Australian National University

Learning Domain-Independent Heuristics over Hypergraphs

William Shen, Felipe Trevizan, Sylvie Thiébaux

The Australian National University

Learn domain-independent heuristics

- **Learn entirely from scratch**
  - Do not use hand-crafted features
    - e.g. Learning Generalized Reactive Policies using Deep Neural Networks [Groshev et al. 2018]
  - Do not rely on existing heuristics as input features
    - e.g. Action Schema Networks: Generalised Policies with Deep Learning [Toyer et. al 2017]
  - Do not learn an improvement for an existing heuristic
    - e.g. Learning heuristic functions from relaxed plans [Yoon et al. 2006]

Learn domain-independent heuristics

- **Generalise to:**
  - different initial states, goals
  - different number of objects
  - **different domains**
    - domains **unseen** during training

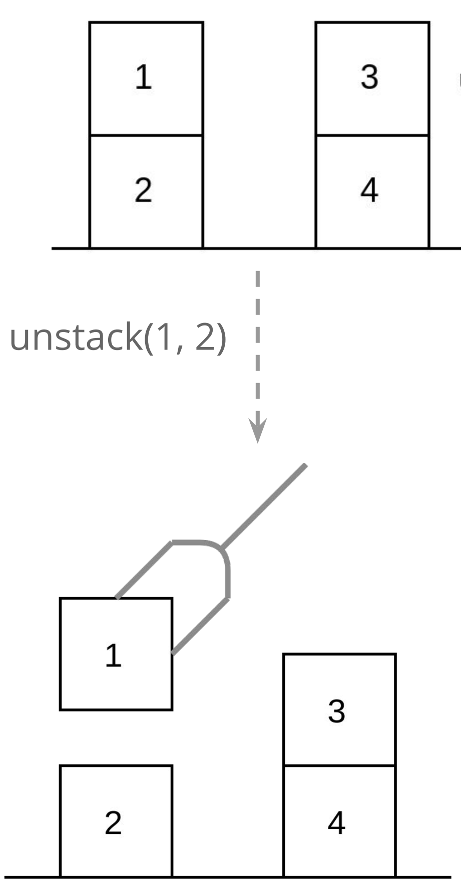
domain-independent!

STRIPS

$$P = \langle F, A, I, G, c \rangle$$

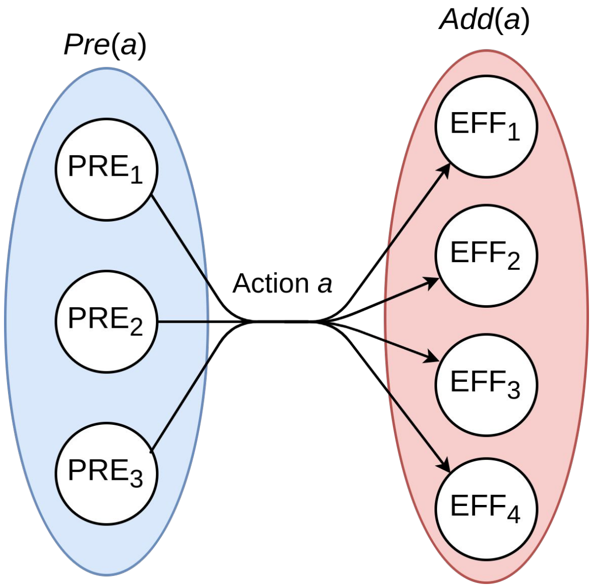
- F is the set of propositions
- A is the set of actions
  - Each action has preconditions, add-effects & delete-effects
- $I \subseteq F$ is the initial state
- $G \subseteq F$ is the goal states
- c is the cost function

unstack(1, 2)
PRE: on(1, 2), clear(1) ...
EFF: holding(1) clear(2)
¬on(1, 2) ...



Hypergraph for the delete relaxation

- **Hyperedge:** edge that joins any number of vertices

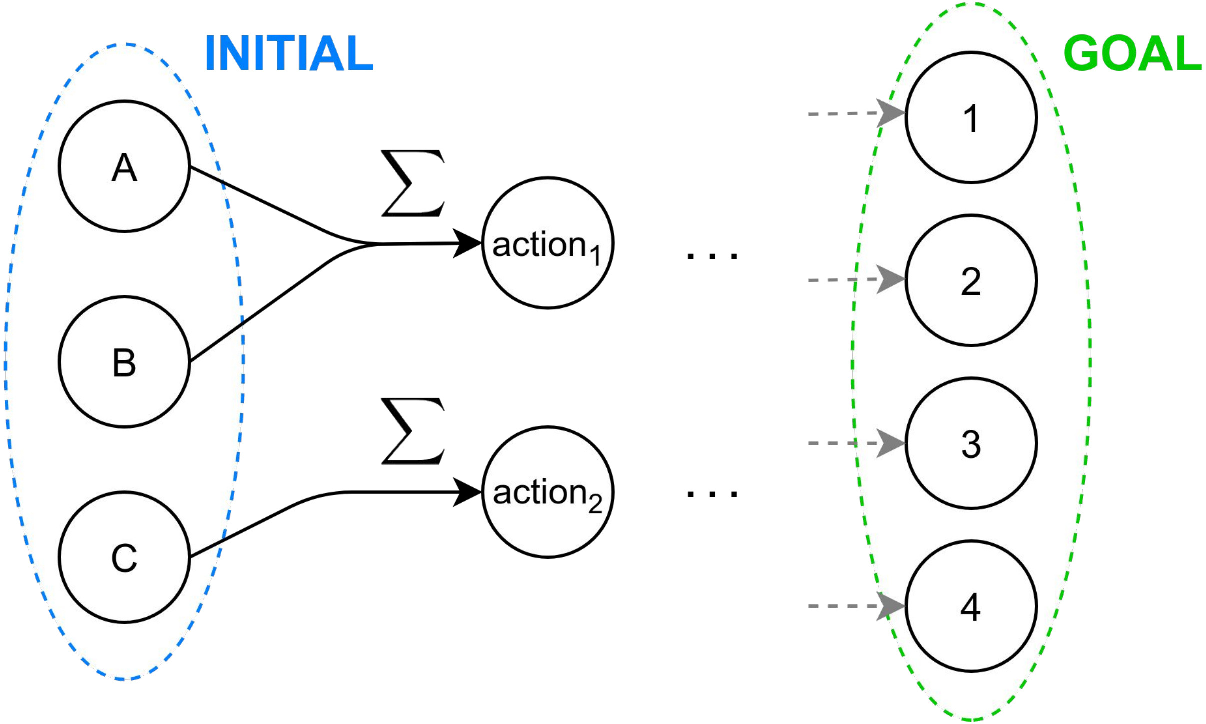


The delete-relaxation P^+ of problem P can be represented by a **hypergraph**

Delete-Relaxation: ignore delete effects for each action

h^{add} heuristic

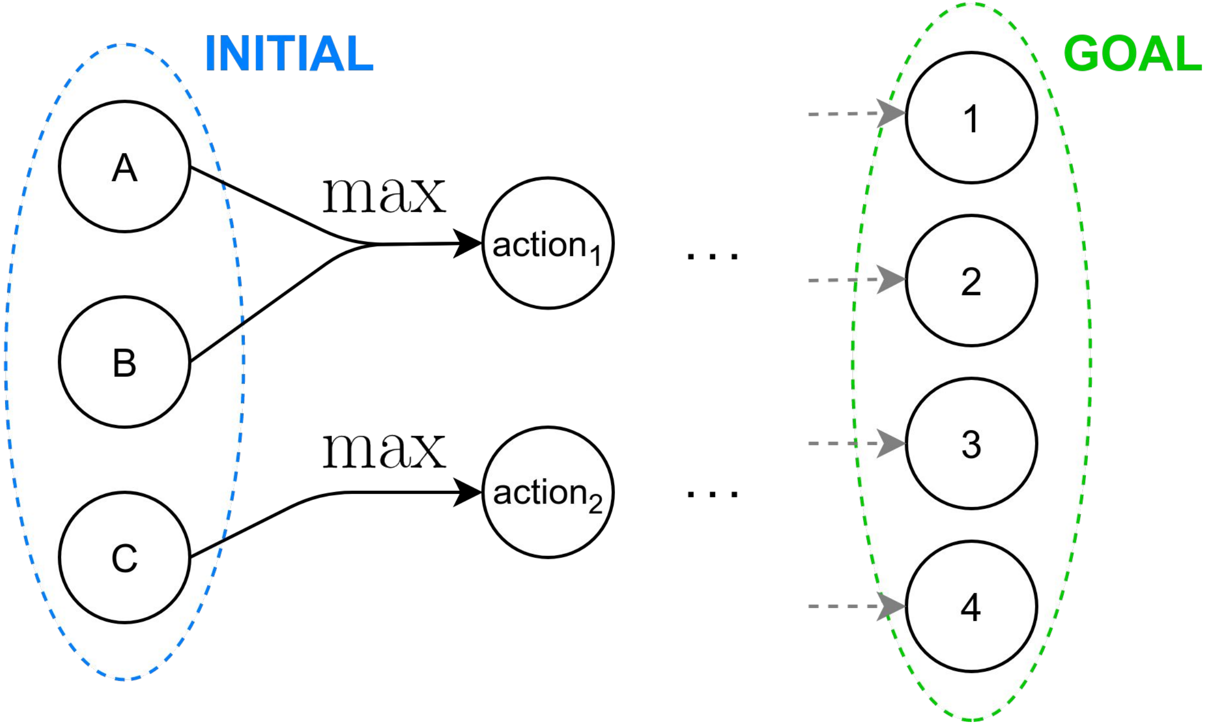
$$h^{\text{add}}(s) = \sum_{g \in G} \underbrace{h^{\text{add}}(s; g)}_{\text{cost of achieving } g}$$



