MOOSE: Satisficing and Optimal Generalised Planning via Goal Regression

LAAS CNRS

RWTH AACHEN UNIVERSITY

Dillon Z. Chen

Till Hofmann

Toryn Q. Klassen

Sheila A. McIlraith

VECTOR INSTITUTE | INSTITUT VECTEUR

UNIVERSITY OF TORONTO

PDDL (STRIPS) Planning

A **domain** is a set of first-order predicates and action schemata $\mathcal{D} = \langle \mathcal{P}, \mathcal{A} \rangle$

A **problem** is a domain, initial state, goal cond. and finite set of objects $P = \langle \mathcal{D}, s^0, g, O \rangle$

A **plan** α is sequence of actions that progresses s^0 to a state satisfying g

Closed World Assumption

Fully Observable Environment

Deterministic Actions

PDDL Planning: Household Robot Example

Domain

action schema

```
(:action move
   :parameters  (?from ?to)
   :precondition (and (atRobot ?from))
   :effect (and  (atRobot ?to)
      (not (atRobot ?from))))

(:action pickUp
   :parameters (?obj ?loc)
   :precondition  (and (at ?obj ?loc)
      (atRobot ?loc) (handFree))
   :effect (and (holding ?obj) (not (at ?obj ?loc))
      (not (handFree)))

(:action putDown
   :parameters (?obj ?loc)
   :precondition (and (holding ?obj) (atRobot ?loc))
   :effect (and (at ?obj ?loc) (handFree)
      (not (holding ?obj))))

...
```

predicate

Problem

```
(:objects dog ball apple mango cake)
(:init
   (hungry dog)
   (at mango bedroom)
   (at cake livingRoom)
   (at apple kitchen)
   (at ball backyard)
   (atRobot backyard)
)
(:goal
   (at cake kitchen)
   (at ball storageRoom)
)
```

objects

initial state

goal condition

Solutions as *Plans*

```
(pickUp ball)
(move backyard storageRoom)
(putDown ball)
(move livingRoom)
(pickUp cake)
(move kitchen)
(putDown cake)
...
```

A plan must be generated for every problem

Problem Statement: Generalised Planning (GP)

A **GP problem** is a set of train and test problems $\langle \mathcal{P}_{train}, \mathcal{P}_{test} \rangle$ from the same domain

A **generalised plan (GPlan) π** is a *program* that

- is *synthesised* from $\mathcal{P}_{train}$
- can be *instantiated* to solve problems in $\mathcal{P}_{test}$

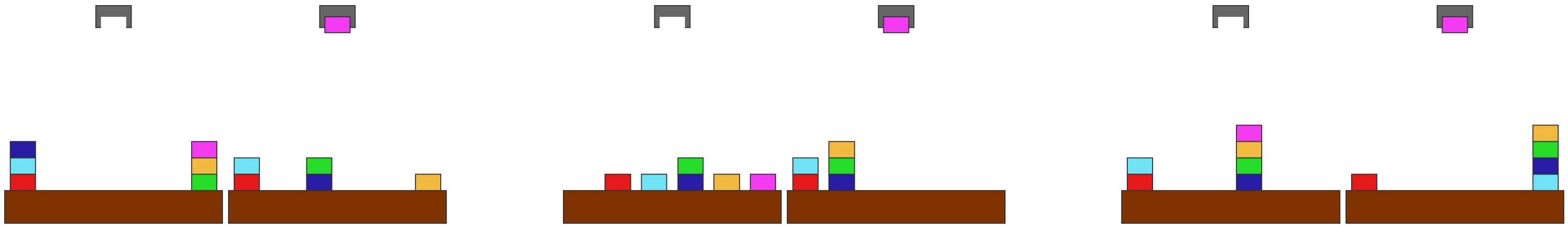
focus on extrapolation setting:

$f(\mathcal{P}_{test}) > f(\mathcal{P}_{train})$

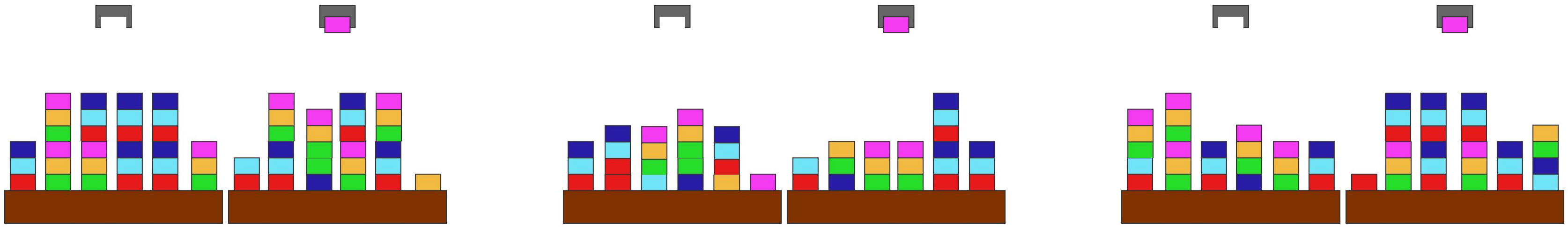
where $f(X)$ denotes the maximum number of objects in X

