The Next-Generation of Planning Heuristics: GNNs and Beyond

Felipe Trevizan

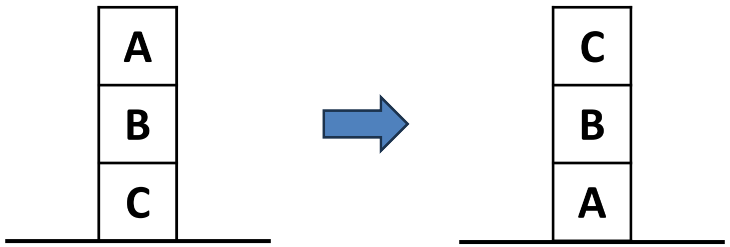
Australian National University

PRL Workshop
2/Jun/2024

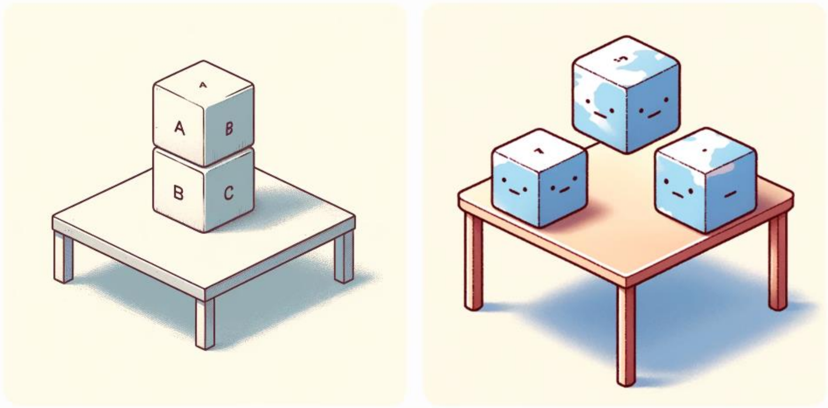
Machine Learning is Ubiquitous

- The public is already getting used to ML systems, e.g., Large Language Models
- LLMs are scoring relatively high in reasoning benchmarks, e.g., Gemini 1.5 Pro:
  - 60.7% (0-shot) on under-graduate physics problems
  - 77.7% (0-shot) on coding problem
- But sometimes they are unable to solve simple planning problems:

You
Draw the solution of the following blocks world problem: initial state on(a,b), on(b,c), on-table(c), clear(a). Goal: on(c,b), on(b,a), on-table(a), clear(c)



DALL·E



Here are the illustrations of the final state of the blocks world problem solution. Let me know if you need any further modifications or additional details!

Machine Learning for Planning

Can Machine Learning solve planning problems? Yes!

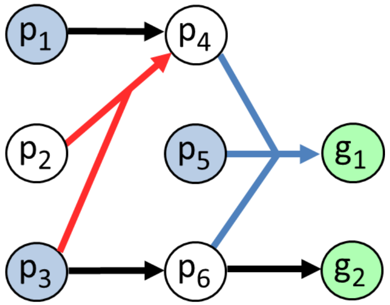
- We can learn a generalized policy that solves all problems of a given domain
- Does not work for every domain
  - Some domains are too hard
  - Limited expressivity for these approaches

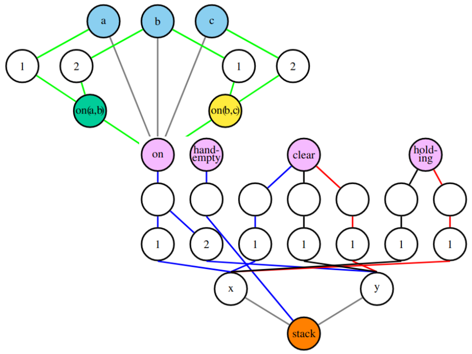
Can Machine Learning help to solve **planning problems?** Yes!

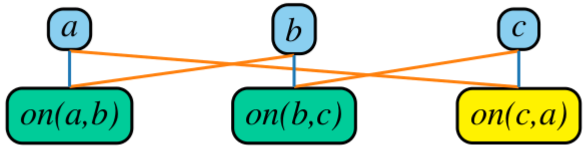
- Learn to guide a search algorithm towards good solutions
- The search algorithm can recover from bad predictions by the ML model
- **Focus of this talk**: graph-based approaches to learn such guidance

Outline

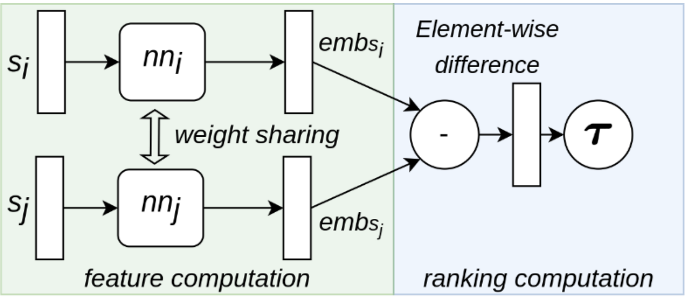
- Three graph-based approaches for learning heuristics

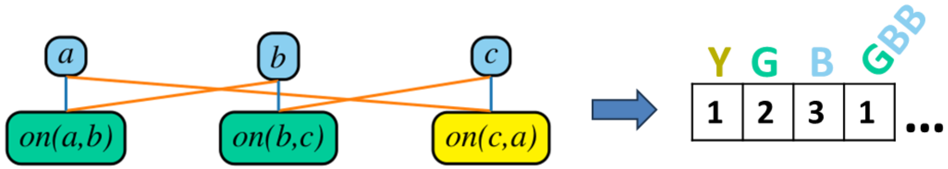






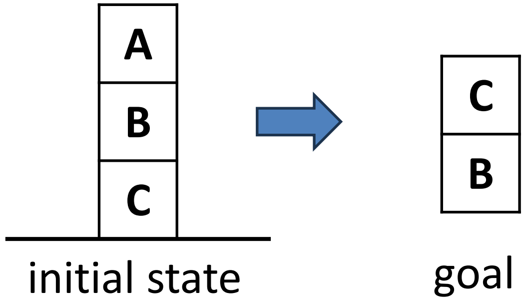
- Two novel methods to use these graphs for planning





AI Planning

Generate a course of action to reach given goals.



Path-finding in a gigantic transition system:

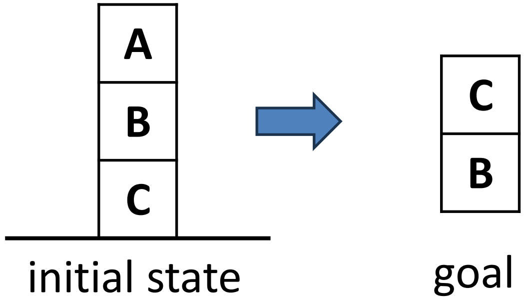
- **states**: world states
- **transitions**: actions
- set of **goal states**

Solution:

- **Plan**: action sequence leading to a goal state

Initial state → action_1 → action_2 → … → action_n → Goal state

STRIPS Representation



STRIPS is supported by **PDDL**:

Domain

Predicates: ← instances of — propositions

on(*block*, *block*)

holding(*block*)

…

Action Schemas: ← instances of — actions

<u>name</u>: stack(*x*, *y*)

<u>precondition</u>: holding(*x*) clear(*y*)

<u>effect</u>:

¬holding(*x*), ¬ clear(*y*) ← delete effects

clear(*x*), hand-empty, on(*x*,*y*) ← add effects

Problem

Objects:

A, B, C ← blocks

Initial State:

clear(A), on(A,B), on(B,C), on-table(C)

Goal:

on(C,B)

Heuristics

- **Heuristics**: cost estimators used to guide search (e.g., A* and GBFS)
  - $h(s)$: estimates the cost of reaching the goal from state s
- **Domain-independent heuristics**: (optimal) solution to a relaxation
- **Delete-relaxation**: remove negative effects
  - too hard to solve optimally
  - sub-optimal solutions computable in polynomial time
  - popular delete-relaxation heuristics are h-ff, h-add, h-max, lm-cut