- Estimate cost of goal as sum of costs of each proposition
- Assumes achieving each proposition is independent
  - Overcounting
  - **Non-admissible!**

$h^{\max}$ heuristic

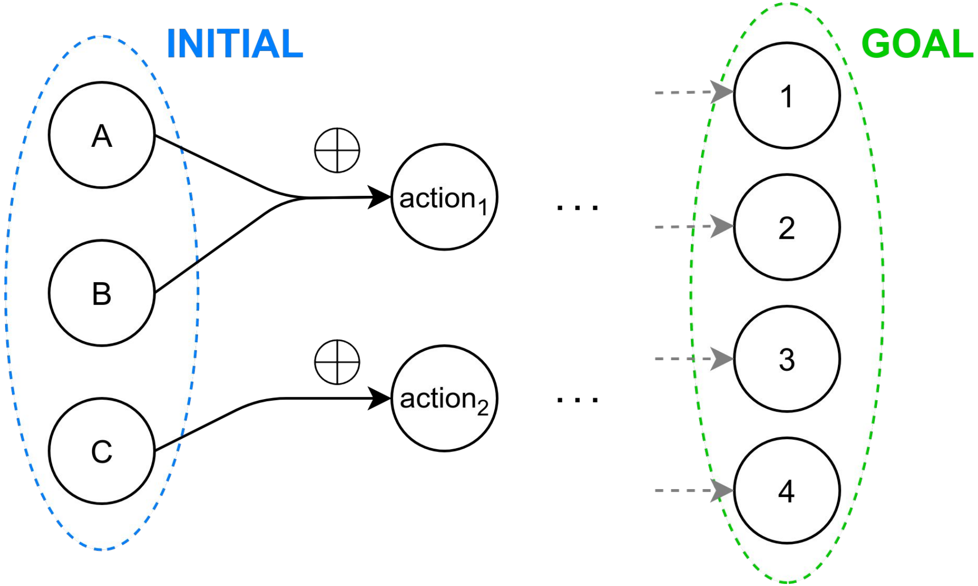
$$h^{\max}(s) = \max_{g \in G} \underbrace{h^{\max}(s; g)}_{\text{cost of achieving } g}$$



- Estimate cost of goal as the most expensive goal proposition
- Admissible but not as informative as h^{add}

Learning Heuristics over Hypergraphs

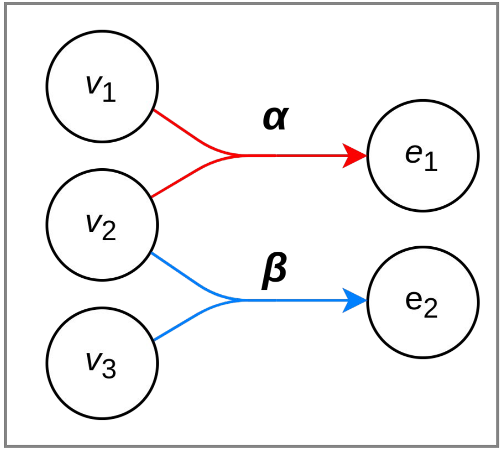
- Learn a function $\oplus$ which better approximates shortest paths



Learning Heuristics over Hypergraphs

- **Learn function *h*: hypergraph → R**

Hypergraph induced by P^+



Input Features

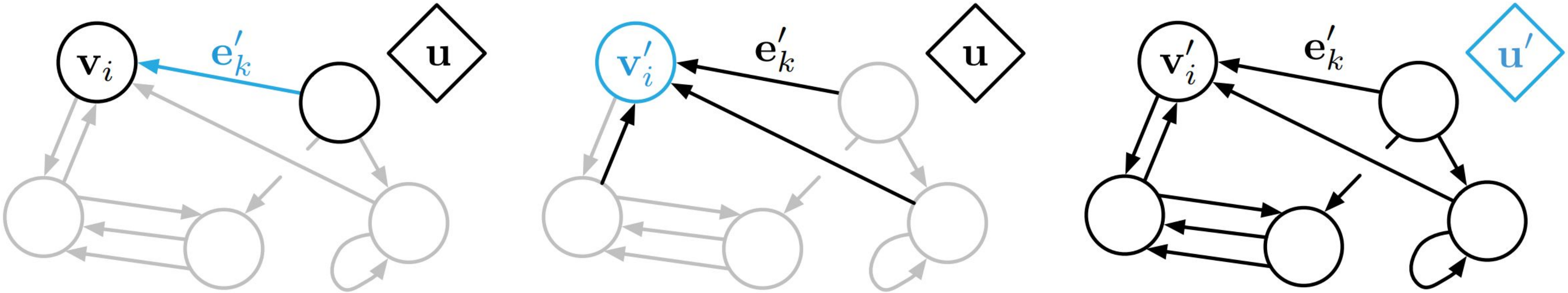
	Cost	#PRE	#ADD
Action	Cost	#PRE	#ADD
e.g. α	1	2	1

	∈ State	∈ Goal
Proposition	∈ State	∈ Goal
e.g. e_2	0	1
v_1	1	0

Hypergraph Networks (HGN)

- Our generalisation of *Graph Networks* [Battaglia et al. 2018] to hypergraphs
- **Hypergraph Network (HGN) Block**
  - Powerful and flexible building block
  - Hypergraph-to-Hypergraph mapping
  - Uses message passing to **aggregate** and **update** features with update/aggregation functions

Hypergraph Networks (HGN)

