Learning and Transferring Relational Instance-Based Policies

Rocío García Durán,
Fernando Fernández and
Daniel Borrajo

Planning and Learning Group (PLG)
Departamento de Informática
Escuela Politécnica Superior
Universidad Carlos III de Madrid

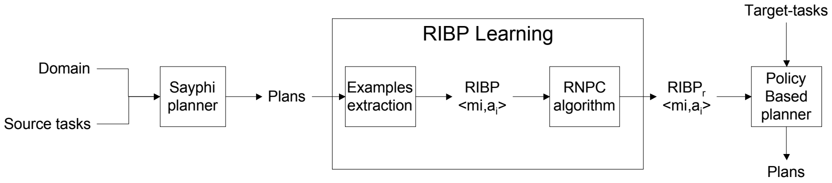
July 14, 2008

Content

Motivation

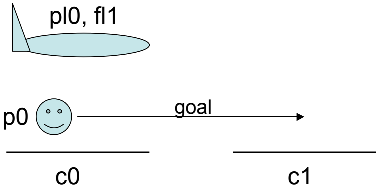
- Planners can solve simple problems optimally, but they can not solve most complex problems
- Solutions:
  - Manually defining efficient domain-independent heuristics
  - Learning control knowledge
- Traditionally, learning has been casted as a transfer learning approach (without explicitly using that terminology)
- Our goal is to transfer knowledge (a relational policy) learned in simple problems (source problems) to help planning systems to solve complex ones (target problems)
- Transferring of a policy is possible given that:
  - The same relational representation is used for all tasks in the same domain (universal policy)
  - A nearest neighbor approach is applied (partial matching)
  - The policy is simplified, so it is fast to use (utility problem)

The learning process



1. **Training**:
    - Generation of plans
    - Extraction of examples
    - Reduction of the examples(RNPC algorithm [Fernández and Isasi, 2004])
2. **Test**: more complex random target problems to solve using the RIBP$_r$.

Deliberative planning



state$_0$:

(at p0 c0) (at pl0 c0)
(fuel-level pl0 fl1)
(next fl0 fl1) (next fl1 fl2)
(next fl2 fl3) (next fl3 fl4)
(next fl4 fl5) (next fl5 fl6)

goal: (goal-at p0 c1)

Solution plan:

action$_0$: (board p0 pl0) $\longrightarrow$ state$_1$: (in p0 pl0), (at pl0 c0), …
action$_1$: (fly pl0 c0 c1 fl1 fl0) $\longrightarrow$ state$_2$: (in p0 pl0), (al pl0 c1), …
action$_2$: (debark p0 pl0 c1) $\longrightarrow$ FINISH

A Relational Instance-Based Policy (RIBP)

A relational policy, π, is defined by:

- $\pi : M \rightarrow A$ is a mapping from a meta-state to an action

Meta-state m_i (s_i+pending goals)		**Action** a_i	
m_0	(at pl0 c0), (at p0 c0), (fuel-level pl0 fl1), (next pl0 pl1)... , (goal-at p0 c1)	a_0	(board p0 pl0 c0)
m_1	(at pl0 c0), (in p0 pl0), (fuel-level pl0, fl1), (next pl0, pl1)..., (goal-at p0 c1)	a_1	(fly pl0 c0 c1 fl1 fl0)
m_2	(at pl0 c1), (in p0 pl0), (fuel-level pl0, fl0), (next pl0, pl1)..., (goal-at p0 c1)	a_2	(debark p0 'pl0 c1)

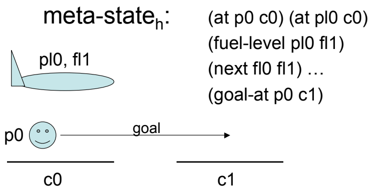
The **RIBP**, π, is defined by a tuple $< P, d >$, where:

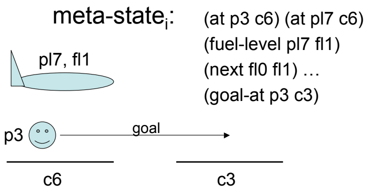
- P: set of tuples $< m_i, a_i >$
- d: relational distance metric
- $\pi(m) = \arg_a \min_{(<m',a'>\in P)} dist(m, m')$

