Satisficing and Optimal Generalised Planning via Goal Regression

Dillon Z. Chen[1,2,3] Till Hofmann[4] Toryn Q. Klassen[1,3] Sheila A. McIlraith[1,3]

[1]Vector Institute, Toronto, Canada
[2]LAAS-CNRS, University of Toulouse, France
[3]University of Toronto, Canada
[4]RWTH Aachen University, Germany

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

Planning and RL: What's the Difference? Both Solve MDPs!

Planning	Reinforcement Learning
model-known	model-free or model-based
goal-conditioned, minimise cost	maximise reward
transitions modelled symbolically	transitions modelled as distributions
search algorithms: A*, GBFS, iLAO*, LRTDP	search algorithms: MTCS, UCT, TD(λ)
heuristic functions (cost-to-go estimator)	value functions (expected reward)
algorithms guided by models	algorithms guided by rewards

Problem Statement — Generalised Planning

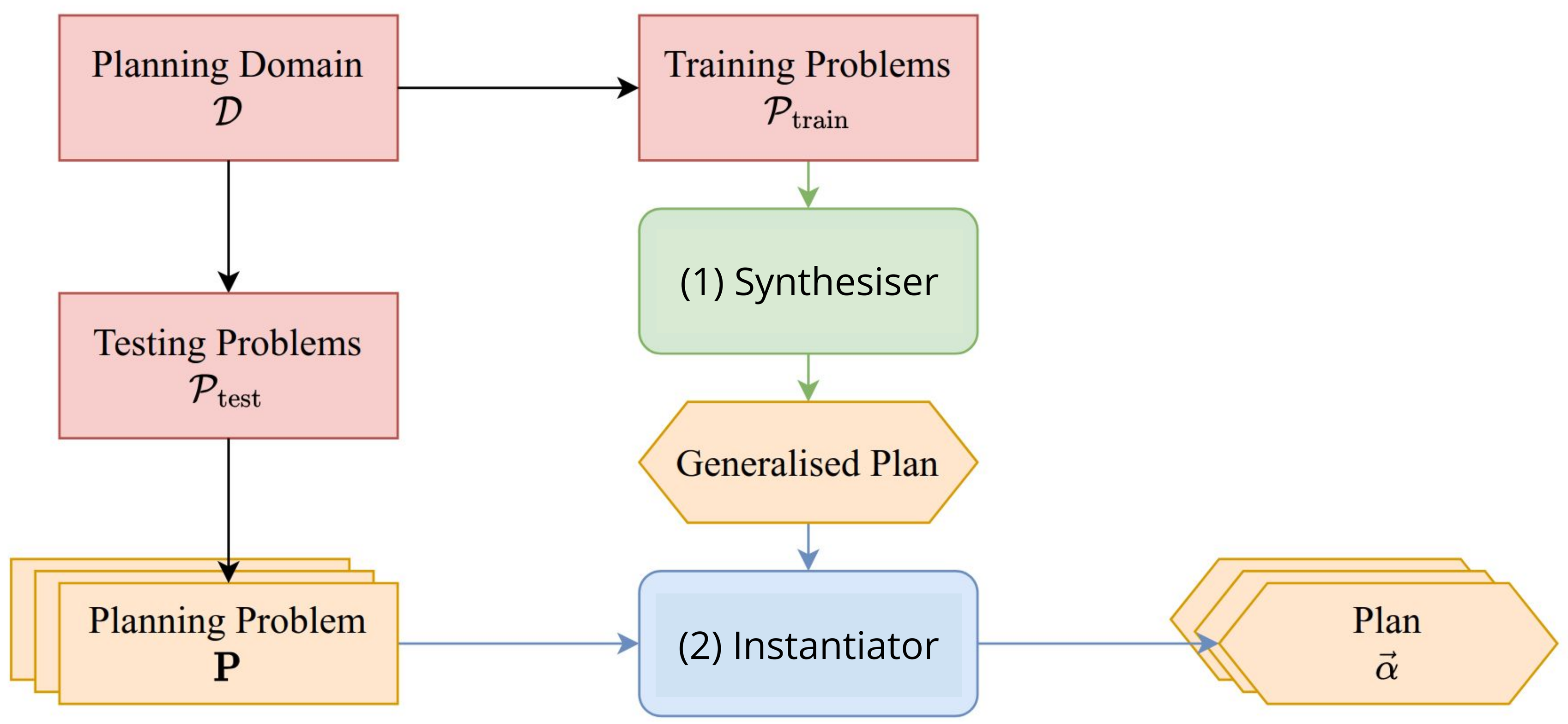
Generalised planning problem:

- a domain $\mathcal{D}$
- training planning problems $\mathcal{P}_{train}$ from $\mathcal{D}$
- testing planning problems $\mathcal{P}_{test}$ from $\mathcal{D}$

Generalised plan (GenPlan): is a *program* π 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



