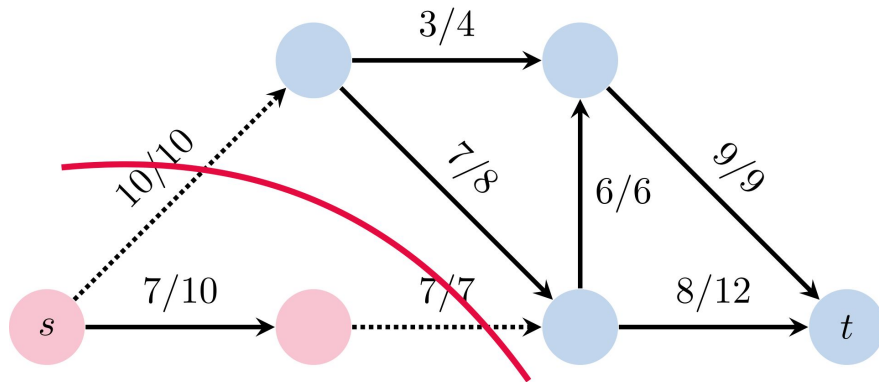
Max-flow Min-cut theorem

Observing data in this way, also yields easy observation of **connections**, hence **theorems**!

Theorem 26.6 (Max-flow min-cut theorem)

If f is a flow in a flow network $G = (V, E)$ with source s and sink t , then the following conditions are equivalent:

1. f is a maximum flow in G .
2. The residual network G_f contains no augmenting paths.
3. $|f| = c(S, T)$ for some cut (S, T) of G .



3

**“Fundamentals
of a method for
evaluating rail
net capacities”**

(Harris & Ross, 1955)



The core problem

- Classical algorithms are designed with *abstraction* in mind, enforcing their inputs to conform to stringent **preconditions**.
 - Keeping the inputs constrained enables an uninterrupted focus on “reasoning”
 - Easily certify the resulting procedure’s correctness, i.e., stringent **postconditions**
- However, we must never forget **why** we design algorithms!
- Unfortunately, this is at **timeless odds** with the way they are designed
 - Let’s study an example from the 1950s.



Original interest in flows

SECRET

SUMMARY

U. S. AIR FORCE

PROJECT RAND

RESEARCH MEMORANDUM

FUNDAMENTALS OF A METHOD FOR EVALUATING
RAIL NET CAPACITIES (U)

T. E. Harris
F. S. Ross

RM-1573

October 24, 1955

Copy No. 157

Air power is an effective means of interdicting an enemy's rail system, and such usage is a logical and important mission for this Arm.

As in many military operations, however, the success of interdiction depends largely on how complete, accurate, and timely is the commander's information, particularly concerning the effect of his interdiction-program efforts on the enemy's capability to move men and supplies. This information should be available at the time the results are being achieved.

<https://apps.dtic.mil/dtic/tr/fulltext/u2/093458.pdf>

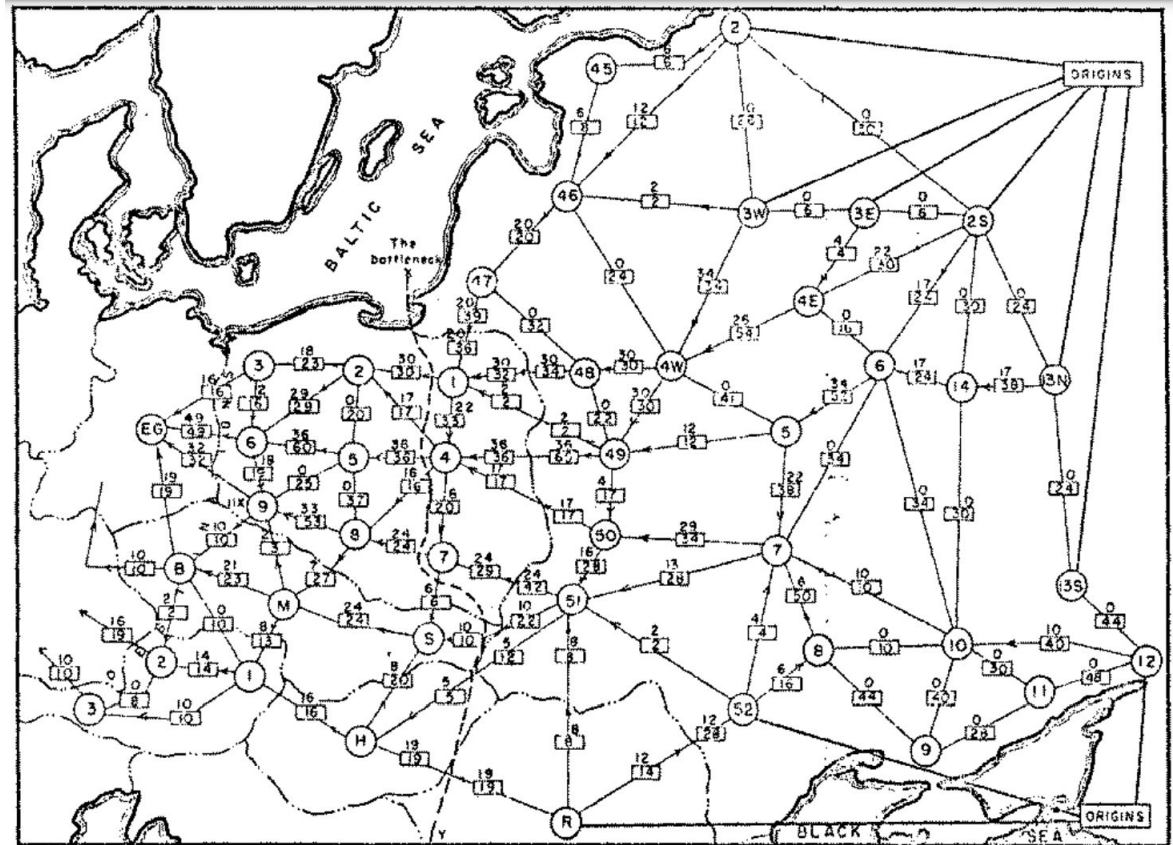


The Warsaw Pact railway network

Find “the bottleneck”, i.e. the **minimum cut**.

