Australian National University

Learning Heuristics for Planning with Hypergraph Networks

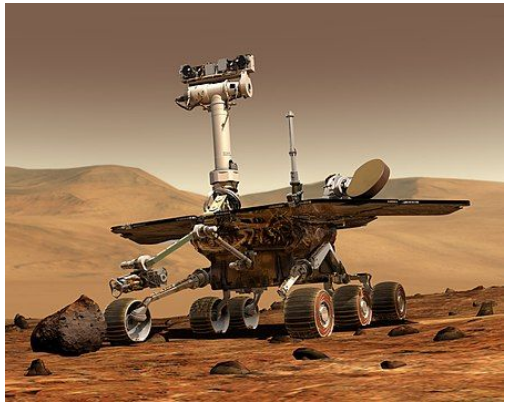
William Shen

Supervisors: Felipe Trevizan, Sylvie Thiébaux

7th November 2019

What is Planning?

- **Decision making** - reasoning about what actions to take



Mars Exploration Rover (2003)

Elevator Control

State-of-the-art Planners

- **Heuristic Search**
  - A heuristic is an estimate of the ‘cost-to-go’ (rules out unpromising regions of the search space)
  - Use a heuristic to guide forward state space search
- **End-to-End Machine Learning**
  - Action-Schema Networks [Toyer et al. AAAI’18]
  - Deep Reactive Policies
  - *Imitate* the actions of an expert (i.e. heuristic search planner)
- What about learning a heuristic?

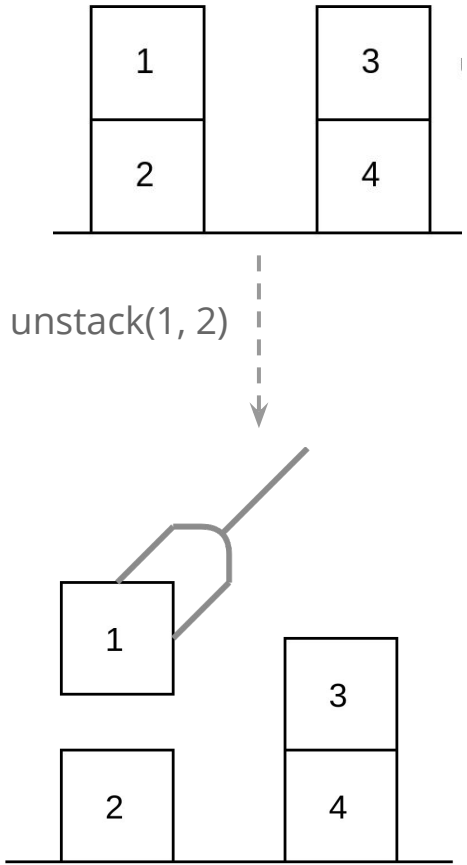
Outline

- **Background on Planning and Heuristics**
- **Contribution: Hypergraph Networks**
  1. Framework for Deep Learning over Hypergraphs
  2. STRIPS-HGN: learning heuristics for planning
- **Experimental Results**
- **Future Work**

STRIPS

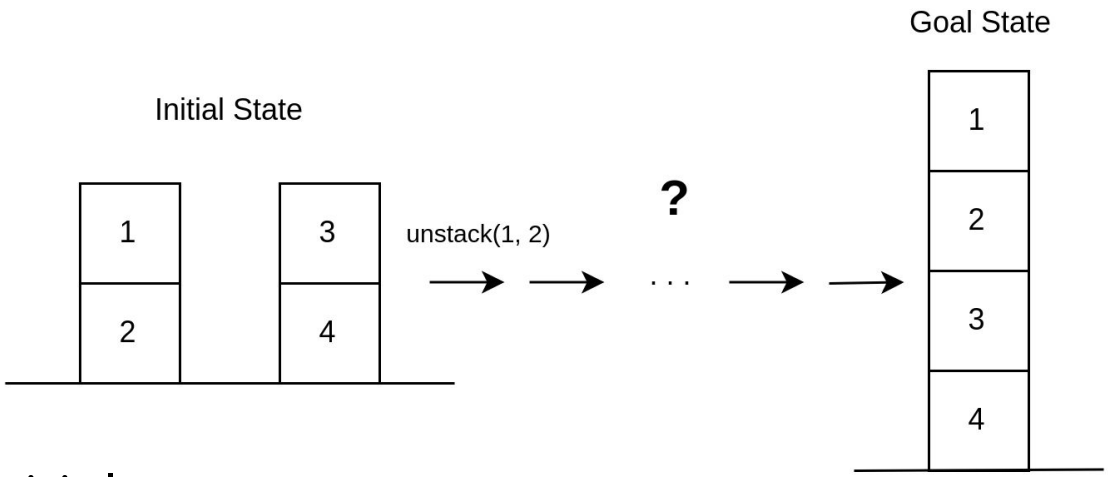
- Finite set of states and actions
- A **state** is composed of positive propositions
  - on(1, 2) ontable(2) ontable(4) etc.
- An **action** has:
  - Preconditions
  - Effects
    - Positive Effects
    - Negative Effects

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



STRIPS

- Initial State
- Goal States
- Cost Function
- **Objective**: move from initial state to a goal state with minimal cost



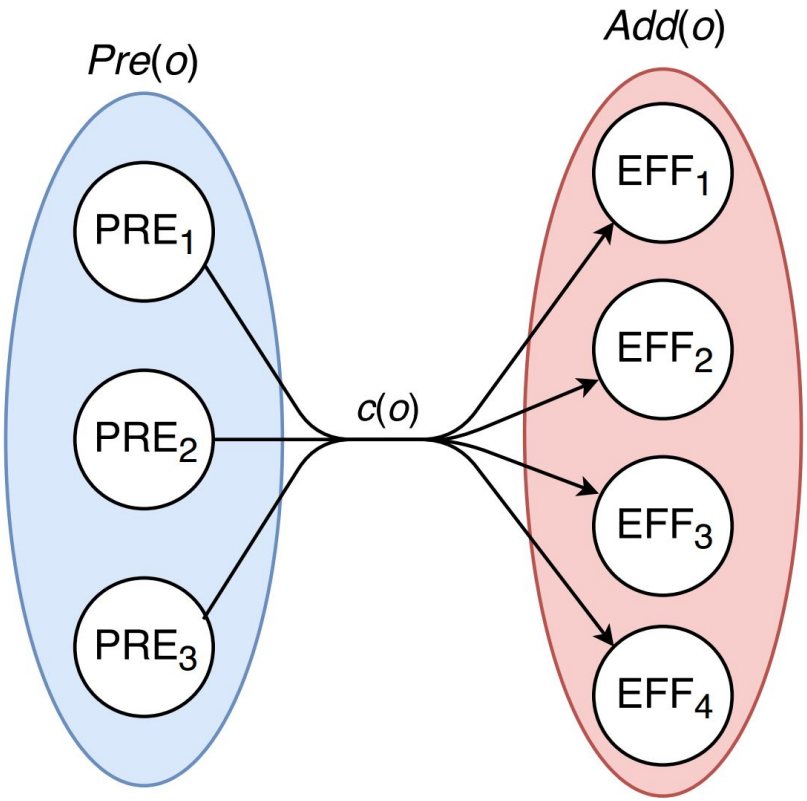
Heuristics

- **Domain-Dependent**
  - Generalises to problems from same domain
- **Domain-Independent**
  - Generalises to problems from multiple domains

- **Delete-Relaxation Heuristics**: ignore negative effects
  - Approximate the shortest path from initial to goal state
    - Polynomial time to calculate a heuristic (vs exponential)

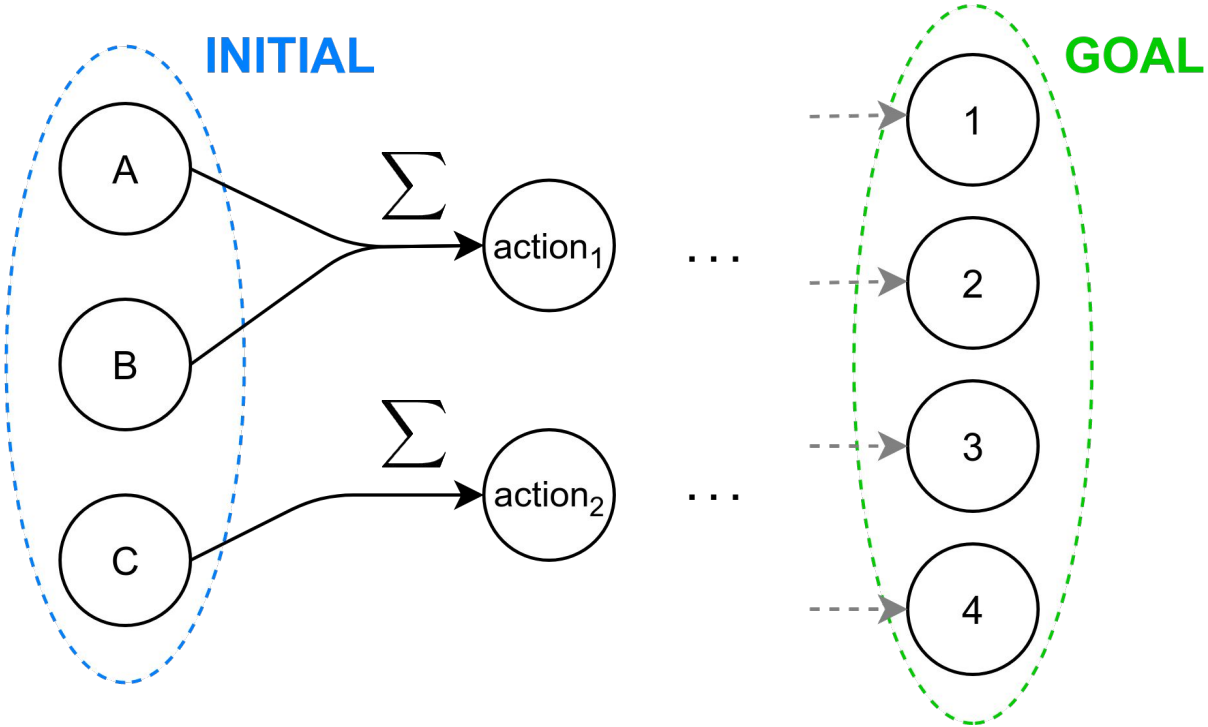
Hypergraphs

- **Hyperedge**
  - Edge that joins any number of vertices
- **STRIPS Action**
  - Preconditions
  - Effects
    - Positive
    - ~~Negative~~ Ignore!
  - Cost