The RIBL distance

- The RIBL distance [Kirsten, Wrobel and Horváth, 2001]
  - $d(m_1, m_2) = \sqrt{\frac{\sum_{k=1}^{K} w_k d_k(m_1,m_2)^2}{\sum_{k=1}^{K} w_k}}$
  - $d_k(m_1, m_2) = \frac{1}{N} \sum_{i=1}^{N} \min_{p \in P_k(m_2)} d'_k(P_k^i(m_1), p)$
  - $d'_k(p_k^1, p_k^2) = \sqrt{\frac{1}{M} \sum_{l=1}^{M} \delta(p_k^1(l), p_k^2(l))}$ where $p_k^i(l)$ is the lth argument of literal p_k^i, and $\delta(p_k^1(l), p_k^2(l))$ returns 0 if both values are the same, and 1 if they are different

An example





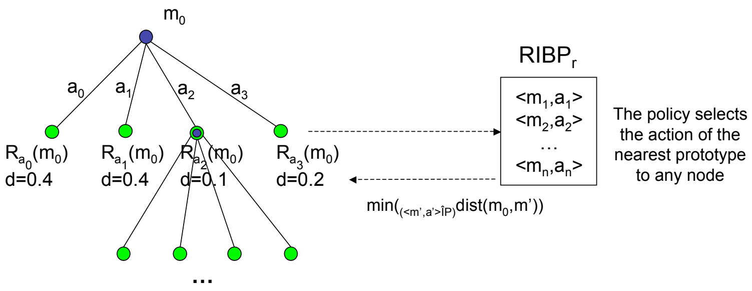
$$d(m_h, m_i) = \sqrt{\frac{d_{at}(m_h,m_i)^2 + d_{in}(m_h,m_i)^2 + d_{fuel-level}(m_h,m_i)^2 + d_{next}(m_h,m_i)^2 + d_{goal-at}(m_h,m_i)^2}{5}} = 0.707$$

- **Problem**: The distance between two instances depends on the similarity between the names of both sets of objects
- **Partial solution**: The meta-states are renamed to keep some kind of relevance level of the objects

Example of use

Sayphi (=MetricFF): heuristic planner with forward search

Heuristic+RIBP$_r$ (greedy algorithm):



a_i: suggested action by the heuristic

$R_{a_i}(m_0)$: renamed meta-state m_0 with the a_i

● Evaluated node with the heuristic

🟢 Renamed node and evaluated node with RIBP$_r$

Experiments in the Zenotravel domain

- Source problems: 250 random problems; 50 with (1,3,1); 100 with (1,3,2); and 100 with (2,3,3), where (planes,cities,persons-goals)
- Training time bound: 180 seconds
- RIBP: 1509 training instances from the solution plans
- RIBP$_r$: average number of prototypes: 18 (after running RNPC 10 times)
- Target problems:
  - 180 problems of different **complexity**. Nine subsets of 20 problems: (1,3,3), (1,3,5), (2,5,10), (4,7,15), (5,10,20), (7,12,25), (9,15,30), (10,17,35) and (12,20,40)
  - 20 problems from the third IPC
- Test time bound: 1800 seconds (standard in the IPC)

Results in the Zenotravel domain (I)

#Problems (#goals)	Approach	Solved	Time	Cost	Nodes
20 (3)	Sayphi	20	0.46	166	683
	RIBP$_r$	20	1.47	275	296
20 (5)	Sayphi	20	0.49	236	868
	RIBP$_r$	20	1.40	291	314
20 (10)	Sayphi	20	5.84	535	3277
	RIBP$_r$	20	8.11	719	743
20 (15)	Sayphi	20	82.54	749	10491
	RIBP$_r$	20	40.69	1285	1307
20 (20)	Sayphi	20	607.32	1293	21413
	RIBP$_r$	20	133.25	2266	2287
20 (25)	Sayphi	13	629.06	961	26771
	RIBP$_r$	20	112.41	1880	1896
20 (30)	Sayphi	8	221.92	712	9787
	RIBP$_r$	20	72.22	1340	1353
20 (35)	Sayphi	0	—	—	—
	RIBP$_r$	15			
20 (40)	Sayphi	1	30.20	120	1420
	RIBP$_r$	6	25.82	298	301

Results in the Zenotravel domain (II)

	solved	time	cost	nodes
Sayphi	18	2578.84	512	30023
RIBP	20	2761.69	1081	1102
RIBP_r	20	111.62	1248	1267