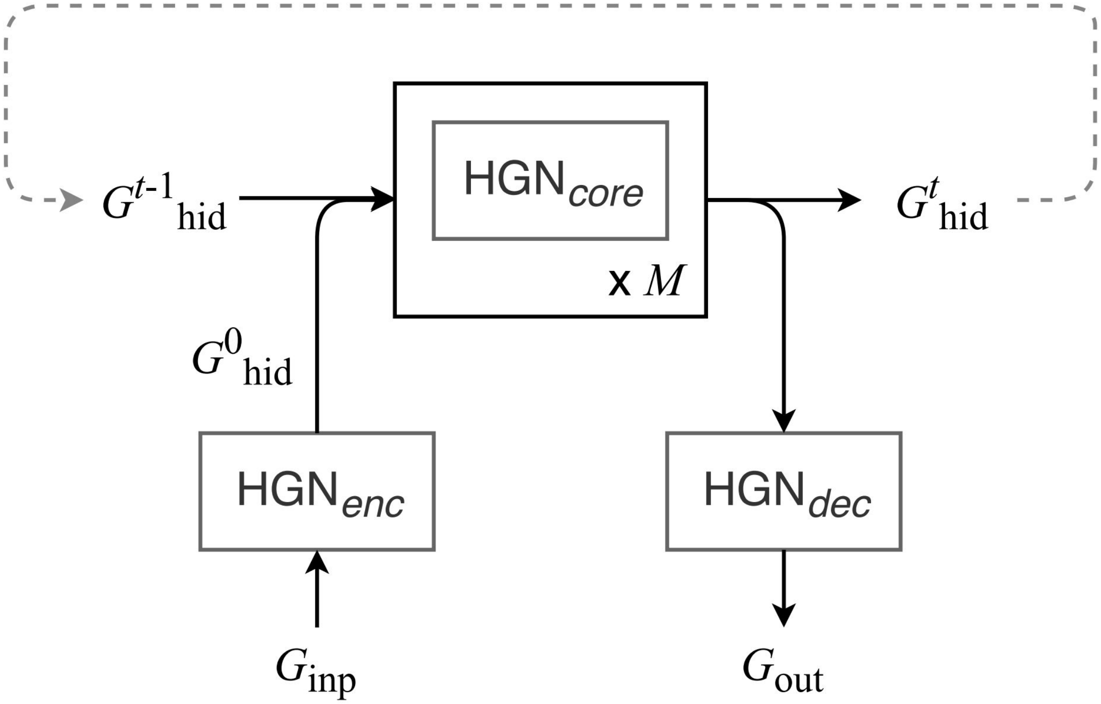


(a) Edge update (b) Node update (c) Global update

Analogous to Message Passing

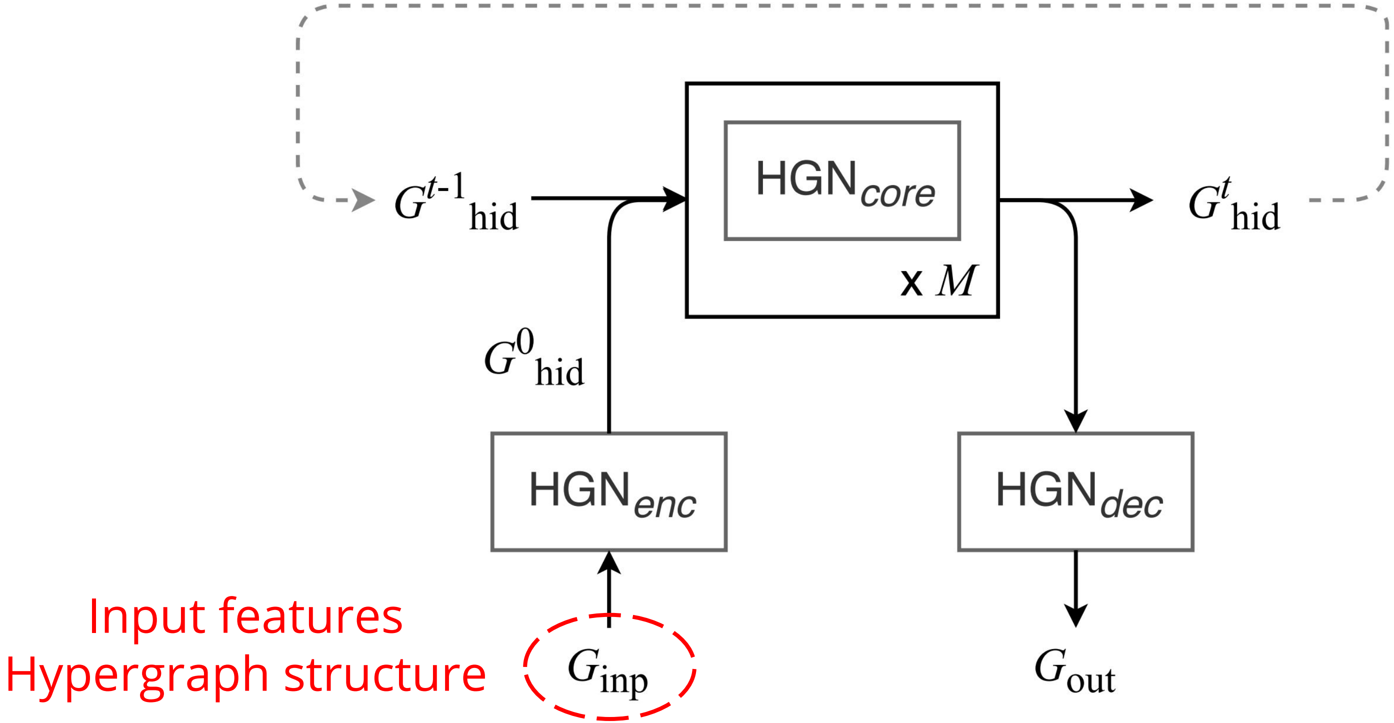
Figure from Battaglia et al. 2018

STRIPS-HGN

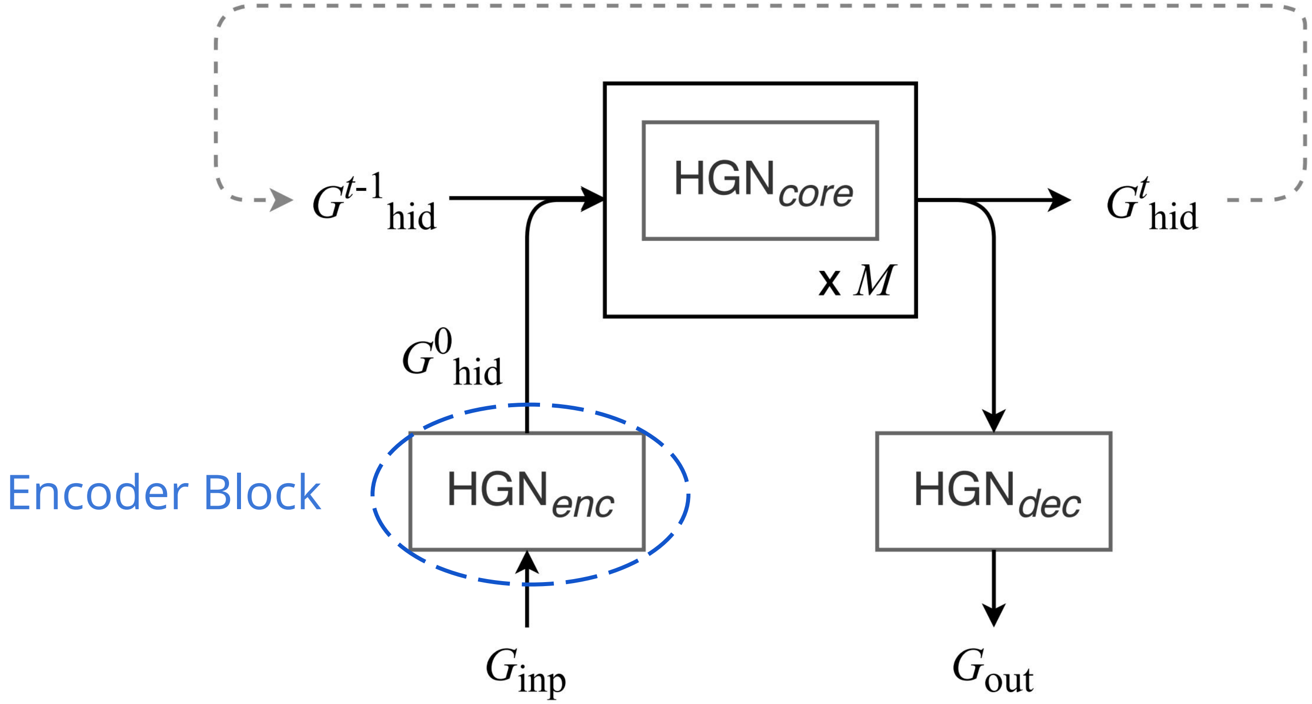


$G^{t\text{-}1}_{\text{hid}}$
HGN$_{core}$
x M
G^{t}_{hid}
G^{0}_{hid}
HGN$_{enc}$
HGN$_{dec}$
G_{inp}
G_{out}

STRIPS-HGN

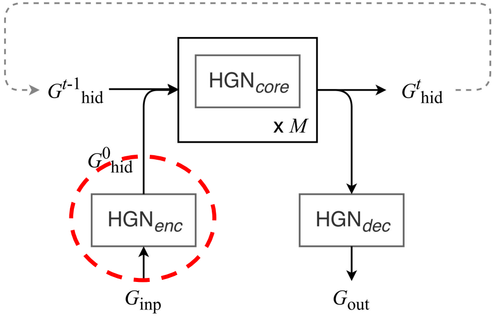

$G^{t\text{-}1}_{\text{hid}}$
HGN$_{core}$
x M
G^{t}_{hid}
G^{0}_{hid}
HGN$_{enc}$
HGN$_{dec}$
Input features
Hypergraph structure
G_{inp}
G_{out}

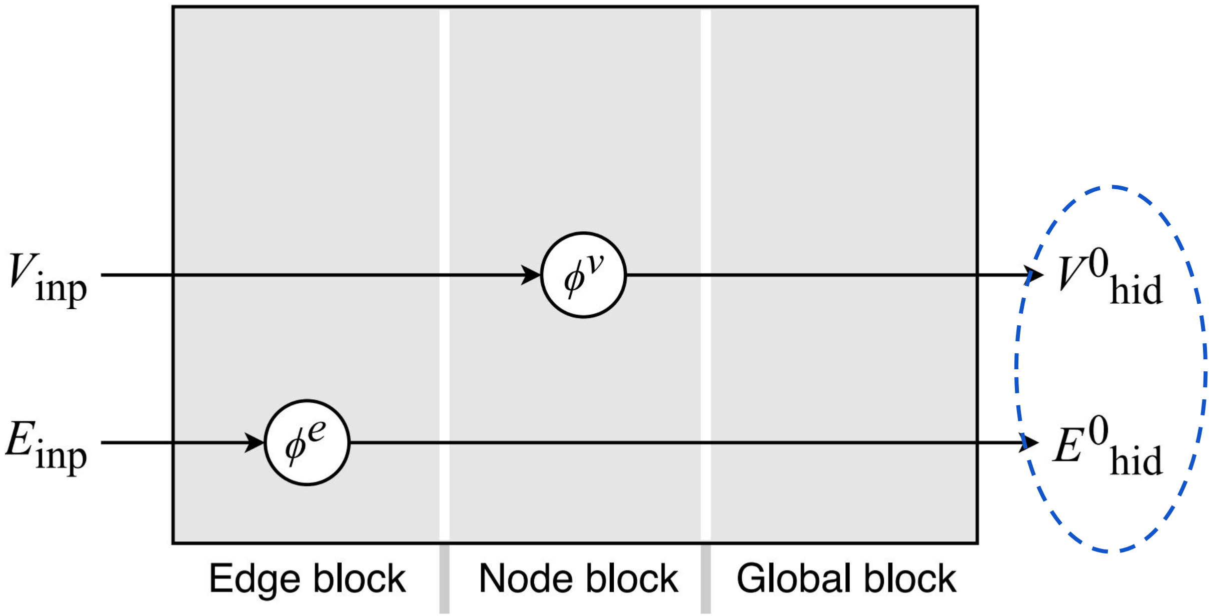
STRIPS-HGN



$G^{t\text{-}1}_{\text{hid}}$
HGN_{core}
x M
G^{t}_{hid}
G^{0}_{hid}
Encoder Block
HGN_{enc}
HGN_{dec}
G_{inp}
G_{out}