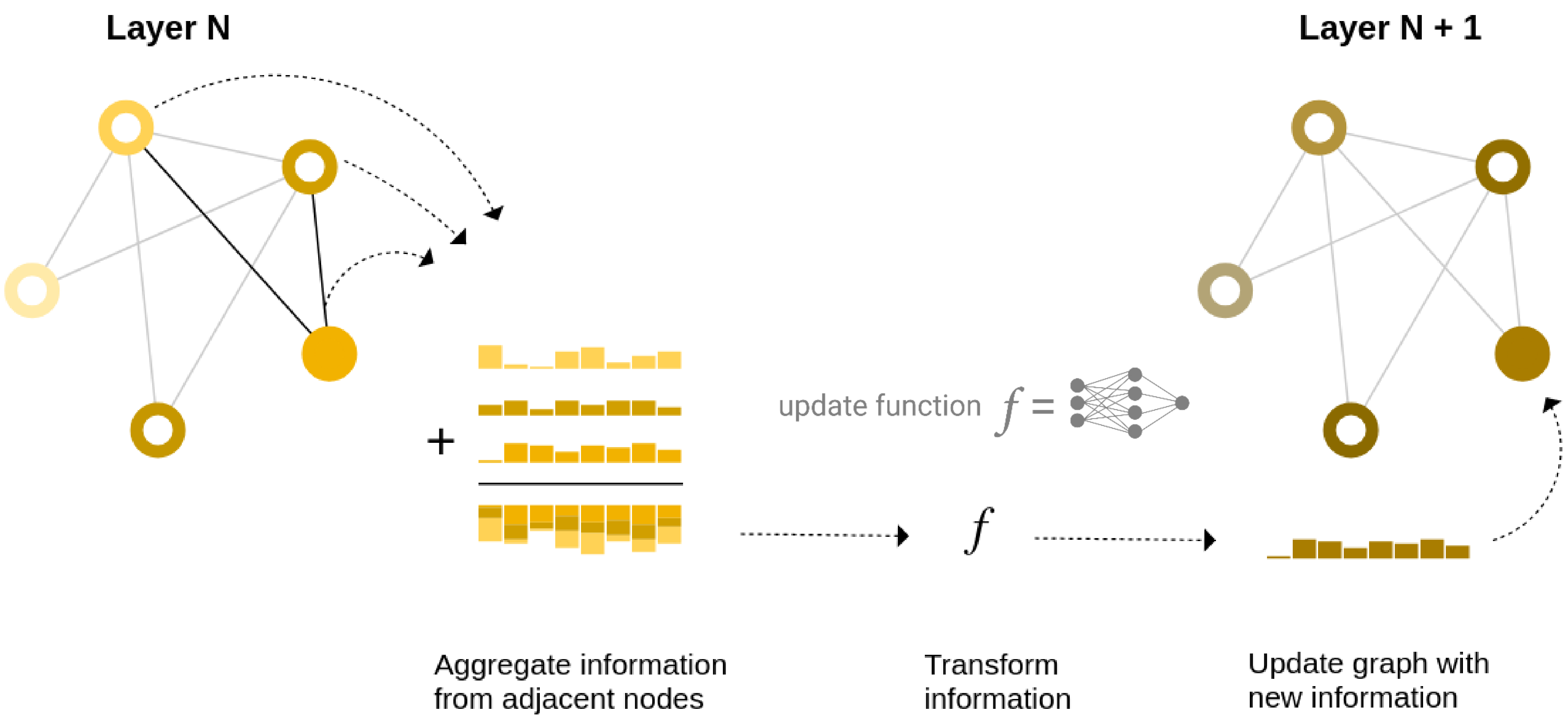
Domain

Predicates:
on(*block*, *block*)
holding(*block*)
…

Action Schemas:
name: stack(*x*, *y*)
precondition: holding(*x*) clear(*y*)
effect:
~~¬holding(*x*), ¬clear(*y*),~~
clear(*x*), hand-empty, on(*x*,*y*)

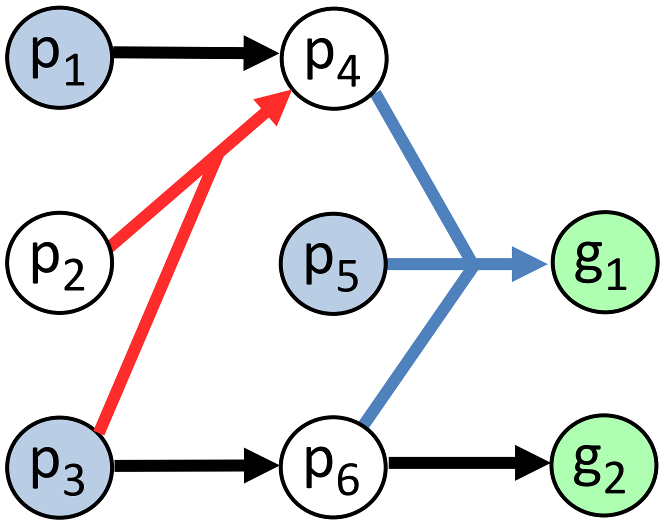
Graph Neural Networks

- Message Passing NN [GSR17]



[GSR17] Neural Message Passing for Quantum Chemistry. J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl. PMLR. 2017.
Image from https://distill.pub/2021/gnn-intro/

STRIPS-HGN: Learning domain-independent heuristics



Reference:

- **Domain-Independent Planning Heuristics with Hypergraph Networks**. William Shen, Felipe Trevizan, and Sylvie Thiébaux. ICAPS 2020.

Code:

- https://github.com/williamshen-nz/STRIPS-HGN

Background: h-max/h-add

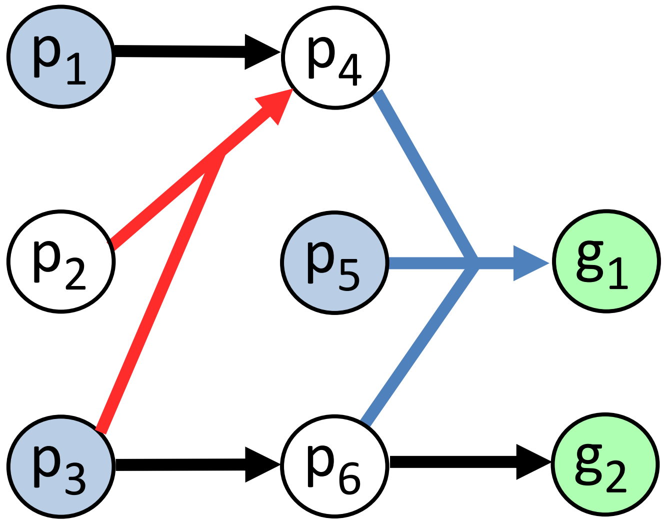
- Delete relaxation + **Ignore interactions between sub-goals**
  - If $G = \{p_1, \cdots, p_n\}$, each p_i is a sub-goal
- $h(s, G)$ **estimates the minimum cost from** s to G:
  - when $G = \{p\}$: $h(s, \{p\}) = \begin{cases} 0 & \text{if } p \in s \\ \infty & \text{if no action adds } p \\ \min_{a \text{ adds } p} h(s, \text{prec}(a)) + c(a) & \text{otherwise} \end{cases}$
  - when $|G| > 1$: $h(s, G) = \max_{p \in G} h(s, \{p\})$ → $h^{\max}(s) = h(s, G)$

    $h(s, G) = \sum_{p \in G} h(s, \{p\})$ → $h^{\text{add}}(s) = h(s, G)$

H-max/add as Shortest Path in a Hypergraph

Previous equations are computing **a shortest path** from each goal proposition to a proposition that is currently true in the following hypergraph:

- **nodes**: propositions
  - green: goal prop.
  - blue: true in state s
- **arcs**: actions
  - head: add effect
  - hypertails: preconditions



$s = \{p_1, p_3, p_5\}$
$\text{h-max}(s, \{g_1\}) = 2$
$\text{h-add}(s, \{g_1\}) = 5$

