Extending Graph Neural Networks for Generalized Stochastic Planning

Ziqi Zhang Florian Geißer

The Australian National University

Computing General Policies for Planning Problems

Policy:

- Mapping from states to actions
- Solution to a specific planning problem

General Policy:

- Applicable to every problem of a given domain class
- Policies commonly represented as Neural Networks

Computing General Policies for Planning Problems

Different architectures support different modeling languages

- PDDL (classical planning / SSPs):
  - STRIPS Hypergraph Networks (Shen et al. 2020)
  - Action Schema Networks (Toyer et al. 2018)
- RDDL (MDPs)
  - SymNet (Garg et al. 2020)
  - TrapsNet (Garg et al. 2019)

Our work: A novel network architecture for planning

- builds on the ideas of TrapsNet
  - but allows fluents of arbitrary size
- is applied to RDDL problems
  - but is in principle independent of the modeling language
- allows to train a graph network on small instances
- evaluation on IPC 2014 domains shows promising results

Relational Markov Decision Process

A Relational Markov Decision Process consists of:

- A set of types T
- state, action, and static predicate symbols F
  - denote relations of and between typed objects
- a lifted transition function P and reward function R
  - describes the dynamics of the first-order MDP

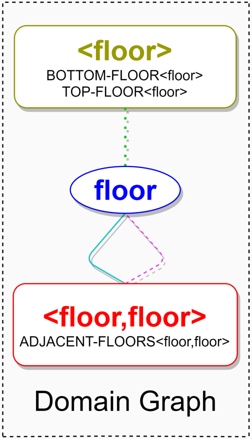
Given a set of objects, applying predicates to type-consistent objects is known as the process of grounding and yields a factored MDP.

Running Example: Elevator domain

- Types: *floor*, *elevator*
- State predicates:
  - *person-waiting-at-floor(elevator, floor)*
  - *elevator-at-floor(elevator, floor)*
- Action predicates:
  - *go-up(elevator)*, *go-down(elevator)*
  - *open-door(elevator)*, *close-door(elevator)*
- Static predicates:
  - *TOP-FLOOR(floor)*
  - *BOTTOM-FLOOR(floor)*
  - *ADJACENT-FLOORS(floor, floor)*

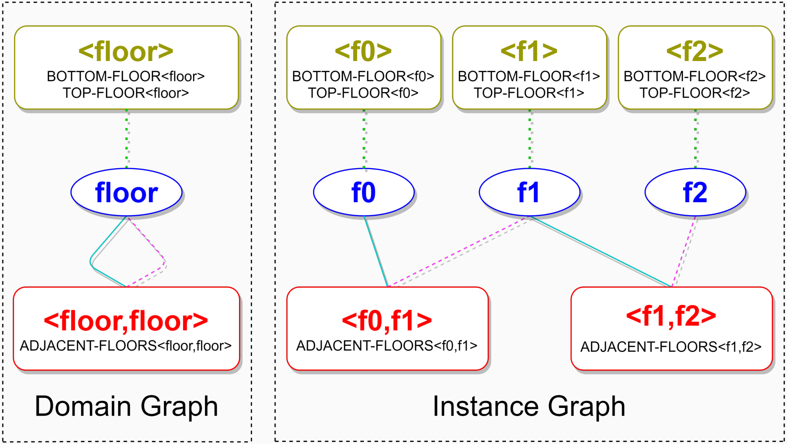
Domain Graph

A domain graph shows relations between types



Instance Graph

An instance graph shows relations between objects



Network Architecture

- Domain/Instance are used in a graph neural network
- Every instance graph vertex v has an initial embedding h_0^v
- Embedding at step $k+1$ is computed by forward pass over the network at step k
  - we use the attention mechanism to aggregate a set of vectors with unknown cardinality
- Final policy is computed by action decoder as a distribution over grounded actions

Empirical Evaluation

- Four domains of the IPPC 2014
- Network trained on the three smallest instances
- We compare against PROST (Keller et al. 2012)
- Our empirical evaluation is only a preliminary study

Evaluation

	Tamarisk domain		Wildfire domain	
Instance	**THTS**	**Network**	**THTS**	**Network**
1(trained on)	-151±19.7	**-146±15.4**	-647.1± 288	**-548.2±300**
2(trained on)	-542±27.7	**-530±24.2**	-11201.3±499	**-10048.1±408**
3(trained on)	**-206±27.3**	-222±23.8	**-1694.3±592**	-1890.2±550
4	-826±28.4	**-822±28.6**	-20126.8±1180	**-14616.3±816**
5	-723.9±41.4	**-655±38**	-2905.0±686	**-1515.6±423**
6	-1071±26.2	**-1045±33.2**	-25866.2±864	**-8862.6±1030**
7	-891±43.2	**-823±41.7**	-8816.2±748	**-6845.2±646**
8	-1285±23.5	**-1251±28.2**	-15811.8±1400	**-11124±910**
9	-902±53.2	**-860±59.9**	-14457.6±629	**-8721.6±906**
10	-1346±36.3	**-1290±39.3**	-21766.5±1110	**-11331.1±619**

	Elevator domain		Sysadmin domain	
Instance	**THTS**	**Network**	**THTS**	**Network**
1(trained on)	**-42.5±2.5**	-45.8±2.6	340.1±3.7	339.5±4.8
2(trained on)	-23.8±2.2	-23.6±2.6	**315.3±7.2**	303.5±9.9
3(trained on)	**-61.6±1.9**	-62.6±1.9	**550.2±14.1**	541.4±13.7
4	**-54.2±4.2**	-96.7±5.1	**495.4±16.3**	459.8±14.3
5	**-64.9±4.6**	-104.2±4.9	581.0±17	**587.9±15.2**
6	**-83.3±3.8**	-120.0±3.8	529.8±15.4	**553.6±15.8**
7	**-79.4±5.4**	-133.2±6.3	611.0±14.7	**683.7±16.1**
8	**-88.2±5.2**	-151.3±5.8	505.3±16.5	**532.1±13.6**
9	**-107.5±5.4**	-160.8±5.5	739.9±17.1	**825.5±14.3**
10	**-66.5±5.9**	-117.0±7.7	553.8±14.4	**606.5±15.6**

Conclusion

- Novel network architecture based on domain graphs
- Allows for problems with arbitrary predicate size
- Generalizes well across instances of different size
- Interesting aspect: network is in principle independent of the modeling language used

References

Garg, Sankalp, Aniket Bajpai, and Mausam (2019). “Size Independent Neural Transfer for RDDL Planning”. In: *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)*. Ed. by Nir Lipovetzky, Eva Onaindia, and David E. Smith. AAAI Press, pp. 631–636.

— (2020). “Symbolic Network: Generalized Neural Policies for Relational MDPs”. In: *International Conference on Machine Learning*, pp. 3397–3407.

Keller, Thomas and Patrick Eyerich (2012). “PROST: Probabilistic Planning Based on UCT”. In: *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*. Ed. by Lee McCluskey, Brian Williams, José Reinaldo Silva, and Blai Bonet. AAAI Press, pp. 119–127.

Shen, William, Felipe Trevizan, and Sylvie Thiébaux (2020). “Learning Domain-Independent Planning Heuristics with Hypergraph Networks”. In: *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)*. Ed. by J. Christopher Beck, Erez Karpas, and Shirin Sohrabi. AAAI Press, pp. 574–584.

Toyer, Sam, Felipe Trevizan, Sylvie Thiébaux, and Lexing Xie (2018). “Action Schema Networks: Generalised Policies with Deep Learning”. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*. AAAI Press, pp. 6294–6301.