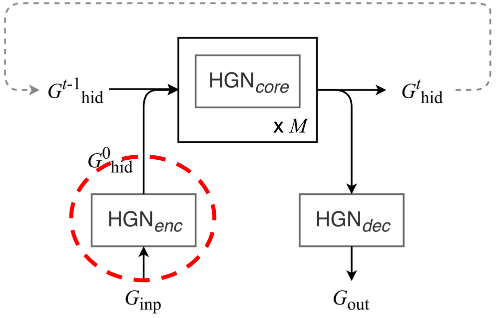
STRIPS-HGN Encoder

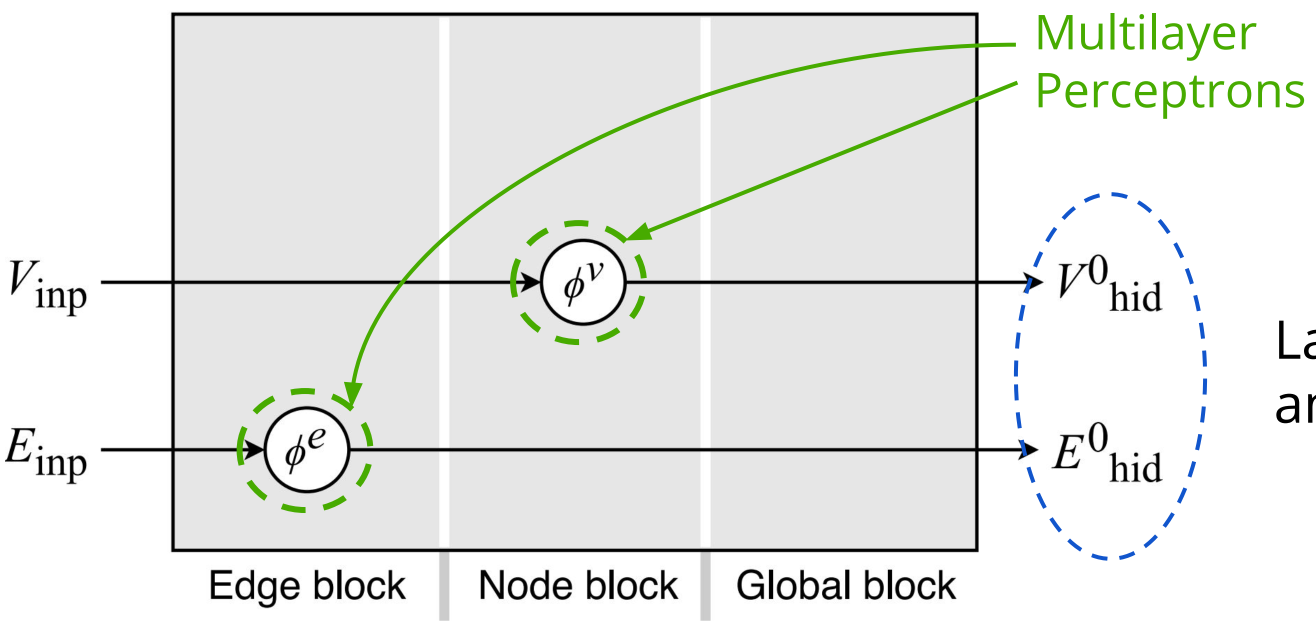




Latent proposition and action features

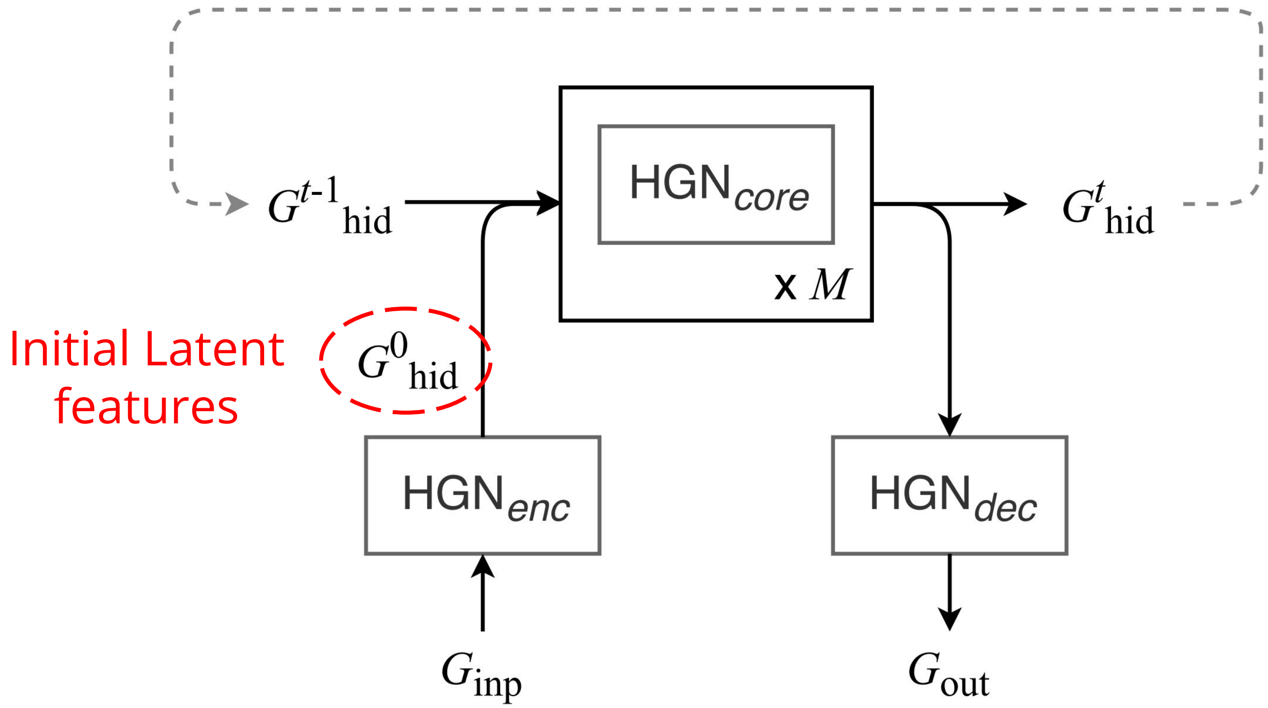
STRIPS-HGN Encoder



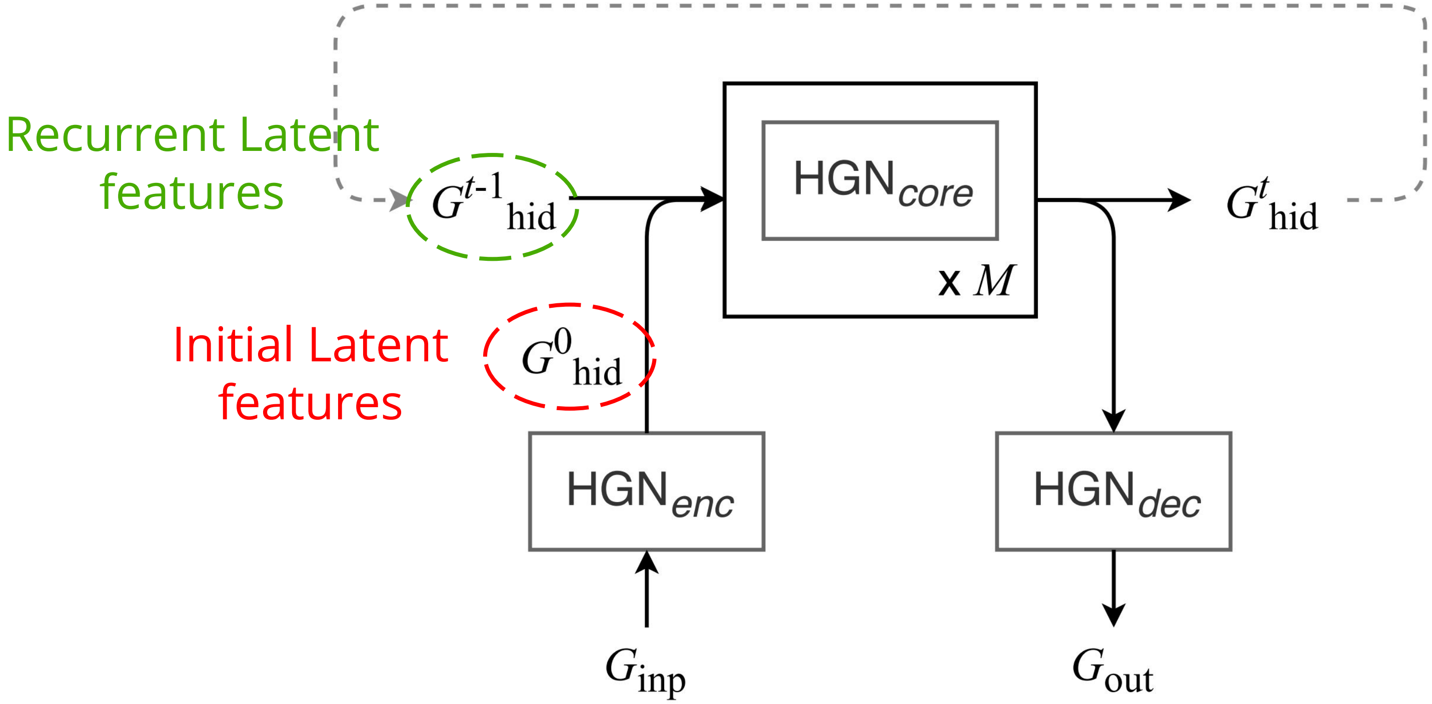


Latent proposition and action features

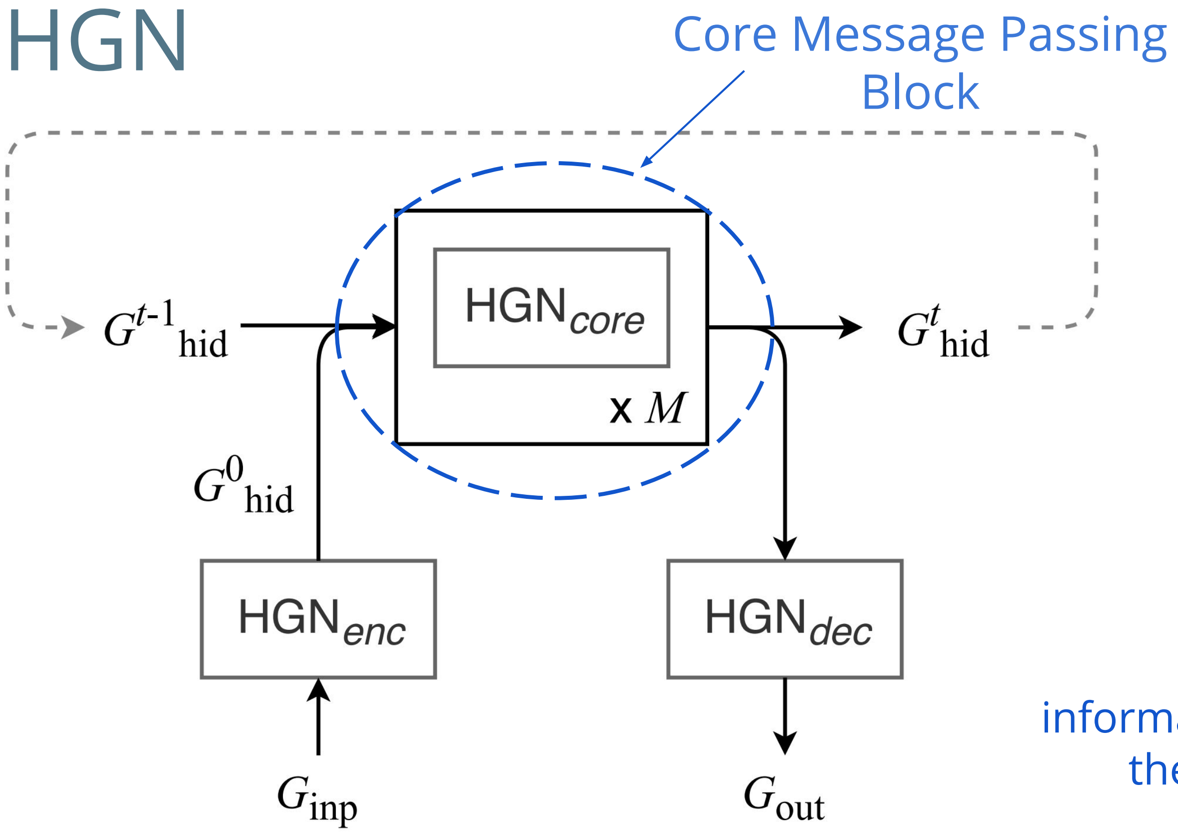
STRIPS-HGN


HGNcore
x M
Gt-1hid
Gthid