As we saw, this is directly related to computing the **maximum flow**.

(this was *intuitively* assumed by Harris & Ross as well)



The core problem, as seen in 1955

II. THE ESTIMATING OF RAILWAY CAPACITIES

The evaluation of both railway system and individual track capacities is, to a considerable extent, an art. The authors know of no tested mathematical model or formula that includes all of the variations and imponderables that must be weighed.* Even when the individual has been closely associated with the particular territory he is evaluating, the final answer, however accurate, is largely one of judgment and experience.



An important issue for the community

- The “core problem” plagues applications of classical combinatorial algorithms to this day!
- Satisfying their preconditions necessitates converting inputs into an **abstractified** form
- If done manually, this often implies *drastic information loss*
 - Combinatorial problem no longer accurately portrays the dynamics of the real world.
 - Algorithm will give a **perfect** solution, but in a **useless** environment
- The data we need to apply the algorithm may be only **partially** observable
 - This can often render the algorithm completely inapplicable.
- An issue of high interest for *both* combinatorial and operations research communities.



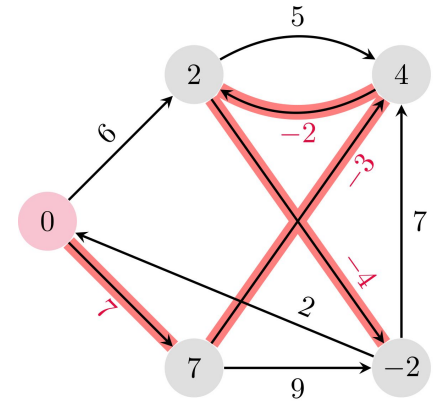
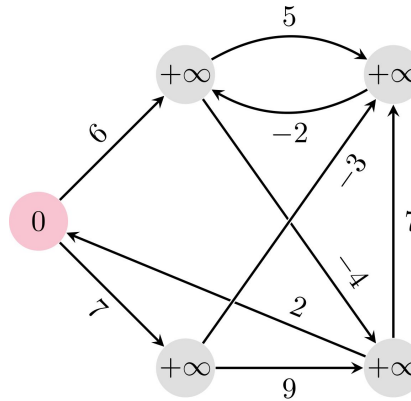
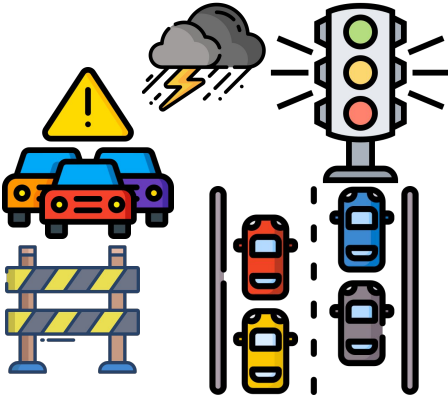
4

Towards a *neurally* spiced solution



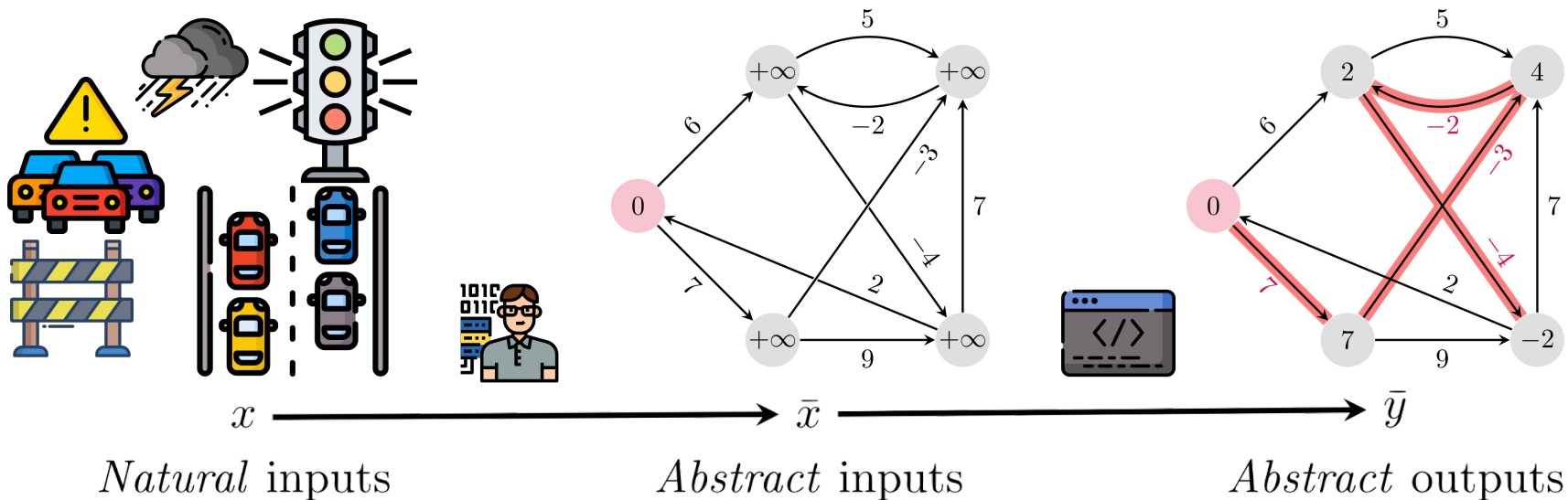
Abstractifying the core problem

- Assume we have *real-world* inputs, but our algorithm only admits *abstract* inputs
 - For now, we assumed **manually** converting from one input to another