Generalised Planning: Blocksworld Example

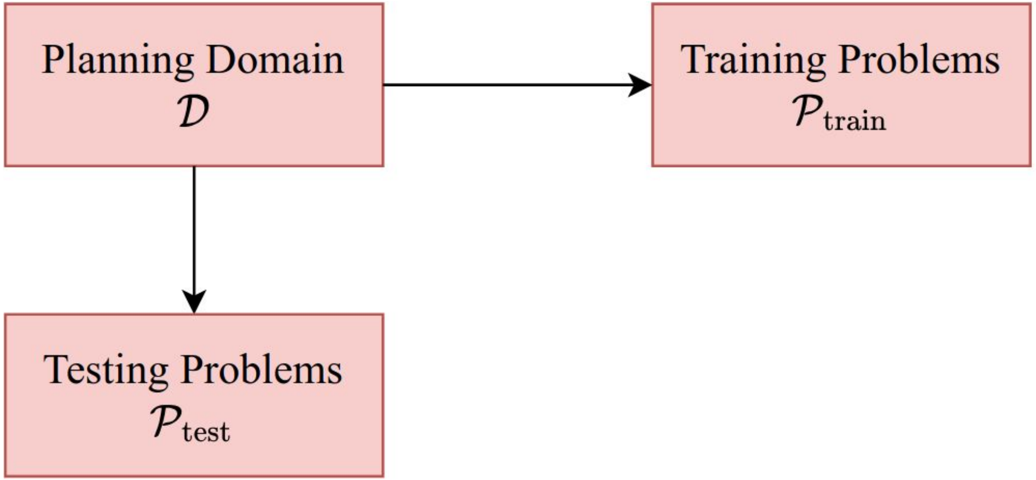
Training Problems $\mathcal{P}_{train}$



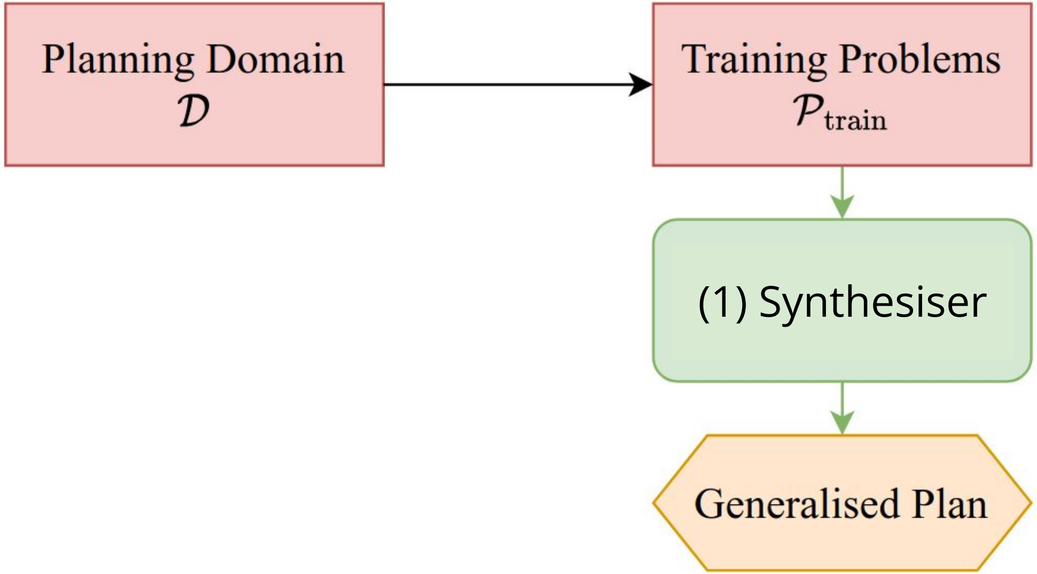
Testing Problems $\mathcal{P}_{test}$: more blocks than seen in training problems



