Graph Learning for Planning:
The Story Thus Far and Open Challenges

Dillon Z. Chen Mingyu Hao Sylvie Thiébaux Felipe Trevizan

LAAS CNRS

GenPlan@AAAI-2025
Philadelphia, PA, USA

Australian National University

Generalisation in Planning

- we want agents that can plan over diverse settings, i.e. generalisation
- one form of generalisation is over problem size*:
    - train on small instances
    - deploy on larger instances

*For other forms of generalisation see
Dillon Z. Chen, Pulkit Verma, Siddharth Srivastava, Michael Katz, Sylvie Thiébaux. *AI Planning: A Primer and Survey (Preliminary Report).* PRL@AAI 2025.

Why Graph Learning for Planning?

- very popular
    - large body of theoretical and empirical results
    - cannot ignore

- handles arbitrarily sized, relational inputs
    - i.e. planning tasks
    - Q: may be able to reason?

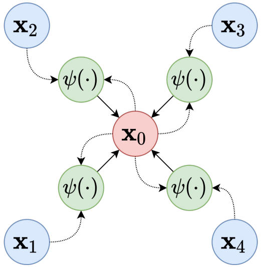
Top 50 keywords

Keyword	Count
Large Language Models	318
Reinforcement Learning	201
Graph Neural Networks	123
Diffusion Models	112
Deep Learning	110

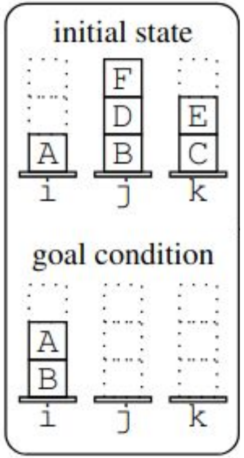
https://github.com/ANLGBOY/ICLR-2024-OpenReview-Ratings

https://github.com/EdisonLeeeee/ICLR2023-OpenReviewData

https://github.com/EdisonLeeeee/ICLR2022-OpenReviewData



Typical Graph Learning for Planning Pipeline



(1) convert planning instance to graph

(2) use graph learning model to embed graph, e.g. GNN

(3) optimisation to learn weights, e.g. learn heuristic function, policies

3 Key Insights

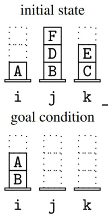
So many models and architectures

some (not all) graph learning architectures for planning:

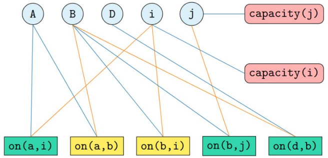
Model/Graph representation	Original intended use (and reference)
ASNets	policy (Toyer et al., 2018, 2020)
STRIPS-HGN	domain-indep. heuristic (Shen et al., 2020)
PLOI	object importance (Silver et al., 2021)
Muninn	value function (Ståhlberg et al., 2022)
GOOSE(SLG)	domain-indep. heuristic (Chen et al., 2024a)
GOOSE(FLG)	domain-indep. heuristic (Chen et al., 2024a)
GOOSE(LLG)	domain-indep. heuristic (Chen et al., 2024a)
GOOSE(ILG)	heuristic (Chen et al., 2024b)
O, A, OA, OP	expressivity checking (Horčík & Šír, 2024)

- how to fairly compare them?
- each has their pros and cons based on usage
- is it still possible to achieve a unified understanding?