Abstractifying the core problem

- Assume we have *real-world* inputs, but our algorithm only admits *abstract* inputs
 - For now, we assumed **manually** converting from one input to another

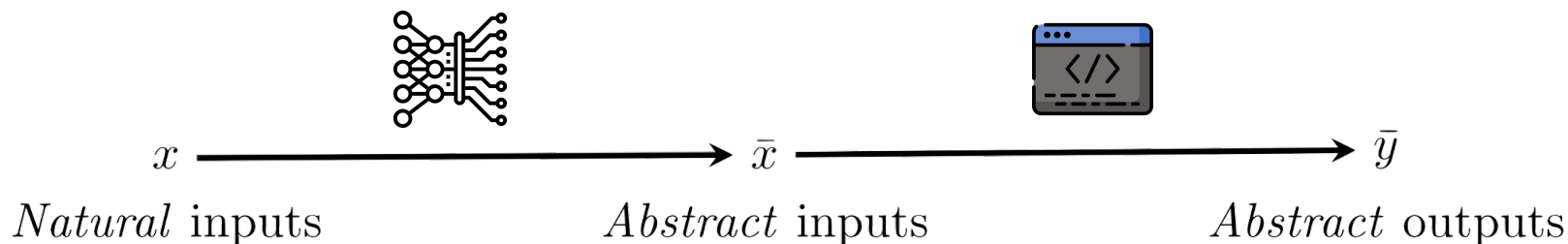


- Whenever we have **manual** feature engineering of **raw** data, **neural nets** are attractive!



Attacking the core problem

- First point of attack: “good old deep learning”
 - Replace human feature extractor with **neural network**
 - Still apply the same combinatorial algorithm



- **First issue:** algorithms typically perform **discrete optimisation**
 - This does not play nicely with gradient-based optimisation that neural nets require.

