Learning More Expressive General Policies for Classical Planning Domains

Simon Ståhlberg[1] Blai Bonet[2] Hector Geffner[1]

[1]RWTH Aachen University, Germany

[2]Universitat Pompeu Fabra, Spain

Introduction

- General policies represent strategies for solving many planning instances
  - E.g., general policy for solving **all** Blocksworld problems

- Three main methods for learning such policies (no "synthesis" methods yet!)
  - **Combinatorial optimization using explicit pool of $\mathcal{C}_2$ features** obtained from domain predicates [B. *et al.*, 2019; Francès *et al.*, 2021]
    - Transparent, can be proved correct, trouble scaling up
  - **Deep learning (DL) using domain predicates but no explicit pool** [Toyer *et al.*, 2020; Garg *et al.*, 2020]
    - Opaque, complex, but scalable
  - **DL exploiting relation between $\mathcal{C}_2$ logic and GNNs** [Barceló *et al.*, 2020; Grohe, 2020; Ståhlberg *et al.*, 2022]
    - **R-GNN architecture** adapted from Max-CSP$[\Gamma]$ [Toenshoff *et al.*, 2021]
    - More transparent and simple, scalable
    - Supervised and non-supervised training
    - **Problem:** insufficient expressivity for generalized planning

In this Work

- Novel relational architecture R-GNN$[t]$, with parameter $t \geq 0$, that combines the R-GNN architecture with a parameterized encoding $A_t(S)$ of planning states S

- As t increases, the expressive power of R-GNN$[t]$ increases, approaching the full expressivity of $\mathcal{C}_3$ logic

- Significant improvements obtained even with $t = 1$, as shown in experiments

- 2- or 3-GNNs and Edge Transformers unfeasible in practice and limited to binary relations:
  - ▷ 2-GNNs: $\Theta(N^2)$ memory, $\Theta(N^3)$ time, $\mathcal{C}_2$ expressivity (yet see below)
  - ▷ 3-GNNs: $\Theta(N^3)$ memory, $\Theta(N^4)$ time, $\mathcal{C}_3$ expressivity
  - ▷ ETs: $\Theta(N^2)$ memory, $\Theta(N^3)$ time, $\mathcal{C}_3$ expressivity [Müller *et al.*, 2024]

Generalized Planning and First-Order STRIPS

- Generalized planning is about finding **general policies** that solve classes of planning problems

- Task is collection $\{P_1, P_2, P_3, \ldots\}$ of ground instances $P_i = \langle D, I_i \rangle$ over common **first-order STRIPS** domain D

- Each instance $P = \langle D, I \rangle$ consists of:
  - ▷ General (reusable) domain D specified with **action schemas** and **predicates**
  - ▷ Instance information I details **objects**, **init** and **goal** descriptions

Distinction between **general** domain D and **specific** instance $P = \langle D, I \rangle$ important for **reusing** action models, and also for **learning** them

Example (Input): 2-Gripper Problem $P = \langle D, I \rangle$ in PDDL

```
(define (domain gripper)
   (:requirements :typing)
   (:types        room ball gripper)
   (:constants    left right - gripper)
   (:predicates   (at-robot ?r - room) (at ?b - ball ?r - room)
                  (free ?g - gripper)  (carry ?o - ball ?g - gripper))

   (:action MOVE
       :parameters    (?from ?to - room)
       :precondition  (at-robot ?from)
       :effect        (and (at-robot ?to) (not (at-robot ?from))))
   (:action PICK
       :parameters    (?obj - ball ?room - room ?gripper - gripper)
       :precondition  (and (at ?obj ?room) (at-robot ?room) (free ?gripper))
       :effect        (and (carry ?obj ?gripper) (not (at ?obj ?room)) (not (free ?gripper))))
   (:action DROP
       :parameters    (?obj - ball ?room - room ?gripper - gripper)
       :precondition  (and (carry ?obj ?gripper) (at-robot ?room))
       :effect        (and (at ?obj ?room) (free ?gripper) (not (carry ?obj ?gripper)))))

(define (problem easy-2balls)
   (:domain   gripper)
   (:objects  roomA roomB - room  B1 B2 - ball)
   (:init     (at-robot roomA) (free left) (free right) (at B1 roomA) (at B2 roomA))
   (:goal     (and (at B1 roomB) (at B2 roomB))))
```

