PDP: A General Neural Framework for Learning Constraint Satisfaction Solvers

Saeed Amizadeh

saamizad@microsoft.com

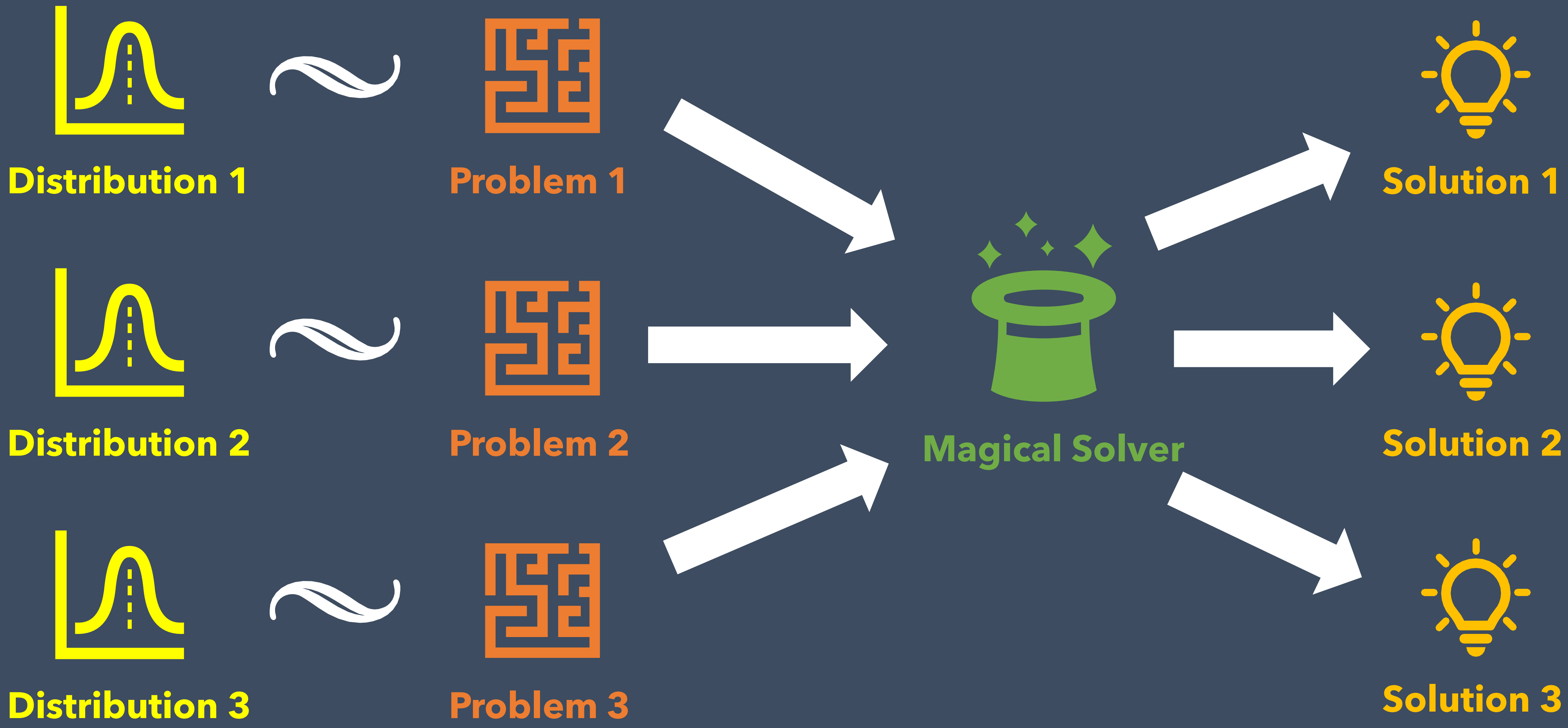
Microsoft

Applied Sciences Lab

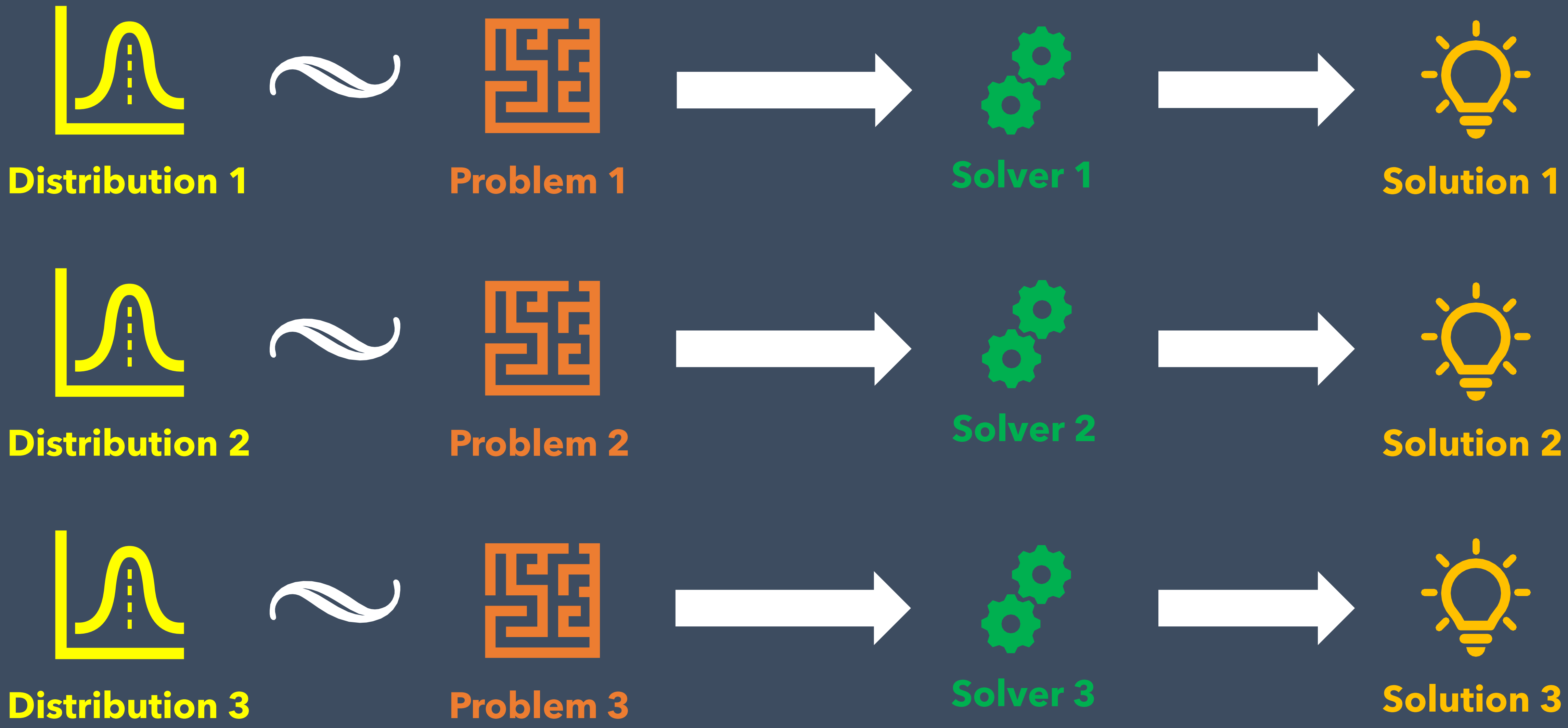
My General Interest

- How **signals** and **symbols** interact in Deep Learning.
- **Neuro-symbolic methods** augment traditional deep learning on continuous input with the power of symbolic calculation.
  - **At the present**: neuro-symbolic visual reasoning, Differentiable First-Order Logic.
  - **In the past**: neural SAT solving.

In the Ideal World...