GP Visualised – Inputs

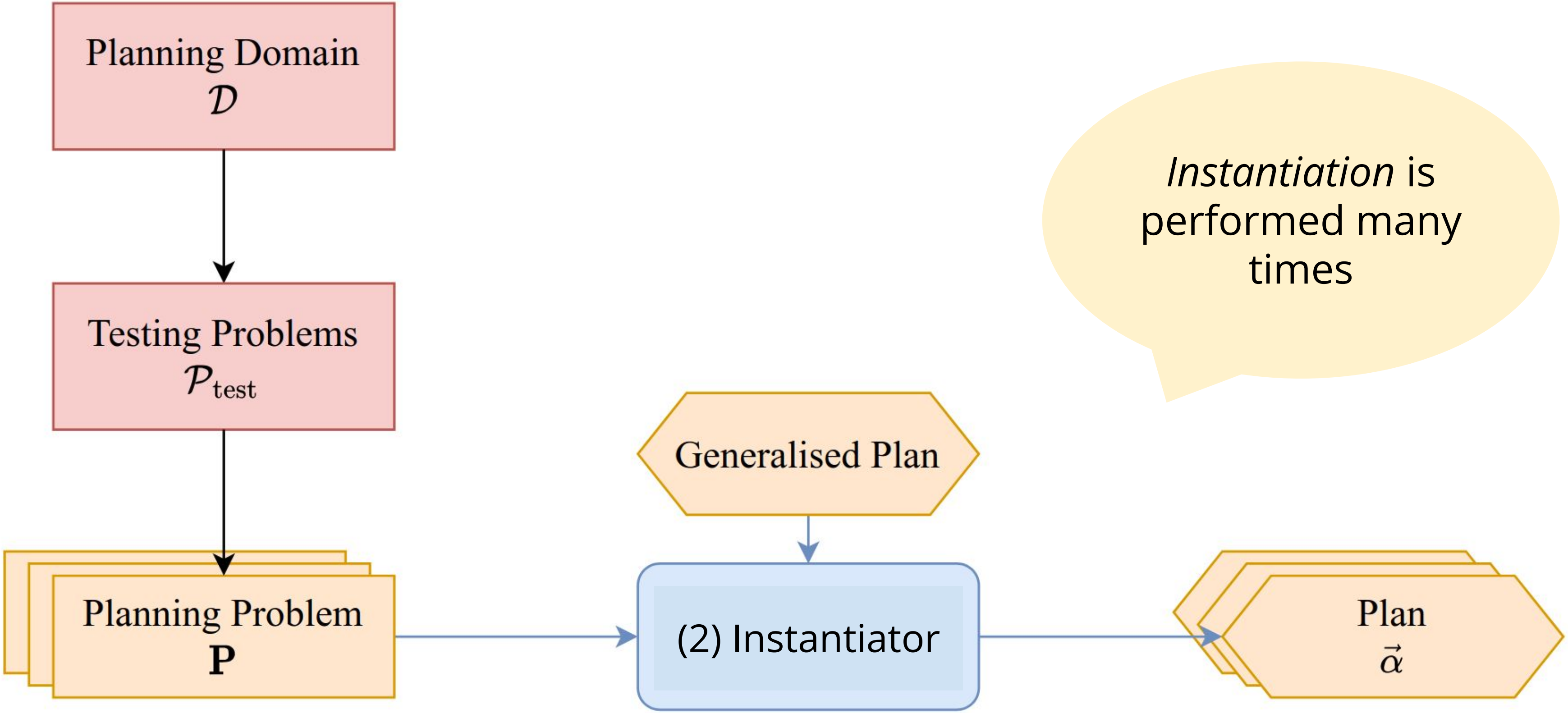


GP Visualised – Step 1: Synthesis

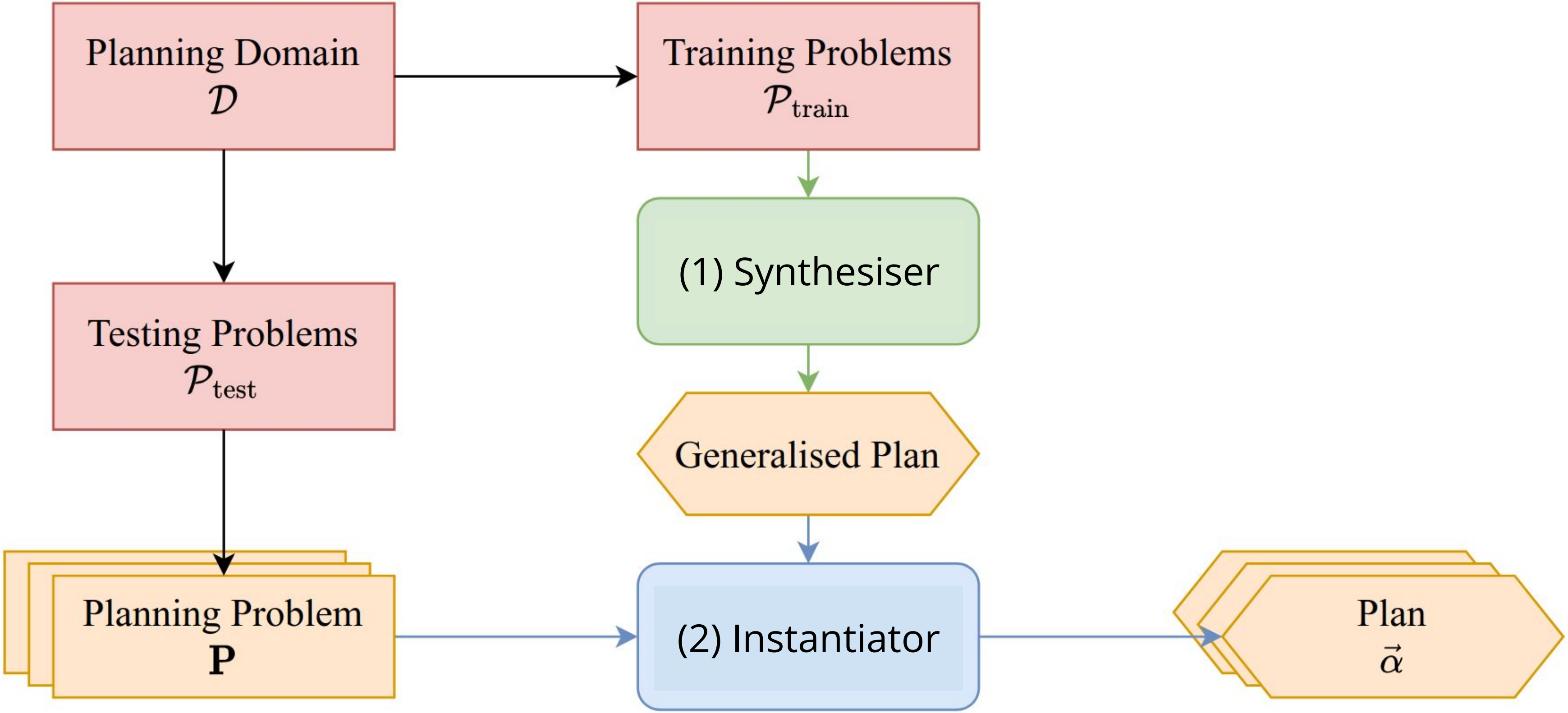


Synthesis is performed once

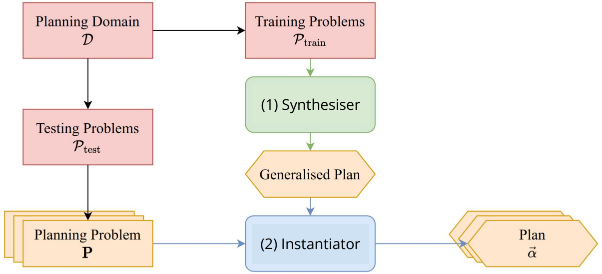
GP Visualised – Step 2: Instantiation



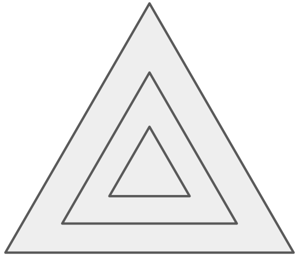
GP Visualised



Metrics



Synthesis Cost
Costs (e.g. data, time, memory) to synthesise a generalised plan

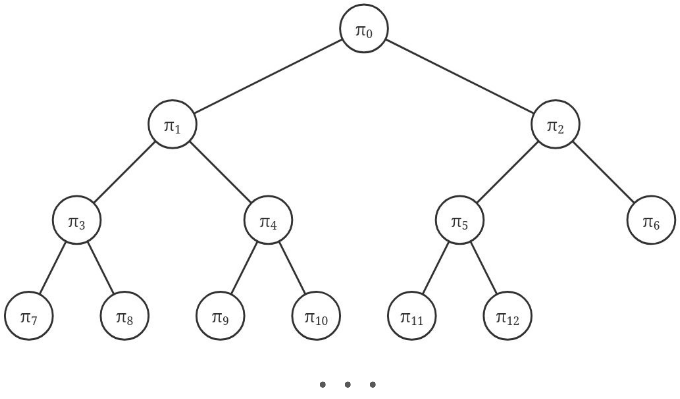


Instantiation Cost
Costs (e.g. time, memory) to instantiate a generalised plan on new problems

Solution Quality
Quality of plans returned from instantiating a generalised plan on new problems

Generalised Planning is Hard!

Generalised Planning is Hard!



- Space of generalised plans is huge
- Some GP models are EXPSPACE-complete [1,2]
- Classical planners are still better than existing generalised planners [3]

[1] Siddharth Srivastava, Shlomo Zilberstein, Neil Immerman, Hector Geffner: Qualitative Numeric Planning. AAAI 2011
[2] Blai Bonet, Hector Geffner: Qualitative Numeric Planning: Reductions and Complexity. J. Artif. Intell. Res. 69: 923-961 (2020)