Relational GNN Architecture for Planning [Ståhlberg et al., 2022-2024]

- Planning state S over STRIPS domain D is a **relational structure:**
  - ▷ Relational symbols given by predicates in D; **shared** by all such states S
  - ▷ Denotation of predicate p given by ground atoms $p(\bar{o})$ true at S
- Adapt architecture of [Toenshoff *et al.*, 2021] for handling relational structures

Relational GNN (R-GNN) Architecture

Input: Set of ground atoms S (state), and objects O
Output: Embeddings $\boldsymbol{f}_L(o)$ for each object $o \in O$

```
Initialize f_0(o) = 0^k for each object o ∈ O
for i ∈ {0, 1, ..., L − 1} do
    for each atom q = p(o_1, o_2, ..., o_m) ∈ S do
        m_{q,o_j} := [MLP_p(f_i(o_1), f_i(o_2), ..., f_i(o_m))]_j
    end for
    for each object o ∈ O do
        f_{i+1}(o) := f_i(o) + MLP_U(f_i(o), agg({{m_{q,o} | o ∈ q, q ∈ S}}))
    end for
end for
```

Parameters: embedding dimension k, rounds L, $\{\mathbf{MLP}_p : p \in D\}$, $\mathbf{MLP}_U$, and aggregator

Ståhlberg, Bonet, and Geffner. Learning More Expressive General Policies for Classical Planning Domains. AAAI-2025.

Final Readout, Value Functions, and Greedy Policies

- **Final readout** is **additive readout** that feeds into final MLP:

$$V(S) \; = \; \mathbf{MLP}\big(\textstyle\sum_{o \in O} \boldsymbol{f}_L(o)\big)$$

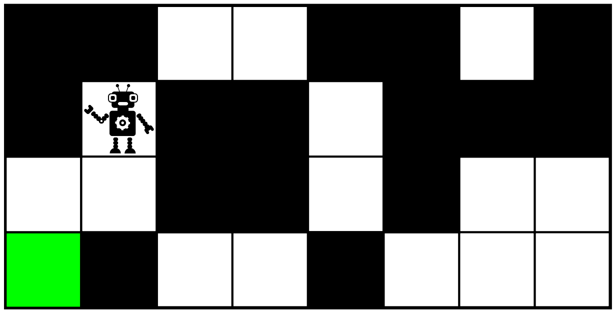
- Training minimize **loss** $L(S) = |V^*(S) - V(S)|$ given by **optimal value function** $V^*(\cdot)$ for small tasks in training set

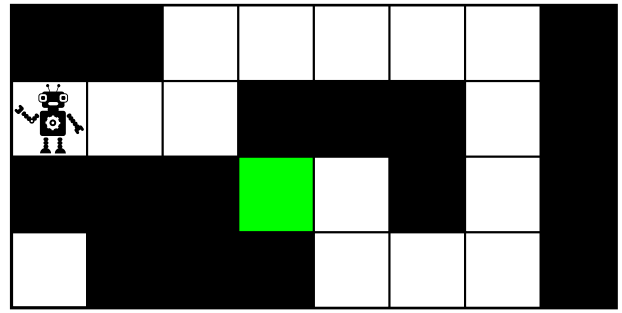
- **Greedy policy** $\pi_V(S)$ chooses action $a = \mathrm{argmin}_{a \in A(S)} \, 1 + V(S_a)$:
  - ▷ If $V(S) = 0$ for goals, and $V(S) = 1 + \min_a V(S_a)$ for non-goals, π_V is **optimal**
  - ▷ If $V(S) = 0$ for goals, and $V(S) \geq 1 + \min_a V(S_a)$ for non-goals, π_V **solves any state** S

  where S_a is result of applying action a in state S

Successful approach for GP, but subject to expressivity of GNNs...