Initial Latent features
G0hid
HGNenc
HGNdec
Ginp
Gout

STRIPS-HGN

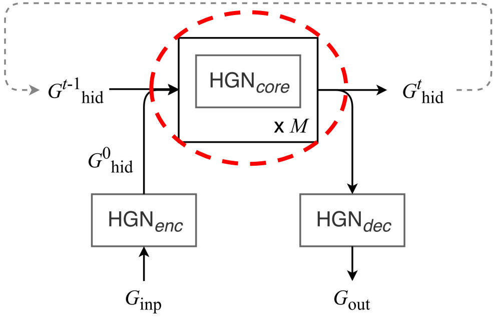

Recurrent Latent features
$G^{t\text{-}1}_{\text{hid}}$
HGN$_{core}$
x M
G^{t}_{hid}
Initial Latent features
G^{0}_{hid}
HGN$_{enc}$
HGN$_{dec}$
G_{inp}
G_{out}

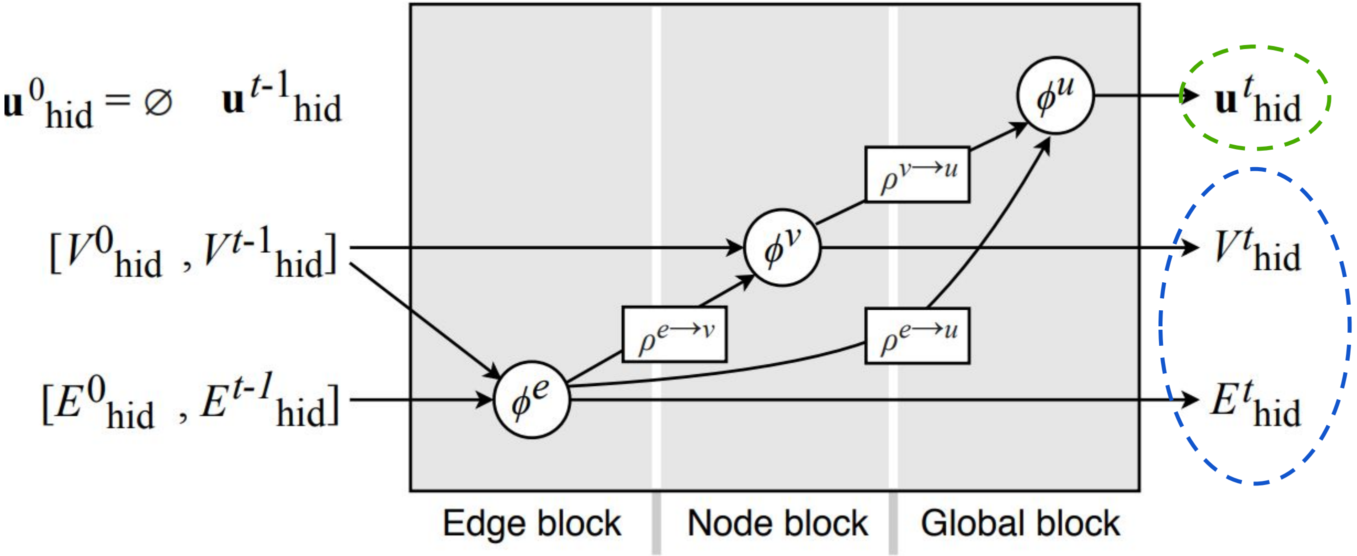
STRIPS-HGN



Propagates information through the hypergraph!