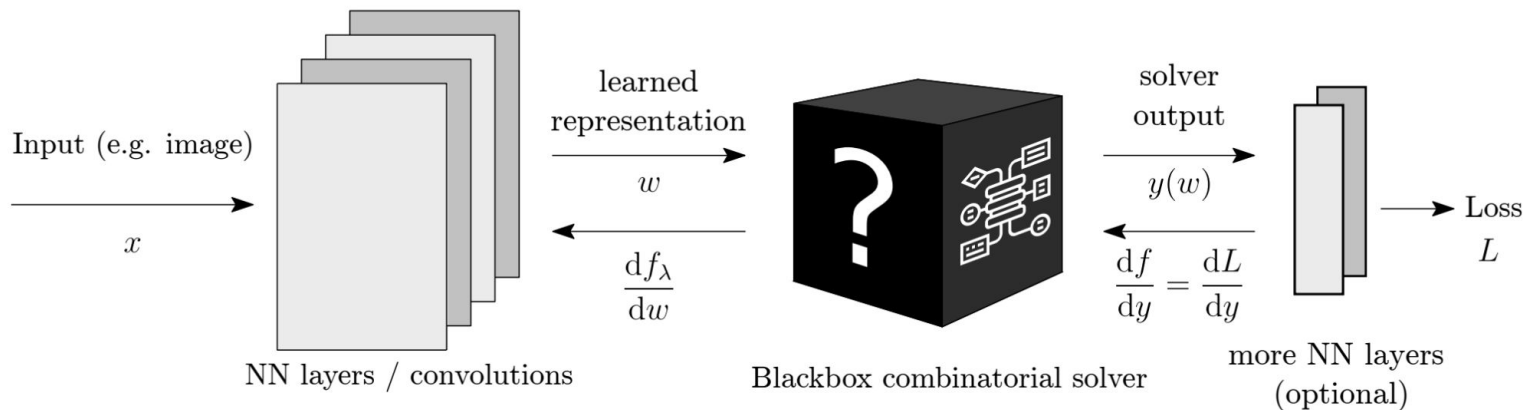


Backpropagating through classical algorithms

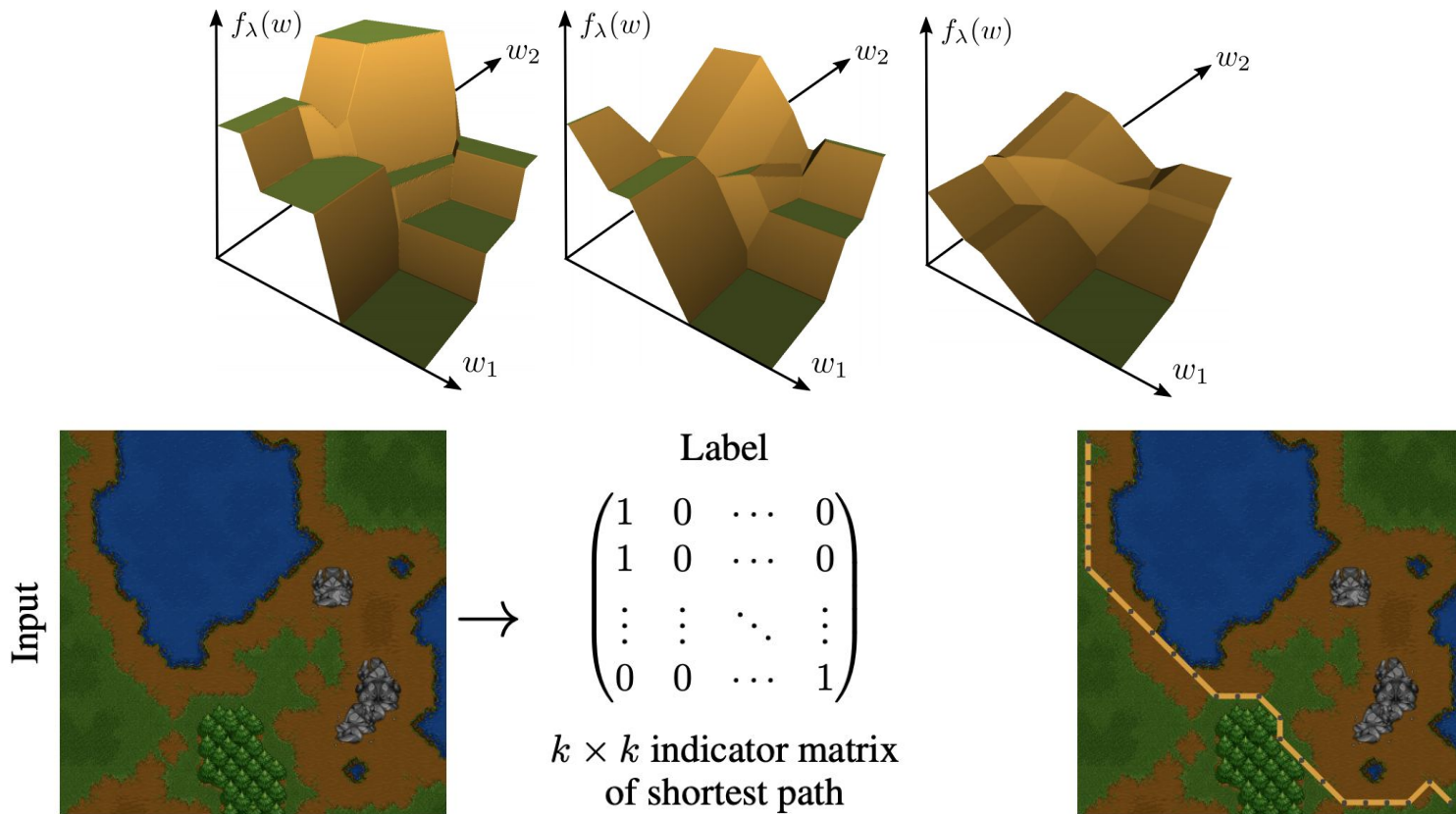
Vlastelica *et al.* (ICLR'20) provide a great approach for differentiating CO solver outputs

DIFFERENTIATION OF BLACKBOX COMBINATORIAL SOLVERS

Marin Vlastelica^{1*}, Anselm Paulus^{1*}, Vít Musil², Georg Martius¹, Michal Rolínek¹



Black-box backprop



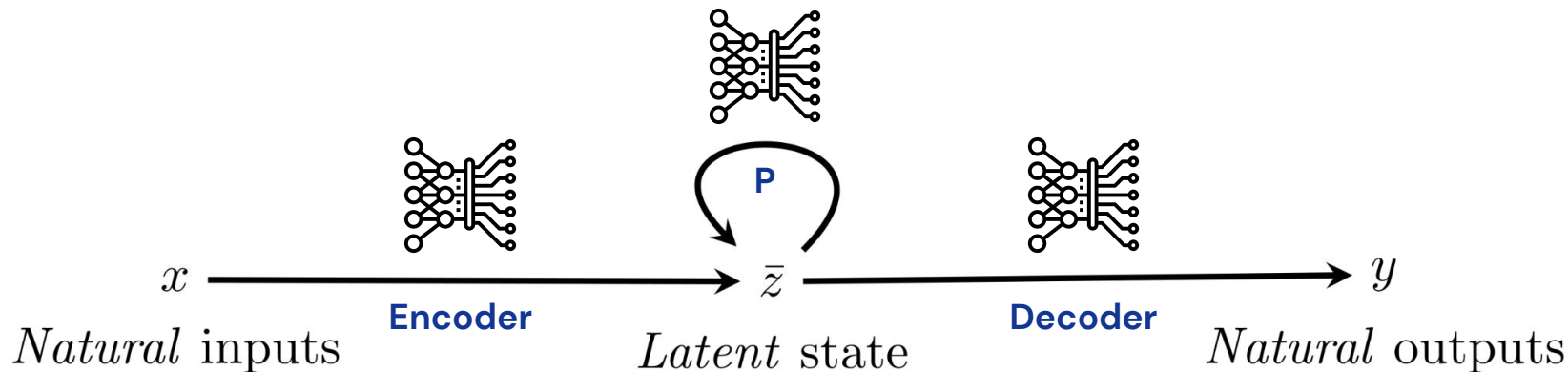
Algorithmic *bottleneck*

- Second (more fundamental) issue: ***data efficiency***
 - Real-world data is often incredibly *rich*
 - We still have to compress it down to **scalar values**
- The algorithmic solver:
 - **Commits** to using this scalar
 - Assumes it is **perfect**!
- If there are insufficient training data to properly estimate the scalars, we hit same issues!
 - Algorithm will give a **perfect** solution, but in a ***suboptimal*** environment



Breaking the bottleneck

- Neural networks derive great flexibility from their **latent** representations
 - They are inherently *high-dimensional*
 - If any component is poorly predicted, others can step in and compensate!
- To *break the bottleneck*, we replace the algorithm with a **neural network**!