[3] Ayal Taitler, Ron Alford, Joan Espasa, Gregor Behnke, Daniel Fiser, Michael Gimelfarb, Florian Pommerening, Scott Sanner, Enrico Scala, Dominik Schreiber, Javier Segovia-Aguas, Jendrik Seipp: The 2023 International Planning Competition. AI Mag. 45(2): 280-296 (2024)

Knowledge Representation Techniques Can Help Solve GP Problems

Goal Regression

- *Goal regression* [4] computes the minimal and sufficient condition for achieving a goal ***g*** via an action ***a***
  - ⇒ efficient policy space search
- PDDL STRIPS goal regression is defined by

$$\boldsymbol{regr(g, a) = (g \setminus add(a)) \cup pre(a)}$$

[4] Raymond Reiter: The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression. Artificial and Mathematical Theory of Computation 1991

Methodology: (1) Synthesising GPlans via Goal Regression

Synthesise a GPlan $\boldsymbol{\pi}$ in the form of a set of first-order rules from $\mathcal{P}_{train}$ by

1. compute optimal plans $\{a_1, ..., a_n\}$ for single goal atoms in some order $\{g_1, ..., g_n\}$ for each training problem $P \in \mathcal{P}_{train}$
2. perform goal regression on goals g_i with corresponding plans π_i to get a set of partial-state, macro-action pairs $\langle \sigma_i, a_i \rangle$ where $a_i = \alpha_1, ..., \alpha_q$
3. lift the set of pairs $\langle \sigma_i, a_i \rangle$ and goals g_i into a set of first-order rules