STRIPS-HGN Processing

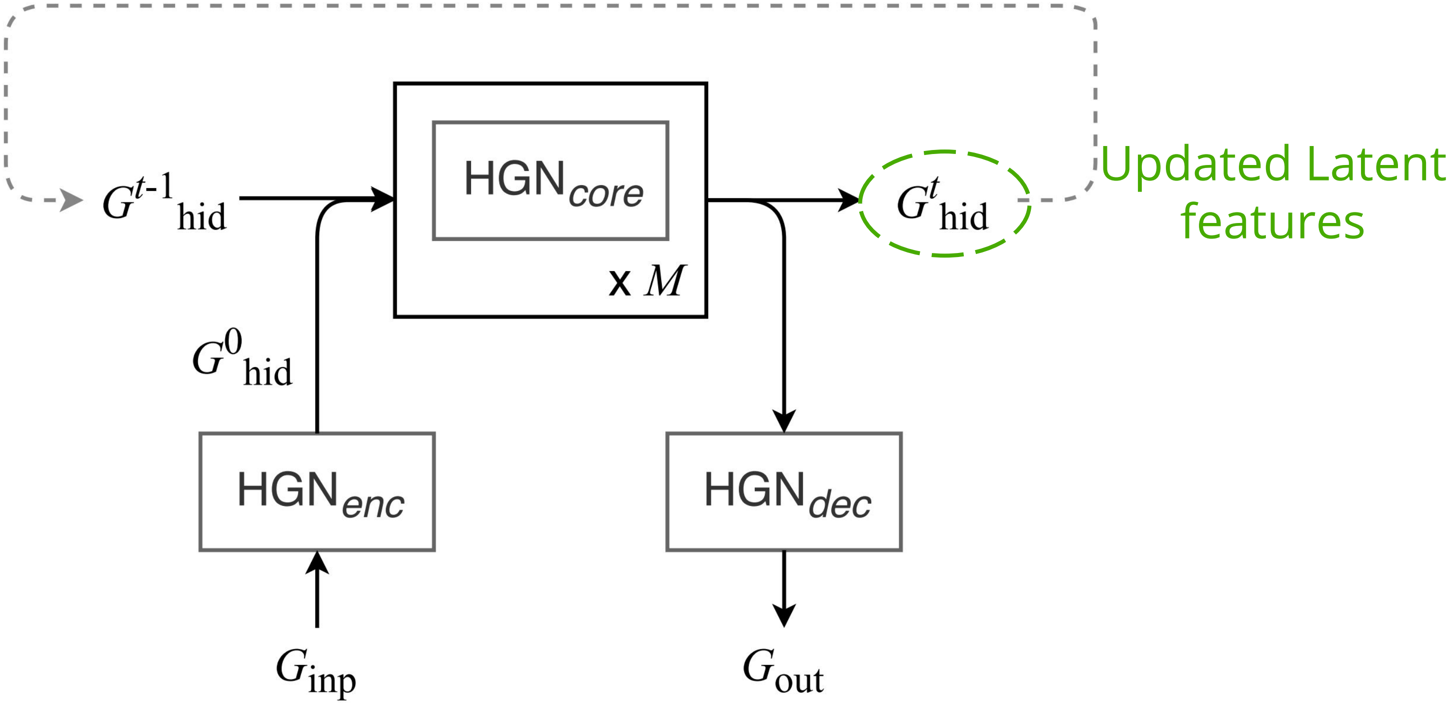




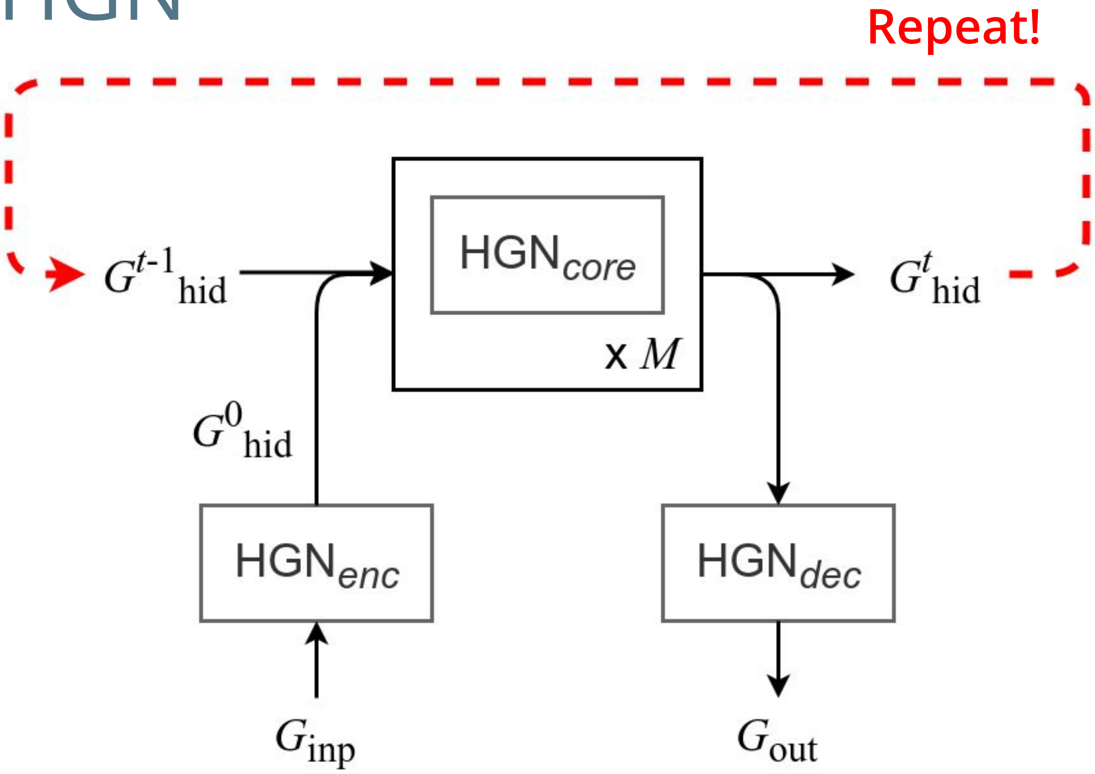
Latent heuristic value!

Updated proposition and action features

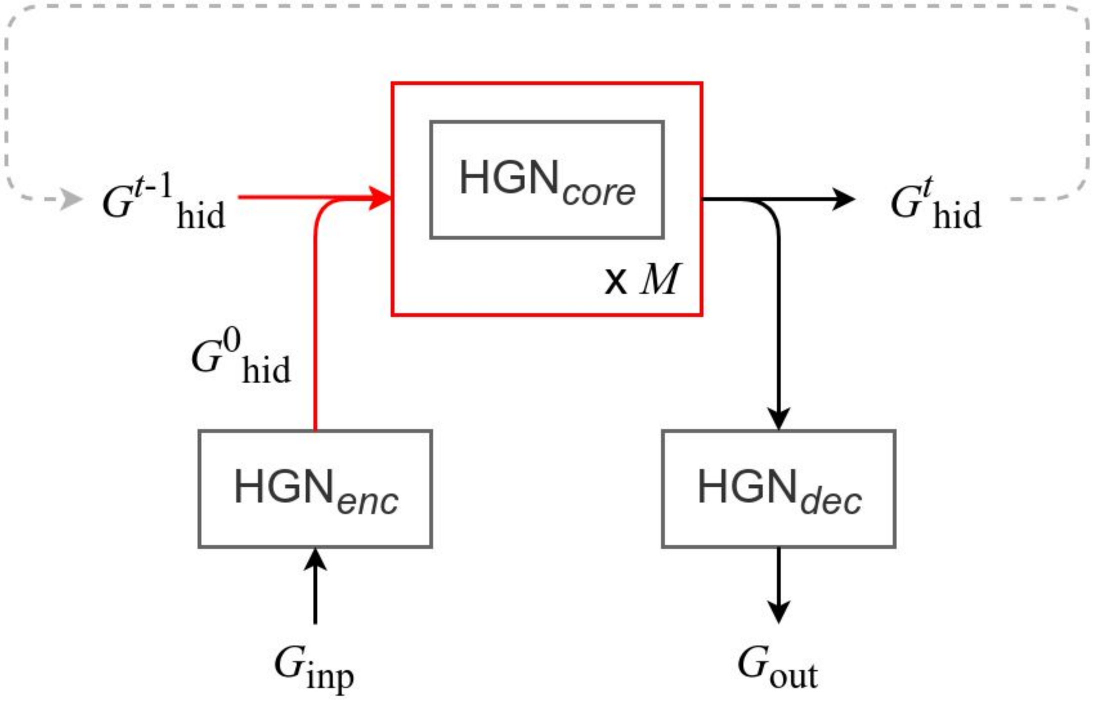
STRIPS-HGN


$G^{t\text{-}1}_{\text{hid}}$
HGN_{core}
x M
G^{t}_{hid}
Updated Latent features
G^{0}_{hid}
HGN_{enc}
HGN_{dec}
G_{inp}
G_{out}

STRIPS-HGN

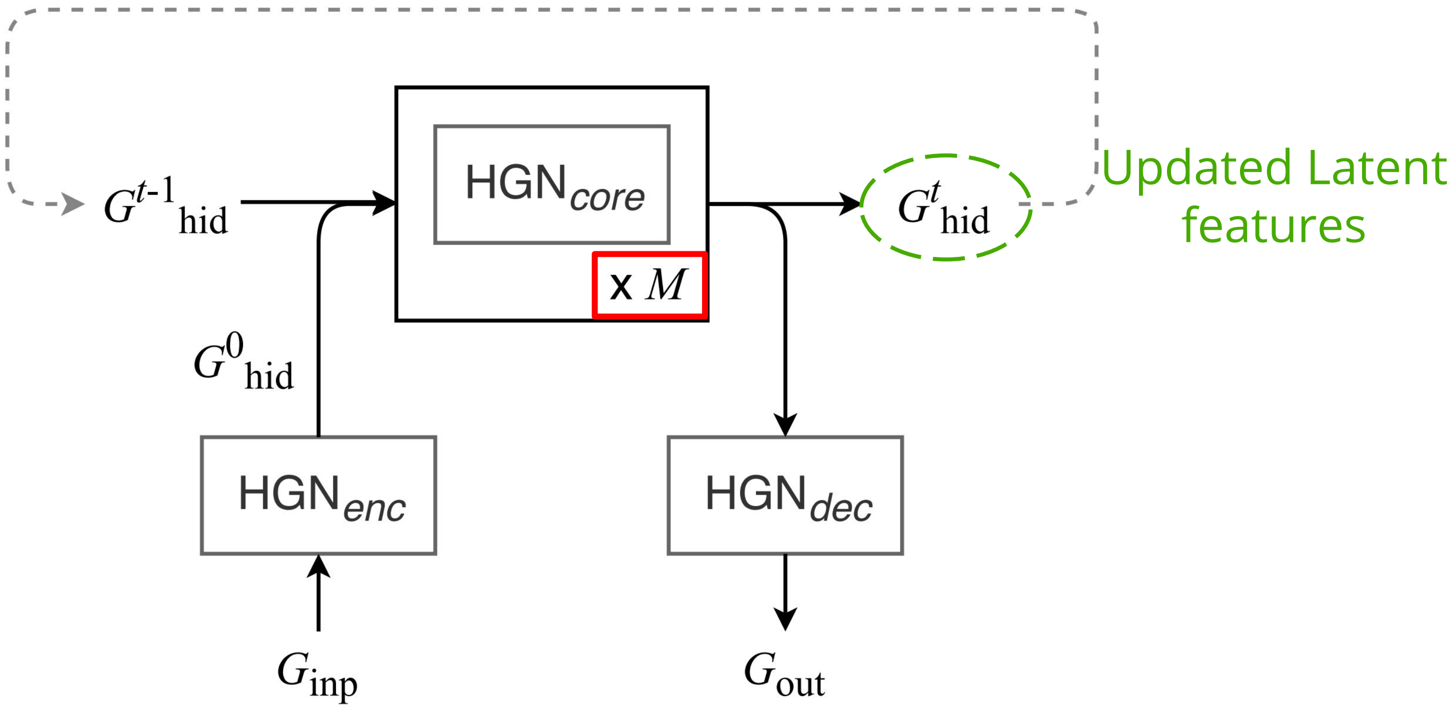

Repeat!
$G^{t\text{-}1}_{\text{hid}}$
HGN_{core}
x M
G^{t}_{hid}
G^{0}_{hid}
HGN_{enc}
HGN_{dec}
G_{inp}
G_{out}