Conclusions

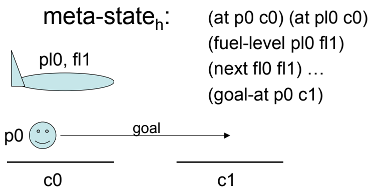
- We have used a Relational Nearest Neighbor approach to learn a reduced policy ($< state - goal, action >$), as control knowledge, to guide the search in planning
- RIBP implements a partial matching of control knowledge
- RIBP$_r$ reduces time and memory in different more complex tasks
- RIBP$_r$ can solve more problems although with worse quality

In the future

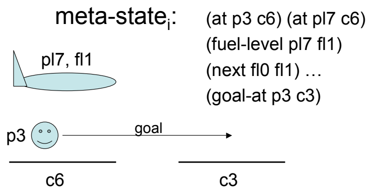
- Extend the experiments to others planners and other domains (IPC08)
- Focus on the distance metric in planning (enumerating variables, splitting goals, creating more relation levels, etc)
- Study and compare different distance metrics for planning domains
- Extend the idea to probabilistic domains where scale-up is more complex

Thank you

An example



(at p0 c0) (at pl0 c0)
(fuel-level pl0 fl1)
(next fl0 fl1) …
(goal-at p0 c1)



(at p3 c6) (at pl7 c6)
(fuel-level pl7 fl1)
(next fl0 fl1) …
(goal-at p3 c3)

$$d(m_h, m_i) = \sqrt{\frac{d_{at}(m_h,m_i)^2+d_{in}(m_h,m_i)^2+d_{fuel-level}(m_h,m_i)^2+d_{next}(m_h,m_i)^2+d_{goal-at}(m_h,m_i)^2}{5}} = 0.707$$

$$d_{at}(m_h, m_i) = \frac{1}{2}(min(1.0, 1.0) + min(1.0, 1.0)) = 1.0$$

d'_{at}	(at p3 c6)	(at pl7 c6)
(at p0 c0)	1.0	1.0
(at pl0 c0)	1.0	1.0

$$d'_{at}((at\ p0\ c0), (at\ p3\ c6)) = \sqrt{\frac{1+1}{2}} = 1.0$$

Renaming the objects

m_h: (at p0 c0)	→	m_h: (at **@p0** **@c0**)	→	m_h: (at **@p0** **@c0**)	→	m_h: (at **@p0** **@c0**)
(at pl0 c0)		(at **@pl0** **@c0**)		(at **@pl0** **@c0**)		(at **@pl0** **@c0**)
(fuel-level pl0 fl1)		(fuel-level **@pl0** fl1)		(fuel-level **@pl0** **fl1**)		(fuel-level **@pl0** **@fl0**)
(next fl0 fl1)		(next fl0 fl1)		(next **fl0** **fl1**)		(next **@fl1** **@fl0**)
(next fl1 fl2)		(next fl1 fl2)		(next **fl1** **fl2**)		(next **@fl0** **@fl2**)
…		…		…		…
(goal-at p0 c1)		(goal-at **@p0** **c1**)		(goal-at **@p0** **@c1**)		(goal-at **@p0** **@c1**)

a_h: (board **p0** **pl0** **c0**)

m_i: (at p3 c6)	→	m_i: (at **@p0** **@c0**)	→	m_i: (at **@p0** **@c0**)	→	m_i: (at **@p0** **@c0**)
(at pl7 c6)		(at pl7 **@c0**)		(at pl7 **@c0**)		(at pl7 **@c0**)
(fuel-level pl7 fl1)		(fuel-level **@pl0** fl1)		(fuel-level **@pl0** **fl1**)		(fuel-level **@pl0** **@fl0**)
(next fl0 fl1)		(next fl0 fl1)		(next **fl0** **fl1**)		(next **@fl1** **@fl0**)
(next fl1 fl2)		(next fl1 fl2)		(next **fl1** **fl2**)		(next **@fl0** **@fl2**)
…		…		…		…
(goal-at p3 c3)		(goal-at **@p0** **c3**)		(goal-at **@p0** **@c1**)		(goal-at **@p0** **@c1**)

a_i: (board **p3** **pl7** **c3**)

Renaming with this order we try to keep some kind of relevance level of the objects to find a better similarity between two instances.