h^{add} heuristic

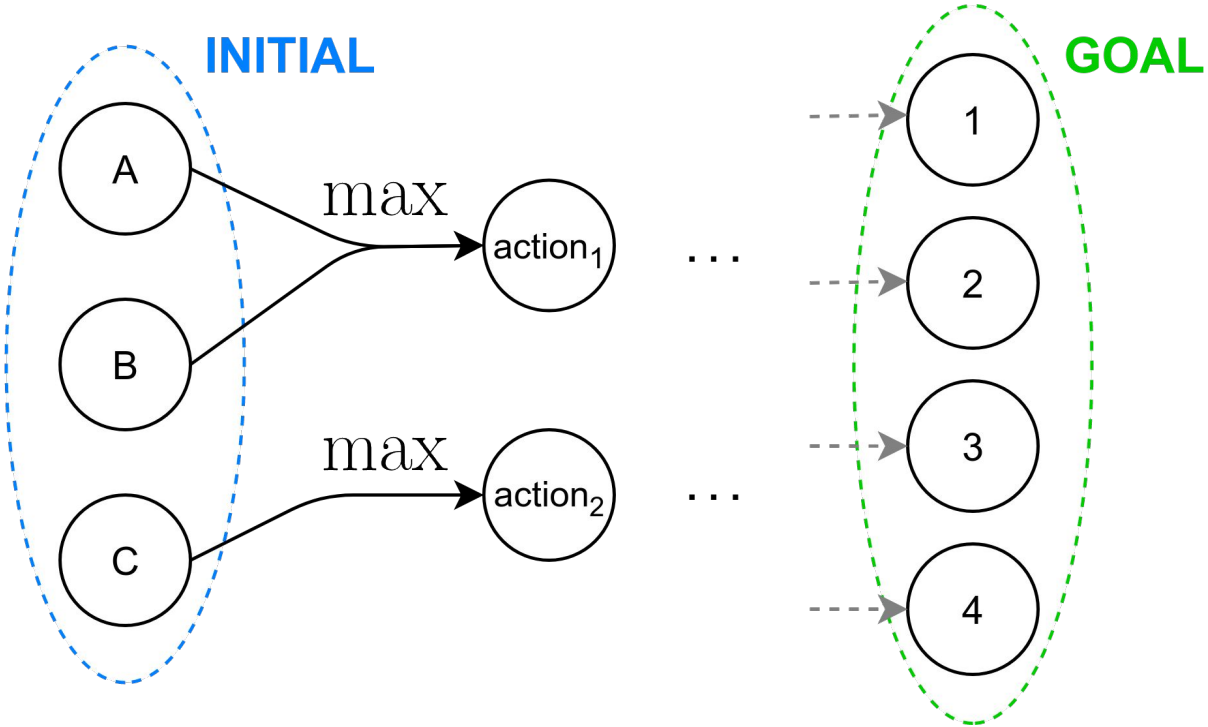
$$h^{\text{add}}(s) = \sum_{g \in G} \underbrace{h^{\text{add}}(s; g)}_{\text{cost of achieving } g}$$



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

$h^{\max}$ heuristic

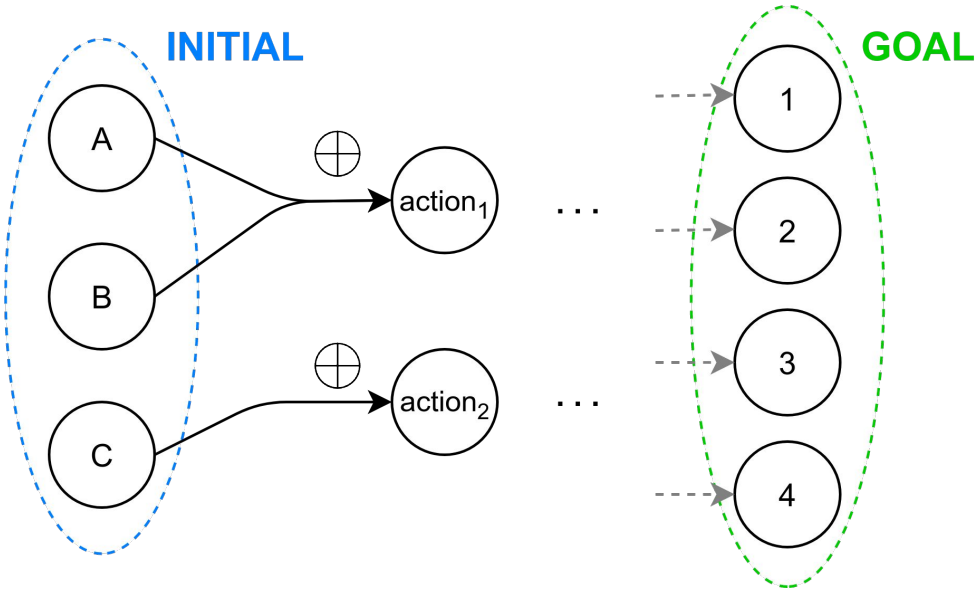
$$h^{\max}(s) = \max_{g \in G} \underbrace{h^{\max}(s; g)}_{\text{cost of achieving } g}$$



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

Learning Hypergraph Heuristics

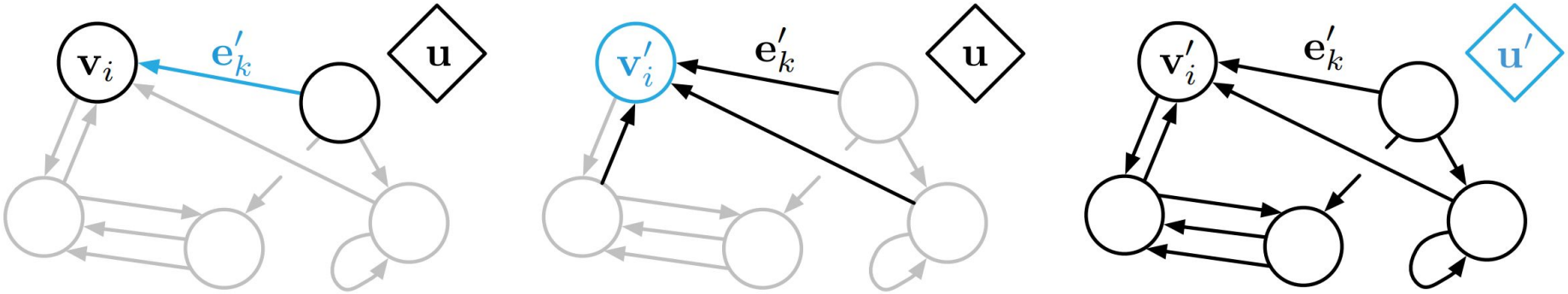
- Learn a function $\oplus$ which better approximates shortest paths



Hypergraph Networks (HGN)

- Our generalisation of Graph Networks [Battaglia et al. 2018] to hypergraphs
- **Generalises and extends** existing deep learning models
- Powerful and flexible building blocks
- Hypergraph Network Block
  - Hypergraph-to-Hypergraph mapping
  - Update functions: compute per-hyperedge and per-vertex updates

Hypergraph Networks (HGN)