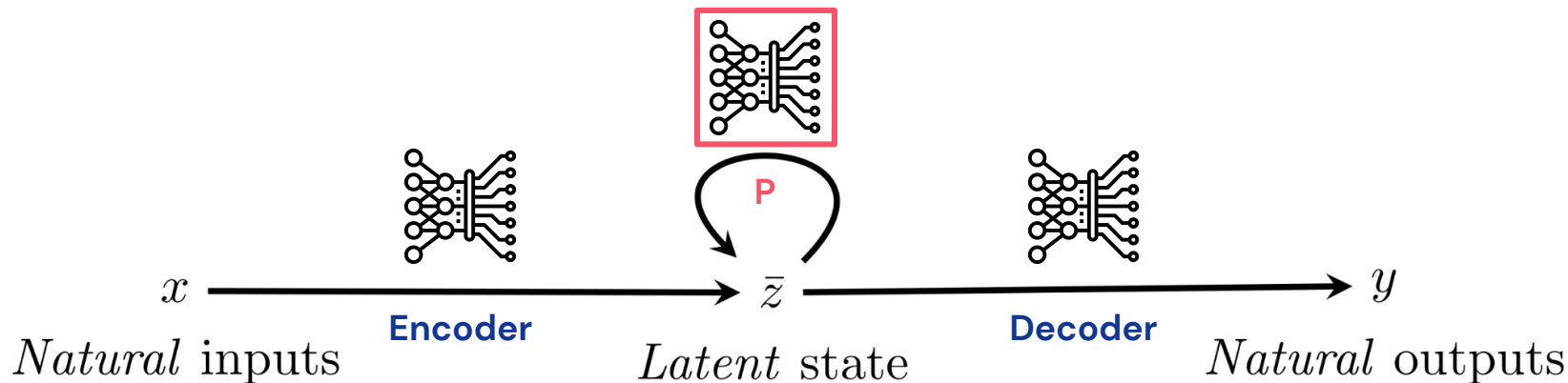


(The setting naturally aligns with *encode-process-decode* (Hamrick et al., CSS'18))



Properties of this construction

- Assuming our **latent**-state NN *aligns* with the steps of an algorithm, we now have:
 - An **end-to-end** neural pipeline which is fully differentiable
 - No scalar-based bottlenecks, hence higher data efficiency.
- How do we obtain **latent-state neural networks** that **align** with algorithms?



DeepMind

5

Algorithmic reasoning



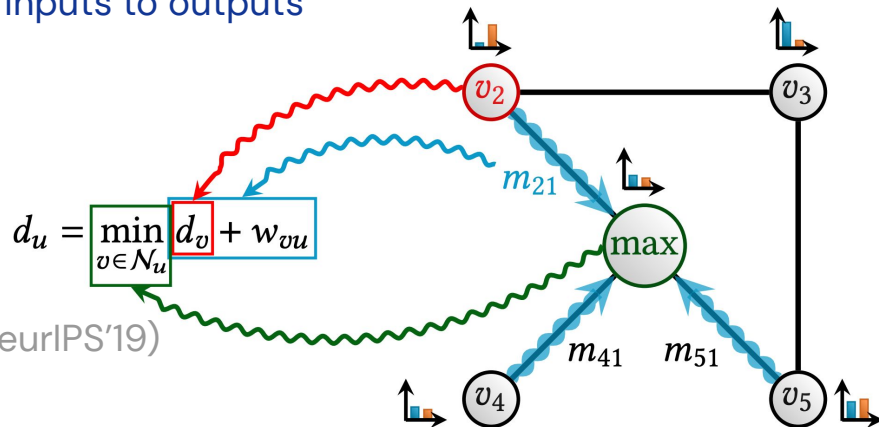
Algorithmic reasoning

- The desiderata for our processor network **P** are slightly different than usual:
 - They are required to imitate the steps of the algorithm *faithfully*
 - This means they must **extrapolate**!
- Neural networks typically **struggle** in the extrapolation regime!
- **Algorithmic reasoning** is an emerging area that seeks to ameliorate this issue
 - Primarily through theoretical and empirical *prescriptions*
 - These guide the neural architectures, inductive biases and featurisations that are useful for extrapolating combinatorially
- This is a **very** active research area, with many key papers published only last year!



tl;dr of algorithmic reasoning

- Graph neural networks (GNNs) align well with **dynamic programming** (Xu et al., ICLR'20)
- Interesting **inductive biases** explored by Veličković et al. (ICLR'20):
 - Encode-**process**-decode from abstract inputs to outputs
 - Favour the **max** aggregation
 - **Strong** supervision on trajectories
- Further interesting work:
 - IterGNNs (Tang et al., NeurIPS'20)
 - Shuffle-exchange nets (Freivalds et al., NeurIPS'19)
 - PGNs (Veličković et al., NeurIPS'20)
- Latest insights: **linear** algorithmic alignment is highly beneficial (Xu et al., ICLR'21)



Further insight

If you would like to know more details about constructing good processor networks:



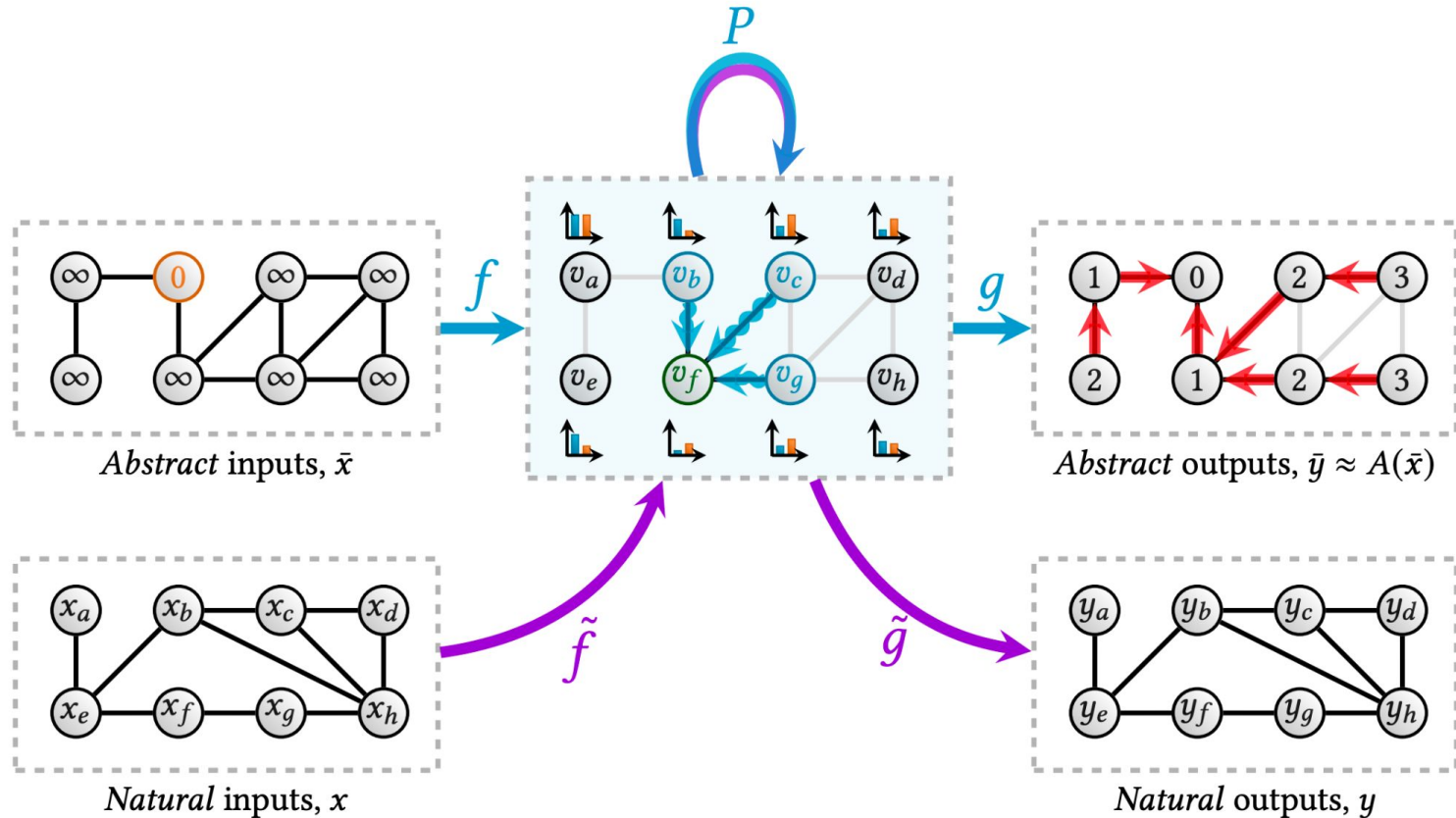
<https://www.youtube.com/watch?v=IPQ6CPoluok>



https://drive.google.com/file/d/1_EQ9Yu7VEkvrHaVHl_WbT5ABvxrSNY-s/view?usp=sharing