Example: Navigation With XY Coordinates





- Navigation in rectangular grid with decoupled coordinates: cells and blocked cells with $\text{CELL}(x, y)$ and $\text{BLOCKED}(x, y)$, position with $\text{AT}(x, y)$, and $\text{ADJ}(i, i + 1)$

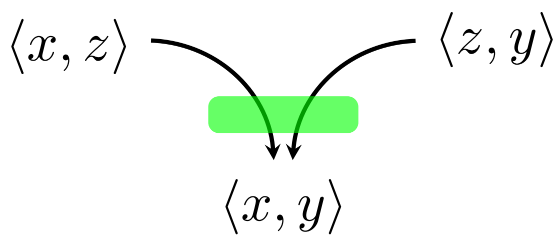
- For computing goal distances (ie V^*), cells (x, y) must "communicate" with neighbors (x, y') and (x', y). In the plain R-GNN, there must be atoms involving $\{x, y, y'\}$ (similarly, $\{x, x', y\}$). No such atoms exists in state S

Expressivity of GNNs

- R-GNNs are instances of (1-)GNNs over undirected graphs
- GNNs compute invariant (resp. equivariant) funcs on graphs (resp. vertices)
- Well-understood **expressivity limitations** in terms of **Weisfeiler-Leman** colorings and **$\mathcal{C}_2$ logic** (formulas with counting quantifiers, and at most 2 variables)
- Eg, join $W(x,y) = \exists z.[R(x,z) \wedge T(z,y)]$ of relations R and T cannot be captured!
- That is, no GNN can "track" such implicit relation $W(x,y)$ on a graph where red and blue edges stand for R and T respectively
- Can augment expressivity with k-GNNs, $k > 1$, that embed k-tuples of vertices:
  - ▷ Expressivity characterized in terms of k-WL colorings
  - ▷ Either k-OWL (less poweful) or k-FWL (more powerful) versions
  - ▷ Related to, respectively, $\mathcal{C}_{k-1}$ and $\mathcal{C}_k$ logics: counting quant., k variables
  - ▷ Infeasible by num. objs. in planning problems: $\Theta(N^k)/\Theta(N^{k+1})$ space/time

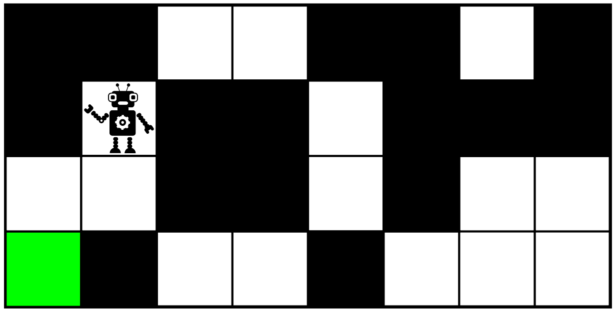
Parametric R-GNN$[t]$ Architecture

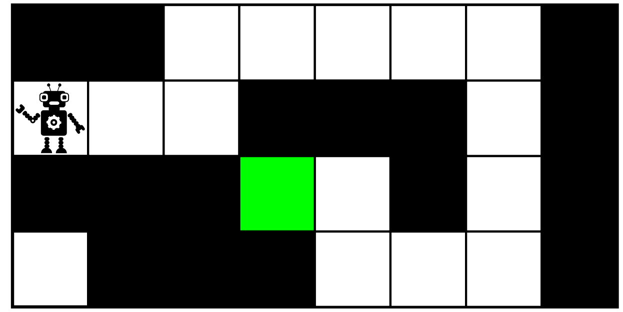
- Same R-GNN architecture, different encoding of planning states S
- Embedding of all objects pairs, like in 2-GNNs: $\Theta(N^2)$ space
  - ▷ Objects in atoms replaced by pairs: $p(a,b) \to p(\langle a,a\rangle, \langle a,b\rangle, \langle b,a\rangle, \langle b,b\rangle)$
  - ▷ Predicate arities expanded from k to k^2
- New composition predicate $\Delta(\langle x,z\rangle, \langle z,y\rangle, \langle x,y\rangle)$:



  - ▷ Set $A_t(S)$ of added Δ-atoms controlled by integer parameter $t \geq 0$
  - ▷ $A_0(S) = \{p(\langle w\rangle^2) \mid p(w) \in S\}$ for $\langle w\rangle^2 = \langle (o_1,o_1), \ldots, (o_i,o_j), \ldots, (o_m,o_m)\rangle$
  - ▷ $A_t(S) = A_0(S) \cup \{\Delta(\langle o,o'\rangle, \langle o',o''\rangle, \langle o,o''\rangle) \mid \langle o,o'\rangle, \langle o',o''\rangle \in R_t\}$
  - ▷ $\langle o,o'\rangle \in R_t$ iff o and o' in some atom in S ($t{=}1$), or $\exists o''[\langle o,o''\rangle, \langle o'',o'\rangle \in R_{t-1}]$ ($t{>}1$)
- R-GNN$[t](S,O) =$ R-GNN$(A_t(S), O^2)$
- **Final readout:** $V(S) = \text{MLP}\big(\sum_{o\in O} \boldsymbol{f}_L(o,o)\big)$ aggregates $|O|$ embeddings

Example: Navigation With XY Coordinates





- Navigation in rectangular grid with decoupled coordinates: cells and blocked cells with $\textsc{cell}(x, y)$ and $\textsc{blocked}(x, y)$, position with $\textsc{at}(x, y)$, and $\textsc{adj}(i, i + 1)$

- After 12 hours of training on 105 random $n \times m$ instances, $mn < 30$, greedy policies achieve coverages of 59.72%, 80.55%, and 100% for R-GNN, R-GNN[0], and R-GNN[1] on instances with **different sets of blocked cells and $nm \leq 32$**

- For computing goal distances (ie V^*), cells (x, y) must "communicate" with neighbors (x, y') and (x', y). In the plain R-GNN, there must be atoms involving $\{x, y, y'\}$ (similarly, $\{x, x', y\}$). No such atoms exists in S, except in R-GNN[t] where $A_t(S)$ includes $\Delta(\langle x, x'\rangle, \langle x', y\rangle, \langle x, y\rangle)$ and $\Delta(\langle x, y\rangle, \langle y, y'\rangle, \langle x, y'\rangle)$

Experiments: Setup

- A learned value function V for domain D defines a **general policy** π_V that at state S selects an unvisited successor state S' with lowest $V(S')$ value

- We implemented in PyTorch, and trained on Nvidia A10s with 24Gb of memory over 12 hours, using Adam, lr=0.0002, batches of size 16, and no regularization. Embedding dimension of $k = 64$, and $L = 30$ layers were used.

- Standard benchmarks from **International Planning Competition (IPC)**

- For each domain and architecture, 3 models were trained, and best model on validation was selected.

- **Baselines:**
  - ▷ Edge Transformer (ET) [Bergen *et al.*, 2021] designed to do triangulations on graphs
  - ▷ R-GNN$_2$ that adds all Δ atoms
  - ▷ 2-GNNs that emulates 2-OWL which captures $\mathcal{C}_3$