graph learning architecture



convert task to graph

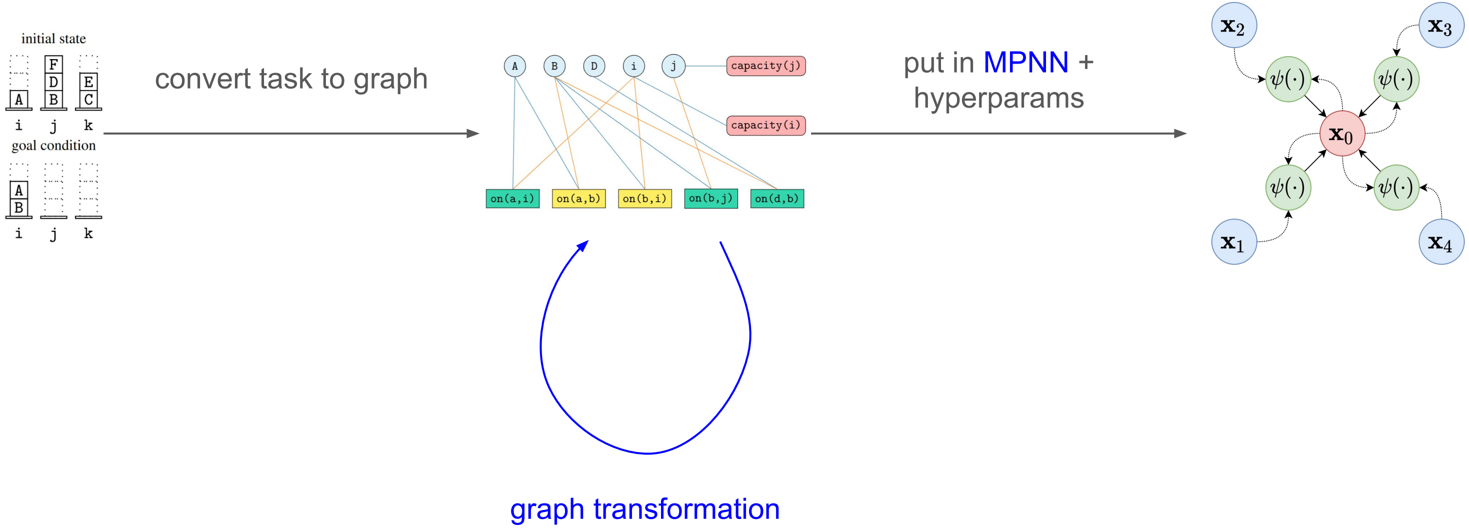


put in GNN +
hyperparams

`SimpleConv`	✓	✓		✓	✓	
`GCNConv` (Paper)	✓	✓			✓	✓
`ChebConv` (Paper)		✓			✓	✓
`SAGEConv` (Paper)	✓			✓	✓	✓
`CuGraphSAGEConv` (Paper)					✓	
`GraphConv` (Paper)	✓	✓		✓	✓	✓
`GatedGraphConv` (Paper)	✓	✓			✓	
`ResGatedGraphConv` (Paper)	✓		✓	✓	✓	✓
`GATConv` (Paper)	✓		✓	✓		✓
`CuGraphGATConv` (Paper)					✓	
`FusedGATConv` (Paper)					✓	
`GATv2Conv` (Paper)	✓		✓	✓		✓
`TransformerConv` (Paper)	✓		✓	✓		✓

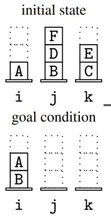
#	**Hyperparameter**	**Type**	**Scope**
1	Number of convolutional layers (n_1)	Categorical	{0,1,. . . ,20}
2	Number of output channels	Integer	{0,1,. . . ,50}
3	Kernel size	Integer	{0,1,. . . ,10}
4	Stride	Integer	{1,2,3}
5	Padding	Integer	{0,1,2}
6	Do a pooling	Boolean	{0,1}
7	Number of full layers (n_2)	Categorical	{0,1,. . . ,30}
8	Size of the full layer	Integer	{0,1,. . . ,500}
9	Dropout rate	Real	[0;1]
10	Activation function	Categorical/Integer	{ReLU (1), Sigmoid (2), Tanh (3)}

equivalent to:

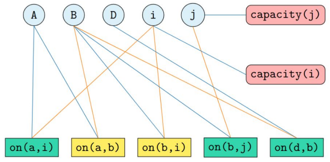


#	Hyperparameter	Type	Scope
1	Number of convolutional layers (n_1)	Categorical	{0,1,. . . ,20}
2	Number of output channels	Integer	{0,1,. . . ,50}
3	Kernel size	Integer	{0,1,. . . ,10}
4	Stride	Integer	{1,2,3}
5	Padding	Integer	{0,1,2}
6	Do a pooling	Boolean	{0,1}
7	Number of full layers (n_2)	Categorical	{0,1,. . . ,30}
8	Size of the full layer	Integer	{0,1,. . . ,500}
9	Dropout rate	Real	[0;1]
10	Activation function	Categorical/Integer	{ReLU (1), Sigmoid (2), Tanh (3)}

at most as expressive as:

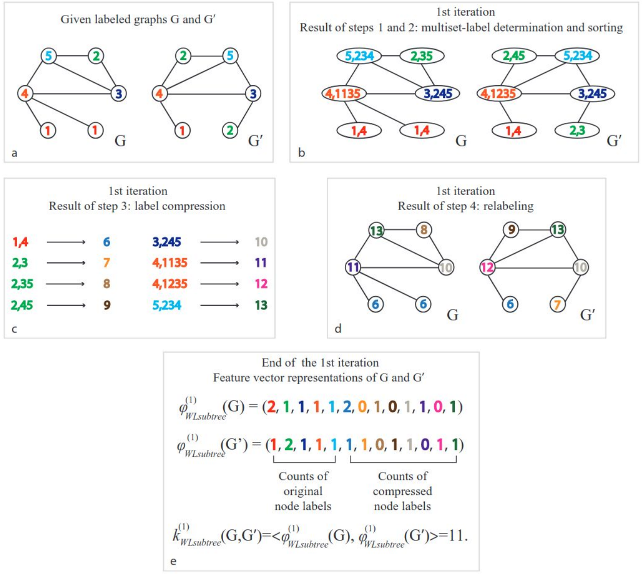


convert task to graph



graph transformation

put in WL kernel



N. Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, K. M. Borgwardt. **Weisfeiler-Lehman Graph Kernels**. JMLR, 2011.

[Insight 1] Graph Learning Model Does Not Matter

Why?

- graph learning architecture = graph representation + GNN
- = graph representation' + MPNN (message passing neural network)
- Morris et al. (AAAI-19) and Xu et al. (ICLR-19) independently pointed out that the WL graph kernel (Shervashidze et al., JMLR-11) is at least as expressive as MPNNs
- also no recent, significant GNN progress (Morris et al., ICML-24)

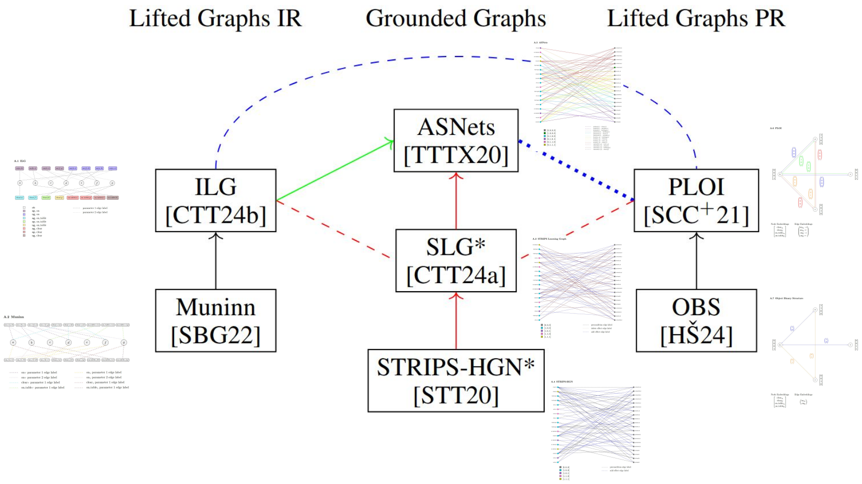
Upshot ⇒ theoretically measure expressivity based on graph representation

Graph Representation Study

In light of this, we

- unify and taxonomise existing graph representations of planning tasks for learning
- **theoretically** compare their representation power

Results Teaser (currently in submission)



X ⟶ Y	Y is strictly more expressive than X
X - - - Y	X and Y are incomparable
X ····· Y	X and Y are incomparable with no h^* difference requirement
X*	X is used for multi-domain heuristics

Extends graph representation + expressivity work from

Dillon Ze Chen, Felipe Trevizan, and Sylvie Thiébaux. *Learning Domain-Independent Heuristics for Grounded and Lifted Planning.* AAAI 2024.

[Insight 2] Deep Learning is Overrated for Symbolic Planning

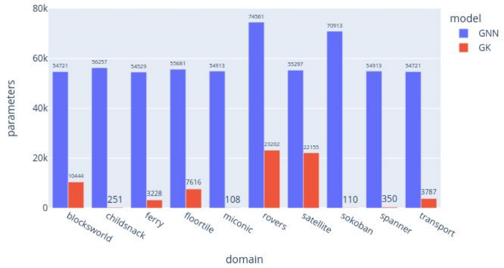
- neural networks are universal compact function approximators
  - whereas planning is inherently abstract and discrete
- neural networks are data hungry
  - not possible to generate new data for new domains
- neural networks are slow and hardware intensive
  - planning is a time-sensitive task, efficiency is a necessity