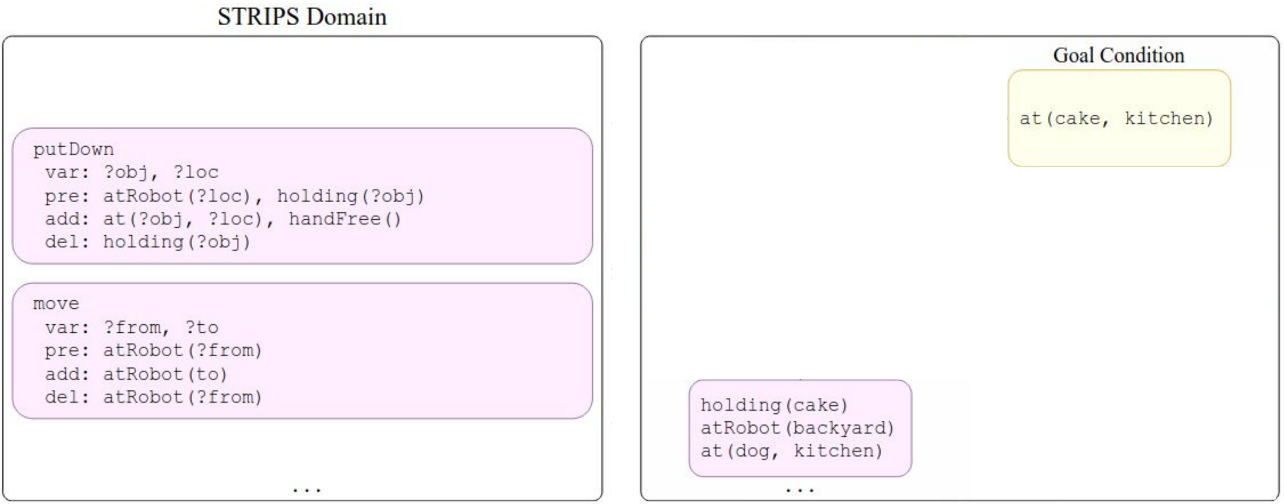
state condition | goal condition | actions

$$\left\{ \exists \{X\} \underbrace{\bigwedge_{i=1,...,m} p_i^s(X_i^s)}_{\text{state condition}} \wedge \underbrace{\bigwedge_{j=1,...,n} p_j^g(X_j^g)}_{\text{goal condition}} \rightarrow \underbrace{\alpha_1(X_1^a), ..., \alpha_q(X_q^a)}_{\text{actions}} \right\}$$

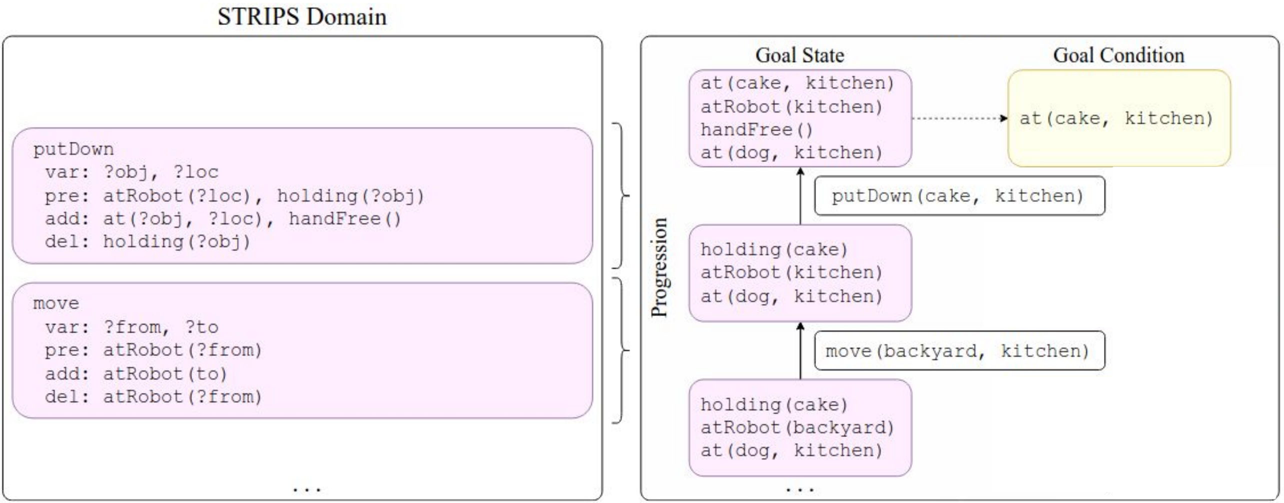
STRIPS Domain

```
putDown
 var: ?obj, ?loc
 pre: atRobot(?loc), holding(?obj)
 add: at(?obj, ?loc), handFree()
 del: holding(?obj)

move
 var: ?from, ?to
 pre: atRobot(?from)
 add: atRobot(to)
 del: atRobot(?from)

...
```