Experiments: Results

Domain	Model	Coverage (%)	Plan Length Total	Median	Mean
Blocks-s	R-GNN	**17 / 17 (100 %)**	674	38	39
	R-GNN[0]	**17 / 17 (100 %)**	670	36	39
	R-GNN[1]	**17 / 17 (100 %)**	684	36	40
	R-GNN$_2$	14 / 17 (82 %)	922	35	65
	2-GNN	**17 / 17 (100 %)**	678	36	39
	ET	16 / 17 (94 %)	826	38	51
Blocks-m	R-GNN	**22 / 22 (100 %)**	868	40	39
	R-GNN[0]	**22 / 22 (100 %)**	830	39	37
	R-GNN[1]	**22 / 22 (100 %)**	834	39	37
	R-GNN$_2$	**22 / 22 (100 %)**	936	39	42
	2-GNN	20 / 22 (91 %)	750	40	37
	ET	18 / 22 (82 %)	966	39	53
Gripper	R-GNN	**18 / 18 (100 %)**	4,800	231	266
	R-GNN[0]	**18 / 18 (100 %)**	1,764	98	98
	R-GNN[1]	11 / 18 (61 %)	847	77	77
	R-GNN$_2$	**18 / 18 (100 %)**	1,764	98	98
	2-GNN	1 / 18 (6 %)	53	53	53
	ET	4 / 18 (22 %)	246	61	61
Miconic	R-GNN	**20 / 20 (100 %)**	1,342	67	67
	R-GNN[0]	**20 / 20 (100 %)**	1,566	71	78
	R-GNN[1]	**20 / 20 (100 %)**	2,576	71	128
	R-GNN$_2$	**20 / 20 (100 %)**	1,342	67	67
	2-GNN	12 / 20 (60 %)	649	54.5	54
	ET	**20 / 20 (100 %)**	1,368	68	68
Visitall	R-GNN	18 / 22 (82 %)	636	29	35
	R-GNN[0]	21 / 22 (95 %)	1,128	35	53
	R-GNN[1]	**22 / 22 (100 %)**	886	35	40
	R-GNN$_2$	20 / 22 (91 %)	739	33	36
	2-GNN	18 / 22 (82 %)	626	32	34
	ET	18 / 22 (82 %)	670	29	37

Domain	Model	Coverage (%)	Plan Length Total	Median	Mean
Grid	R-GNN	9 / 20 (45 %)	109	11	12
	R-GNN[0]	12 / 20 (60 %)	177	11	14
	R-GNN[1]	**15 / 20 (75 %)**	209	13	13
	R-GNN$_2$	10 / 20 (50 %)	124	11.5	12
	2-GNN	6 / 20 (30 %)	82	11.5	13
	ET	1 / 20 (5 %)	15	15	15
Logistics	R-GNN	10 / 20 (50 %)	510	51	51
	R-GNN[0]	9 / 20 (45 %)	439	48	48
	R-GNN[1]	**20 / 20 (100 %)**	1,057	52	52
	R-GNN$_2$	15 / 20 (75 %)	799	52	53
	2-GNN	0 / 20 (0 %)	–	–	–
	ET	0 / 20 (0 %)	–	–	–
Rovers	R-GNN	9 / 20 (45 %)	2,599	280	288
	R-GNN[0]	**14 / 20 (70 %)**	2,418	153	172
	R-GNN[1]	**14 / 20 (70 %)**	1,654	55	118
	R-GNN$_2$	11 / 20 (55 %)	2,225	239	202
	2-GNN	Unsuitable domain: ternary predicates			
	ET				
Vacuum	R-GNN	**20 / 20 (100 %)**	4,317	141	215
	R-GNN[0]	**20 / 20 (100 %)**	183	9	9
	R-GNN[1]	**20 / 20 (100 %)**	192	9	9
	R-GNN$_2$	**20 / 20 (100 %)**	226	9	11
	2-GNN	Unsuitable domain: ternary predicates			
	ET				
Visitall-xy	R-GNN	5 / 20 (25 %)	893	166	178
	R-GNN[0]	15 / 20 (75 %)	1,461	84	97
	R-GNN[1]	**20 / 20 (100 %)**	1,829	83	91
	R-GNN$_2$	19 / 20 (95 %)	2,428	116	127
	2-GNN	12 / 20 (60 %)	1,435	115	119
	ET	3 / 20 (15 %)	455	138	151

Experiments: #Objects in Training / Validation, and Test Sets

Domain	Training / Validation	Test
Blocks	4–9	10–20
Gripper	2–14	16–50
Logistics	2–5 / 3–5	15–19 / 8–11
Miconic	2–20 / 1–10	11–30 / 22–60
Rovers	2–3 / 3–8	3 / 21–39
Vaccum	8–38 / 11–6	40–93 / 6–10
Visitall	1–21	100

Expressivity of the R-GNN$[t]$ Architecture

- The architecture R-GNN$[t]$ has the capability to capture compositions of binary relations that can be expressed in $\mathcal{C}_3$

Definition ($\mathcal{C}_3$-Joins). *Let σ be relational language. The class $\mathcal{J}_3 = \mathcal{J}_3[\sigma]$ of relational joins is the* **smallest class** *of formulas that satisfies:*