Blueprint of algorithmic reasoning

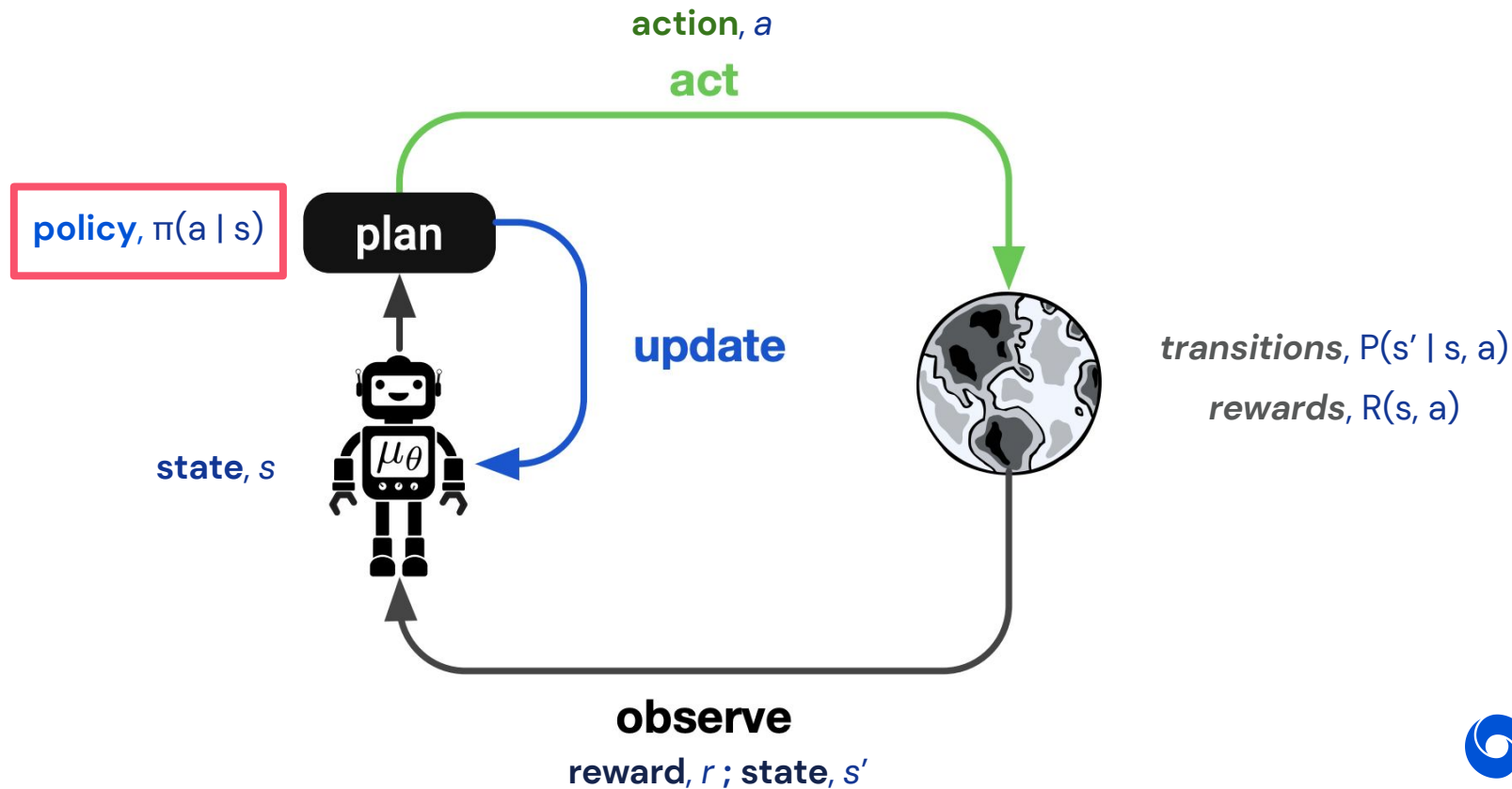


6

An algorithmic *implicit planner*



Reinforcement learning (RL) setting



Intro to value iteration

- Value Iteration: *dynamic programming* algorithm for **perfectly** solving an RL environment

$$v^{(t+1)}(s) = \max_{a \in \mathcal{A}_s} r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) v^{(t)}(s')$$

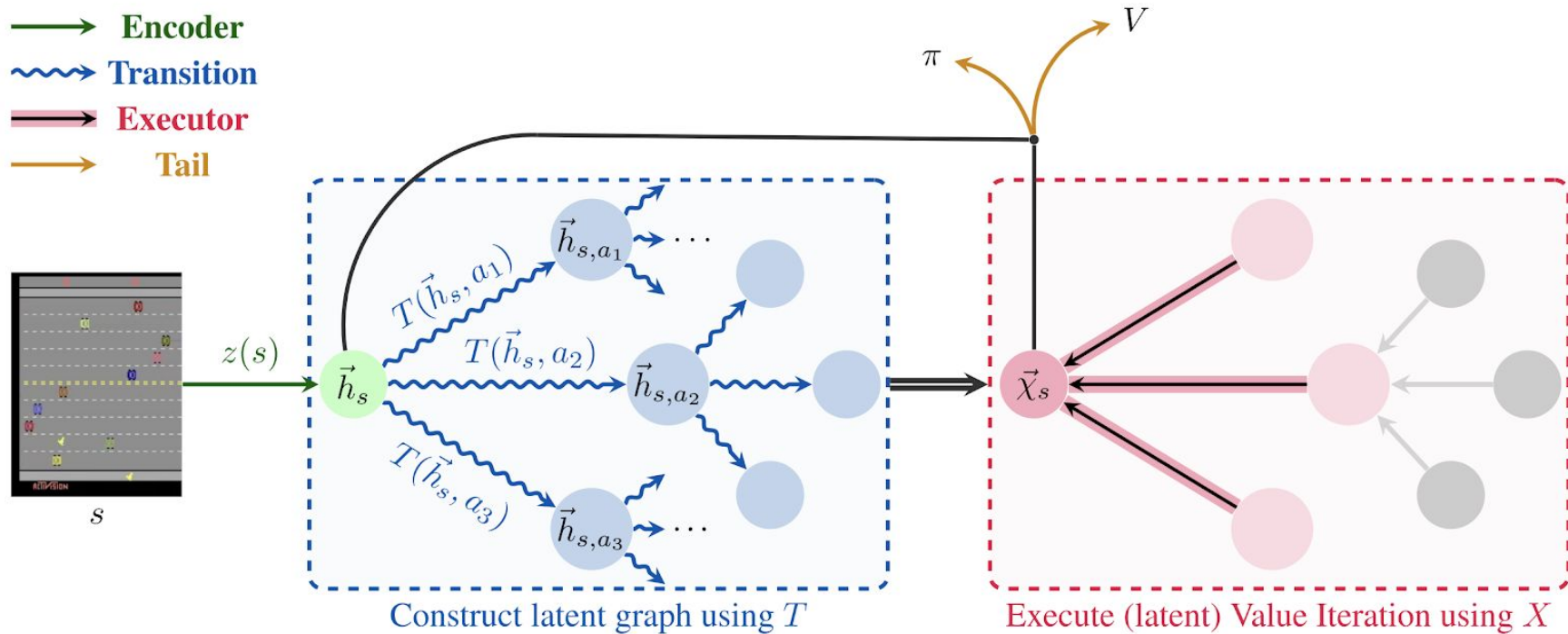
- **Guaranteed** to converge to *optimal* solution (fixed-point of Bellman optimality equation)!