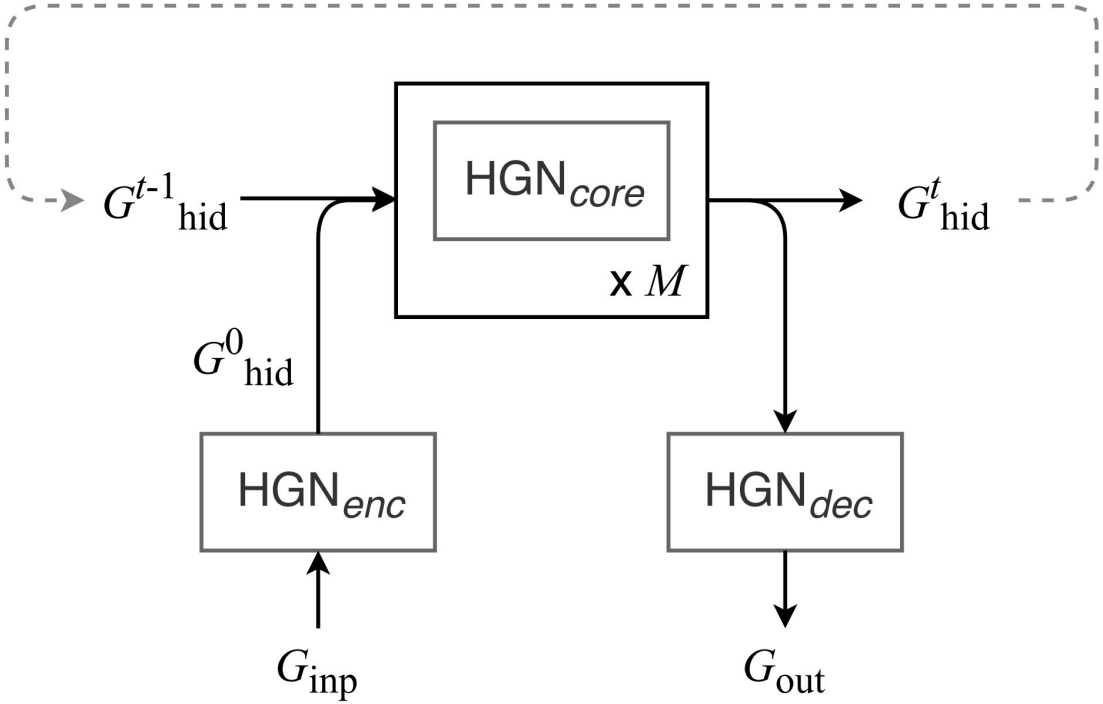


(a) Edge update (b) Node update (c) Global update

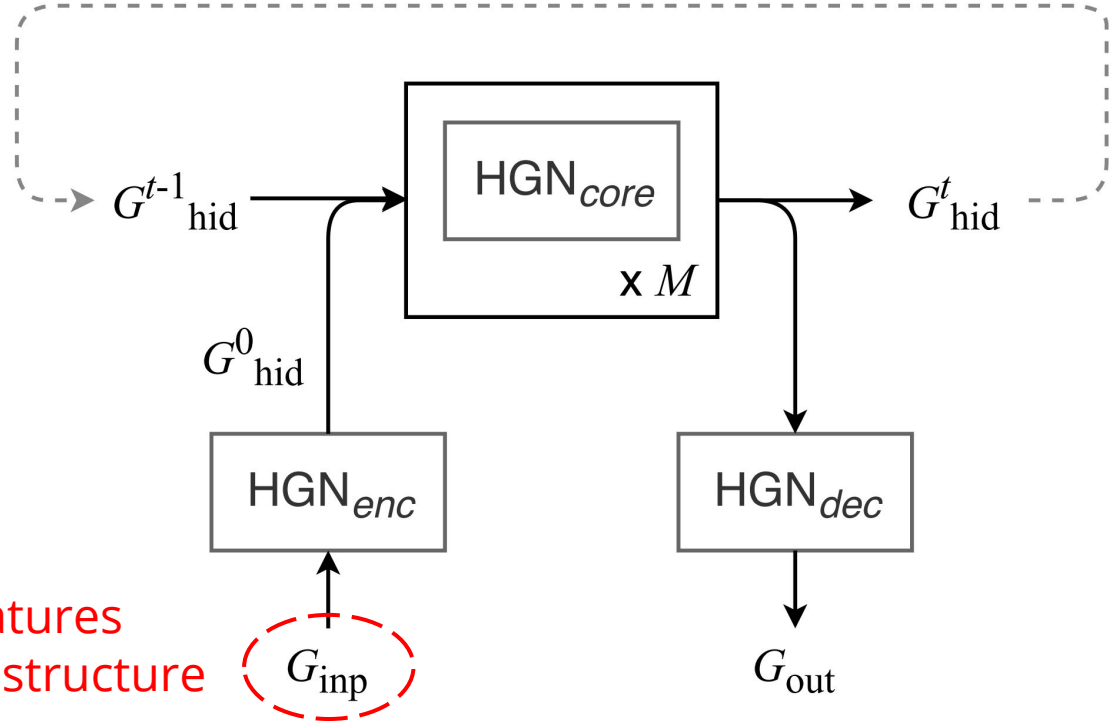
Analogous to Message Passing

Figure from Battaglia et al. 2018

STRIPS-HGN

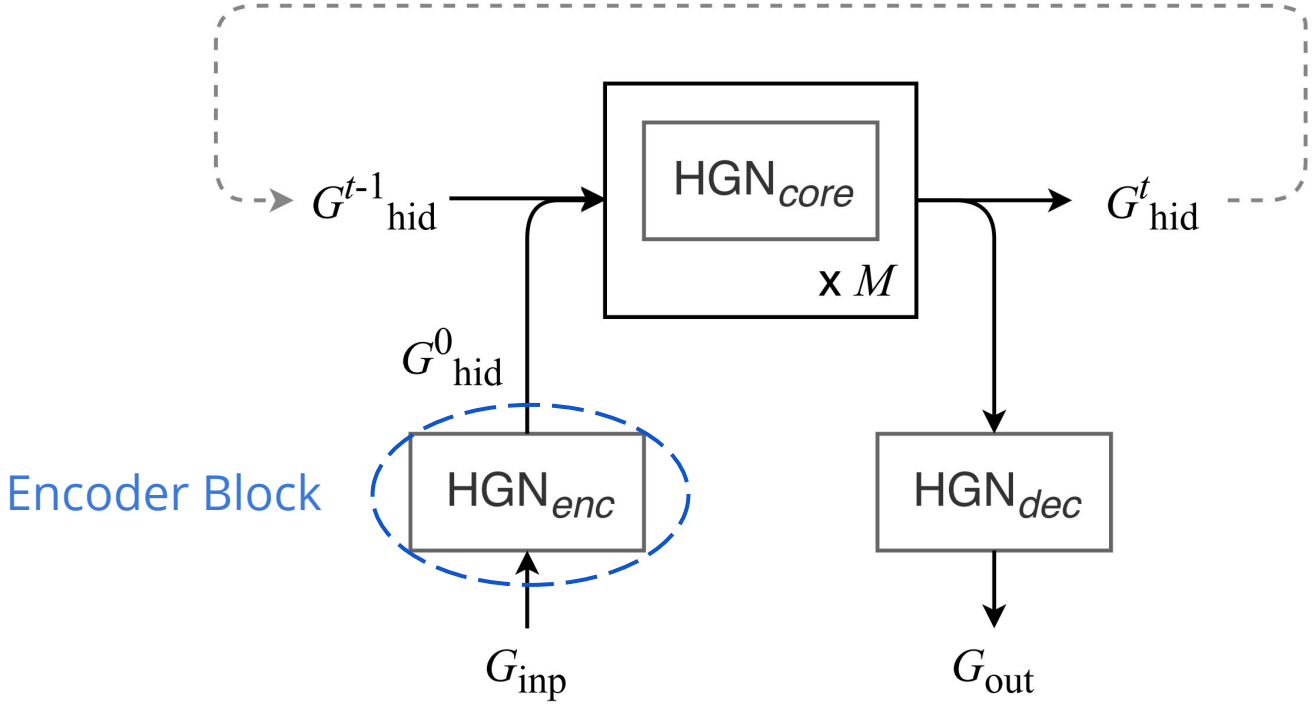

G^{t-1}_{hid}
HGN$_{core}$
x M
G^{t}_{hid}
G^{0}_{hid}
HGN$_{enc}$
HGN$_{dec}$
G_{inp}
G_{out}