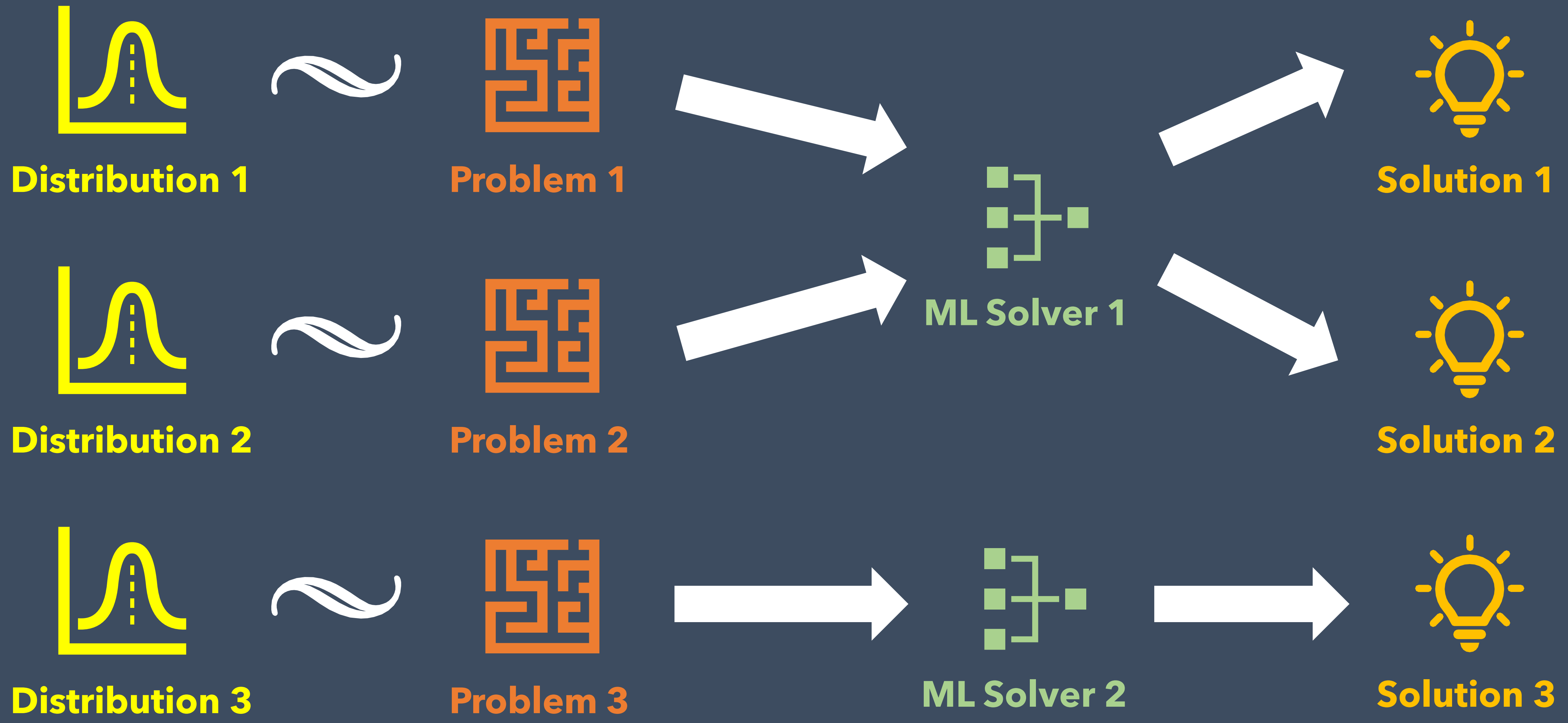
In the Real World…



But What If...

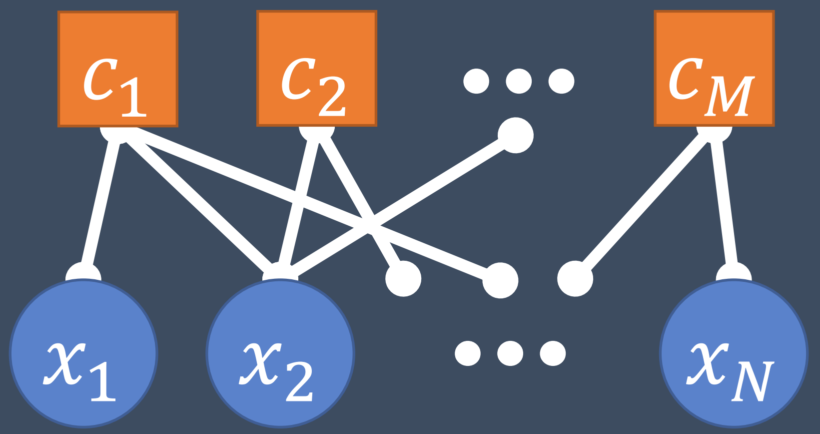
- For a new problem distribution, instead of **manually designing a new solver algorithm** from scratch, we could just **train a new solver model** from data.
- The trained model could further pick up from data the **problem solving strategies** that were overlooked by the human algorithm designers.

The Allure of Machine Learning



CSP & SAT

- A set of **Variables**: $X = \{x_i\}_{i=1}^{N}$
  - $x_i \in \mathcal{X}$, where $\mathcal{X}$ is a discrete set of values (for SAT $\mathcal{X} = \{0,1\}$).
- A set of **Constraints**: $C = \{c_a\}_{a=1}^{M}$
  - $c_a : \mathcal{X}^{|\partial a|} \longrightarrow \{0,1\}$
  - ∂a = a set of variables participating in c_a



Factor Graph Representation (FGR)

The Neuro-Symbolic Approach

Classical ML

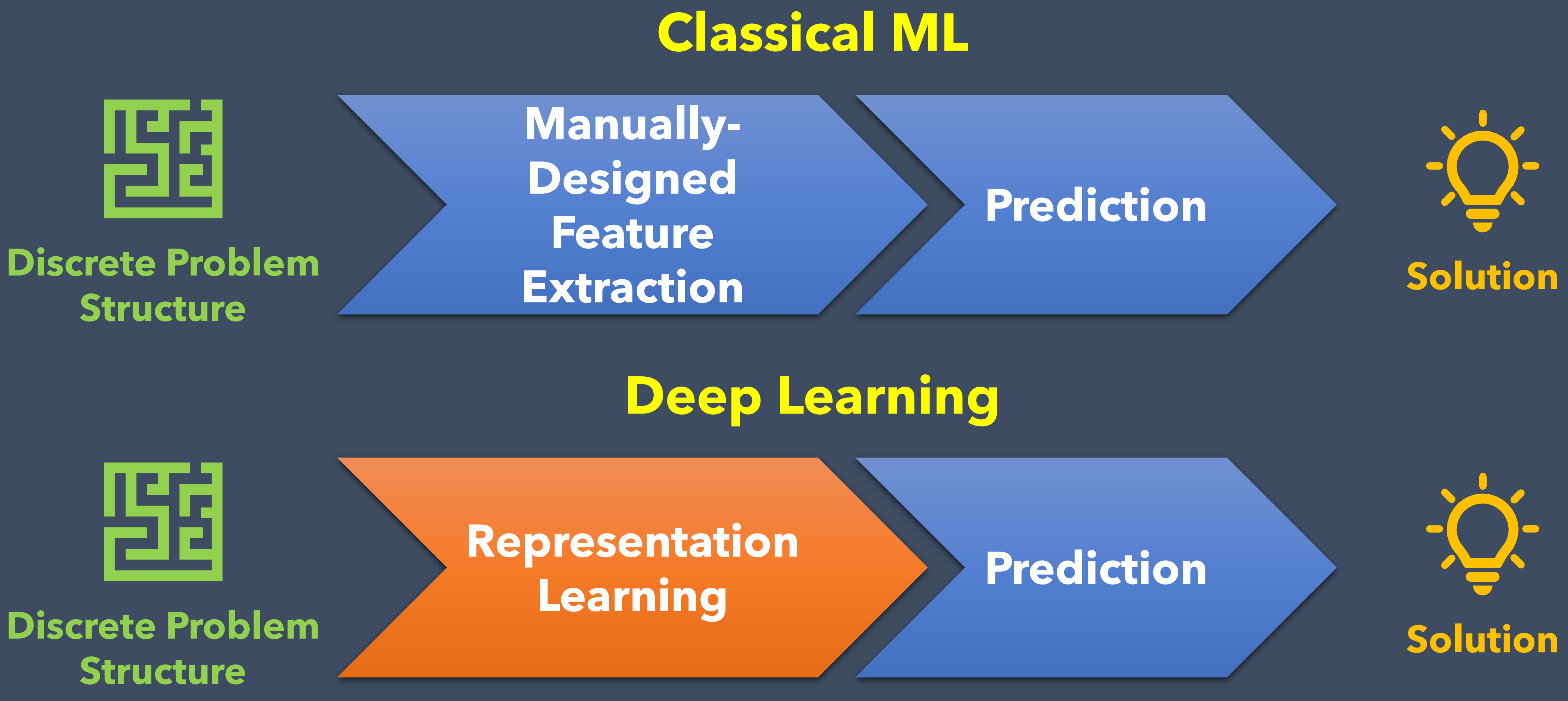
- ✓ Can perform **learning & soft-computing**
- ✗ Does not work with **discrete structures**
- ✗ Not necessarily **scale-invariant**