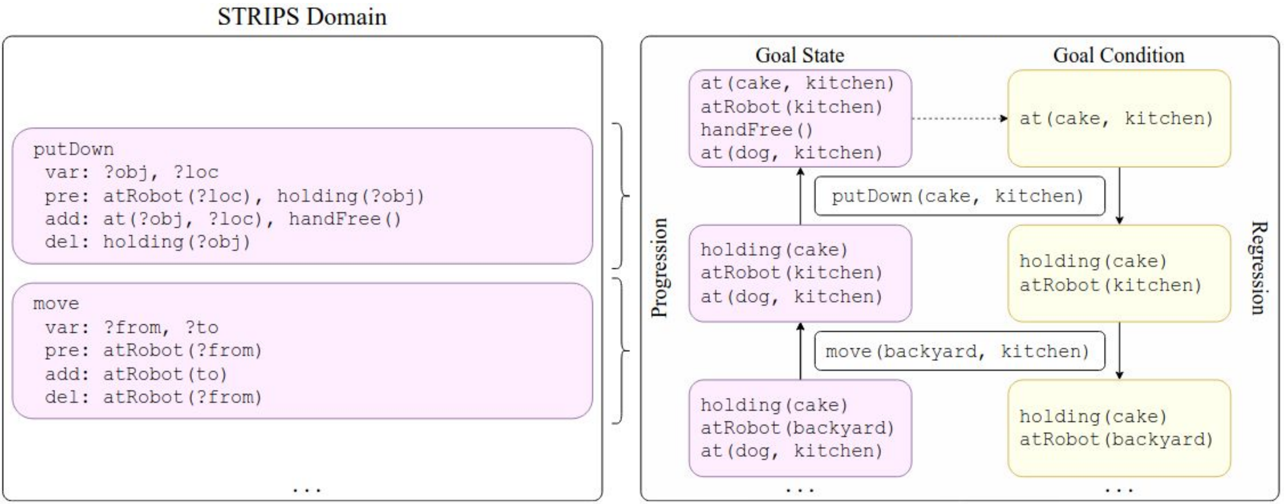
transportation domain



initial state
and
goal condition

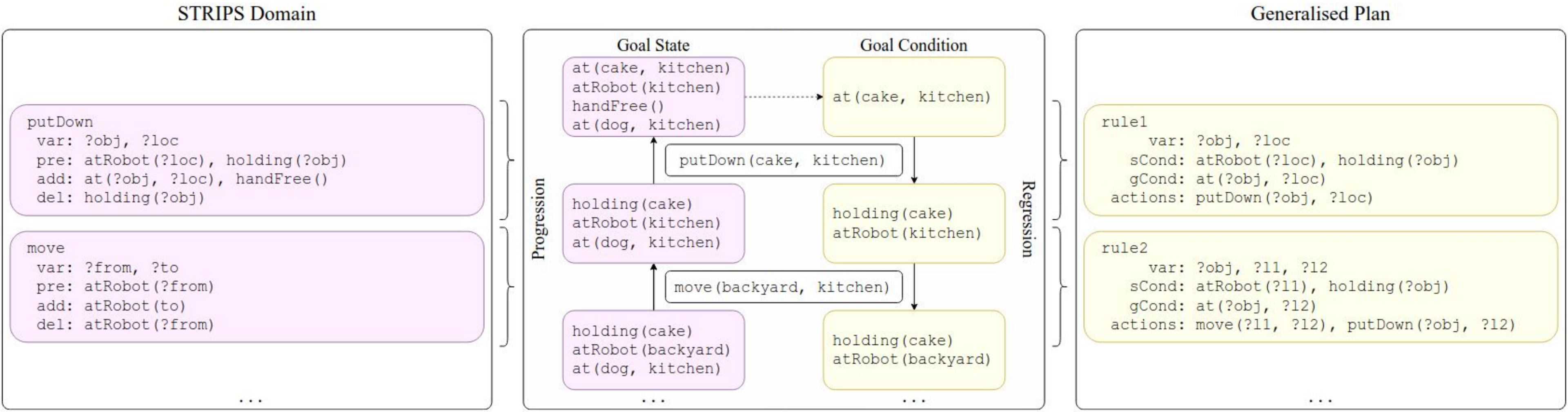


find a plan and progress the initial state



regress the goal with the plan

lift the regressed states into rules



Methodology: (2) Instantiating GPlans via Database Algorithms

speed
focused GP

Instantiate a GPlan $\boldsymbol{\pi}$ on a problem $\boldsymbol{P} \in \mathcal{P}_{test}$ by treating it as a policy

1. set $\boldsymbol{s} = \boldsymbol{s_0}$ and while the goal has not been achieved, repeat the following steps
2. ground a lifted rule where $\wedge_{i=1,\dots,m} p_i^s(X_i^s)$ holds in $\boldsymbol{s}$ and $\wedge_{j=1,\dots,n} p_j^g(X_j^g)$ holds in $\boldsymbol{g} \setminus \boldsymbol{s}$
3. apply corresponding sequence of actions $a_1(X_1^a), \dots, a_q(X_q^a)$ on $\boldsymbol{s}$

ground with
first-order query
algorithms

Methodology: (3) Instantiating GPlans via Search

Instantiate a GPlan $\boldsymbol{\pi}$ on a problem $\boldsymbol{P} \in \mathcal{P}_{test}$ with search space pruning via PDDL axioms

quality focused GP

1. encode axioms that detect unachieved goals

$$p_{ug}(X) \text{ :- } p_g(X) \wedge \neg p(X)$$

2. encode axioms that restrict action application based on learned rules

$$(\alpha_1)_\pi(X) \text{ :- } \bigwedge_{i=1,\dots,m} p_i^s(X_i^s) \wedge \bigwedge_{j=1,\dots,n} (p_j^g)_{ug}(X_i^g)$$

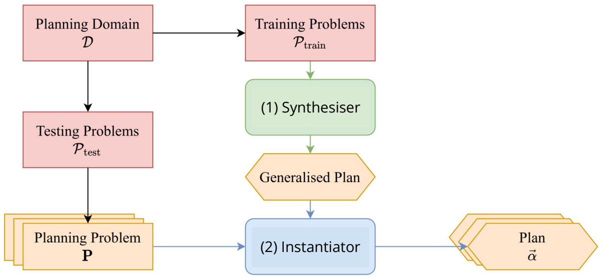
3. feed transformed PDDL problem into a planner that supports axioms