STRIPS-HGN

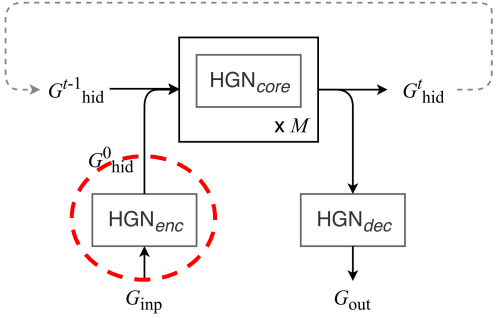


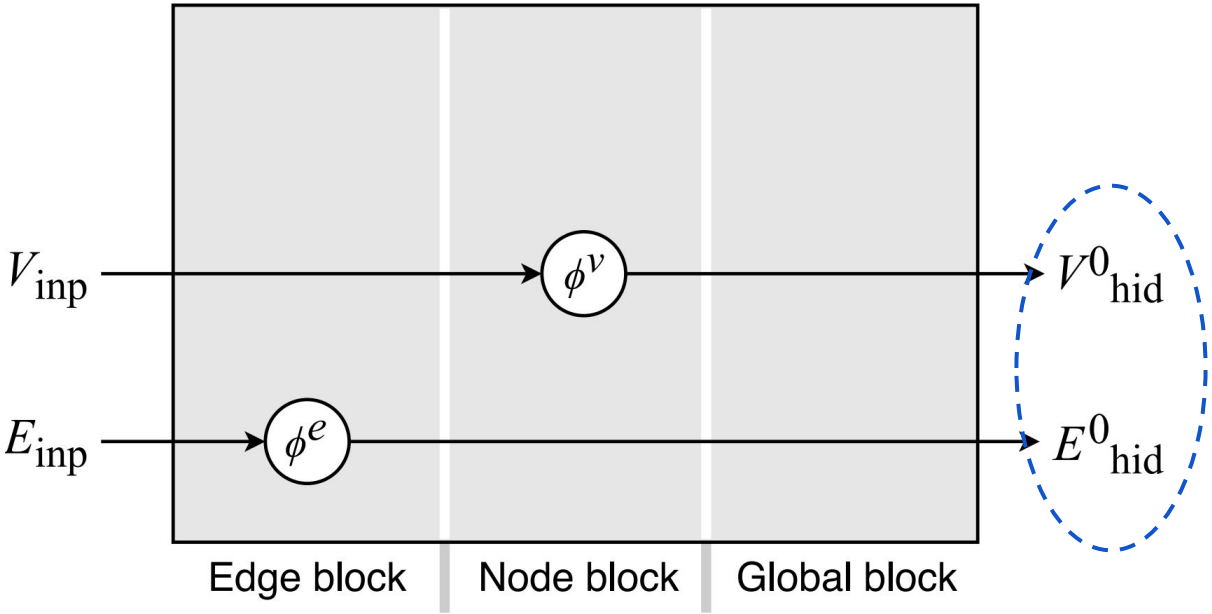
Input features
Hypergraph structure

STRIPS-HGN


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

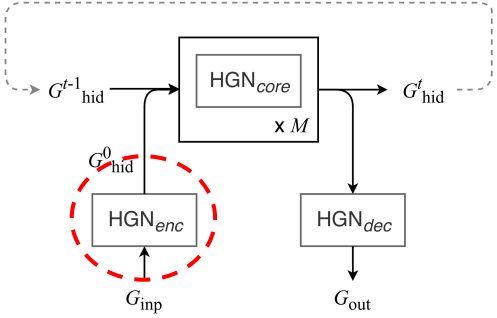
STRIPS-HGN Encoder

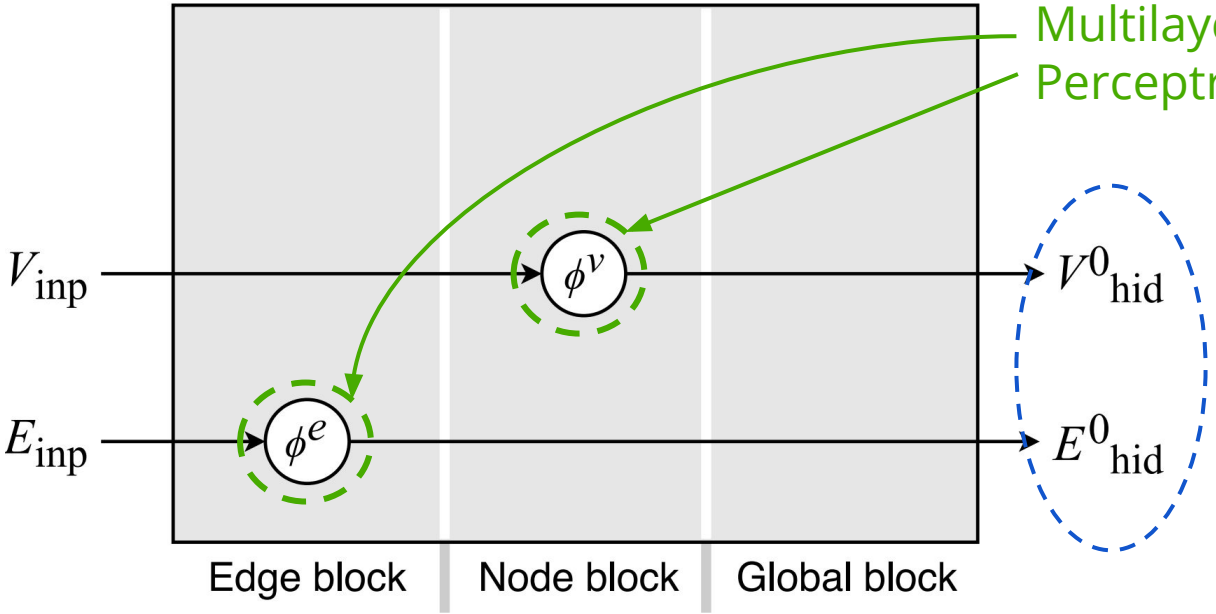




Latent proposition and action features

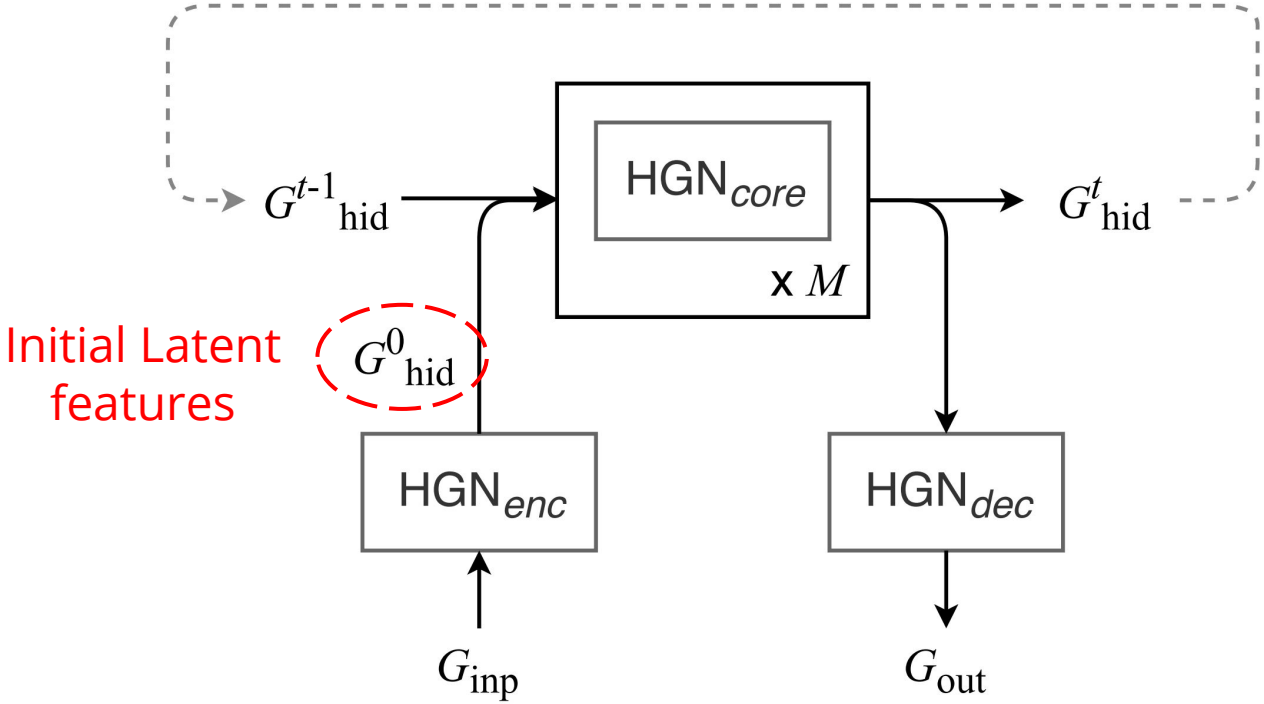
STRIPS-HGN Encoder



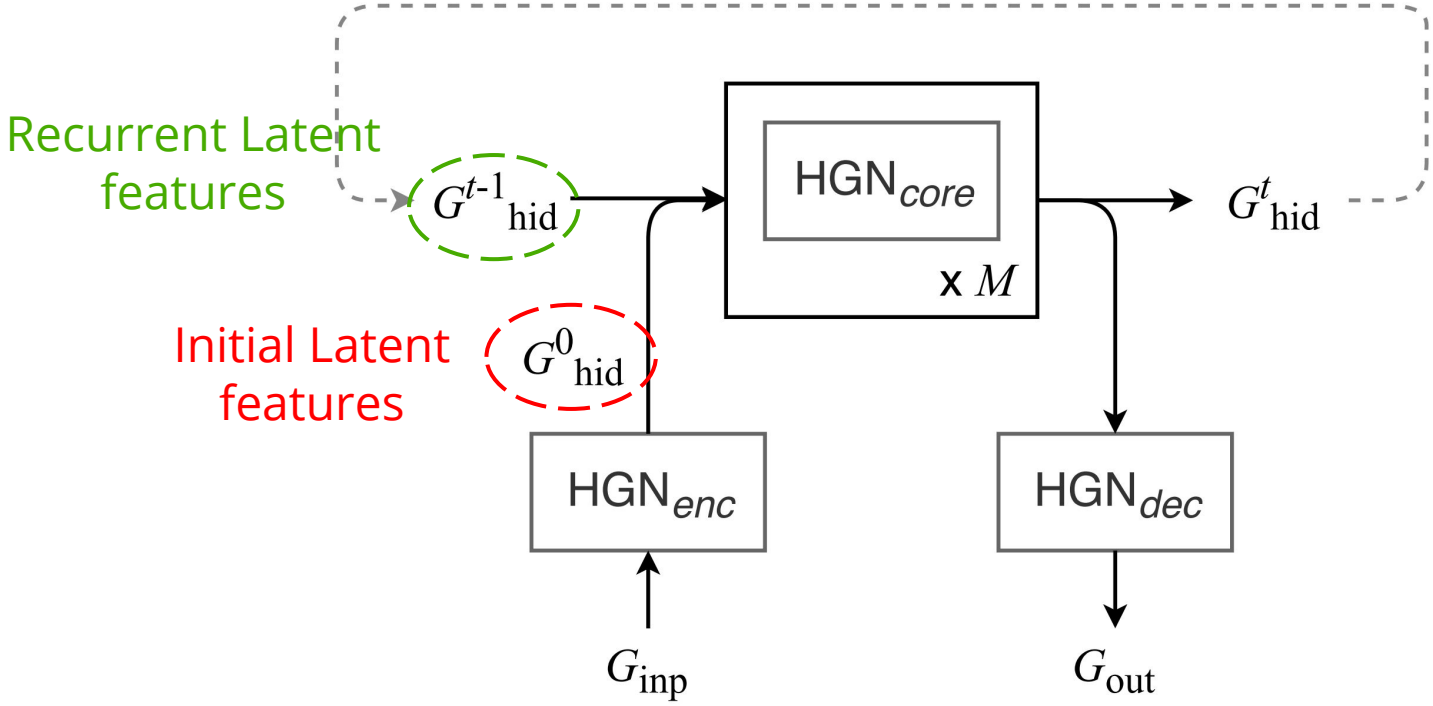


Latent proposition and action features

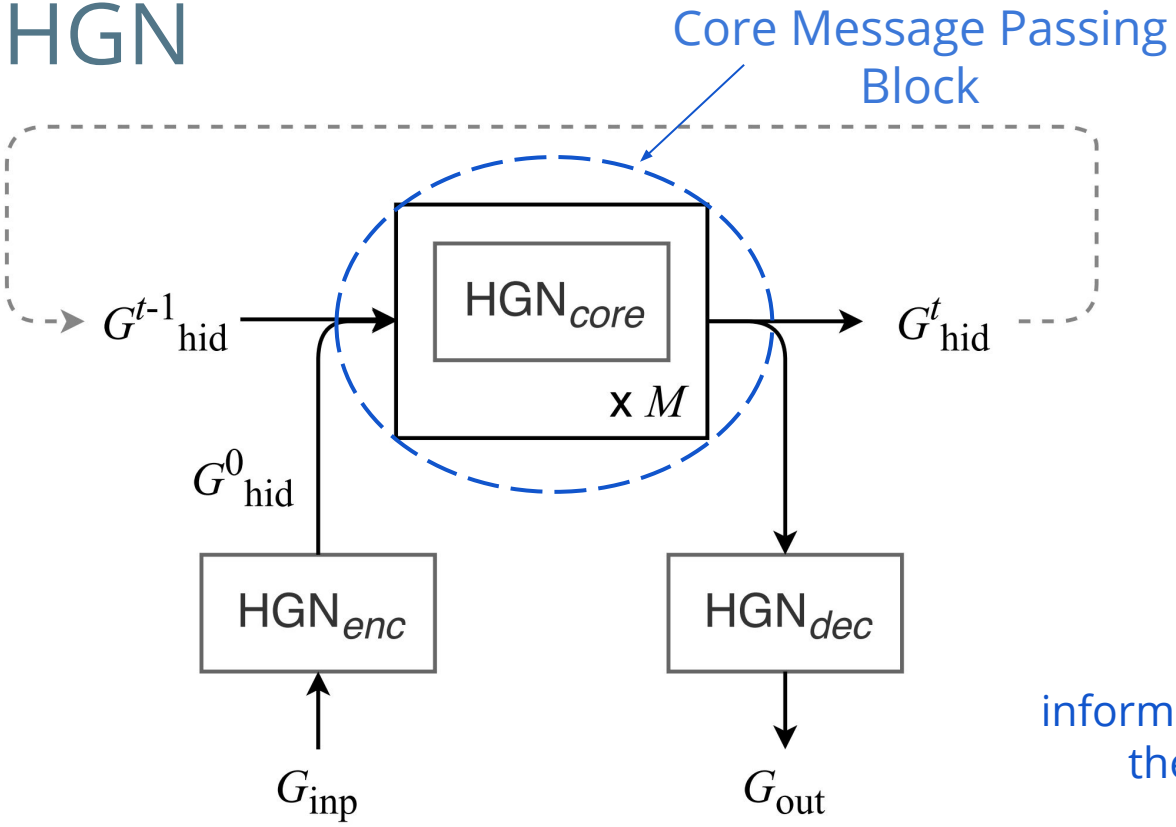
STRIPS-HGN


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

STRIPS-HGN

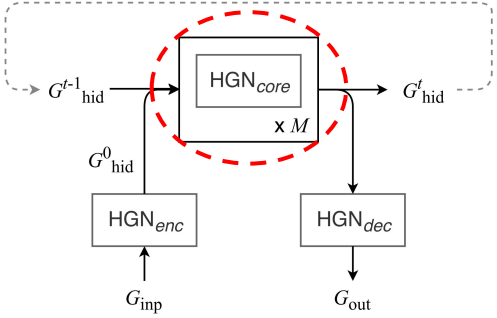

Recurrent Latent features
G^{t-1}_{hid}
HGN$_{core}$
x M
G^{t}_{hid}
Initial Latent features
G^{0}_{hid}
HGN$_{enc}$
HGN$_{dec}$
G_{inp}
G_{out}