Classical Solver

- ✗ Can not do **learning & soft-computing**
- ✓ Understands **symbols** & **discrete structures**
- ✓ **Scale-invariant**

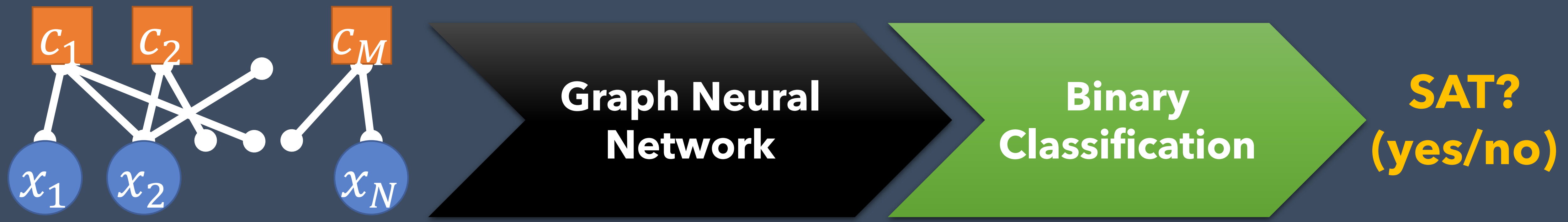
- **Approach A**: Start from a ML model → make it understand discrete CSP structure.
- **Approach B**: Start from a classical solver → make it incorporate ML.

Approach A: How to Represent the **Problem Structure**



Approach A: NeuroSAT [Selsam et al, 2018]

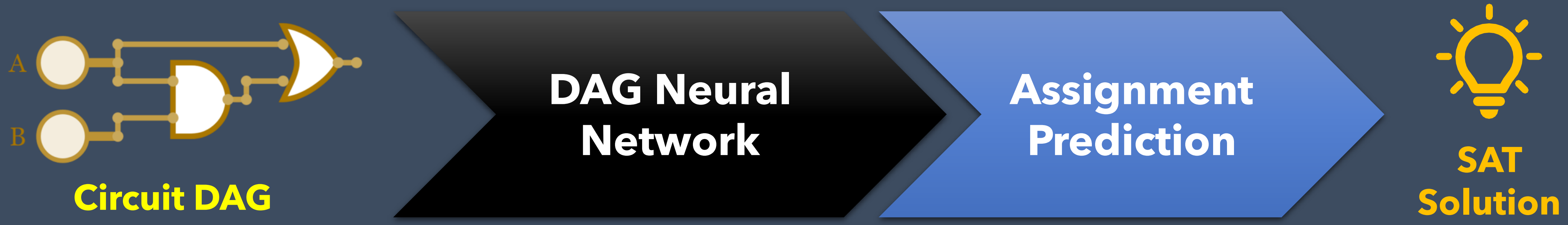
- NeuroSAT learns **structural patterns** in **FGR** that predict **satisfiability**.
- Uses **Graph Neural Networks (GNN)** for representation learning.



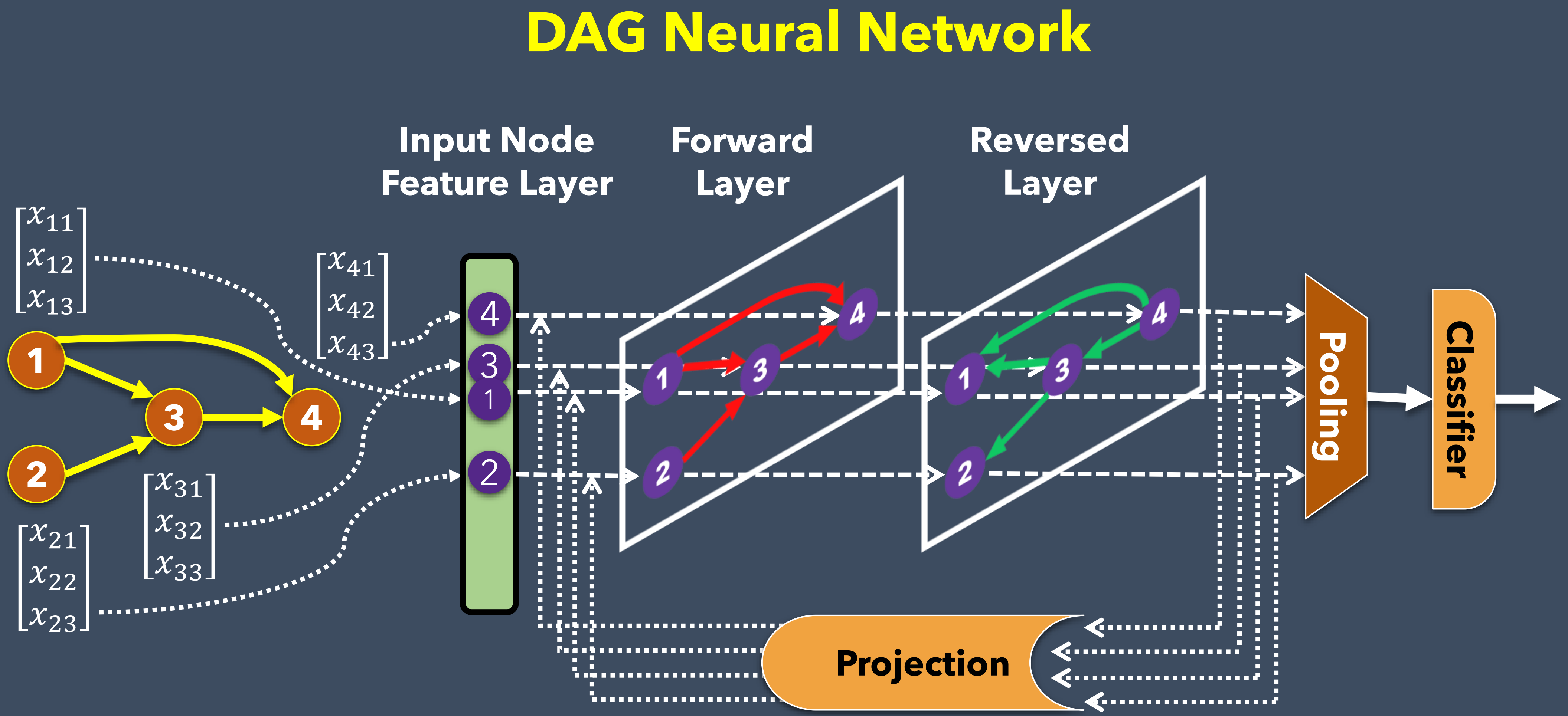
FGR

Approach A: Neural Circuit-SAT [Amizadeh et al, 2018]

- Intuition: a **circuit** has more useful **structural signals** in it compared to the flat CNF.
- Uses **Directed-Acyclic Graph (DAG) Neural Network** for representation learning.
- Trains **directly** toward solving the SAT problem via **Energy Minimization**.