Methodology Summary

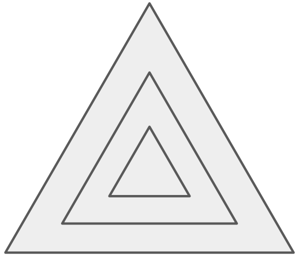
1. Synthesise plans via goal regression
   - goal regression greatly reduces synthesis costs
2. Instantiate plans via policy execution with conjunctive query algorithms
   - database query algorithms greatly reduce instantiation costs
3. Instantiate plans via search with PDDL axiom encodings
   - search provides optimal solutions and high solution quality

Experimental Results

Recall: 3 Primary GP Metrics



Synthesis Cost
Costs (e.g. data, time, memory) to synthesise a generalised plan



Instantiation Cost
Costs (e.g. time, memory) to instantiate a generalised plan on new problems

Solution Quality
Quality of plans returned from instantiating a generalised plan on new problems

Synthesis Experiments

Satisficing Planning Experiments

Optimal Planning Experiments

Synthesis Experiments

- 8 PDDL benchmark domains
- Compare against *3 configurations* of the Sketch Learner [5] generalised planner
- 32 GB memory
- 12 hour runtime limit
- 5 repeats per domain

[5] Dominik Drexler, Jendrik Seipp, Hector Geffner: Learning Sketches for Decomposing Planning Problems into Subproblems of Bounded Width. ICAPS 2022

Synthesis Results

Average time and memory usage (↓)

MOOSE uses <1GB memory and synthesises GPlans for all domains

	Time (s)				Memory (MB)			
	SLEARN-0	SLEARN-1	SLEARN-2	MOOSE	SLEARN-0	SLEARN-1	SLEARN-2	MOOSE
Barman	-	-	-	**202**	-	-	-	**184**
Ferry	21	12	**2**	9	184	134	76	**52**
Gripper	**3**	9	45	10	66	142	391	**64**
Logistics	-	-	-	**71**	-	-	-	**73**
Miconic	57	**1**	3	12	381	56	125	**52**
Rovers	-	-	-	**534**	-	-	-	**187**
Satellite	-	-	1559	**514**	-	-	7598	**82**
Transport	-	**12**	**12**	21	-	114	129	**80**

Synthesis Experiments

Satisficing Planning Experiments

Optimal Planning Experiments

Satisficing Planning Experiments

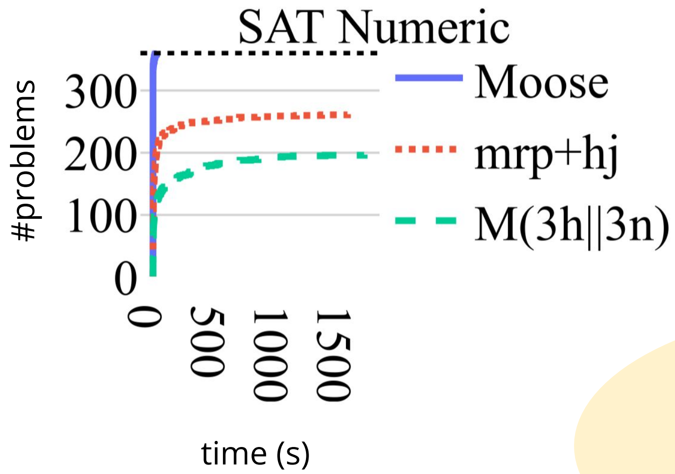
- 8 classical domains and 4 numeric domains
- Compare against:
  - Classical planners: Sketch Learner [5], LAMA [6]
  - Numeric planners: ENHSP(mrp+hj) [7], ENHSP(M(3h||3n) [8]
- 8 GB memory
- 30 minute runtime limit
- 5 repeats per problem

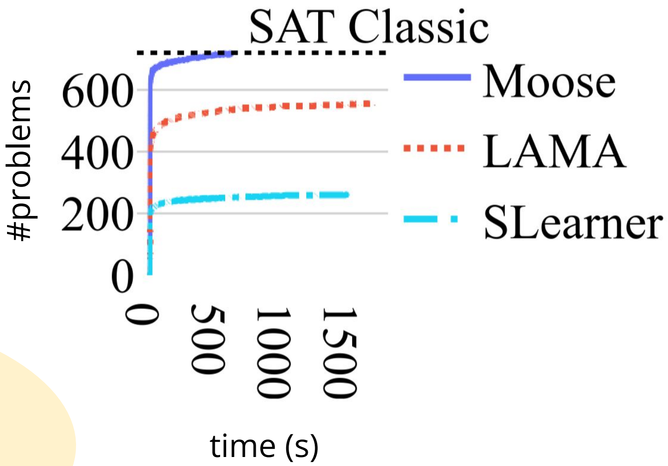
[5] Dominik Drexler, Jendrik Seipp, Hector Geffner: Learning Sketches for Decomposing Planning Problems into Subproblems of Bounded Width. ICAPS 2022
[6] Silvia Richter, Matthias Westphal: The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. J. Artif. Intell. Res. 39: 127-177 (2010)
[7] Enrico Scala, Alessandro Saetti, Ivan Serina, Alfonso Emilio Gerevini: Search-Guidance Mechanisms for Numeric Planning Through Subgoaling Relaxation. ICAPS 2020
[8] Dillon Z. Chen, Sylvie Thiébaux: Novelty Heuristics, Multi-Queue Search, and Portfolios for Numeric Planning. SOCS 2024