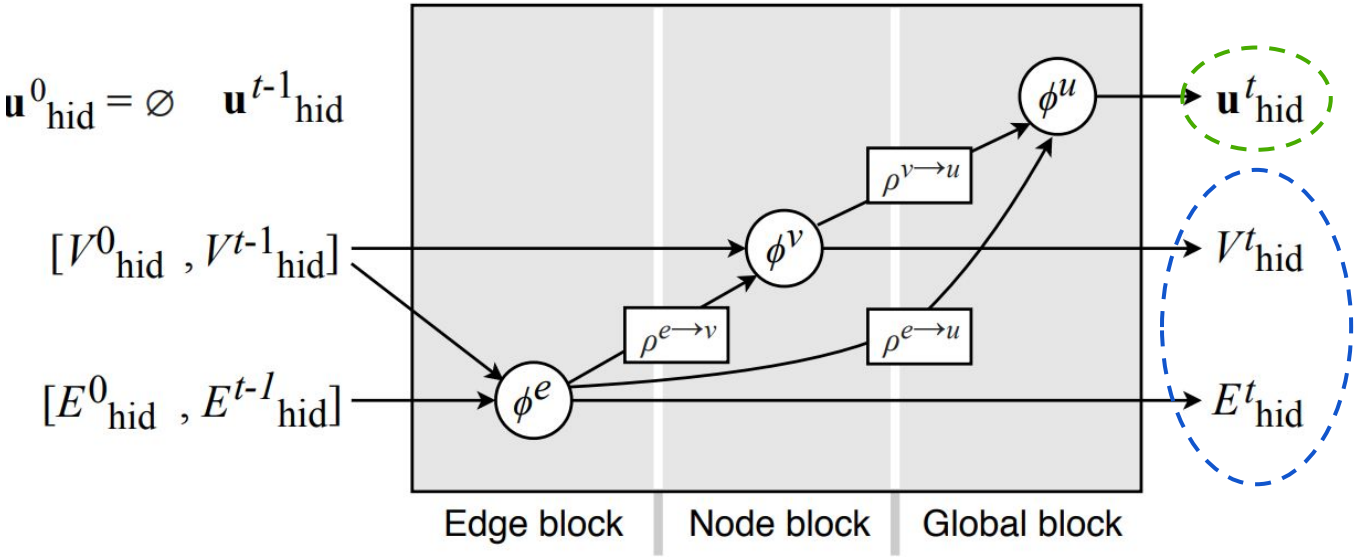
STRIPS-HGN



Propagates information through the hypergraph!

STRIPS-HGN Processing

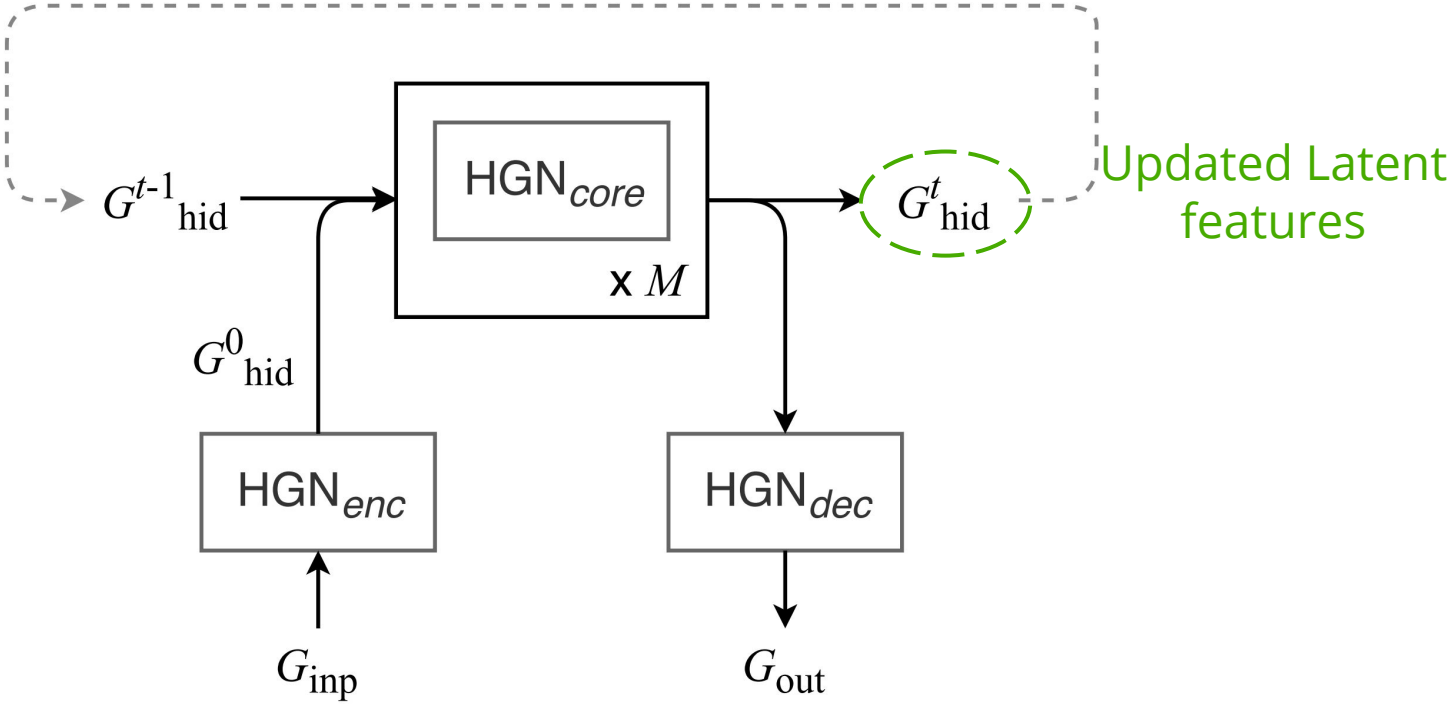




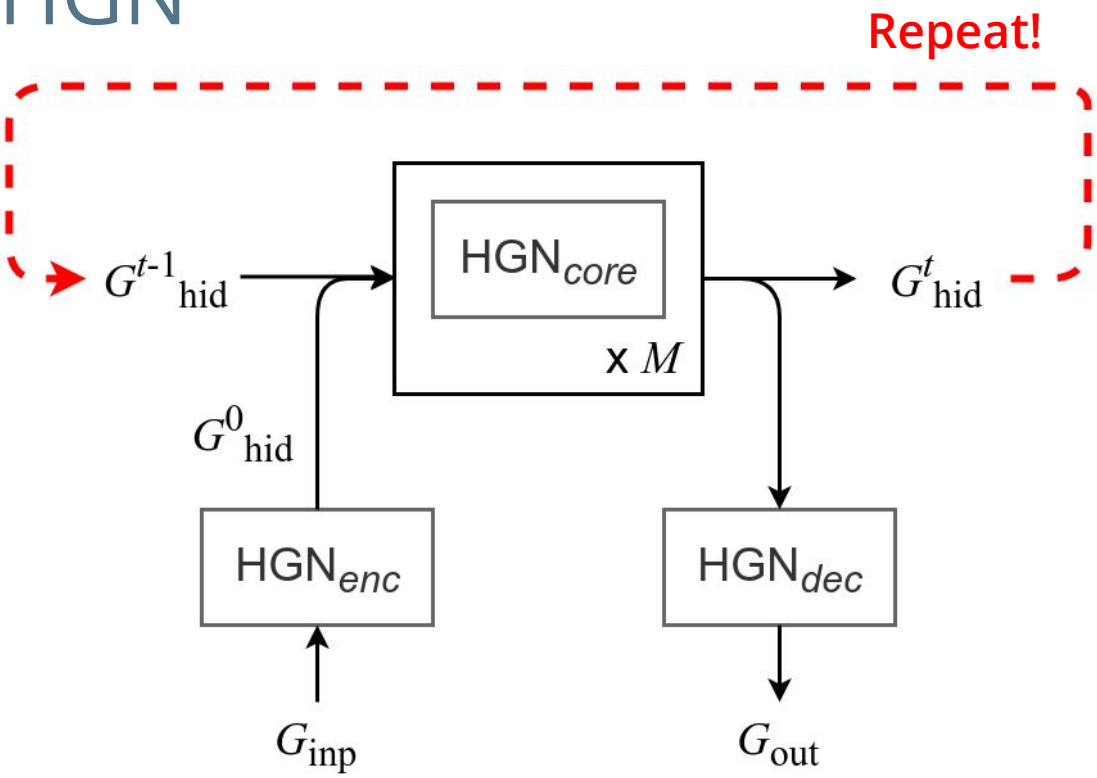
Latent heuristic value!