Approach A: Neural Circuit-SAT [Amizadeh et al, 2018]



Approach A: Pros & Cons

- **Pros**:
  - It's a **generic approach** that can theoretically be applied to any CSP/SAT problem distribution.
- **Cons**:
  - It does not employ **useful inductive biases** present in classical solvers.
  - **Generalization** to **larger-scale problems** at the test time is not straightforward.

The Neuro-Symbolic Approach

Classical ML

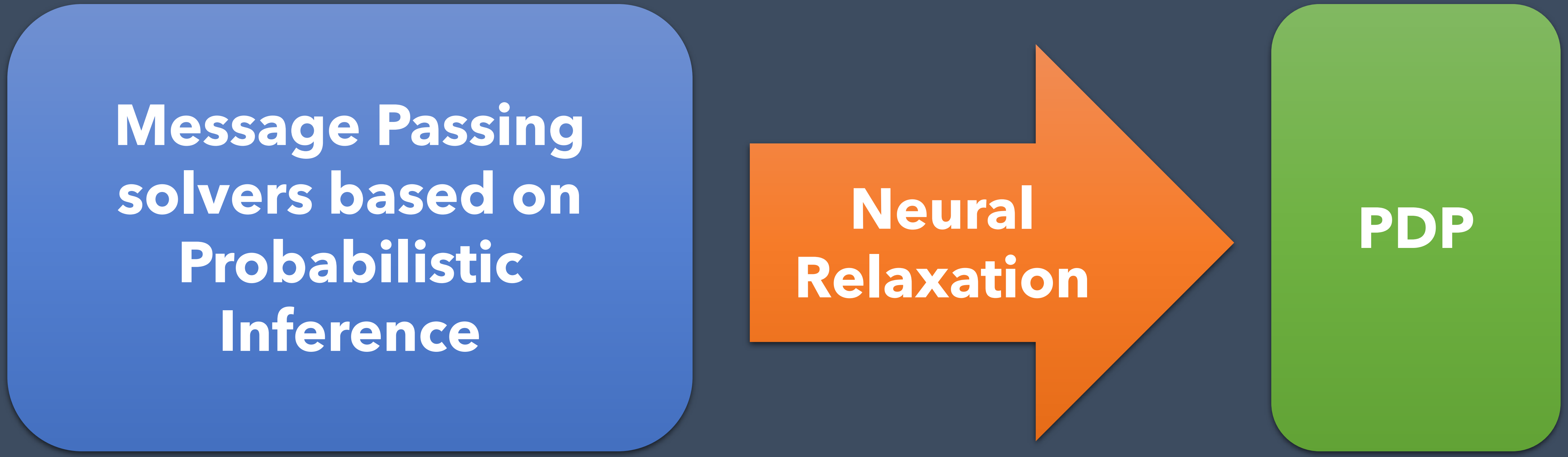
- Can perform **learning & soft-computing**
- Does not work with **discrete structures**
- Not necessarily **scale-invariant**

Classical Solver

- Can not do **learning & soft-computing**
- Understands **symbols** & **discrete structures**
- **Scale-invariant**

- **Approach A**: Start from a ML model → make it understand discrete CSP structure.
- **Approach B**: Start from a classical solver → make it incorporate ML.

PDP Belongs to **Approach B** Group!

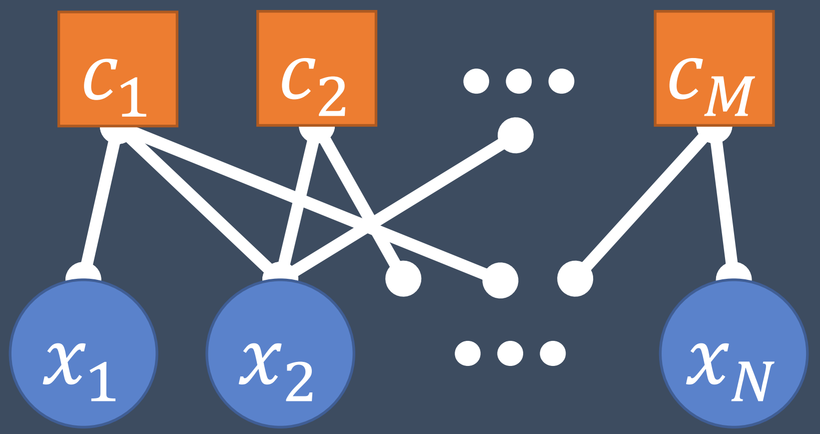


SAT Solving as Probabilistic Inference

- Solving a SAT problem can be formulated as:

$$X^* = \arg\max_X \frac{1}{Z} \cdot \prod_{a=1}^{M} \phi_a(x_{\partial a})$$

$$\phi_a(x_{\partial a}) = \max\left[c_a(x_{\partial a}), \epsilon\right]$$



Factor Graph
Graphical Model

Generalized Message Passing (GMP)

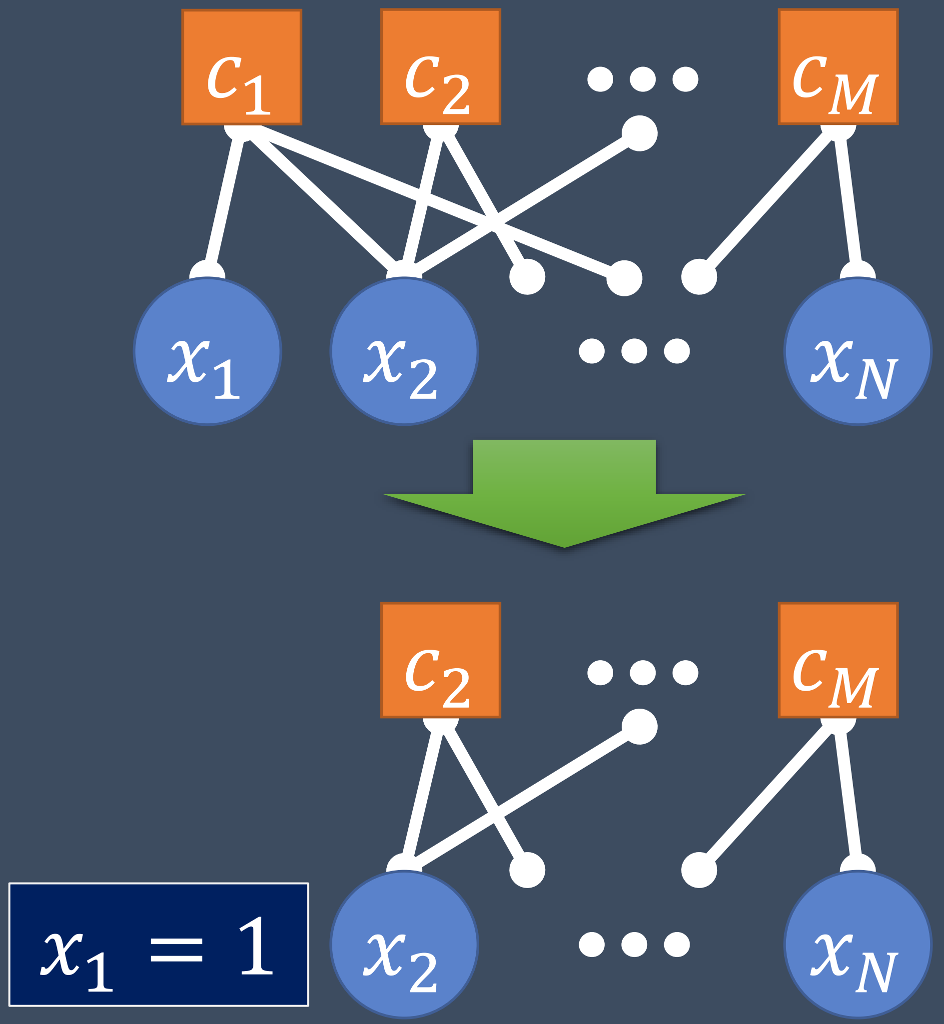
Step 1 – Iterative message passing

$$m_{a\to i}^{(t)} = \mathcal{F}_a\left(\left\{m_{j\to a}^{(t-1)} : j \in \partial a \setminus i\right\}\right) \quad m_{i\to a}^{(t)} = \mathcal{G}_i\left(\left\{m_{b\to i}^{(t-1)} : b \in \partial i \setminus a\right\}\right)$$