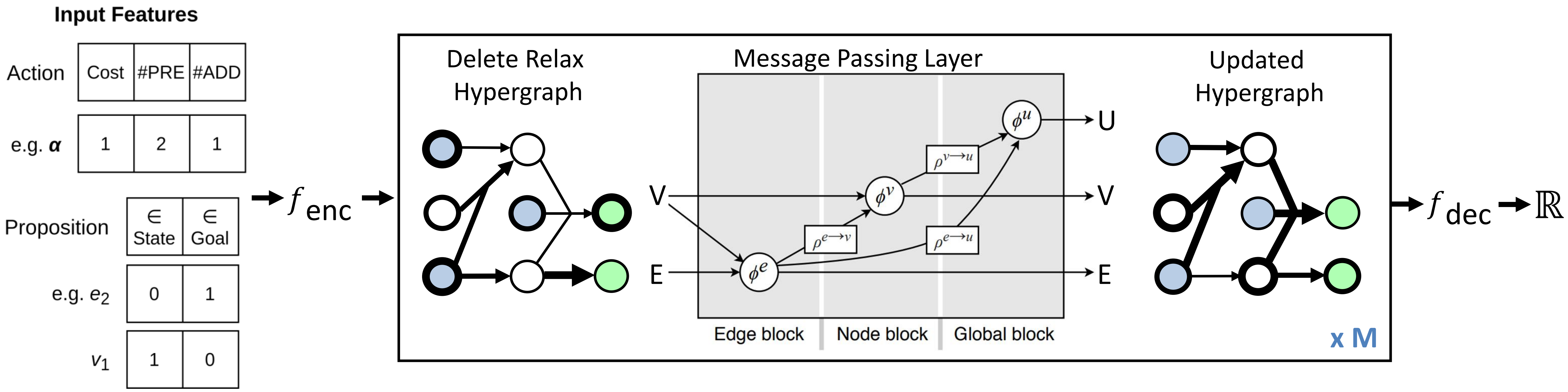
No well-defined distance function because of the hypertails

- Fixed by using **max** or **sum** to resolve hypertails

Can we learn a distance function for these hypergraphs from scratch?

STRIPS-HGN

- MPNN extended to support hypergraphs
- We use the implicit hypergraph from h-max/add



Update functions

- Learned MLPs: $f_{\text{enc}}, f_{\text{dec}}, \phi^e, \phi^v, \phi^u$
- Aggregation functions: $\rho^{e\rightarrow v}, \rho^{e\rightarrow u}, \rho^{v\rightarrow u}$ ← Element-wise sum

Training

- **Example generation**
  - Generate optimal plan π_i for problems P_i, $i \in \{1, \ldots, n\}$
  - Training samples (G, h*(s)) for each state s encountered in π_i
- **Weight optimization**
  - Regression problem
  - Mean Squared Error loss: $L_w(B) = \frac{1}{|B|} \sum_{(G,h^*(s)) \in B} \frac{1}{M} \sum_{t \in \{1,\ldots,M\}} (h_t^w(G) - h^*(s))^2$

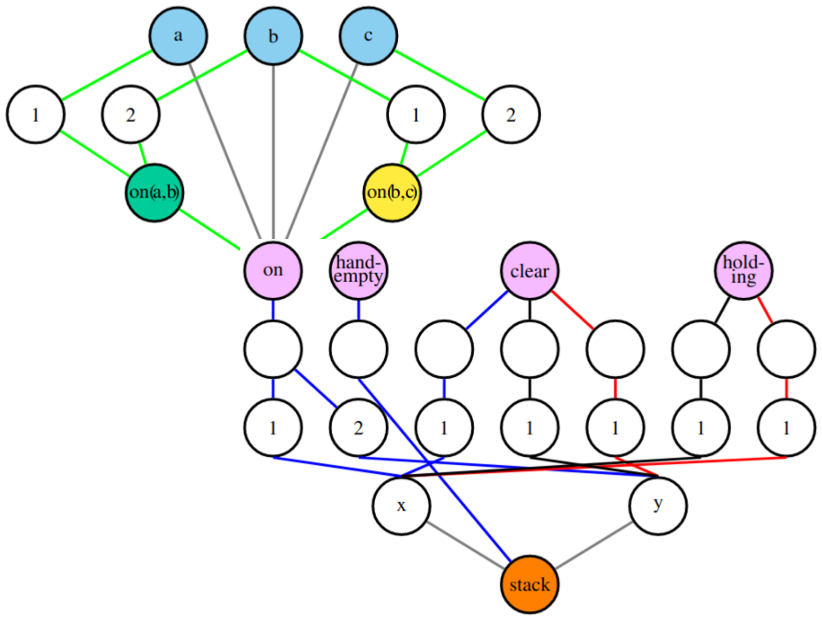
Experiment

Domain-specific setting (few-shot learning):
- Evaluation done using **unseen problems** of the domain used for training

Domain-independent setting (zero-shot learning):
- Evaluation done using problems of an **unseen domain**
- E.g., trained on BW and Gripper problems and evaluated on Zeno problems

	blind	h-max	h-add	STRIPS-HGN spec.	STRIPS-HGN indep.
blocksworld (100)	78	68	100	95	60
gripper (17)	12	10	10	16	5
zeno (60)	37	33	60	43	16
sum (177)	127	111	170	154	81

Lifted Learning Graph: Improving Domain-Independent Learning



Reference:

- **Learning Domain-Independent Heuristics for Grounded and Lifted Planning**. Chen, D., Thiébaux, S. and Trevizan, F. In Proc. of 38th AAAI Conference on Artificial Intelligence. 2024.

Code:

- https://github.com/dillonzchen/goose

Motivation

STRIPS-HGN has a drawback:

- It builds the complete hypergraph to do message passing
  - h-max/h-add do this implicitly
- Each message passing step is expensive

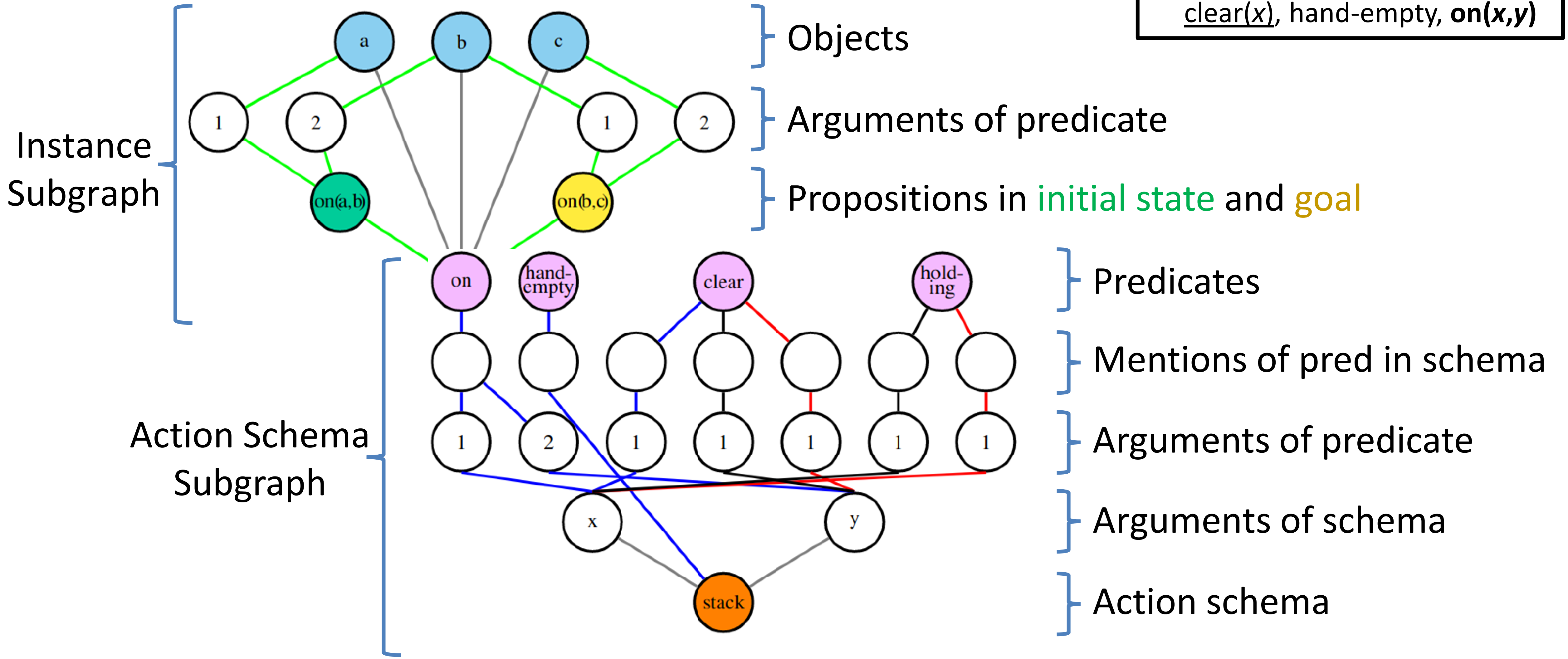
We want a graph that scales up better:

- Compact even for large instances
- Still represents the properties of domains and problems

Idea: design a graph based on the **lifted representation**

Lifted Learning Graph

stack(*x*, *y*)
precondition: holding(*x*) clear(*y*)
effect:
¬holding(*x*), ¬ clear(*y*),
clear(*x*), hand-empty, **on(*x*,*y*)**



Domain-Independent Experiment (2)

- **Training**: previous IPCs domains except evaluation domains
- **STRIPS-HGN** is trained as a **domain-specific** heuristic
- Greedy Best-First Search (GBFS) is used for evaluating heuristics

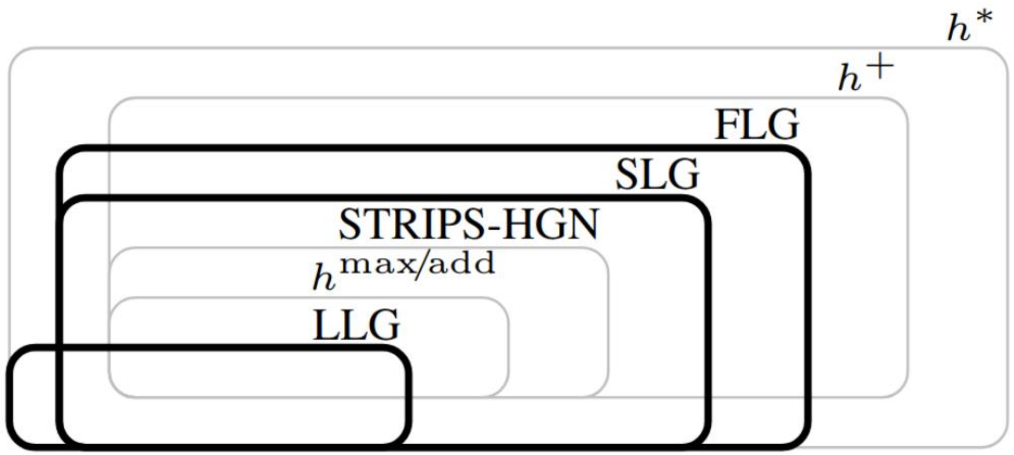
	blind	h^{FF}	d. spec. HGN	d. indep. LLG	d. indep. SLG	d. indep. FLG
blocksworld (90)	–	19	–	6	9	8
ferry (90)	–	90	–	2	28	22
gripper (18)	1	18	5	9	5	3
n-puzzle (50)	–	36	–	–	6	3
sokoban (90)	74	90	10	15	45	40
spanner (90)	–	–	–	–	–	–
visitall (90)	–	6	25	–	16	41
visitsome (90)	3	26	33	15	73	65
sum (608)	78	285	73	47	182	182

Grounded graphs defined in the same paper as LLG

Theoretical Results

We have characterized expressiveness of our networks:

- **LLG cannot represent h-max/h-add**
- STRIPS-HGN can represent h-max/h-add
- None of them can represent optimal solution to delete-free problem

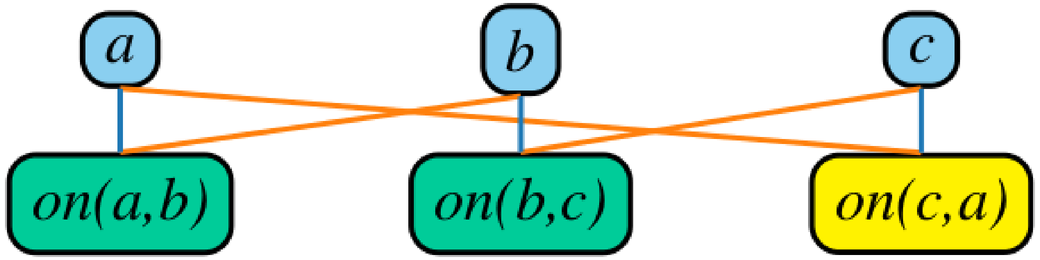


These results are based on the connection between MPNN and the Weisfeiler-Lehman algorithm for graph isomorphism/color-refinement:

– MPNNs are at most as powerful as color refinement [XWL19]

[XWL19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In International Conference on Learning Representations. 2019

Instance Learning Graph:
Learning Domain-Specific Heuristics



Reference:

- **Return to Tradition: Learning Reliable Heuristics with Classical Machine Learning.** Chen, D., Trevizan, F. and Thiébaux, S. In Proc. of 34th Int. Conf. on Automated Planning and Scheduling (ICAPS). 2024.

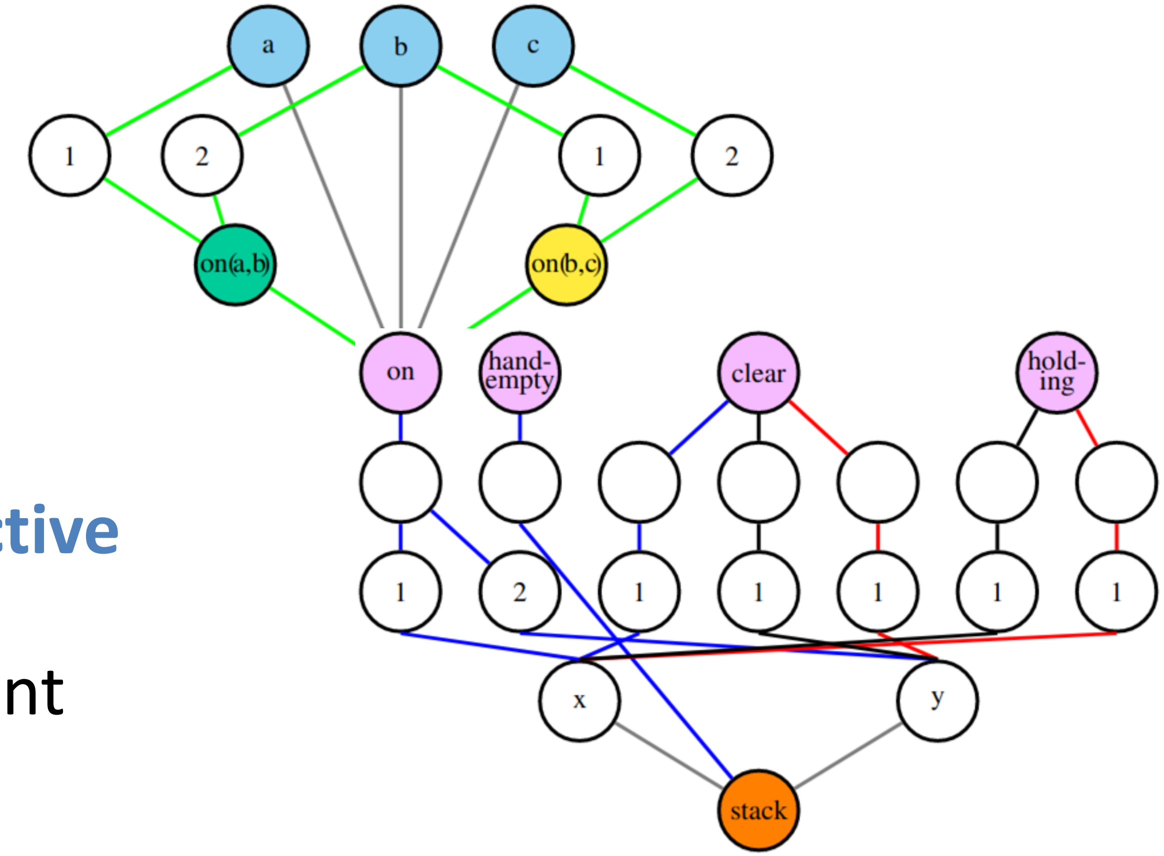
Code:

- https://doi.org/10.5281/zenodo.10757383

Motivation

Domain-independent knowledge is:

- hard to learn
- expensive to encode



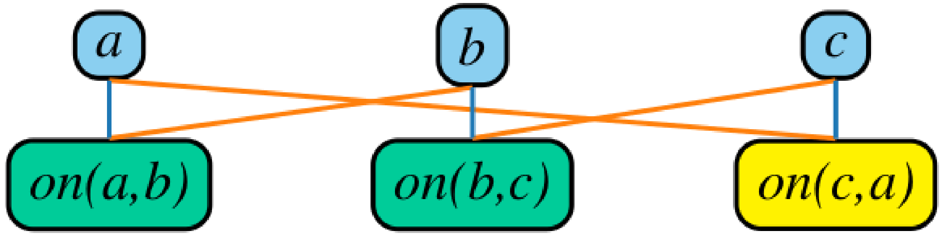
Domain-specific knowledge is **more effective**

- no need to encode actions (schemas)
- algorithms remain domain-independent

Idea for domain-specific graph:

- Simplify LLGs since the action schemas and predicates will remain the same for all problems of the same domain

Instance Learning Graph



- Nodes: all objects and all propositions in the initial state and goal condition
- Edges: between a proposition and the objects used to instantiate it
- Colors (labels):
  - Edges: position of the object in the predicate associated with proposition
  - Nodes:
    - Objects
    - Achieved goal proposition
    - Achieved proposition
    - Unachieved goal proposition

× set of predicates, e.g., (AP,on) and (UG,on)

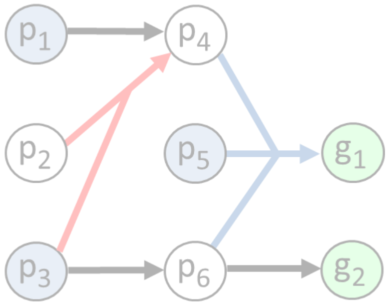
Domain-Specific Experiments

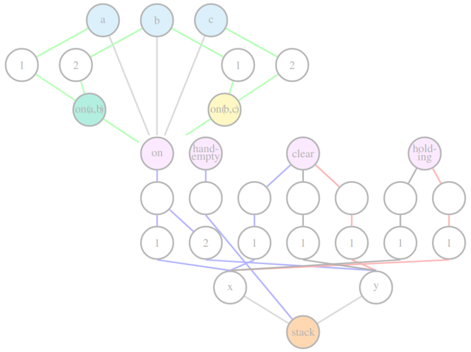
- Using the IPC 2023 Learning Track problems and methodology:
  - 99 problems in increasing order of difficult for training
  - 30 problems for evaluation for each difficulty

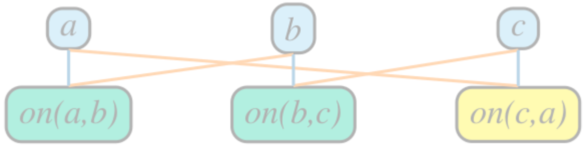
	h^{FF}	HGN				Lifted LG (LLG)				**Instance LG (ILG)**			
domain	sum	easy	med.	hard	sum	easy	med.	hard	sum	easy	med.	hard	sum
blocksworld	28	22	–	–	22	30	–	–	30	30	20	–	50
childsnack	26	6	–	–	6	19	–	–	19	18	–	–	18
ferry	68	26	2	–	28	30	30	1	61	30	30	1	61
floortile	12	–	–	–	0	1	–	–	1	–	–	–	0
miconic	90	30	30	–	60	30	30	15	75	30	30	16	76
rovers	34	25	–	–	25	29	2	–	31	25	1	–	26
satellite	65	13	–	–	13	27	2	–	29	26	1	–	27
sokoban	36	27	–	–	27	26	1	–	27	27	1	–	28
spanner	30	30	–	–	30	30	4	–	34	30	6	–	36
transport	41	22	–	–	22	30	8	–	38	30	9	–	39
sum	430	201	32	0	233	252	77	16	345	246	98	17	361

Outline

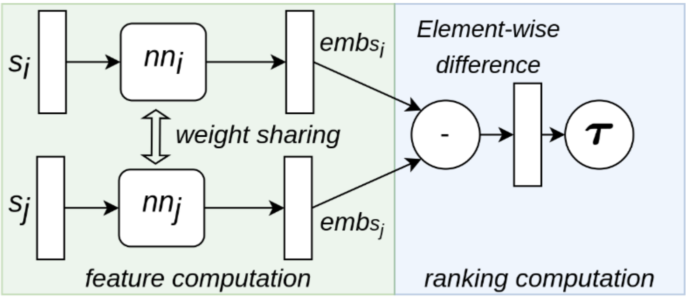
- Three Graphs-based approaches for Learning Heuristics

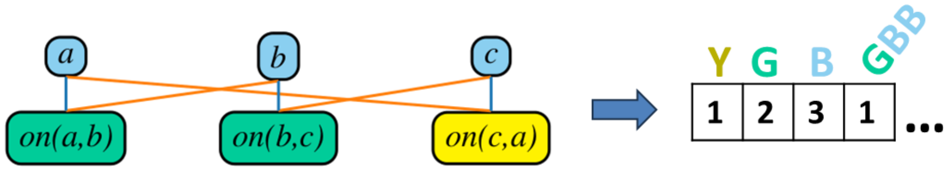




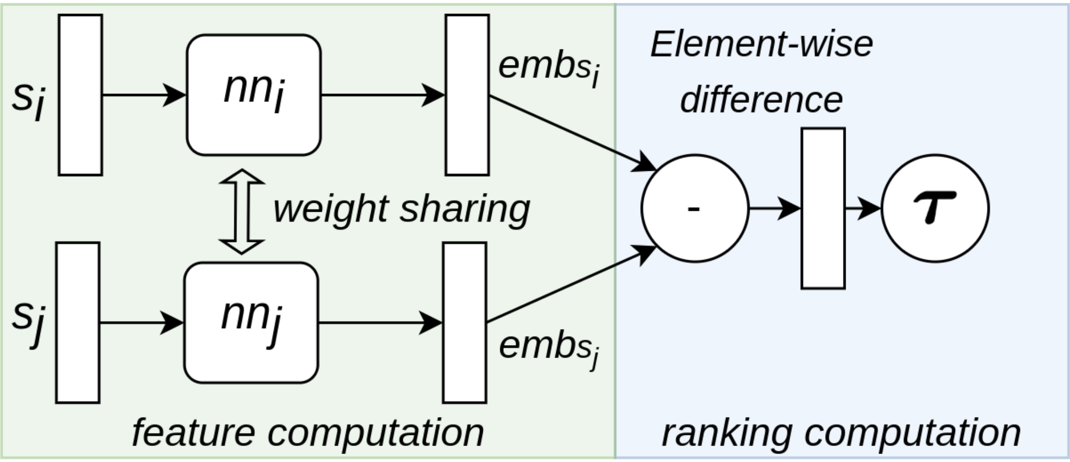


- Two new methods to use these graphs for planning





Optimal Ranker: Learning a Ranking Function



Reference:

- **Guiding GBFS through Learned Pairwise Rankings**. Hao, M., Trevizan, F., Thiébaux, S., Ferber, P. and Hoffmann, J. In Proc. of 33rd Int. Joint Conf. on AI (IJCAI). 2024.