Satisficing Planning Results

Cumulative coverage (↑)

The number of problems (*y-axis*) that a planner solves within *n* seconds (*x-axis*)





MOOSE solves almost all problems faster than the baselines

Satisficing Planning Results

Coverage per domain (↑)

Domain	M(3h‖3n)	MRP+HJ	MOOSE
NFerry	60	61	**90.0**
NMiconic	63	71	**90.0**
NMinecraft	30	66	**90.0**
NTransport	44	64	**90.0**
∑ (360)	197	262	**360.0**

Domain	SLEARN-0	SLEARN-1	SLEARN-2	LAMA	MOOSE
Barman	0.0	0.0	0.0	49	**90.0**
Ferry	15.0	67.0	60.0	69	**90.0**
Gripper	59.6	50.8	33.0	65	**90.0**
Logistics	0.0	0.0	0.0	77	**89.6**
Miconic	68.8	72.6	67.8	77	**90.0**
Rovers	0.0	0.0	0.0	66	**90.0**
Satellite	0.0	29.2	34.6	89	**90.0**
Transport	0.0	63.0	46.8	66	**90.0**
∑ (720)	143.4	282.6	242.2	558	**719.6**

Synthesis Experiments

Satisficing Planning Experiments

Optimal Planning Experiments

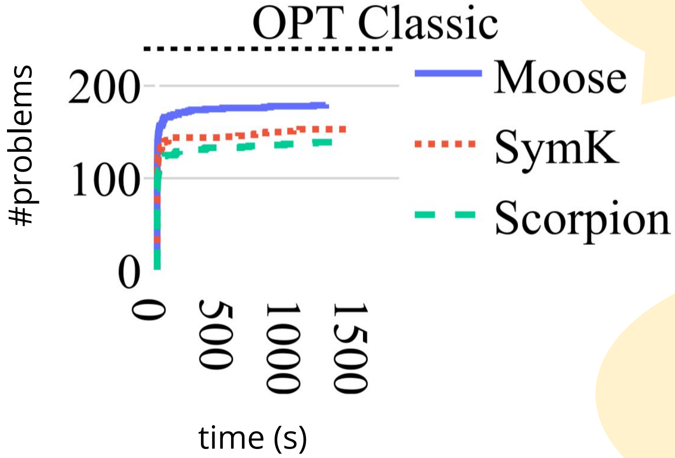
Optimal Planning Experiments

- 8 classical domains
- use SymK [9] as downstream planner that supports PDDL axioms
- Compare against SymK without axioms and Scorpion [10]
- 8 GB memory
- 30 minute runtime limit
- 5 repeats per problem

[9] David Speck, Jendrik Seipp, Álvaro Torralba: Symbolic Search for Cost-Optimal Planning with Expressive Model Extensions. J. Artif. Intell. Res. 82 (2025)
[10] Jendrik Seipp, Thomas Keller, Malte Helmert: Saturated Cost Partitioning for Optimal Classical Planning. J. Artif. Intell. Res. 67: 129-167 (2020)

Optimal Planning Results

Cumulative coverage (↑)



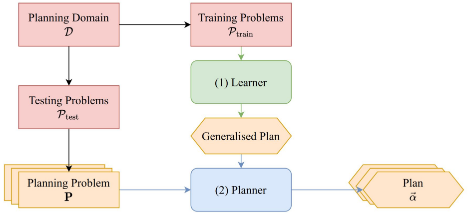
MOOSE solves more problems optimally in total

MOOSE usually improves upon its base planner (SymK)

Coverage table by domain (↑)

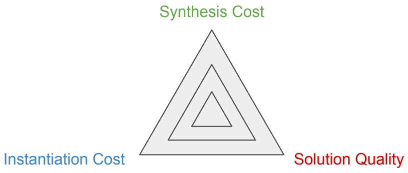
Domain	SCORPION	SYMK	MOOSE
Barman	0	12	**24.6**
Ferry	17	18	**30.0**
Gripper	7	**30**	27.0
Logistics	**22**	10	15.0
Miconic	**30**	**30**	**30.0**
Rovers	18	**20**	**20.0**
Satellite	**26**	21	21.4
Transport	**20**	13	15.0
$\sum$ (240)	140	154	**183.0**

Summary Slide



Problem

Synthesise generalised plans for solving families of planning problems



Method

Synthesise via goal regression
→ improve synthesis efficiency
Instantiate via database query algorithms
→ improve planning speed
Instantiate via encoding rules as pruning axioms
→ improve solution quality

	Time (s)				Memory (MB)			
	SLEARN-0	SLEARN-1	SLEARN-2	MOOSE	SLEARN-0	SLEARN-1	SLEARN-2	MOOSE
Barman	-	-	-	**202**	-	-	-	**184**
Ferry	21	12	**2**	9	184	134	76	**52**
Gripper	**3**	9	45	10	66	142	391	**64**
Logistics	-	-	-	**71**	-	-	-	**73**
Miconic	57	**1**	3	12	381	56	125	**52**
Rovers	-	-	-	**534**	-	-	-	**187**
Satellite	-	-	1559	**514**	-	-	7598	**82**
Transport	-	**12**	**12**	21	-	114	129	**80**

Theory

See paper for soundness and completeness theorems

Experiments

Improvements on the 3 metrics of synthesis cost, instantiation cost, and solution quality