Synthesising GenPlans via Goal Regression

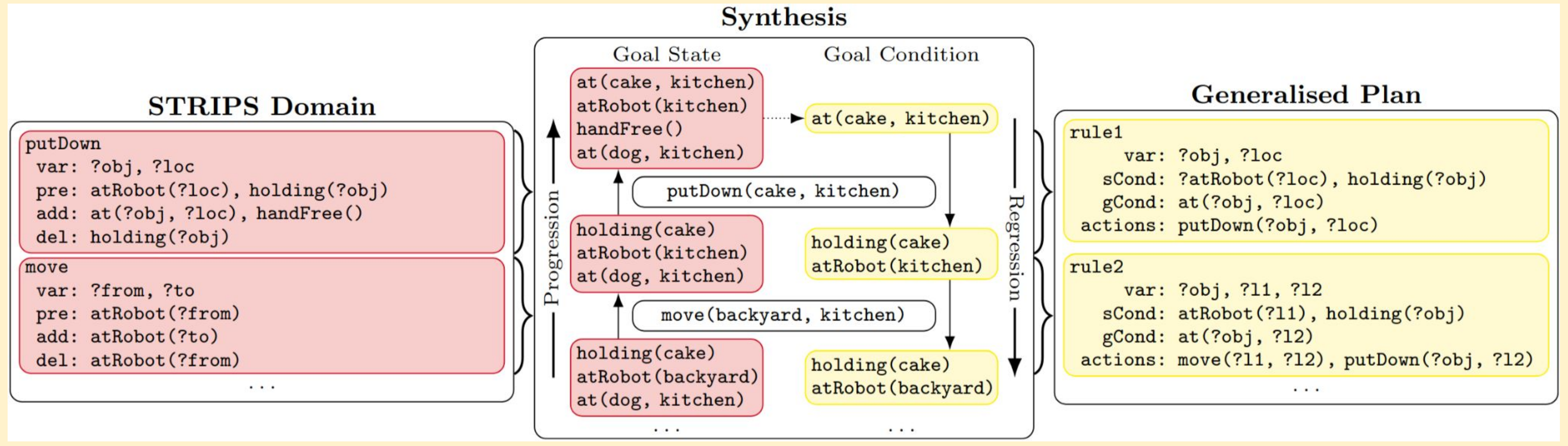
Synthesise a GPlan π in the form of a set of first-order rules from $\mathcal{P}_{train}$ by

1. compute optimal plans $\{\alpha_1, \ldots, \alpha_n\}$ for single goal atoms in some order $\{g_1, \ldots, g_n\}$ 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 $\langle \sigma_i, \alpha_i \rangle$ where $\alpha_i = a_1, \ldots, a_q$
3. lift the set of pairs $\langle \sigma_i, \alpha_i \rangle$ and goals g_i into a set of first-order rules

$$\{\exists\{X\} \underbrace{\wedge_{i=1,\ldots,m} p_i^s(X_i^s)}_{\text{state condition}} \wedge \underbrace{\wedge_{j=1,\ldots,n} p_j^g(X_j^g)}_{\text{goal condition}} \rightarrow \underbrace{a_1(X_1^a), \ldots, a_q(X_q^a)}_{\text{actions}}\}$$

- *Goal regression* 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

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



Instantiating GenPlans via Database Algorithms

speed focused GP

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

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

ground with first-order query algorithms

Instantiating GenPlans via Search

quality focused GP

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

1. encode axioms that detect unachieved goals

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

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

$$(a_1)_\pi(X) :- \wedge_{i=1,\ldots,m} p_i^s(X_i^s) \wedge \wedge_{j=1,\ldots,n} (p_j^g)_{ug}(X_j^g)$$

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

Experiments

Benchmarks: HUGE numbers of objects

	Max #Training Objects	Max #Testing Objects
Barman	27	853
Ferry	8	1461
Gripper	5	48500
Logistics	29	1260
Miconic	11	1950
Rovers	36	596
Satellite	43	402
Transport	17	354

Synthesis Experiments

Average time and memory usage (↓)

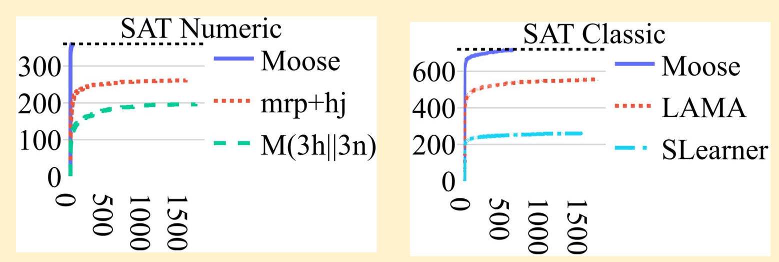
	Time (s)				Memory (MB)			
	SLEARN-0	SLEARN-1	SLEARN-2	MOOSE	SLEARN-0	SLEARN-1	SLEARN-2	MOOSE
Barman	-	-	-	**202**	-	-	-	**184**
Ferry	21	12	**2**	9	184	134	76	**52**
Gripper	**3**	9	45	10	66	142	391	**64**
Logistics	-	-	-	**71**	-	-	-	**73**
Miconic	57	**1**	3	12	381	56	125	**52**
Rovers	-	-	-	**534**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

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

Satisficing Planning Experiments

Cumulative coverage (↑)

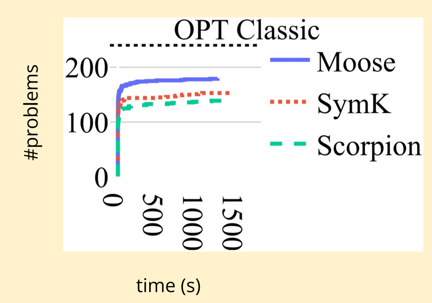
The number of problems (*y-axis*) that a planner solves within *n* seconds (*x-axis*)



MOOSE solves almost all problems faster than the baselines

Optimal Planning Experiments

Cumulative coverage (↑)



MOOSE solves more problems optimally in total

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
∑ (240)	140	154	**183.0**

MOOSE usually improves upon its base planner (SymK)