Classical Machine Learning ≫ Deep Learning for Planning

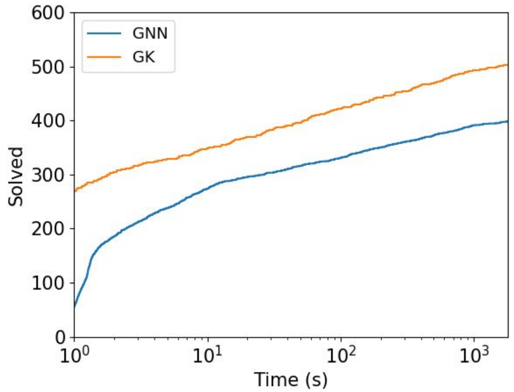
Dillon Ze Chen, Felipe Trevizan, and Sylvie Thiébaux. *Return to Tradition: Learning Reliable Heuristics with Classical Machine Learning.* ICAPS 2024.

- *Idea*: we use Graph Kernels (GKs) for our learning models
- GKs at least as **expressive** as Graph Neural Networks (GNNs)
- orders of magnitude cheaper to train and evaluate

GNN vs GK train time (s)
Note the LOG scale

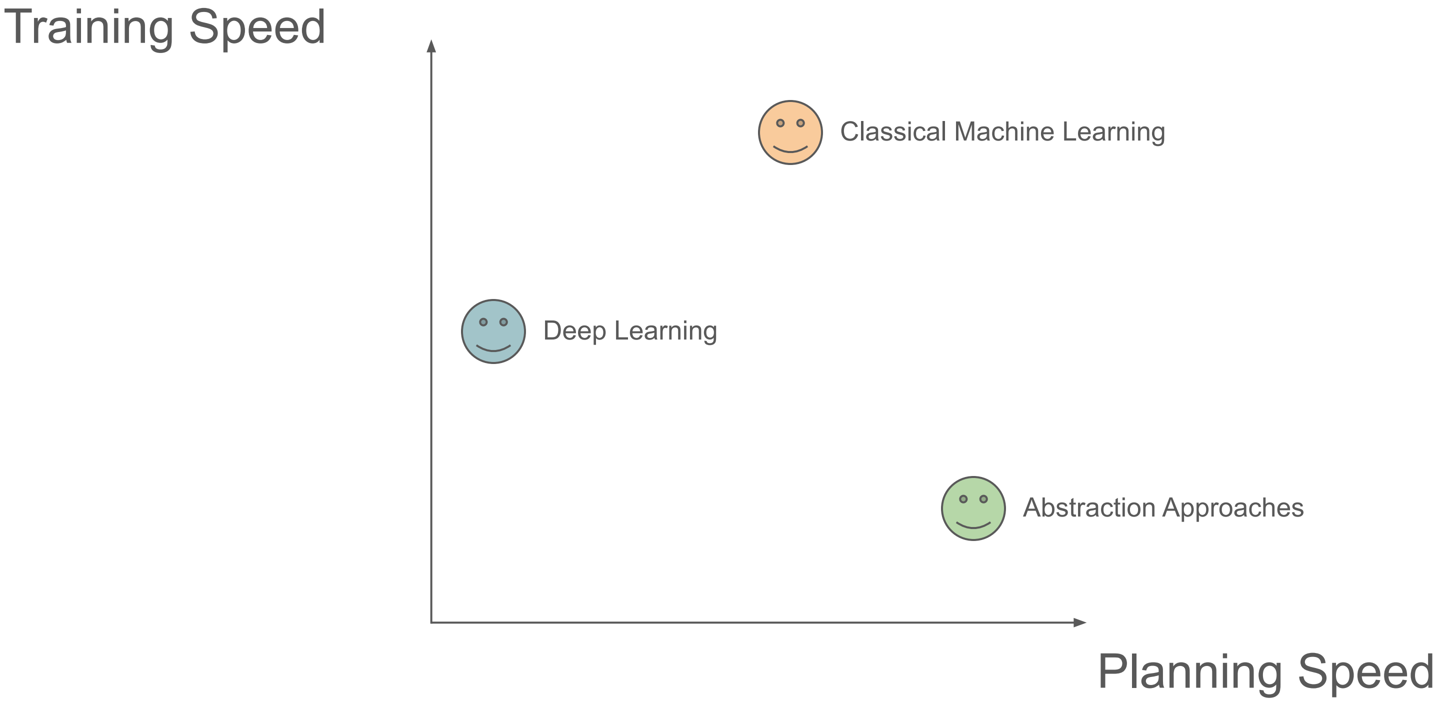


GNN vs GK parameters



Cumulative coverage over time
GNNs also have access to GPUs

Fixing our earlier slide…



What should we learn?