1. *$\{R(x,y), \neg R(x,y)\} \subseteq \mathcal{J}_3$ for relation R in σ,*
2. *$\mathcal{J}_3$ is closed under conjunctions and disjunctions, and*
3. *$\exists y[\phi(x,y) \wedge \phi(y,z)] \in \mathcal{J}_3$ if $\{\phi(x,y), \phi(y,z)\} \subseteq \mathcal{J}_3$.*

Notation $\phi(x,y)$: ϕ is a formula whose free variables are among $\{x,y\}$

Theorem. *Let σ be relational language, and let $\mathcal{D} \subseteq \mathcal{J}_3$ be* **finite collection** *of $\mathcal{C}_3$-joins. There is parameter $\langle t, k, L\rangle$, where k is embedding dimension and L is number of layers, and network N in* R-GNN$[\sigma, t, k, L]$ *that* **computes** $\mathcal{D}$

Conclusions

- Novel parametric architecture R-GNN[t] that **provably** increases the expressivity of the relational R-GNN architecture
- R-GNN[t] embeds all pair of objects but does a bounded number of triangulations, determined by the value of parameter $t \geq 0$
- In benchmarks, a small value of $t = 1$ achieves best results
- Other ways to increase expressivity, like k-GNNs for $k \geq 2$, in either the OWL or FWL setting are infeasible in practice due to high number of objects: $\Theta(N^k)$ space, $\Theta(N^{k+1})$ time
- **Future work:** consider use of indexicals/markers that can be moved around as an alternative to increase the expressivity in GNN architectures for planning

References

[Bajpai et al., 2018] Bajpai, A. N., Garg, S., et al. (2018). Transfer of deep reactive policies for mdp planning. In *Advances in Neural Information Processing Systems*, pages 10965–10975.

[Barceló et al., 2020] Barceló, P., Kostylev, E., Monet, M., Pérez, J., Reutter, J., and Silva, J.-P. (2020). The logical expressiveness of graph neural networks. In *Proc. of the 8th International Conference on Learning Representations (ICLR 2020)*. OpenReview.net.

[Bergen et al., 2021] Bergen, L., O'Donnell, T., and Bahdanau, D. (2021). Systematic generalization with edge transformers. *Advances in Neural Information Processing Systems*, 34:1390–1402.

[Bonet et al., 2019] Bonet, B., Francès, G., and Geffner, H. (2019). Learning features and abstract actions for computing generalized plans. In *Proc. of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, pages 2703–2710. AAAI Press.

[Frances et al., 2021] Frances, G., Bonet, B., and Geffner, H. (2021). Learning general planning policies from small examples without supervision. In *Proc. AAAI*, pages 11801–11808.

[Grohe, 2021] Grohe, M. (2021). The logic of graph neural networks. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–17.

[Müller et al., 2024] Müller, L., Kusuma, D., Bonet, B., and Morris, C. (2024). Towards principled graph transformers. In *Proc. of the 38th Annual Conference on Neural Information Processing Systems (NeurIPS 2024)*.

[Ståhlberg et al., 2022a] Ståhlberg, S., Bonet, B., and Geffner, H. (2022a). Learning general optimal policies with graph neural networks: Expressive power, transparency, and limits. In *Proc. ICAPS*, pages 629–637.

[Ståhlberg et al., 2022b] Ståhlberg, S., Bonet, B., and Geffner, H. (2022b). Learning generalized policies without supervision using GNNs. In *Proc. KR*, pages 474–483.

[Ståhlberg et al., 2023] Ståhlberg, S., Bonet, B., and Geffner, H. (2023). Learning general policies with policy gradient methods. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 647–657.

[Toenshoff et al., 2021] Toenshoff, J., Ritzert, M., Wolf, H., and Grohe, M. (2021). Graph neural networks for maximum constraint satisfaction. *Frontiers in Artificial Intelligence and Applications*, 3:580607.

[Toyer et al., 2018] Toyer, S., Trevizan, F., Thiébaux, S., and Xie, L. (2018). Action schema networks: Generalised policies with deep learning. In *AAAI*.