Updated proposition and action features

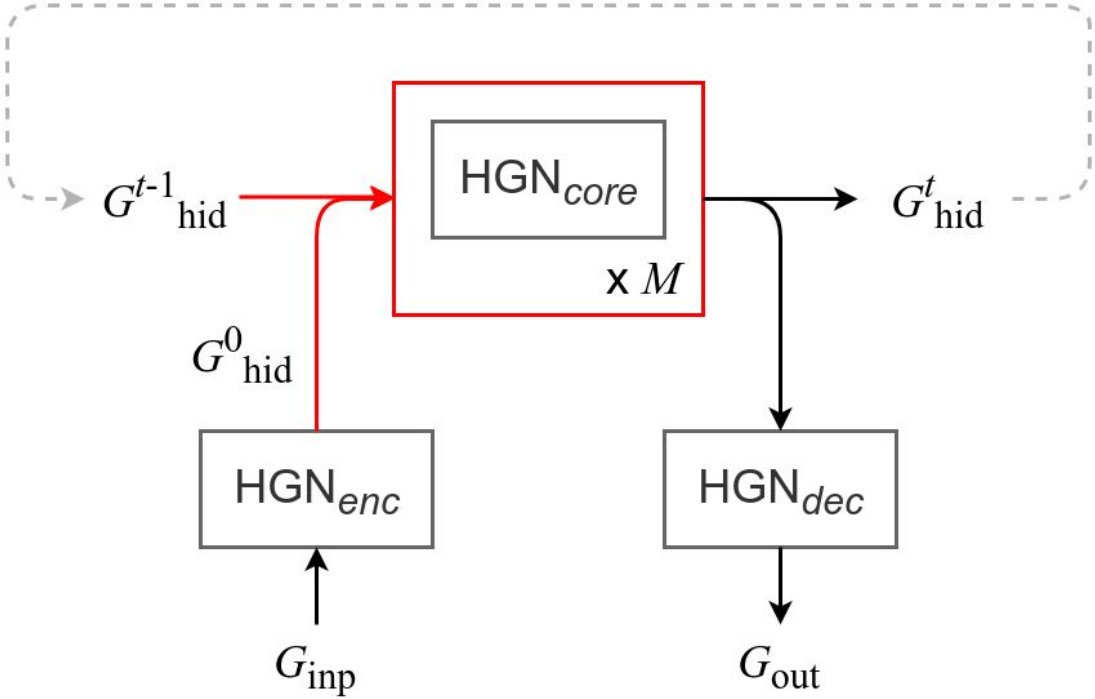
STRIPS-HGN


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

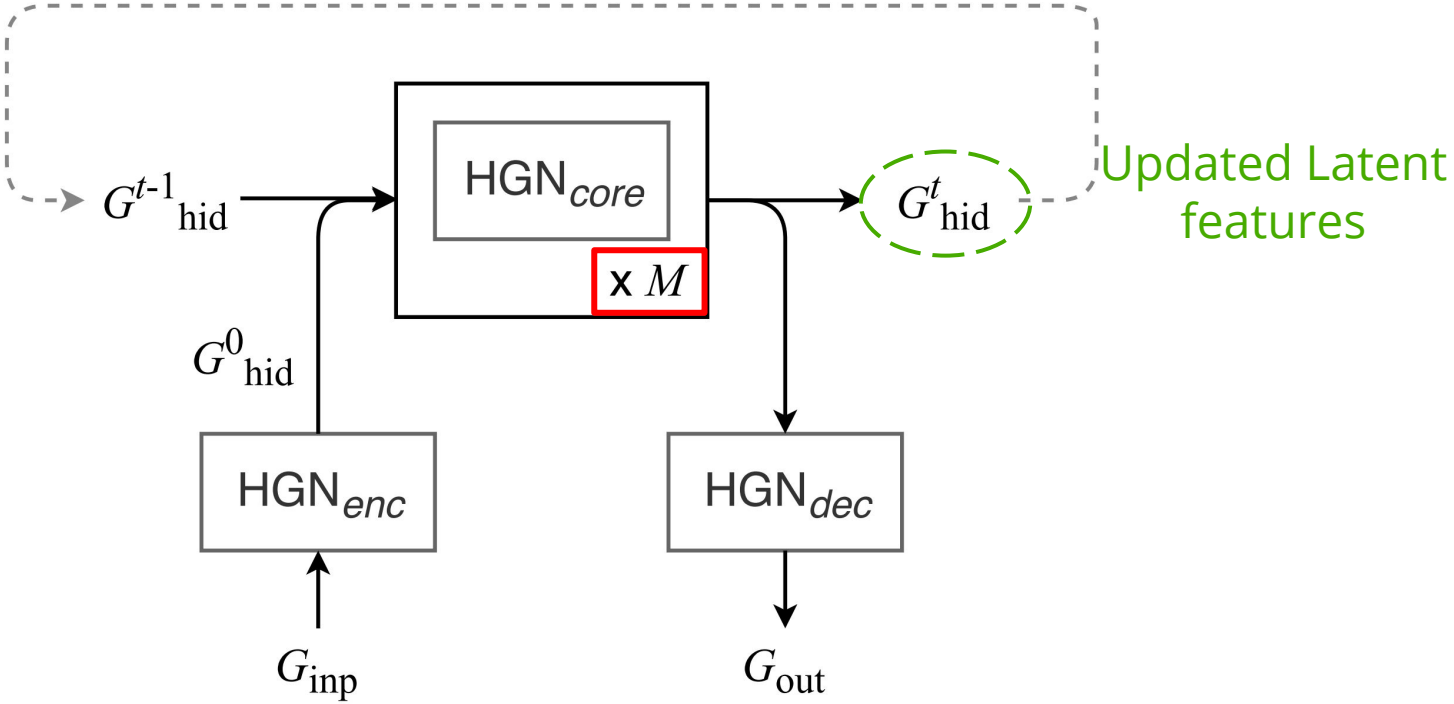
STRIPS-HGN


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

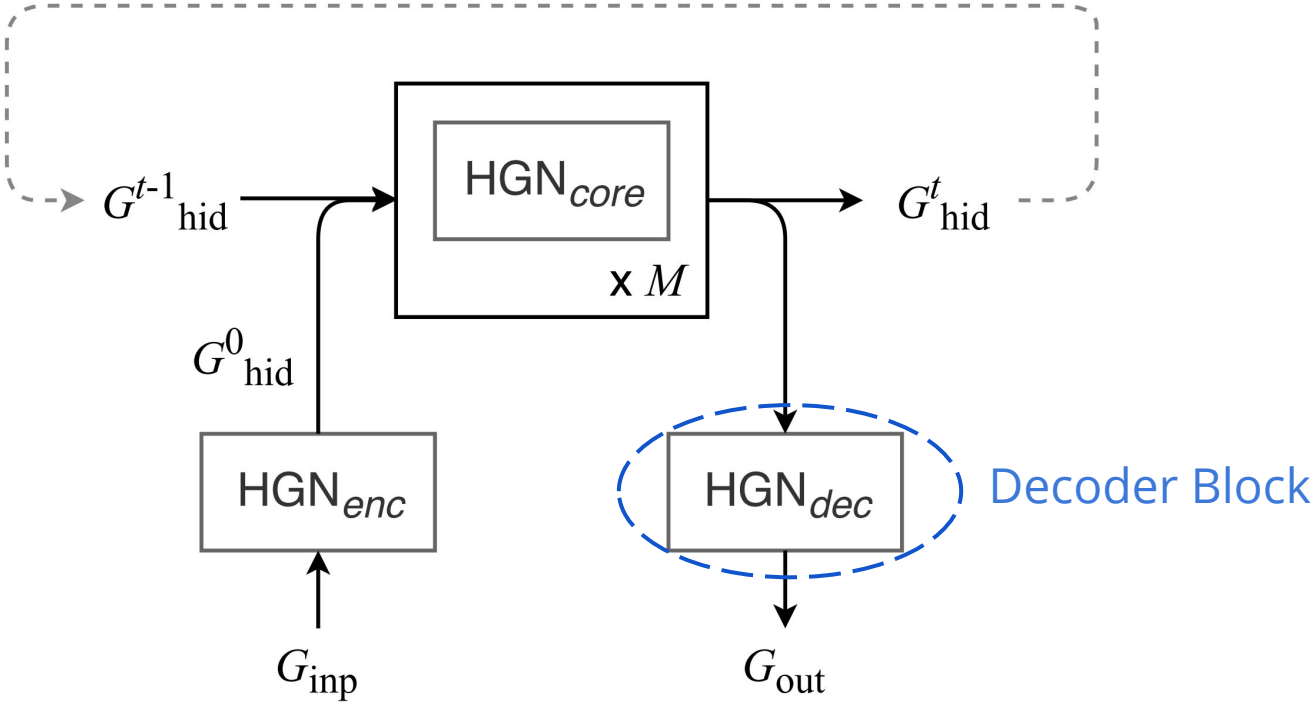
STRIPS-HGN


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

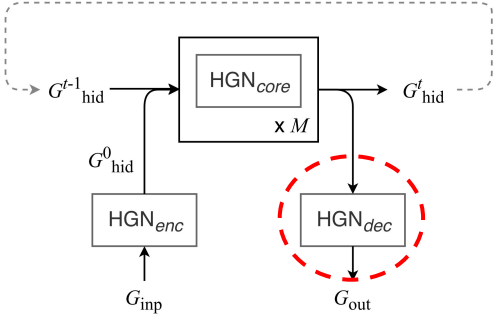
STRIPS-HGN

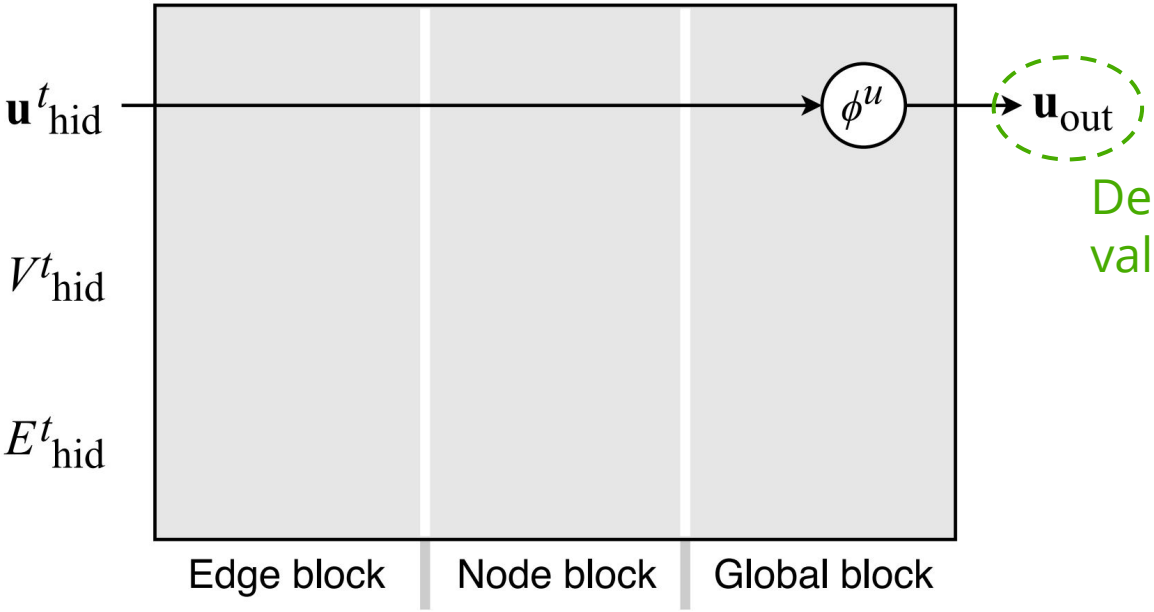

G^{t-1}_{hid}
HGN$_{core}$
x M
G^{t}_{hid}
Updated Latent features
G^{0}_{hid}
HGN$_{enc}$
HGN$_{dec}$
G_{inp}
G_{out}