STRIPS-HGN

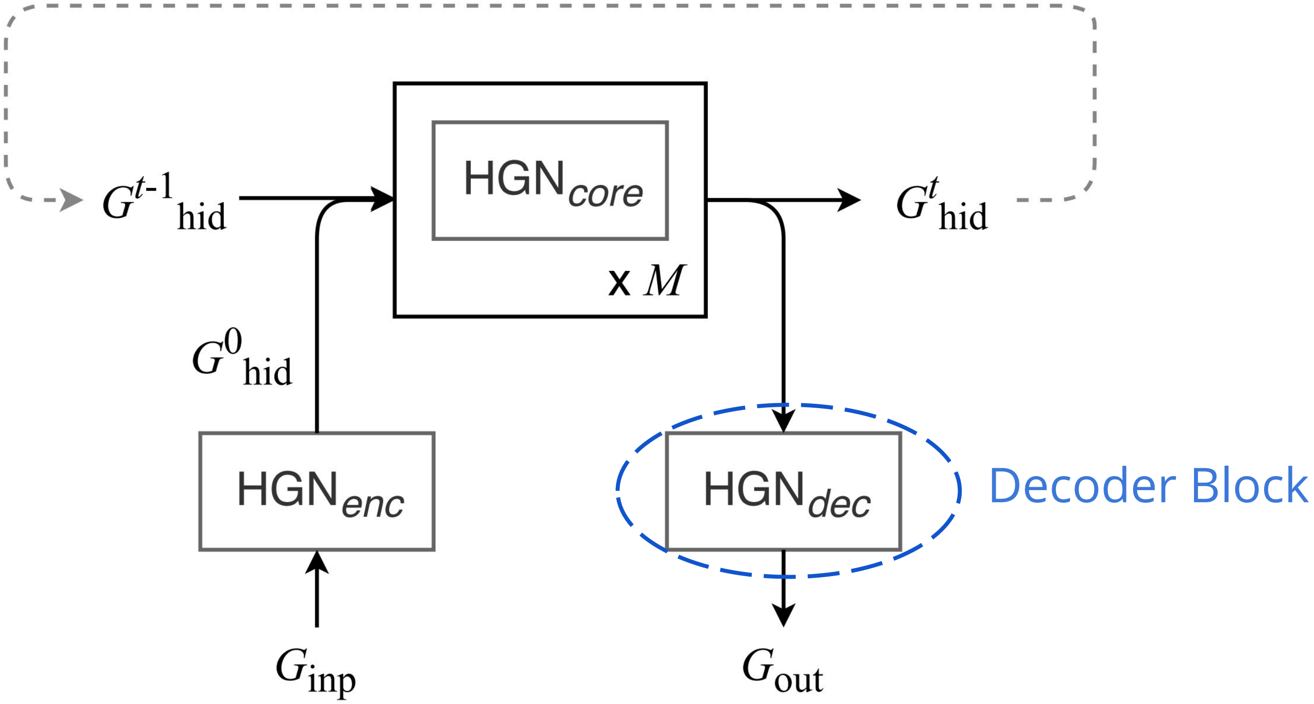


$G^{t\text{-}1}_{\text{hid}}$
HGN_{core}
x M
G^{t}_{hid}
G^{0}_{hid}
HGN_{enc}
HGN_{dec}
G_{inp}
G_{out}

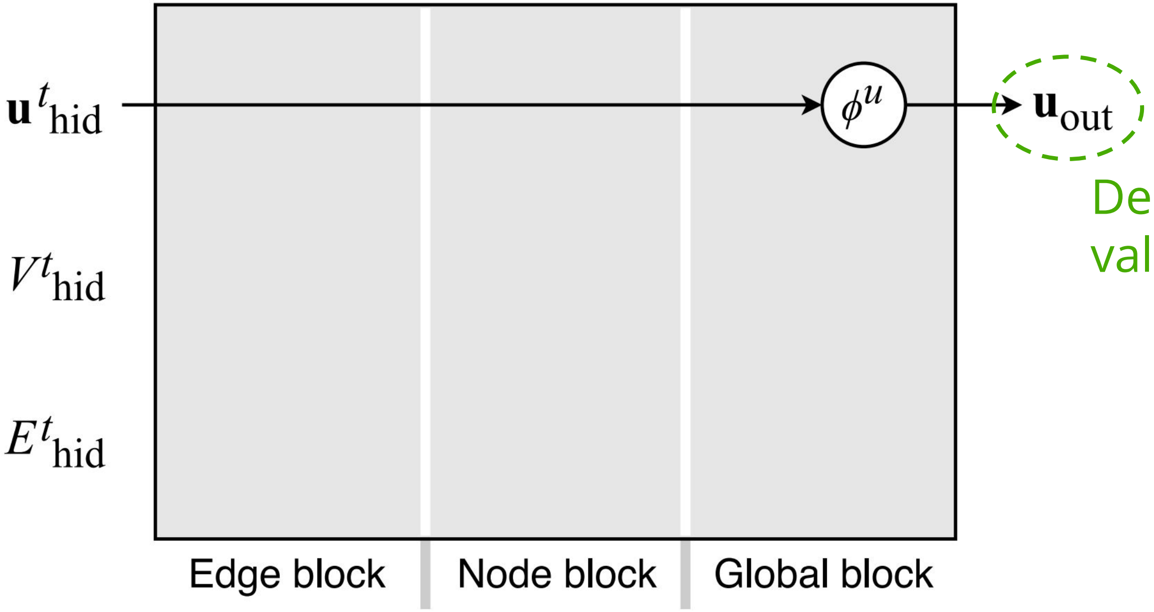
STRIPS-HGN


$G^{t\text{-}1}_{\text{hid}}$
HGN$_{core}$
x M
G^{t}_{hid}
Updated Latent features
G^{0}_{hid}
HGN$_{enc}$
HGN$_{dec}$
G_{inp}
G_{out}

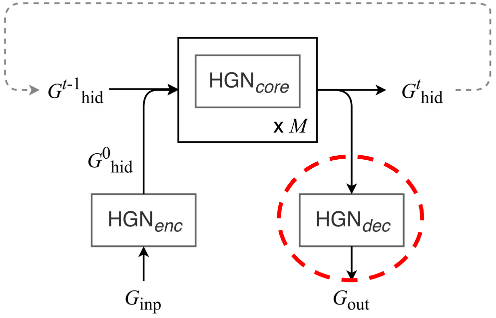
STRIPS-HGN


$G^{t\text{-}1}_{\text{hid}}$
HGN_{core}
x M
G^{t}_{hid}
G^{0}_{hid}
HGN_{enc}
HGN_{dec}
Decoder Block
G_{inp}
G_{out}

STRIPS-HGN Decoder



Decoded heuristic value (real number)



Training a STRIPS-HGN

- **Input Features** - learning from scratch
  - Proposition: `[proposition in current state, proposition in goal state]`
  - Action: `[cost, #preconditions, #add-effects]`
- **Generate Training Data**
  - Run an optimal planner for a set of training problems
  - Use the states encountered in the optimal plans
  - Aim to learn the optimal heuristic value
- **Train using Gradient Descent, treat as regression problem**

Experimental Results

- **Evaluate using A* Search**
- **Baseline Heuristics**
  - h^{add} (inadmissible), h^{max}, blind and Landmark Cut (admissible)
- **STRIPS-HGN:** h^{HGN}
  - Train and evaluate on a **single CPU core**
  - Run core block 10 times (i.e., M = 10)
  - Powerful generalisation but slower to compute