Generalized Message Passing (GMP)

Step 2 – Sequential decimation

- Pick the variable with the largest **certainty criterion**.
- Set it to a value according to its **polarity spin**.
- **Simplify** the factor graph.
- Go back to Step 1.



Generalized Message Passing (GMP)

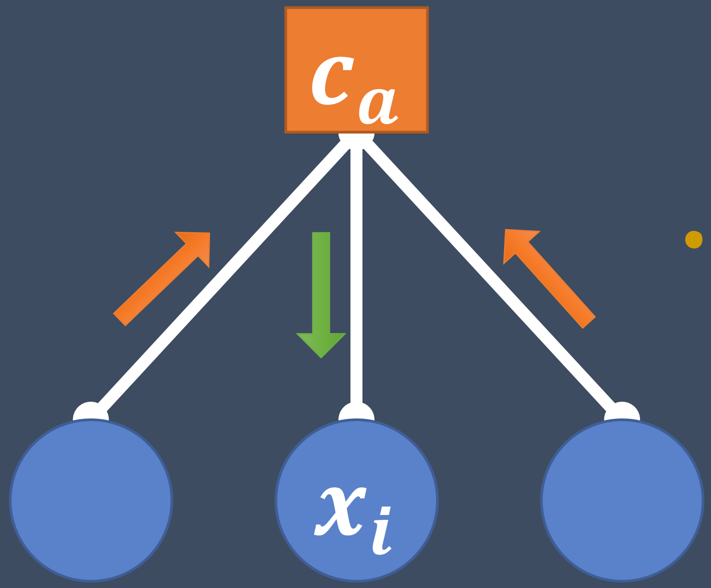
- GMP is a **generic template** for many well-known algorithms characterized by the **choice of $\mathcal{F}$ and $\mathcal{G}$**:
  - Belief Propagation (aka Sum-Product) Algorithm
  - Max-Product Algorithm
  - Min-Sum Algorithm
  - Warning Propagation Algorithm
  - Survey Propagation Algorithm (SP)

Relaxing GMP Toward a Neural Model

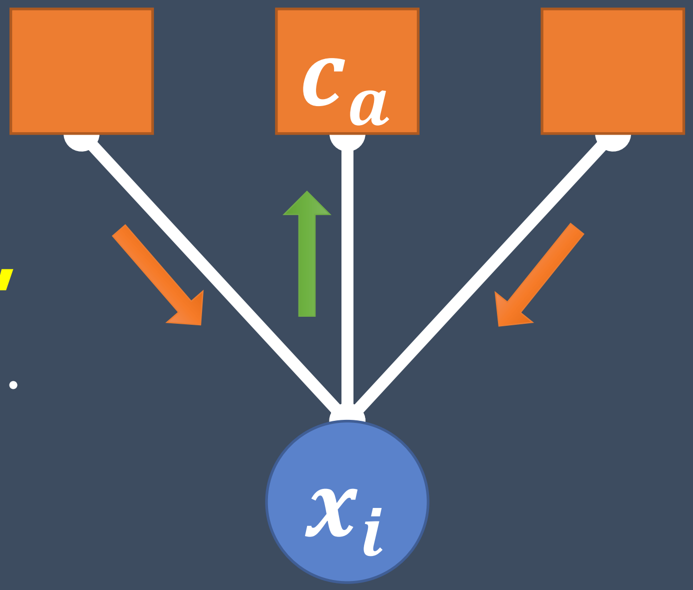
GMP	Its Neural Relaxation
Message = a **scalar** value in $\mathbb{R}$	Message = a **vector** in $\mathbb{R}^d$
Decimation only runs **after** MP is **converged.**	Decimation & MP run **concurrently.**
Sequential decimation affects **only one variable** at a time.	**All variables** are affected during neural decimation.
Sequential decimation = **Fixing** a variable to a value	Neural decimation = Transforming messages in a **stateful manner.**

Propagation Decimation Prediction (PDP)

Step 1 – Propagation



- The propagators are **stateless, feed-forward** neural networks.

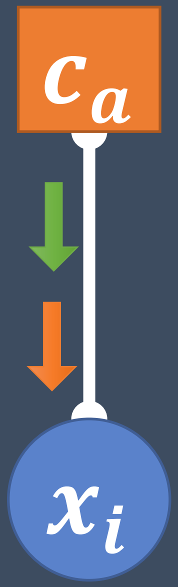


$$p_{a \to i}^{(t)} = \Psi_{\gamma}\left(\left\{d_{j \to a}^{(t-1)} : j \in \partial a \setminus i\right\}\right)$$

$$p_{i \to a}^{(t)} = \Psi_{\theta}\left(\left\{d_{b \to i}^{(t-1)} : b \in \partial i \setminus a\right\}\right)$$

Propagation Decimation Prediction (PDP)

Step 2 – Decimation



- The decimators are **stateful, recurrent** neural networks.

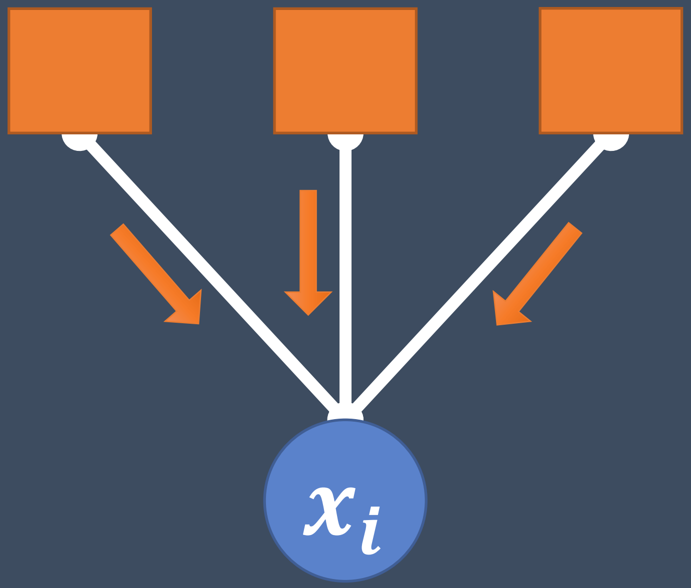
$$d_{a\to i}^{(t)} = \Phi_{\omega}\left(p_{a\to i}^{(t)}, d_{a\to i}^{(t-1)}\right)$$

$$d_{i\to a}^{(t)} = \Phi_{\nu}\left(p_{i\to a}^{(t)}, d_{i\to a}^{(t-1)}\right)$$