STRIPS-HGN


G^{t-1}_{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** - whether proposition is in initial or goal state
  - Learning heuristics from scratch!
- **Generate Training Data**
  - Run an optimal planner for a set of problems
  - Use the states encountered in the optimal plans
  - Aim to learn the optimal heuristic value
- **Train using Gradient Descent**
  - Adam Optimiser with Mean Squared Error loss
  - Custom Stratified K-Fold with Binning

Experimental Results

- **Evaluate using A* Search**
- **Baseline Heuristics**
  - h^{add} (inadmissible), h^{max} and Landmark Cut (admissible)
- **STRIPS-HGN:** $h^{spatial}$
  - Run core block 10 times
  - Powerful generalisation but slower to compute

Experiments

Domain-Dependent Same size problems	**Domain-Dependent** Problems with different sizes
Domain-Independent Known domains	**Domain-Independent** Unknown domains

Experiments

Domain-Dependent 1 Same size problems	Domain-Dependent Problems with different sizes
Domain-Independent 2 Known domains	**Domain-Independent** 3 Unknown domains

8-puzzle

- Train on 10 random problems
- Evaluate on 50 random problems
- Training time: 100 minutes (10 minutes per network)
- **Learn a Problem-Size Dependent Heuristic!**

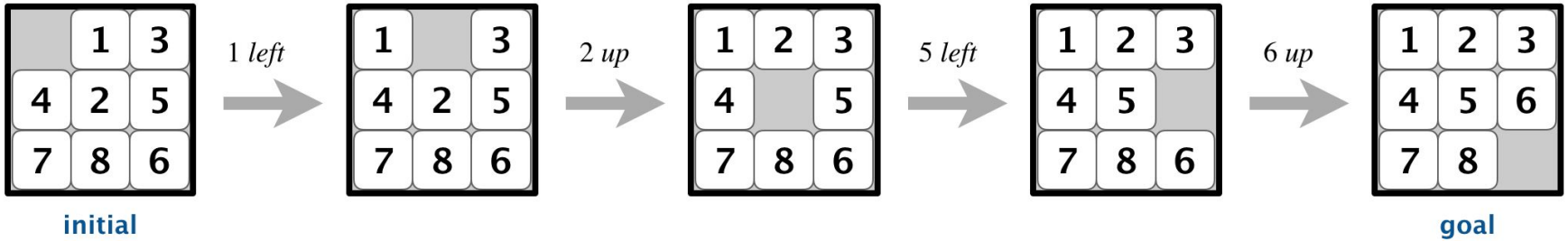
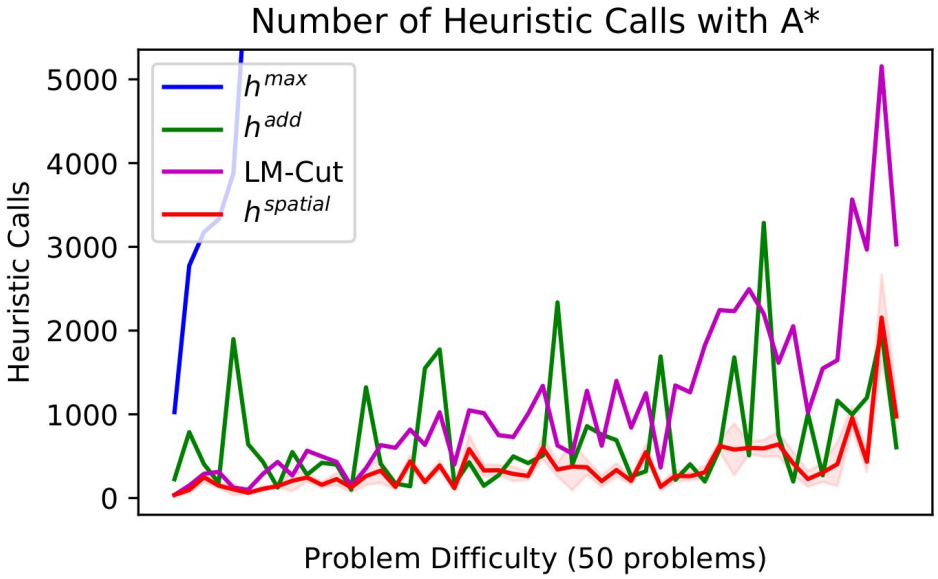
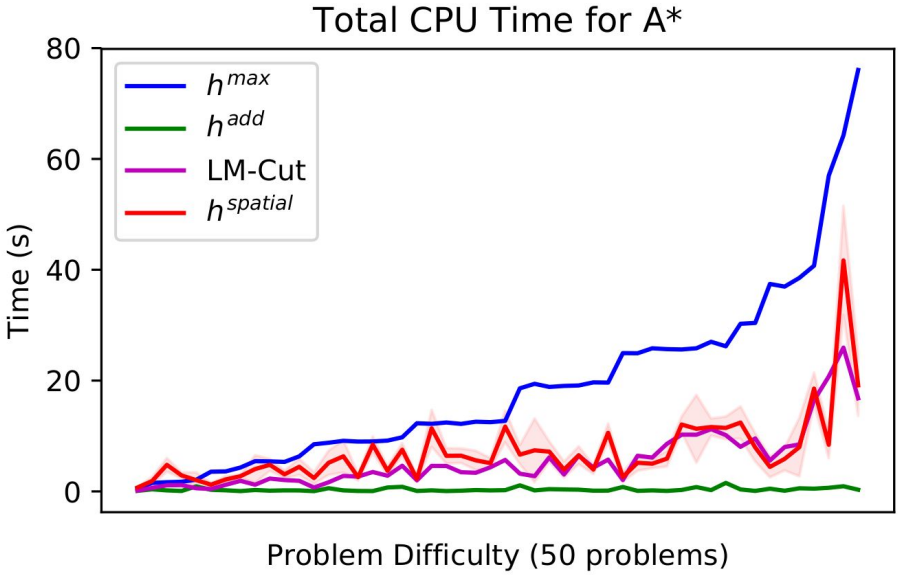


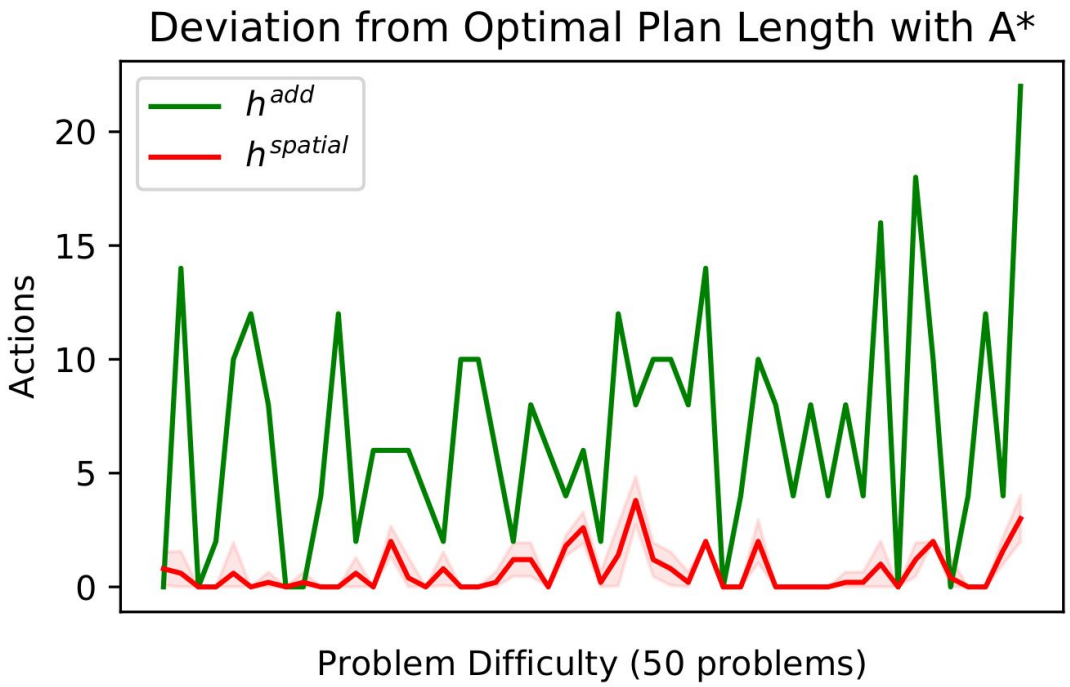
Figure from: https://www.cs.princeton.edu/courses/archive/spring18/cos226/assignments/8puzzle/index.html

Results - 8-puzzle





Results - 8-puzzle



Lower the better!