Evaluation on domains we trained on

Training

Zenotravel

10 small Training Problems

2-3 cities

Gripper

3 small Training Problems

1-3 balls

Blocksworld

10 small Training Problems

4-5 blocks

Testing

Gripper

18 larger Testing Problems

4-20 balls

Blocksworld

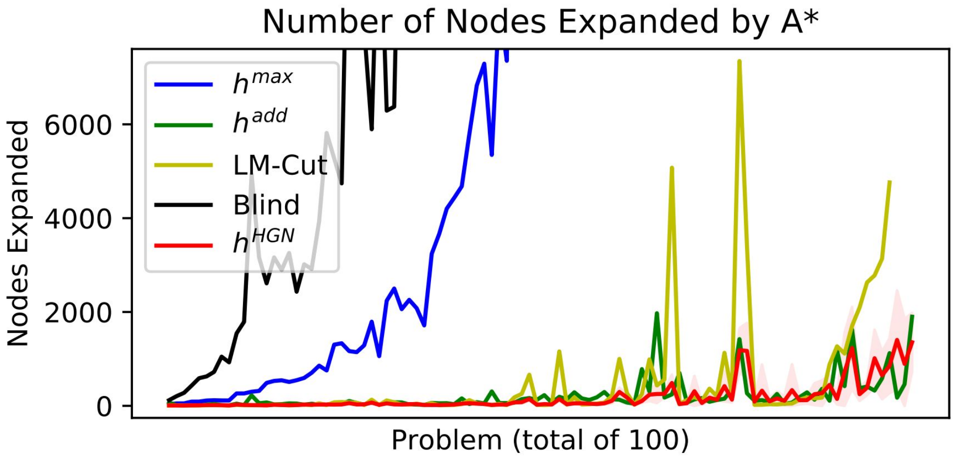
100 larger Testing Problems

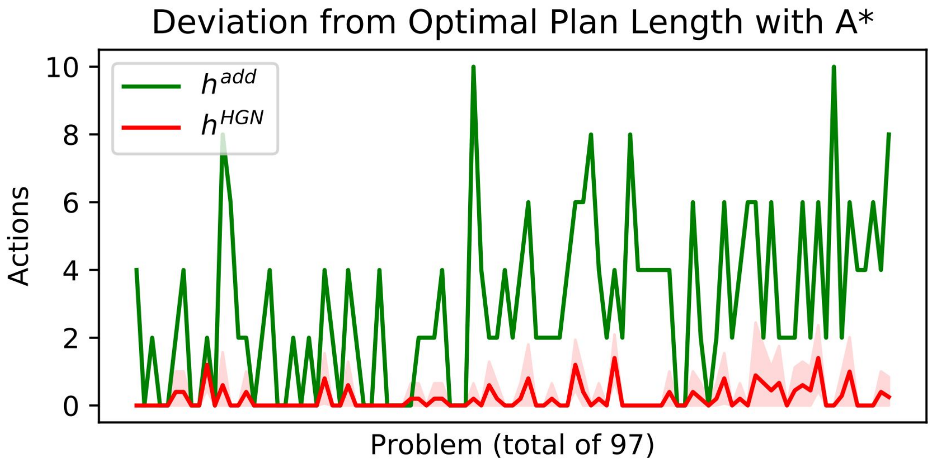
6-10 blocks

- Train and evaluate **a single network** on 3 domains.
- **Training time:** 15 min

Blocksworld (trained on)

Train on Zenotravel, Gripper & **Blocksworld**





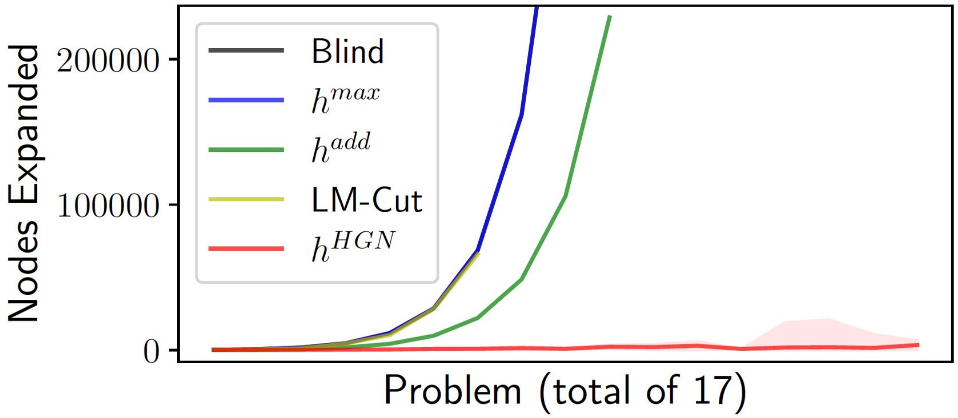
95% confidence interval shown for h^{HGN} over 10 repeated experiments.

Australian National University

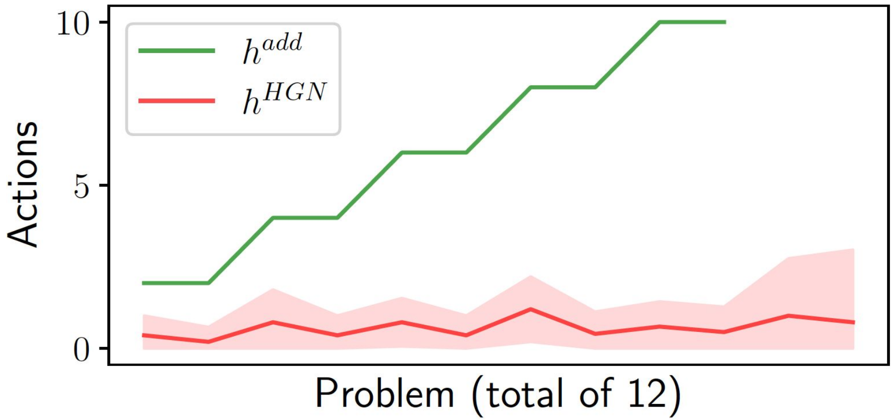
Gripper (trained on)

Train on Zenotravel, **Gripper** & Blocksworld

Number of Nodes Expanded by A*



Deviation from Optimal Plan Length with A*



Evaluation on domains we did **not** train on

Training

Zenotravel

10 small Training Problems

2-3 cities

Gripper

3 small Training Problems

1-3 balls

Testing

Blocksworld

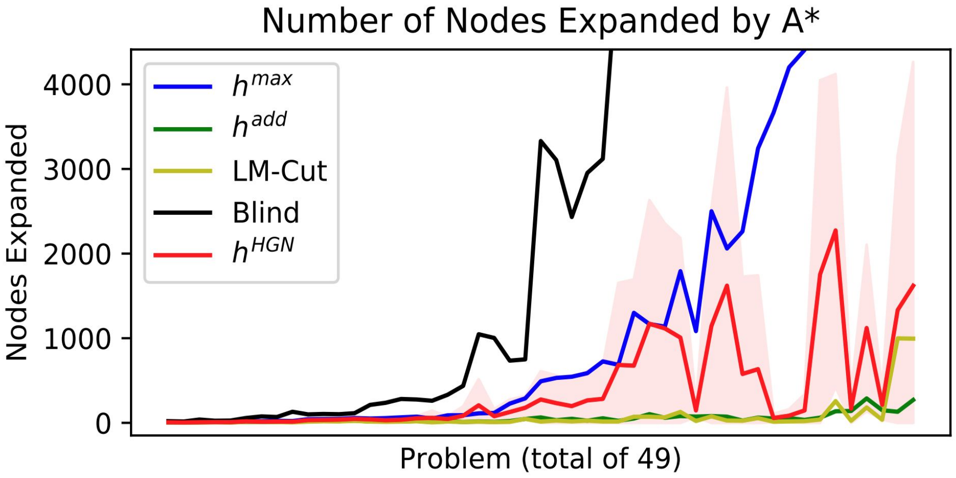
50 Testing Problems

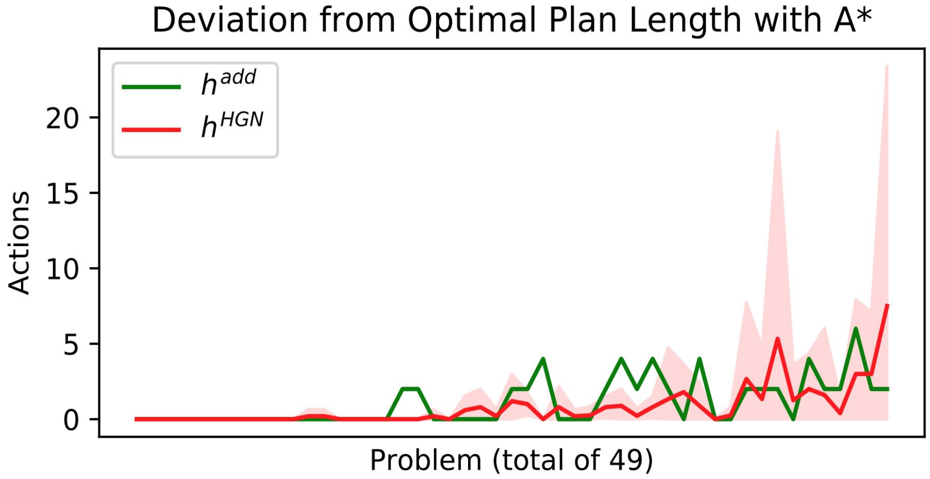
4-8 blocks

- Train a single network on 2 domains. Evaluate on **new unseen domain.**
- **Training time:** 10 min

Blocksworld (not trained on)

Train on Zenotravel and Gripper **only.**





Future Work

- **Speeding up a STRIPS-HGN**
  - Slow to evaluate - bottleneck
  - Optimise Hypergraph Networks implementation
  - Take advantage of multiple cores or use GPUs for parallelisation
- **Improve Generalisation Performance**
  - Use richer set of input features
  - Careful study of hyperparameter space, similar to [Ferber et al. 2020]

Thanks!