Heuristics?

Policies?

- Should we learn the optimal heuristic/policy?

But this may not be possible, e.g. P to solve, NP-hard to optimise domains

- Then what about the satisficing heuristic/policy?

This may not be known a priori for each domain.

*Synthesis approaches compute satisficing knowledge but requires expanding entire state spaces

We can relax the learning target

- When we (learn to) make decisions, we don't always try to compute optimal costs/rewards
- e.g. Should I drive to Philadelphia from Toronto for a 2-day workshop or take the plane?
  - I know that planes are a lot faster than cars for long distances
  - I don't need to compute/estimate time costs to bother consider taking a car:

```
time_cost(go(Toronto, Philadelphia, car)) = 8.5 hours

time_cost(go(home, Toronto_airport, car); go(Toronto_airport, Philadelphia, plane)) = 1.5~2.5 hours
```

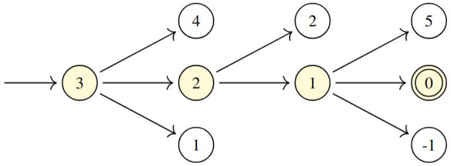
"decision makers can satisfice either by finding optimum solutions for a simplified world, or by finding satisfactory solutions for a more realistic world" – Herbert A. Simon

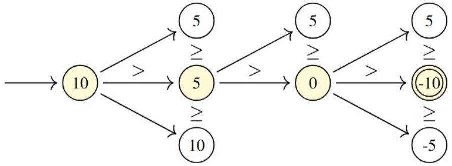
[Insight 3] Learn Rankings Instead of Hard Targets

Instead of learning to compute optimal costs per state, just learn to **compare** states

- Relax the optimisation criteria → no longer learning an NP-hard target
- Get additional data from plan traces for free
- Reduce overfitting to target values vs. optimal heuristic learners

Relevant refs:
[Garrett and Kaelbling and Lozano-Pérez, IJCAI-16]
[Chrestien et al., NeurIPS-23]
[Hao et al., IJCAI-24]
[Chen and Thiébaux, NeurIPS-24]





5 Open Challenges

Typical Machine Learning Problems

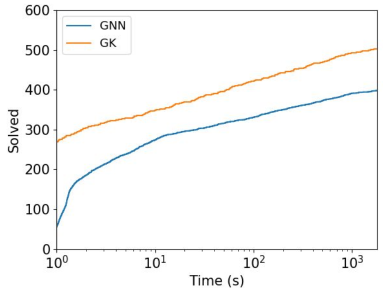
(1) Expressivity	Expressivity determines what domains your model can solve. e.g. a neural network (PTIME) cannot solve Sokoban (PSPACE)
(2) Generalisation	There is minimal generalisation theory with learning to plan. ⇒ lots of research opportunities
(3) Optimisation	Choice of optimisation depends heavily on the domain. e.g. {optimal, satisficing} x {heuristic, policy, sketches}?
(4) Collecting Data	When do we know we have collected enough data? e.g. Blocksworld training data only 1 tower, do I need data for N
(5) Fair Comparisons	Model performance is not robust to training data and parameters ~ More metrics when learning/generalisation is involved: N data, training speed etc.

Graph Learning for Planning:

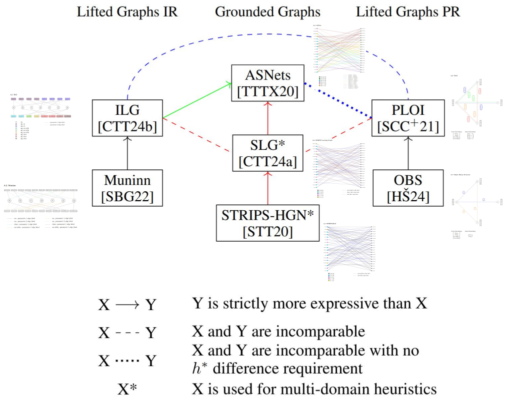
The Story Thus Far and Open Challenges

Dillon Z. Chen   Mingyu Hao   Sylvie Thiébaux   Felipe Trevizan

[Insight 1] Your Graph Learning Model Does Not Matter



[Insight 2] Deep Learning is Overrated for Symbolic Planning



[Insight 3] Learn Rankings Instead of Hard Targets