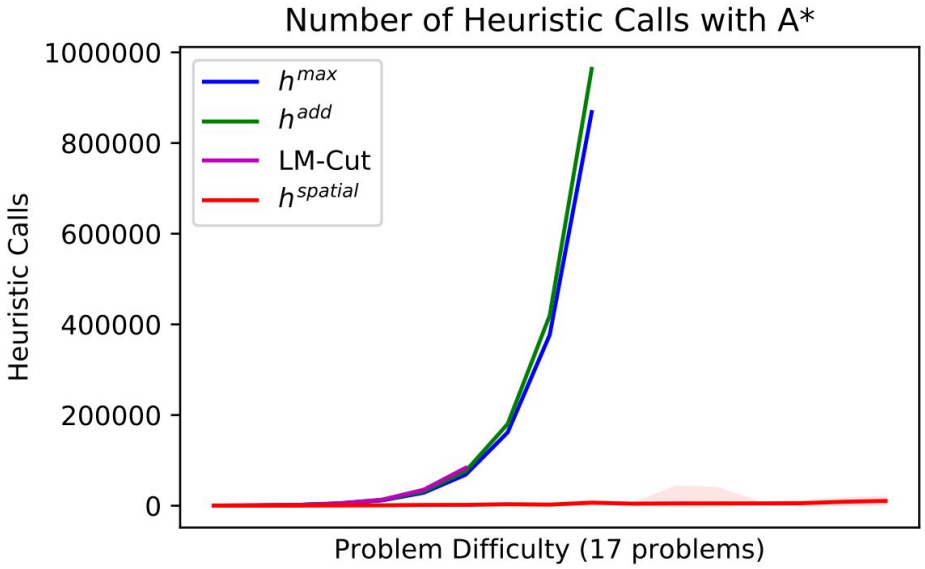
Training/Evaluating on Multiple Domains

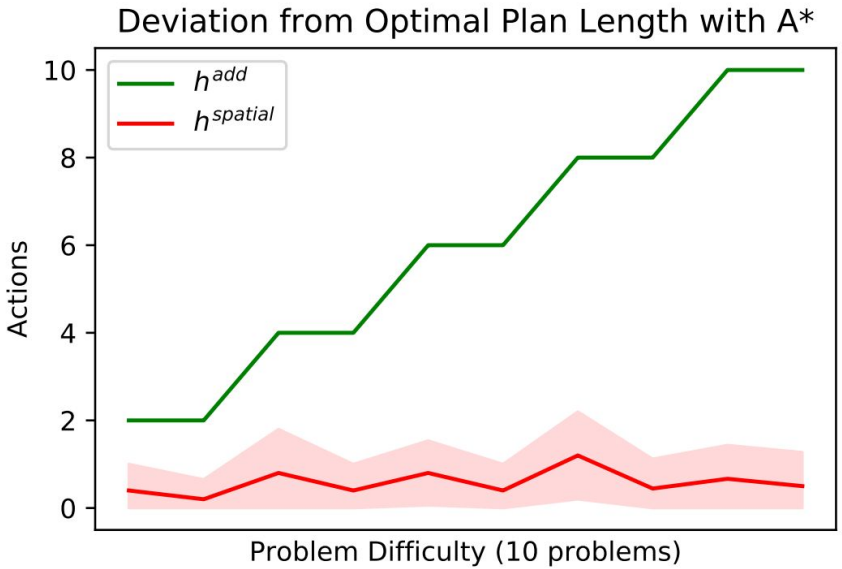
- Train and evaluate a <u>single network</u> on 3 domains
- **Training time:** 150 minutes (15 minutes per network)
- Learn a Domain-Independent Heuristic

Training and Testing

Blocksworld	Zenotravel	Gripper
10 Small Training Problems	10 Small Training Problems	3 Small Training Problems
100 Larger Testing Problems	60 Larger Testing Problems	17 Larger Testing Problems

Multi-Domain Gripper





Evaluating on Unseen Domains

Training

Zenotravel

10 small Training Problems

Gripper

3 small Training Problems

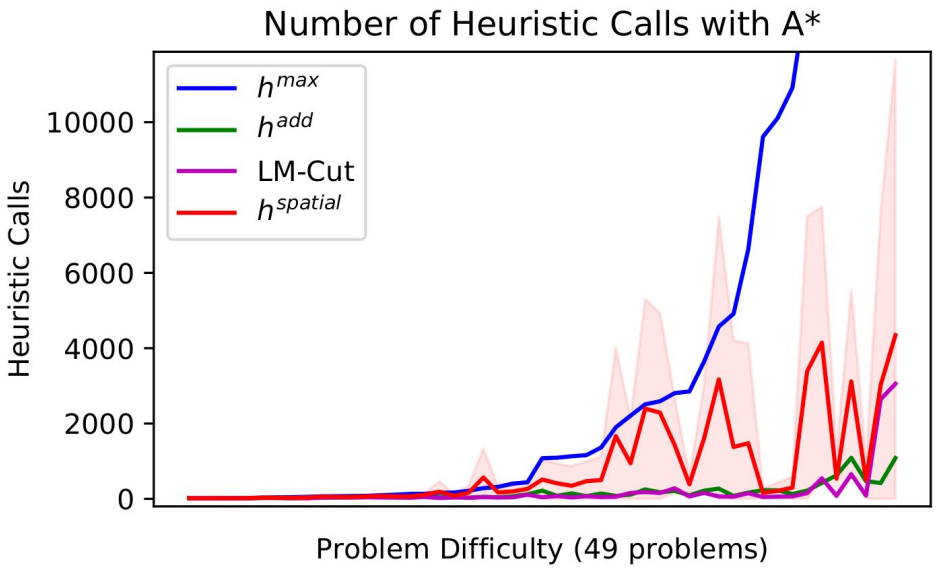
Testing

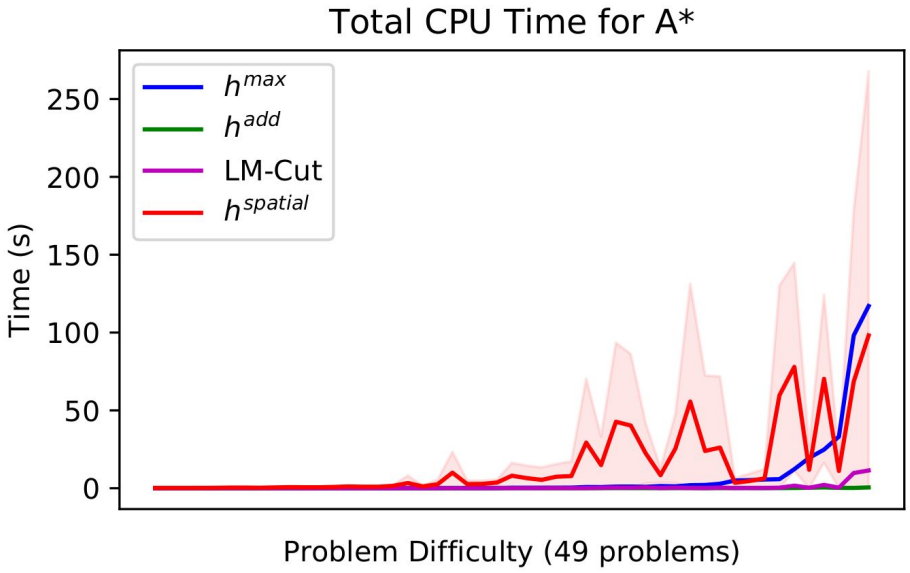
Blocksworld

50 Testing Problems (4-8 blocks)

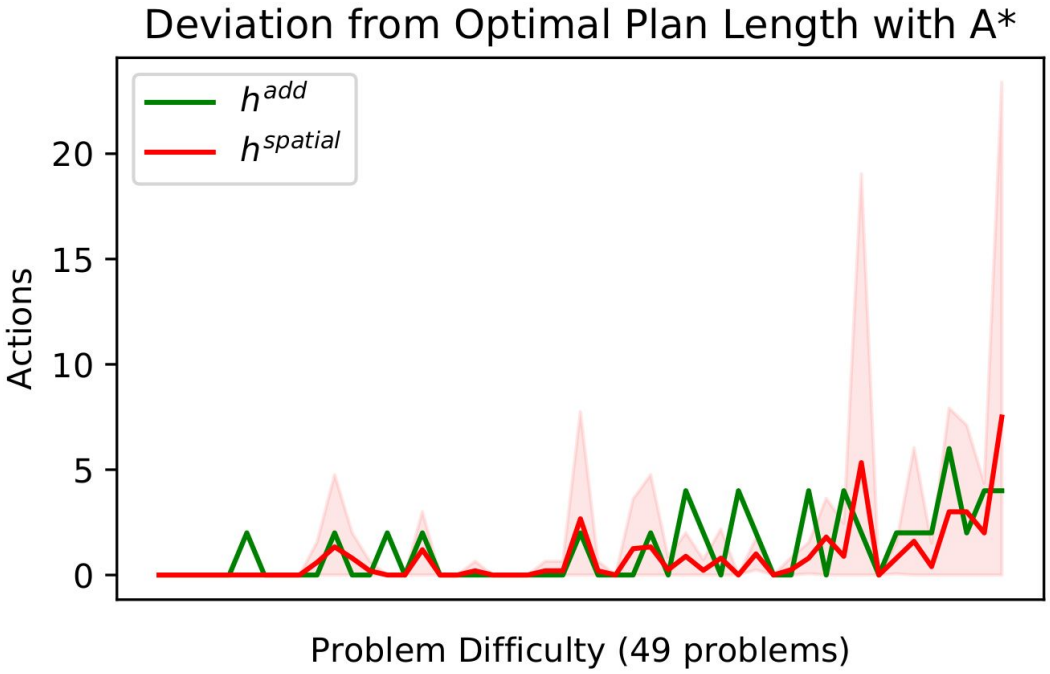
- **Training time:** 100 minutes (10 minutes per network)
- Learn a Domain-Independent Heuristic

Unseen Blocksworld





Unseen Blocksworld



Future Work

- **Using richer input features**
  - e.g. Derived from existing planning heuristics
- **Speeding up a STRIPS-HGN**
  - Currently limited by Hypergraph size → pruning?
  - Learn a policy instead - i.e. actions
- **Extend STRIPS-HGN to Probabilistic Planning**
  - Existing heuristics are either expensive to compute or use *determinisation*

Thanks

Any questions?

References

- Battaglia et al. 2018, Relational inductive biases, deep learning, and graph networks.
- Toyer, S.; Trevizan, F. W.; Thiébaux, S.; and Xie, L., 2019. ASNets: Deep Learning for Generalised Planning.
- Slide 3 Mars Exploration Rover https://www.nasa.gov/centers/ames/research/exploringtheuniverse/spiffy.html
- Slide 3 Elevator: https://pixabay.com/photos/elevators-lobby-entrance-1756630/