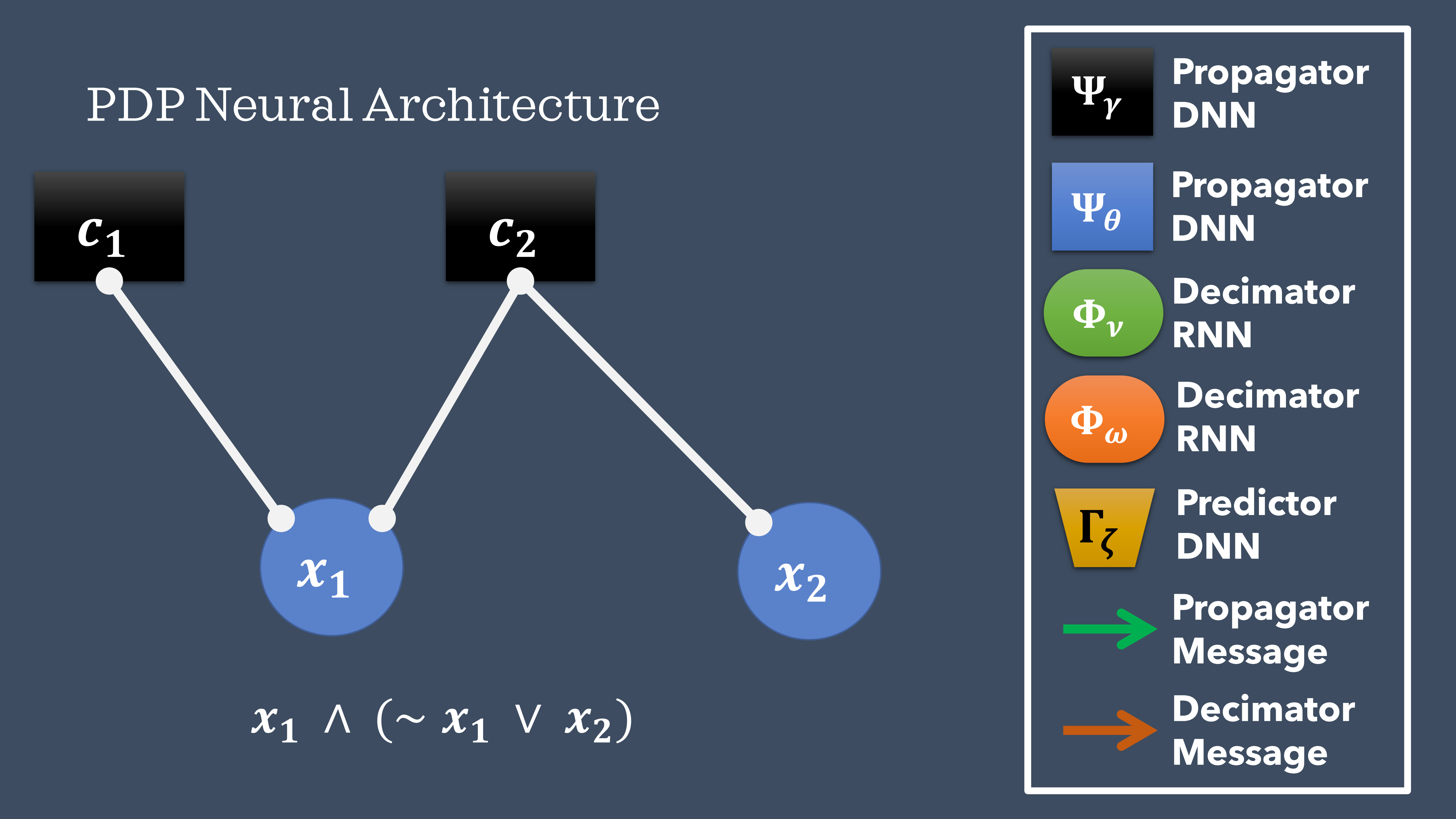
Propagation Decimation Prediction (PDP)

Step 3 – Prediction

- At each time t, the Prediction Step predicts a **soft assignment** for each variable in **[0, 1]**.

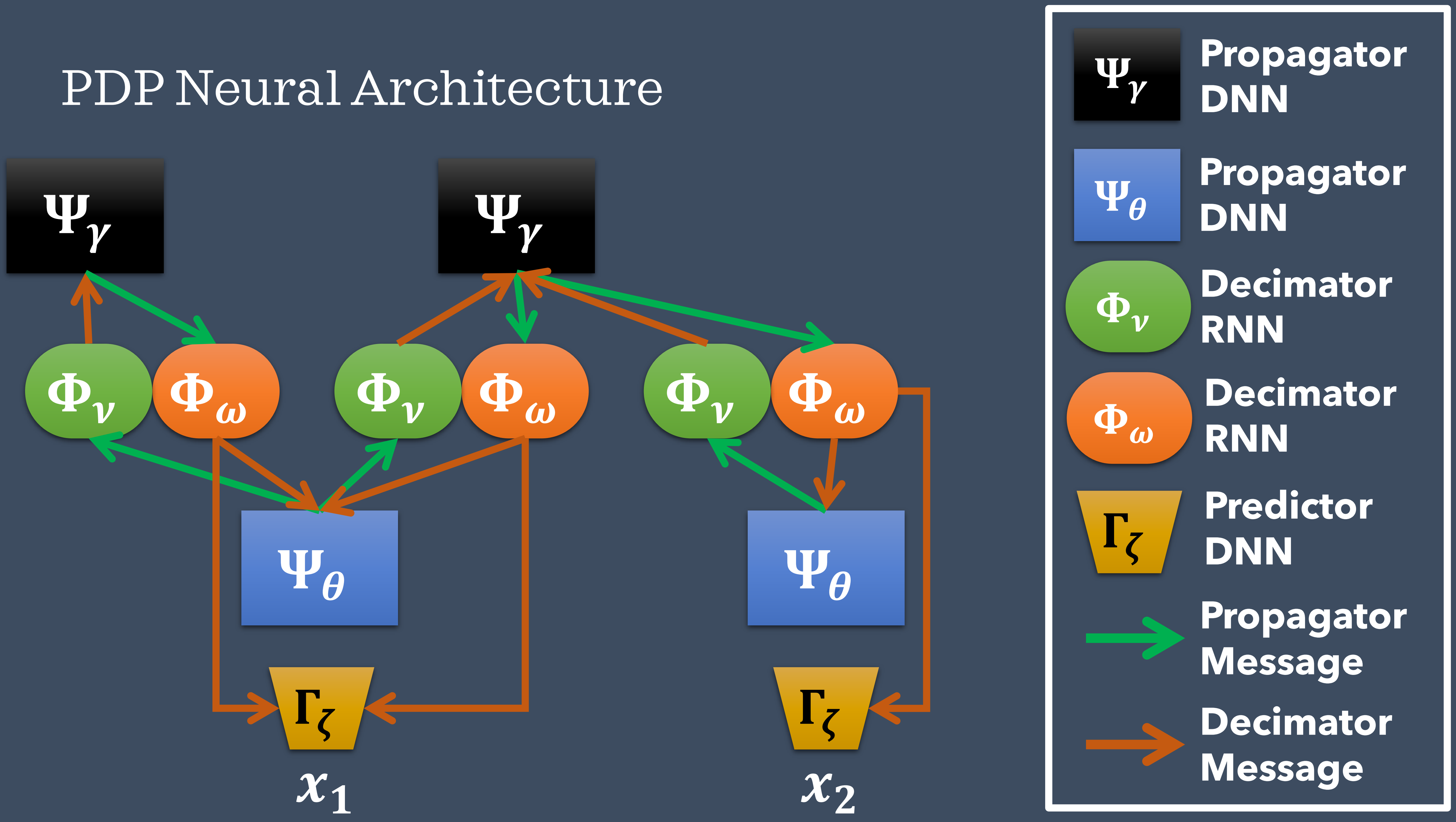


$$x_i^{(t)} = \Gamma_\zeta\left(\left\{d_{b\to i}^{(t)} : b \in \partial i\right\}\right)$$