Code:

- https://zenodo.org/records/11107790

Motivation

Greedy Best-First Search (GBFS):

open-list $:= \{s_0\}$
while open-list $\neq \emptyset$ **do**
    $s :=$ state $s' \in$ open-list with smallest $h(s')$
    remove s from open-list
    mark s as visited
    for all successor s' of s **do**
        if s' is a goal state **then** solution found!
        add s' to the open-list if it is not visited
    end for
end while

What if we change h(s) to:

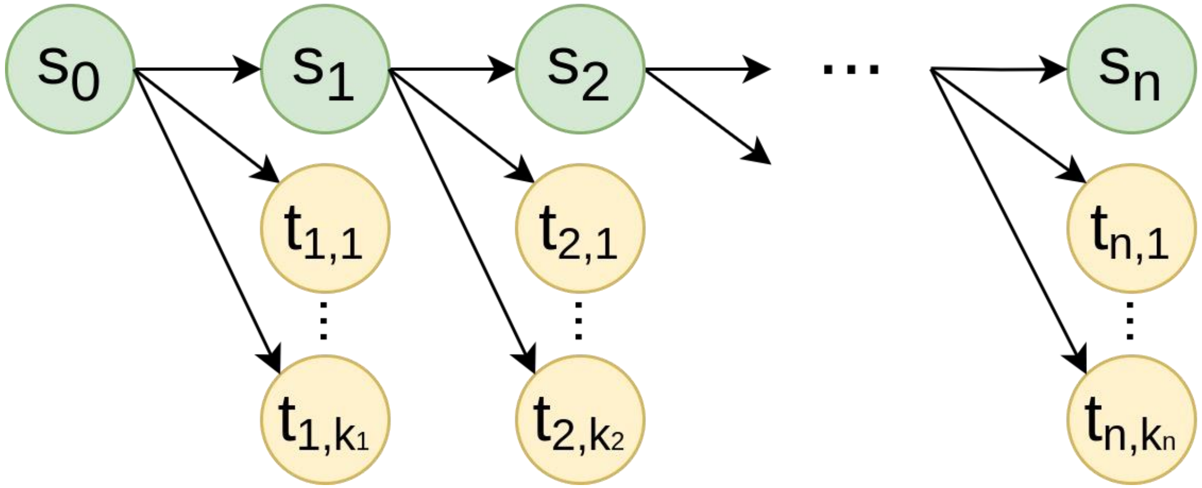
- 10 × h(s) ?
- log(1+h(s)) ?

The solution will not change because GBFS uses the heuristic to order/rank states!

Idea: learn a ranking between states instead of a heuristic (goal distance estimator)

Learning a Ranking between States

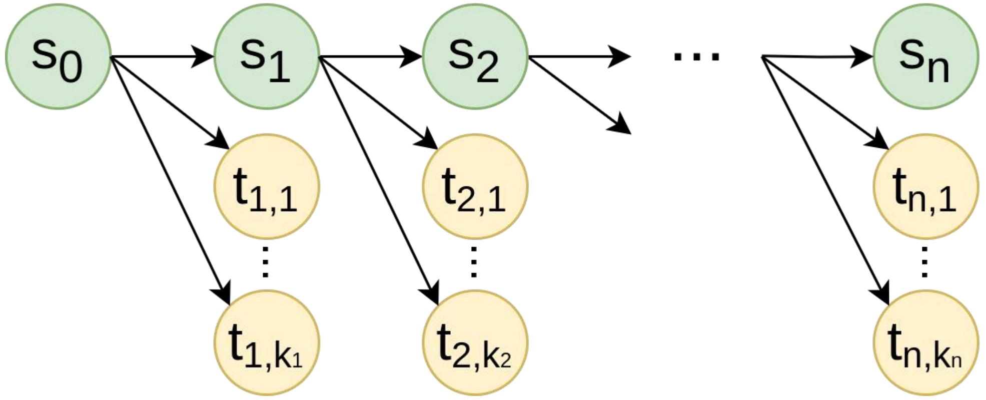
- Given two states s and s' **learn if s is better than or equal to s'** or s' is better than or equal to s
- **Advantages**
  - It is a classification problem instead of a regression problem
  - **More data for free**: no need to compute $h^*(t_{i,j})$ for training



Instead, say that s_i **is better than** $t_{i,j}$ for all i

Optimal Ranking

- We can go one step further: **learn a total quasi-order**, i.e., satisfies
  - Totality, transitivity and reflexivity

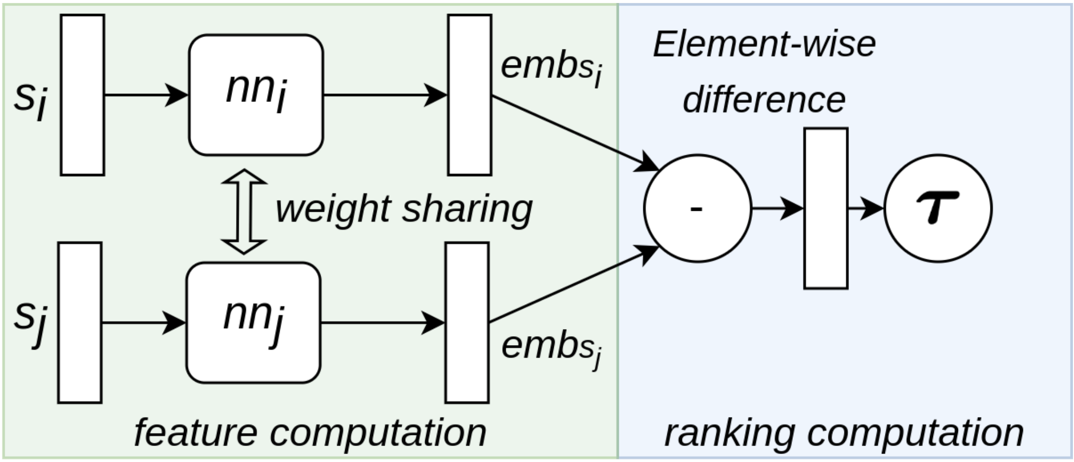


- **Optimal Ranking**: total quasi-order between the states in the optimal plan and their siblings
- **Even more data for free**:
  - an optimal plan of size n contains $O(n^2b)$ ordered pairs (**branching factor**)
  - due transitivity, we need only O(nb) pairs to encode all pairs

Learning and using Optimal Rankings

Learn using Direct Ranker [KWP20]

- Bring your own NN to compute embeddings
- **Learns how to compare states**:
  - $r(s_i,s_j) = \sigma(\vec{w} \cdot (emb_i - emb_j))$
  - s_i is better than or eq to s_j if $r(s_i,s_j) \leq 0$
- Guarantees total-quasi order



Use in GBFS by converting $r(s_i,s_j)$ to a **global ranking function** $\hat{r}(s)$:

- $r(s_i,s_j) \leq 0$ iff $\hat{r}(s_i) \leq \hat{r}(s_j)$
- **Smaller values of $\hat{r}(s)$ are preferred**

[KWP20] Koppel, M.; Segner, A.; Wagener, M.; Pensel, L.; Karwath, A.; and Kramer, S. 2020. Pairwise Learning to Rank by Neural Networks Revisited: Reconstruction, Theoretical Analysis and Practical Performance. ECML PKDD. 237–252

Domain-Specific Experiments (2)

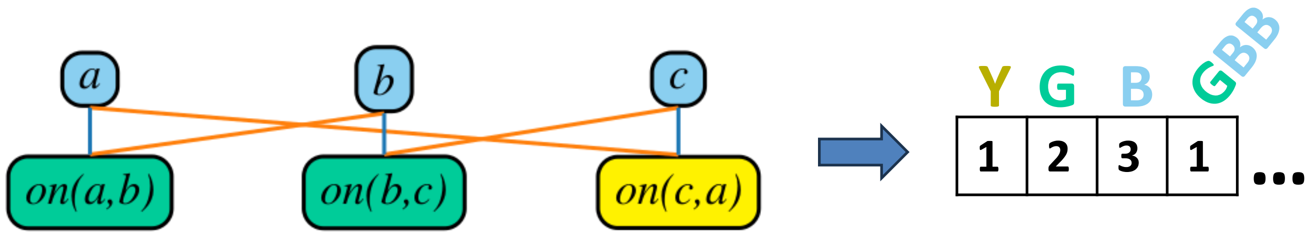
- Same IPC Learning Track 2023 setting as before

domain	h^{FF} sum	HGN easy	HGN med.	HGN hard	HGN sum	OptRank(HGN) easy	OptRank(HGN) med.	OptRank(HGN) hard	OptRank(HGN) sum	ILG easy	ILG med.	ILG hard	ILG sum	OptRank(ILG) easy	OptRank(ILG) med.	OptRank(ILG) hard	OptRank(ILG) sum
blocksworld	28	22	–	–	22	23	–	–	23	30	20	–	50	30	30	9	69
childsnack	26	6	–	–	6	25	–	–	25	18	–	–	18	21	1	–	22
ferry	68	26	2	–	28	30	8	–	38	30	30	1	61	30	30	3	63
floortile	12	–	–	–	0	1	–	–	1	–	–	–	0	1	–	–	1
miconic	90	30	30	–	60	30	27	1	58	30	30	16	76	30	30	16	76
rovers	34	25	–	–	25	28	–	–	28	25	1	–	26	28	–	–	28
satellite	65	13	–	–	13	30	1	–	31	26	1	–	27	30	5	–	35
sokoban	36	27	–	–	27	30	1	–	31	27	1	–	28	30	2	–	32
spanner	30	30	–	–	30	30	14	–	44	30	6	–	36	30	30	1	61
transport	41	22	–	–	22	28	–	–	28	30	9	–	39	30	1	–	31
sum	430	201	32	0	233	254	52	1	307	246	98	17	361	260	129	29	418

31.7% Increase

15.8% Increase

WL-Kernel:
GNNs features for Classical ML



Reference:

- **Return to Tradition: Learning Reliable Heuristics with Classical Machine Learning.** Chen, D., Trevizan, F. and Thiébaux, S. In Proc. of 34th Int. Conf. on Automated Planning and Scheduling (ICAPS). 2024.