$$V^*(s) = \max_{a \in \mathcal{A}} \left(R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) V^*(s') \right)$$

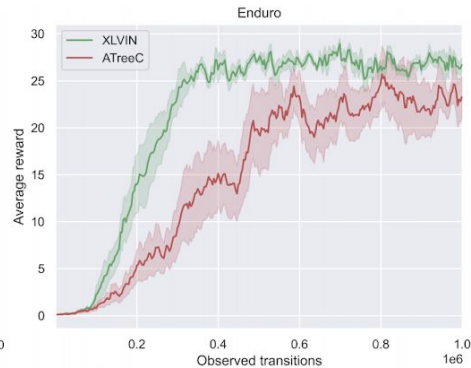
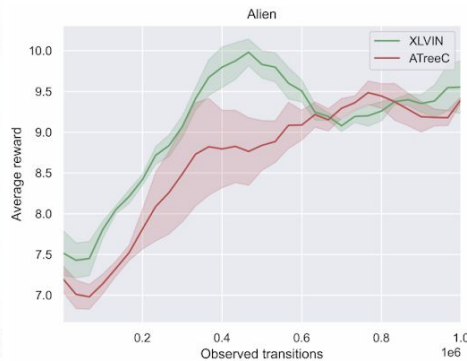
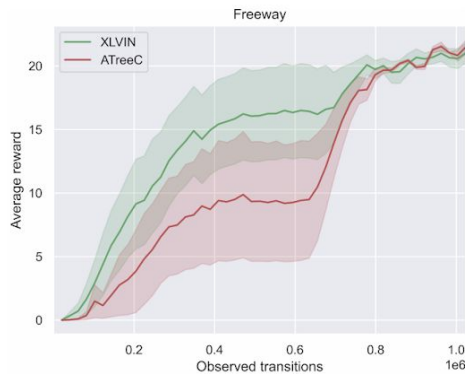
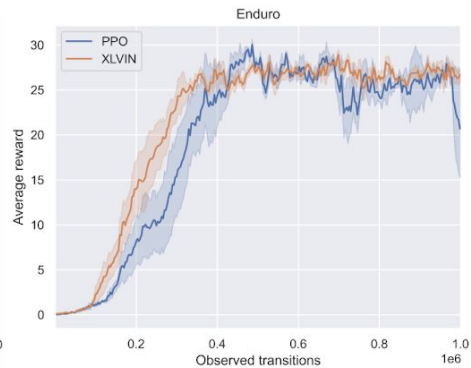
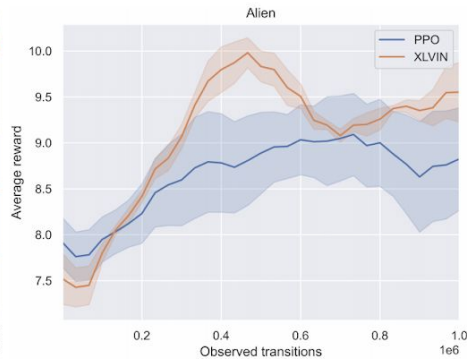
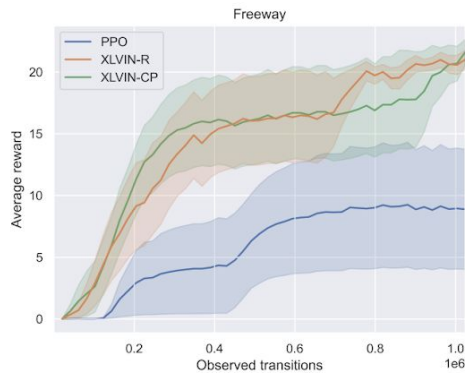
- BUT requires **full knowledge** of underlying MDP (P / R)
 - Prime target for our previously studied blueprint :)



XLVIN (Deac et al., 2020)



Results on *low-data* Atari



Want to know more?

Combinatorial optimization and reasoning with graph neural networks

Quentin Cappart¹, Didier Chételat², Elias Khalil³, Andrea Lodi²,
Christopher Morris², and Petar Veličković^{*4}

¹Department of Computer Engineering and Software Engineering, Polytechnique Montréal

²CERC in Data Science for Real-Time Decision-Making, Polytechnique Montréal

³Department of Mechanical & Industrial Engineering, University of Toronto

⁴DeepMind

Our 43-page survey on GNNs for CO!

<https://arxiv.org/abs/2102.09544>

Section 3.3. details algorithmic reasoning,
with comprehensive references.

Combinatorial optimization is a well-established area in operations research and computer science. Until recently, its methods have focused on solving problem instances in isolation, ignoring the fact that they often stem from related data distributions in practice. However, recent years have seen a surge of interest in using machine learning, especially graph neural networks (GNNs), as a key building block for combinatorial tasks, either as solvers or as helper functions. GNNs are an inductive bias that effectively encodes combinatorial and relational input due to their permutation-invariance and sparsity awareness. This paper presents a conceptual review of recent key advancements in this emerging field, aiming at both the optimization and machine learning researcher.



DeepMind

Thank you!

`petarv@google.com | https://petar-v.com`

In collaboration with Charles Blundell, Raia Hadsell, Rex Ying, Matilde Padovano, Andreea Deac, Ognjen Milinković, Pierre-Luc Bacon, Jian Tang, Mladen Nikolić, Christopher Morris, Quentin Cappart, Elias Khalil, Didier Chéatalat, Andrea Lodi, Lovro Vršek, Mile Šikić, Lars Buesing, Matt Overlan, Razvan Pascanu and Oriol Vinyals