PDP Neural Architecture
c_1
c_2
x_1
x_2
$x_1 \wedge (\sim x_1 \vee x_2)$
Ψ_γ
Propagator DNN
Ψ_θ
Propagator DNN
Φ_ν
Decimator RNN
Φ_ω
Decimator RNN
Γ_ζ
Predictor DNN
Propagator Message
Decimator Message

PDP Neural Architecture



PDP: Unsupervised Training

- PDP is trained in **unsupervised fashion** via **Discounted Accumulated Energy Minimization**.

$$\mathcal{E}(X) = \log Z - \sum_{a=1}^{M} \log \widetilde{\phi}(x_{\partial a})$$

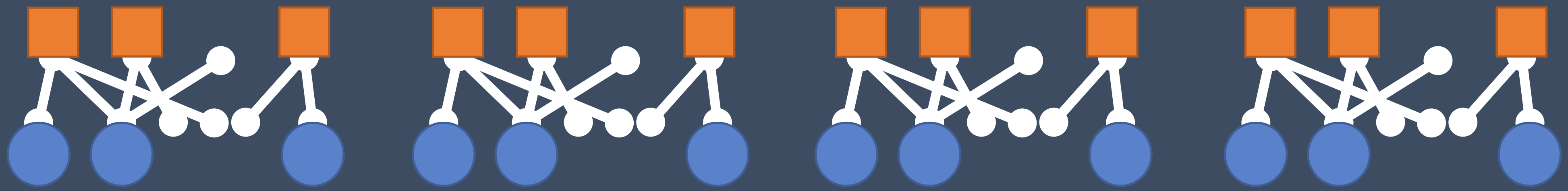
Differentiable surrogate for the constraint potential

$$\mathcal{L}_\lambda\left(X^{(t)}\right) = \sum_{t=1}^{T_{max}} \lambda^{(T_{max}-t)} \cdot \mathcal{E}\left(X^{(t)}\right)$$

Encourages the model to find the solution faster.

Parallelization & Batch Replication

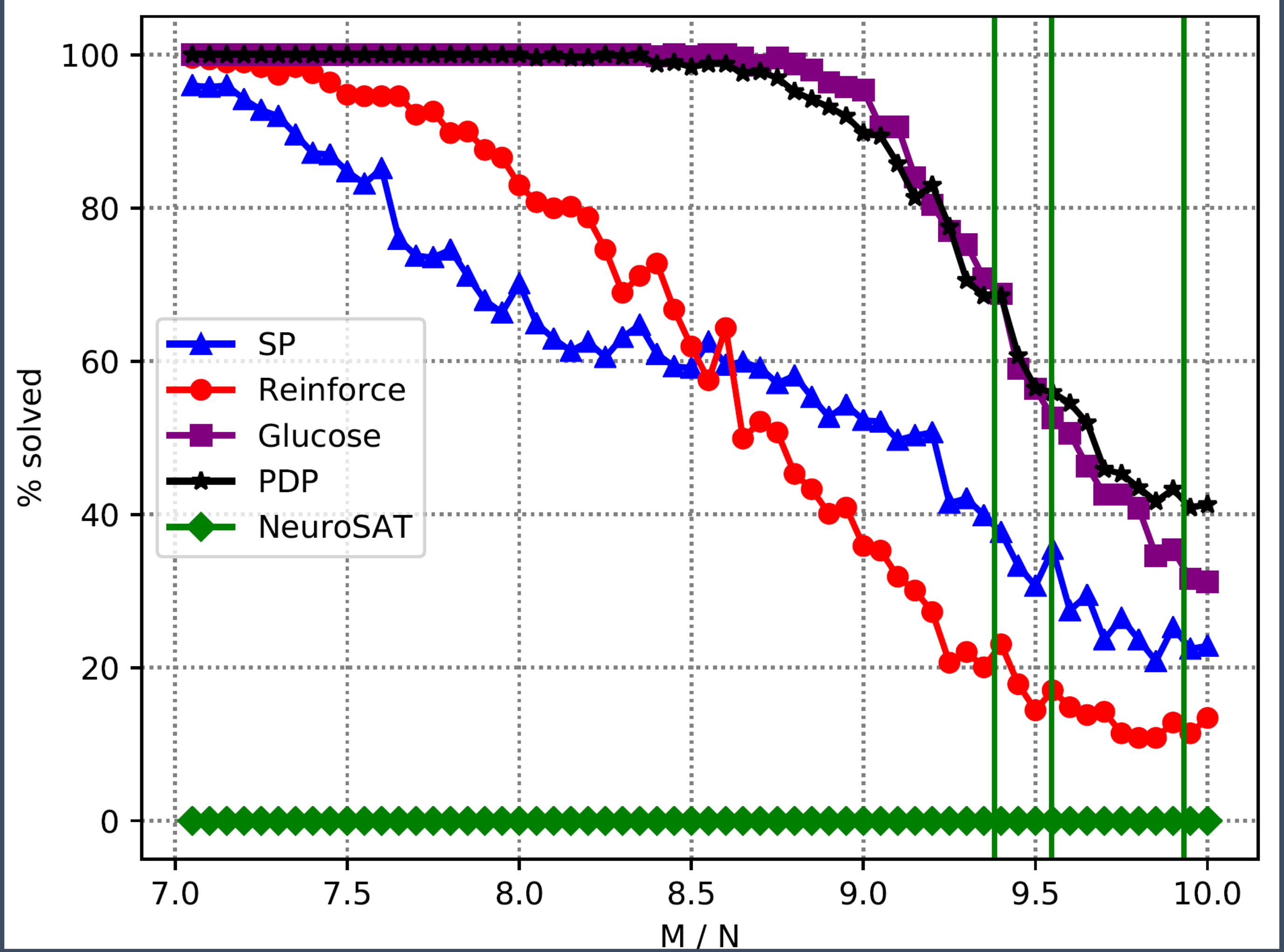
- **Parallelization**: we can run PDP on multiple problem instances **in parallel** by **concatenating** their factor graphs into a big one.