Code:

- https://doi.org/10.5281/zenodo.10757383

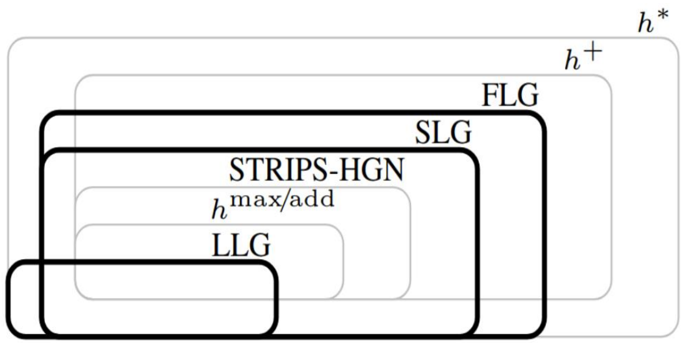
Motivation

We have been using **GNNs** so far but they have some **drawbacks**

- Several hyperparameters
- Several parameters to be learned

Recall from our theoretical results regarding LLGs:

- MPNNs are at most as powerful as color refinement



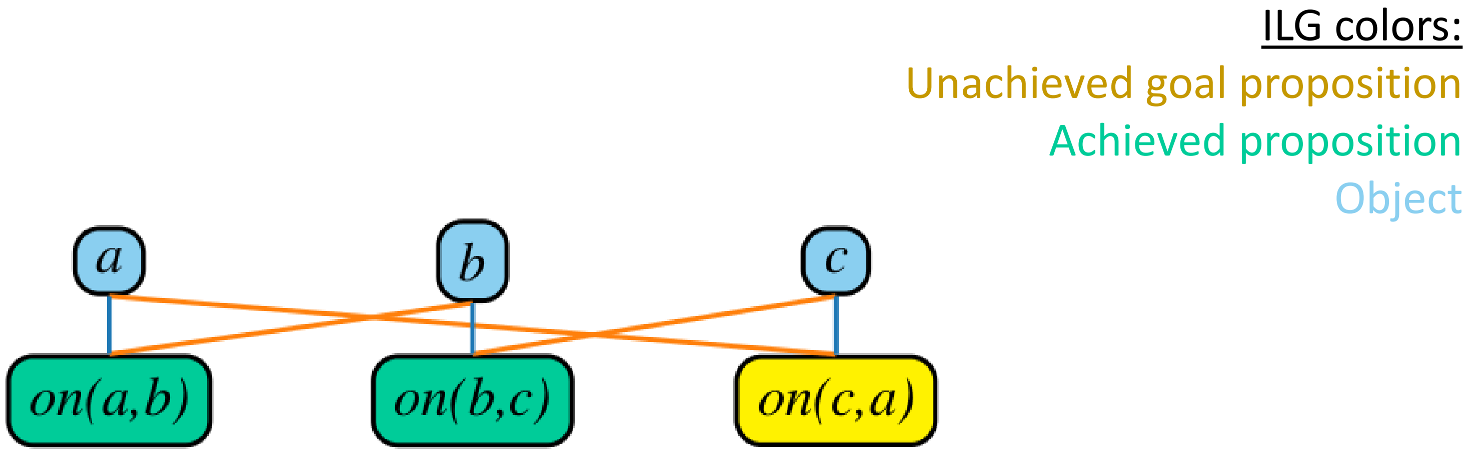
Idea: Use color refinement directly

- Generate features with same expressiveness power as the GNNs learned embeddings
- Use classical (non-NN) ML algorithms

WL Algorithm

The Weisfeiler-Leman algorithm graph isomorphism test based in color (label) refinement [LW68]

- At each iteration, the new color of a nodes is defined based on its own color and its neighbors' color
- Repeat for k iterations



Example: **on(a,b)** colors

0. green
1. green, {{blue, blue}}
2. (green, {{blue, blue}}), {{(blue, {{green, yellow}}), (blue, {{green, green}})}}

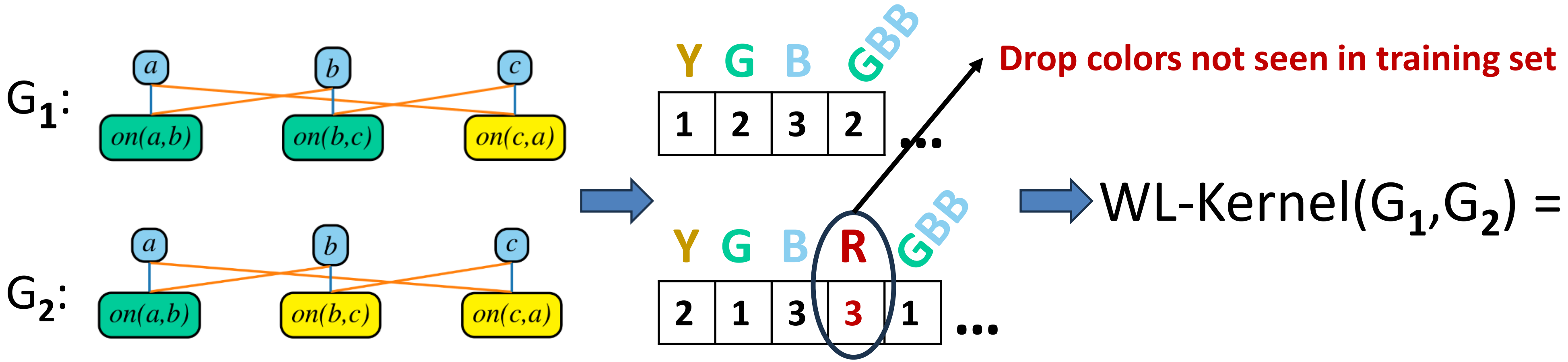
[LW68] Leman, A.; and Weisfeiler, B. 1968. A reduction of a graph to a canonical form and an algebra arising during this reduction. Nauchno-Technicheskaya Informatsiya.

WL Graph Kernel

A kernel k(x,y) in ML is a function measuring the "similarity" between x and y

WL Graph Kernel [SSV11]:

- Compute the WL colors for all nodes for all graphs in the training set
- Represent new graphs as a histogram of its WL colors **over the known colors**
- Compare two graphs by the dot product of their histograms



$\text{WL-Kernel}(G_1,G_2) = 1\times2 + 2\times1 + \ldots$

[SSV11] Shervashidze, N.; Schweitzer, P.; Van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-lehman graph kernels. Journal of Machine Learning Research.

Domain-Specific Experiments (3)

- Same IPC Learning Track 2023 setting as before
- GPR: Gaussian Process Regression

	h^{FF}	LAMA First	ILG	GPR WL(ILG)
blocksworld	28	61	63	75
childsnack	26	35	23	29
ferry	68	68	70	76
floortile	12	11	0	2
miconic	90	90	89	90
rovers	34	67	26	37
satellite	65	89	31	53
sokoban	36	40	33	38
spanner	30	30	46	73
transport	41	66	32	29
sum	430	557	413	502

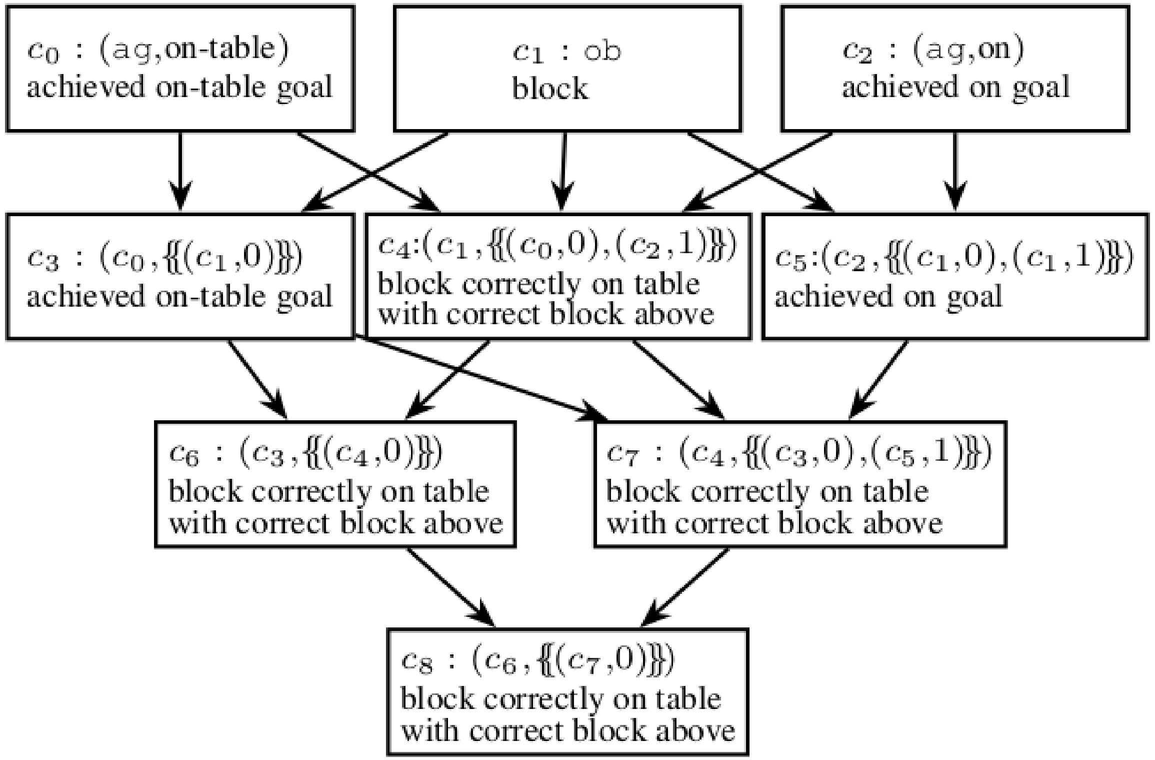
21.5% Increase

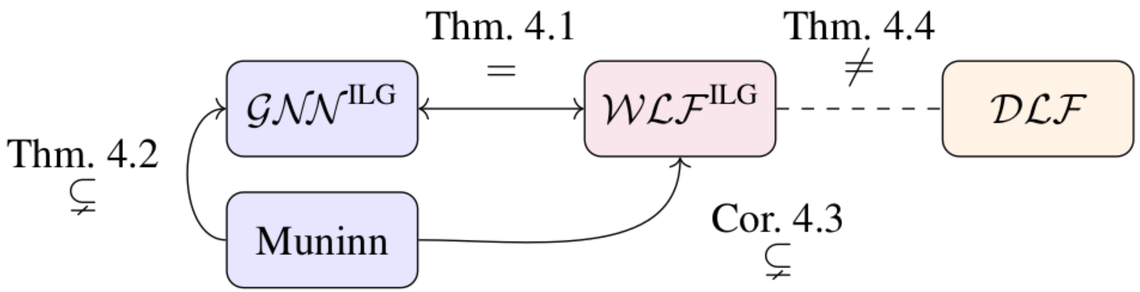
Interpreting WL Features and Theoretical Results

For details on this come to our talk on

Tuesday 15:00-16:30 (Planning & Learning)

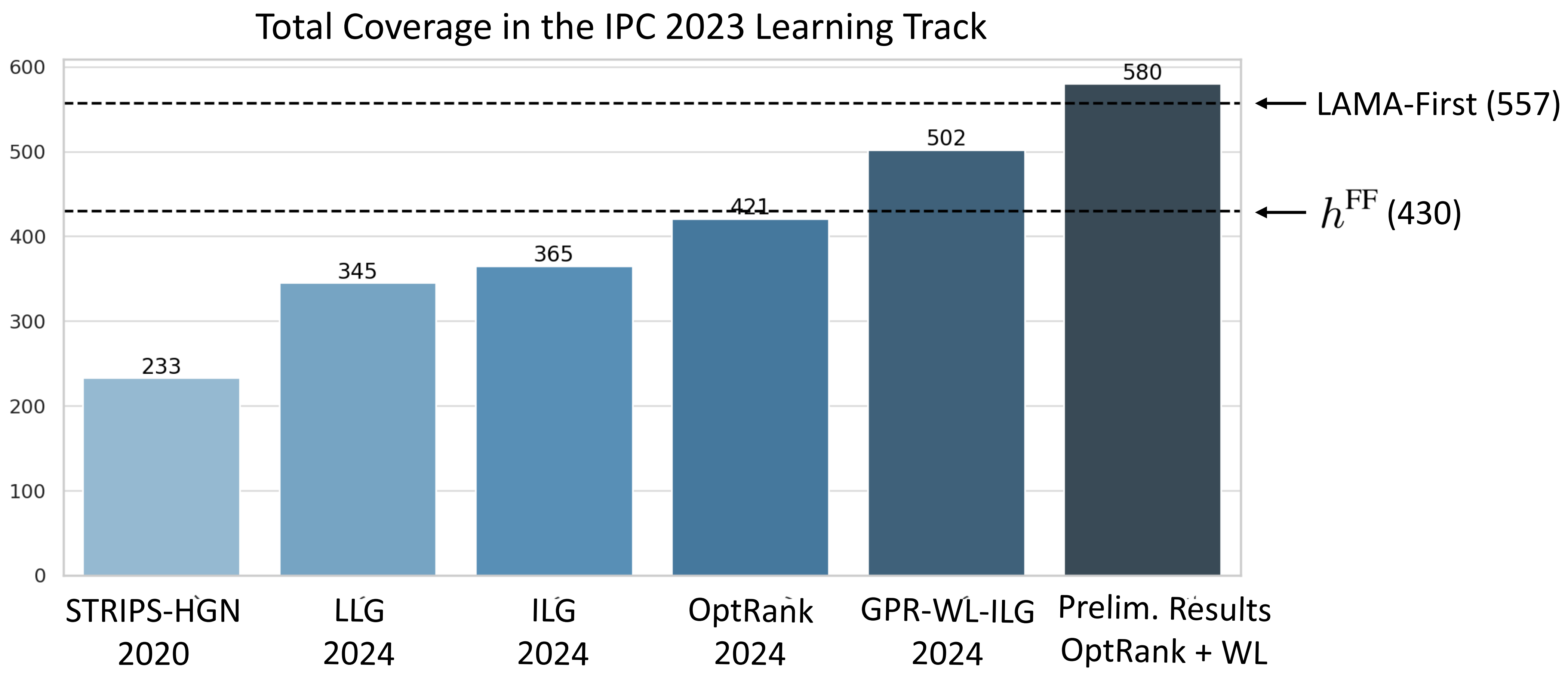
Return to Tradition: Learning Reliable Heuristics with Classical Machine Learning





Take away message

ML-based heuristics have the potential to replace classical planning heuristics for **sub-optimal planning**



But there are several challenges

- **Some domains are still challenging for ML**
  - From the IPC 23 Learning Track: Floor title, Rovers, Satellite, Transport and Child Snack
- **Computing training data (optimal plans) is expensive**
  - Try to get even more data for free
- **Curriculum Learning**, e.g., how to generate problems for training?
  - IPC provided problems in increasing order of difficulty
- **Continual Learning**
  - Improve the model during search (evaluation) when better solutions are found

Thank you

and to my collaborators and students:

- Dillon Chen
- Florian Geisser
- Joerg Hoffmann
- Malte Helmert
- Mingyu Hao
- Patrick Ferber
- Sylvie Thiébaux
- William Shen

Questions?