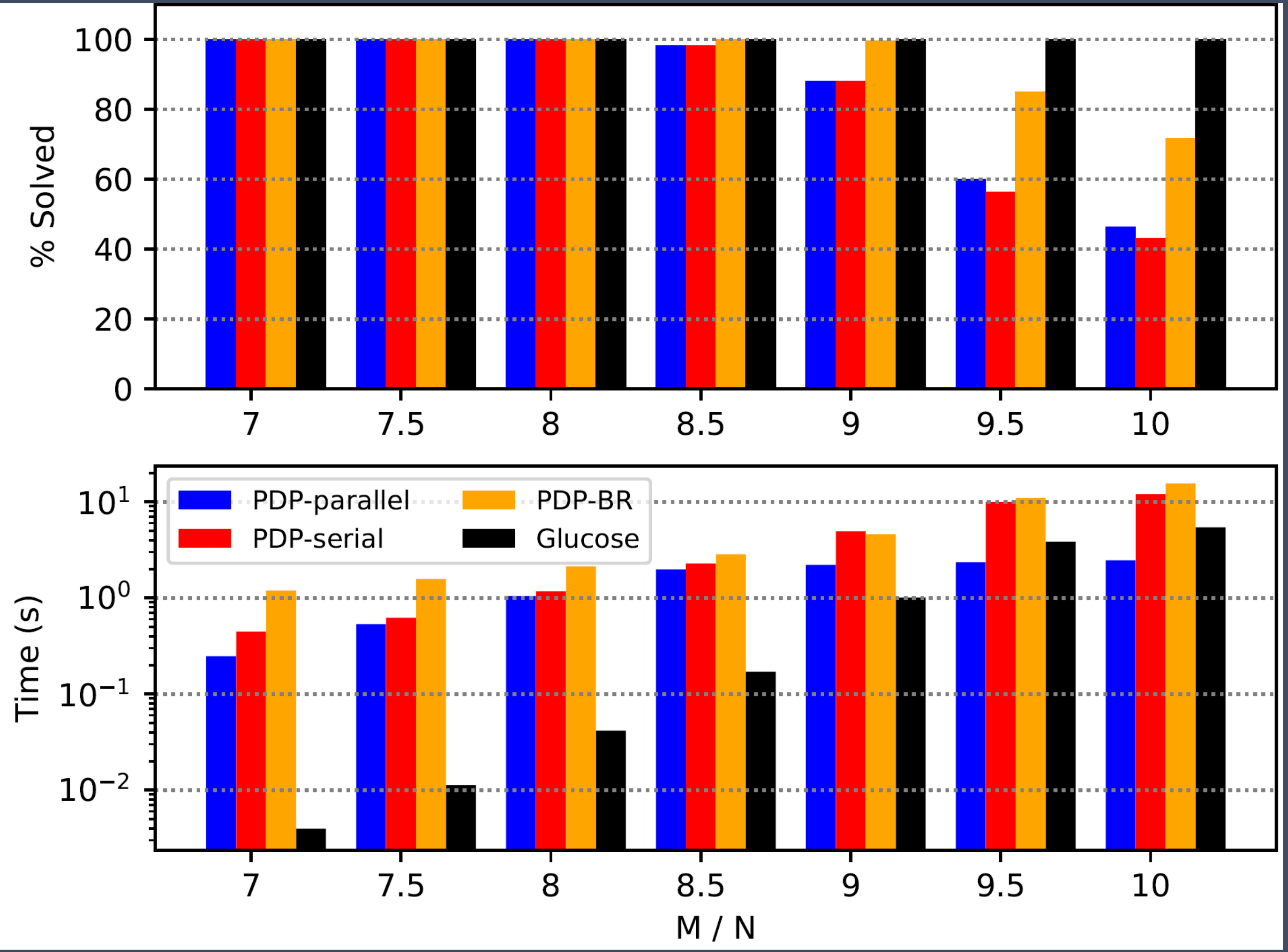
- **Batch Replication**: we can **replicate** the **same problem** multiple times in a batch s.t. each replica starts with a **different initial message values**, so that we can find a solution faster.

Experimental Results: Uniform Random k-SAT

- Generated 500 random 4-SAT problems with 100 variables for each $^{M}/_{N}$ **ratio**.
- Set $T_{max} = 1000$ for PDP-based methods which translates to **3s timeout** threshold for Glucose.



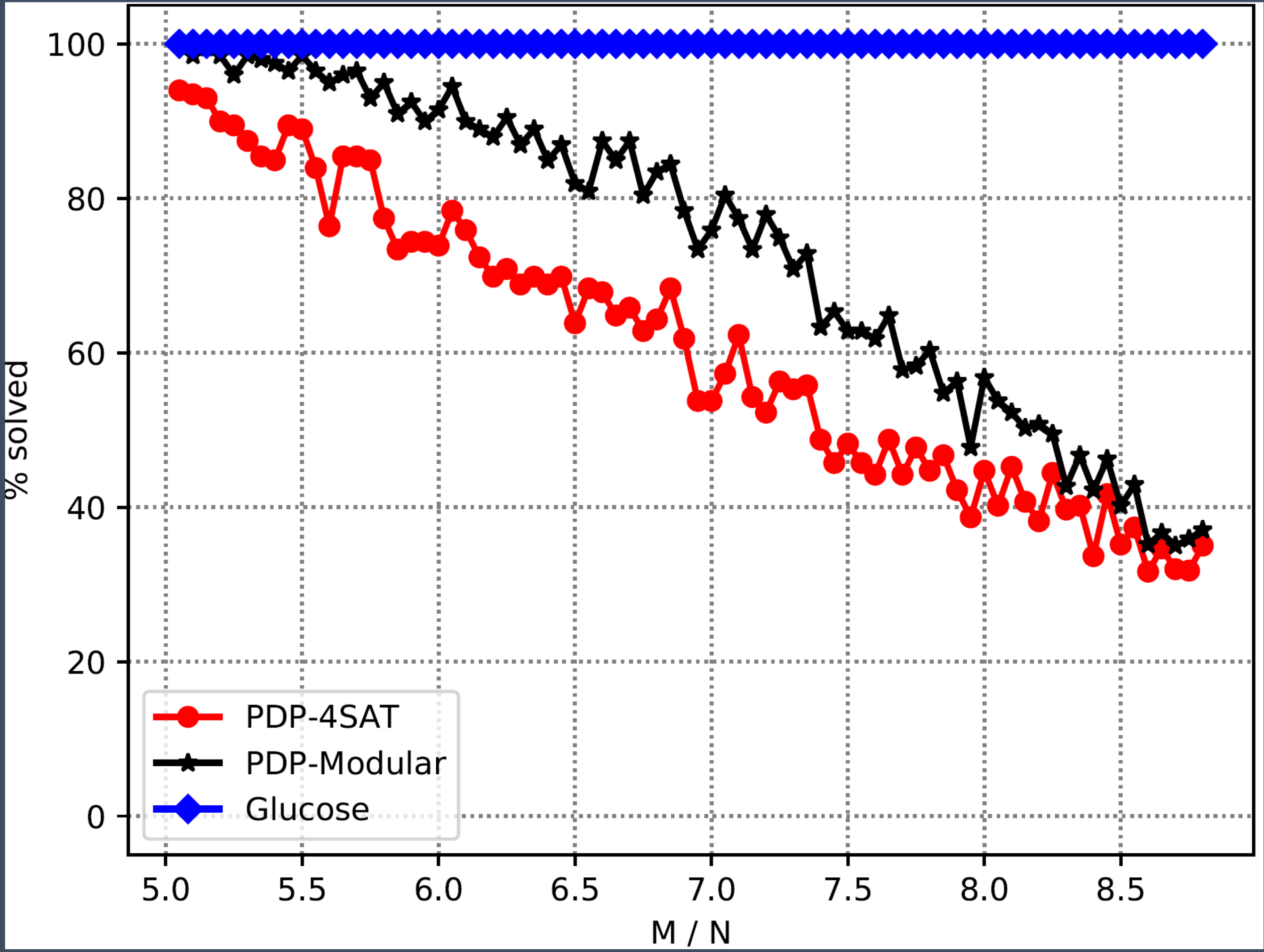
Experimental Results: Uniform Random k-SAT



- Eliminated Glucose's timeout.
- Compared it against:
  - PDP Parallel
  - PDP Serial
  - PDP Serial + Batch Replication
- Glucose wins but **Batch Replication** significantly improves serial PDP.

Experimental Results: Pseudo-Industrial Random k-SAT

- Many industrial SAT problems have **modular structure**.
- Used **Community Attachment** [Giraldez-Cru & Levy, 2016] model to generate modular SAT.
- PDP is capable of **adapting** to a new problem distribution.



Insights & Takeaways

Obviously, we are still **far away** from performing on par with **industrial solvers**, but…

- The ML approach to SAT provides us with **generic solution frameworks** that can **adapt** to **new problem distributions**.
- Approach B is **superior** to Approach A, because it enables us to encode informative **inductive biases** into the model.
- **Neural Relaxation** is a powerful methodology to arrive at Approach B frameworks.
- PDP serves as a **generic template** capable of realizing fully-neural as well as **hybrid models**.
- PDP is **highly parallel** and further enables us to implement classical **restart** via **batch replication**.

Important Directions Ahead

- Approaching other aspects of SAT via the ML approach, e.g. providing **proof of UNSAT**.
- Incorporating other powerful classical techniques such as **backtracking** into the neural framework.
- **Ideally, we want a generic neural framework with a right balance between ML components and powerful classical techniques that is end-to-end differentiable/trainable.**

Thank You!

- My co-authors:

Sergiy Matusevych, **Microsoft**

Markus Weimer, **Microsoft**

- The paper:
  - https://arxiv.org/abs/1903.01969
- The open-source code:
  - https://github.com/microsoft/PDP-Solver