Symbolic Network: Generalized Neural Policies for Relational MDPs

Sankalp Garg

ICML 2020

Joint Work with Aniket Bajpai, Mausam

Data Analytics & Intelligence Research

Indian Institute of Technology, Delhi

(https://www.cse.iitd.ac.in/dair)

Overview

- Focus on Relational MDP: Compact first order representation
  - Goal: Find generalized policy to run out-of-the-box on new problem instance
  - Attractive: If learned, sidesteps the “curse of dimensionality”
  - Introduced in 1999 [Boutilier et al], but research died down because the problem is too hard
  - No relational planners participated in International Probabilistic Planning Competition (IPPC) since 2006!

- First neural model to generalize policies for RMDP in RDDL [Sanner 2010]
  - We learn a policy on small problem sets using Neural Network
  - Given any new problem, we output a (good enough) policy without retraining

Running Example

Image courtesy: [Scott Sanner](), [RDDL Tutorial]()

State Variables (18):

Burning (x1, y1), Burning (x2, y1), Burning (x3, y1),
Burning (x1, y2), Burning (x2, y2), Burning (x3, y2),
Burning (x1, y3), Burning (x2, y3), Burning (x3, y3)

Out-of-fuel (x1, y1), Out-of-fuel (x2, y1), Out-of-fuel (x3, y1),
Out-of-fuel (x1, y2), Out-of-fuel (x2, y2), Out-of-fuel (x3, y2),
Out-of-fuel (x1, y3), Out-of-fuel (x2, y3), Out-of-fuel (x3, y3)

Actions (19):

Cut-out (x1, y1), Cut-out (x2, y1), Cut-out (x3, y1),
Cut-out (x1, y2), Cut-out (x2, y2), Cut-out (x3, y2),
Cut-out (x1, y3), Cut-out (x2, y3), Cut-out (x3, y3)

Put-out (x1, y1), Put-out (x2, y1), Put-out (x3, y1),
Put-out (x1, y2), Put-out (x2, y2), Put-out (x3, y2),
Put-out (x1, y3), Put-out (x2, y3), Put-out (x3, y3)

Finisher

Markov Decision Process: MDP

- $m \times n$ field – 2^{2*m*n} states
- With different targets as well!

Difficulties

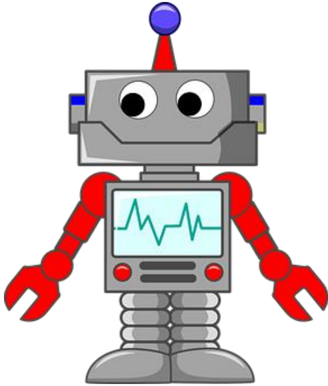
- Curse of dimentionality : Difficult to represent states
- For learning policy (π), we need to learn #actions in order of #states.

Relational Markov Decision Process: RMDP

Compact representation considering that real life objects share properties.

Represented with set of state variables:

- Burning$(?x, ?y)$



For $m \times n$ field:

- 2 state predicates
- Number of states are still the same, but representation is compact

Relational Markov Decision Process: RMDP

- $\mathcal{C}$: A set of classes denoting objects (e.g. Coordinate x, Coordinate y)
- $\mathcal{SP}$: A set of state predicates
  - $Fluent$: Changes with time (e.g. Burning, Out-of-Fuel)
  - $Non-fluent$: Static with time (e.g. X-Neighbor, Y-Neighbor)
- $\mathcal{A}$: A set of action templates (e.g. Put-out, Cut-out)
- $\mathcal{O}$: A set of objects (e.g. x1, x2, y1, y2)
- $\mathcal{T}$: Transition function template

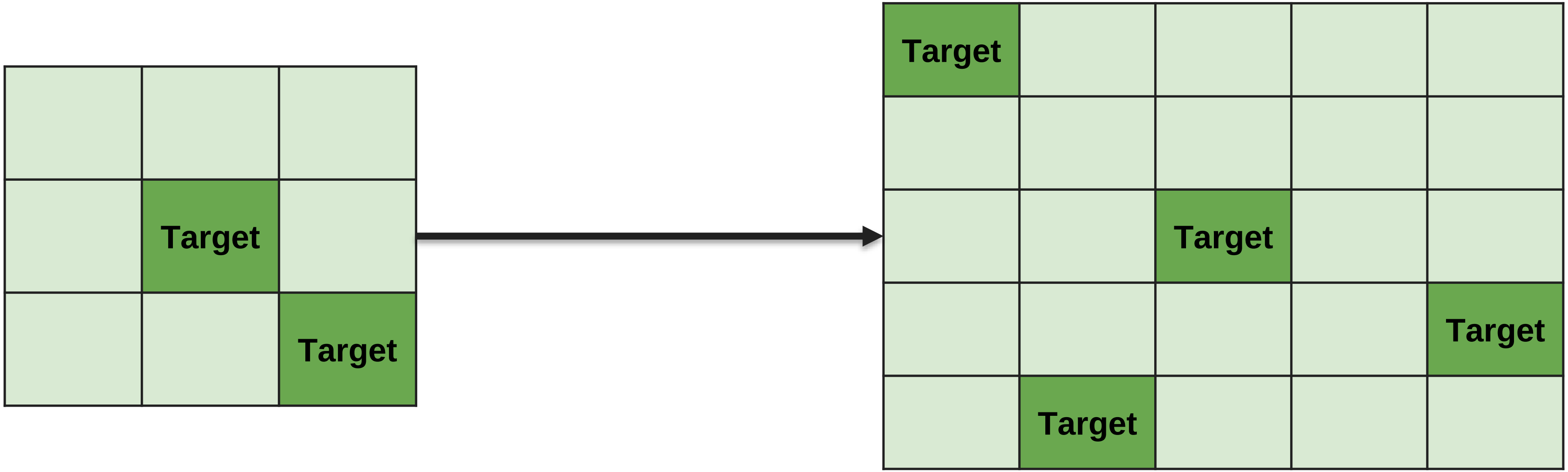
$$p(burning(x_i, y_i) = true) = \frac{1}{1+e^{4.5-k}} \text{ , } where\ k = \#\ neighbours\ on\ fire$$

Still want to learn a policy $\pi : \mathcal{S} \rightarrow A$,

➢ But this time utilize the compact representation to share information

Problem

- Learn a generalized policy π^D which works on all instances of domain D.
- Should be able to solve any RMDP instance of D without human interference.
- Policy should be learnt on some small problem instances (fixed set)
- Learnt policy should work out-of-the-box on larger problem instance.



Overview of SymNet

- **Problem Representation:**

  Instance $\longrightarrow$ Graph

- **Representation Learning:**

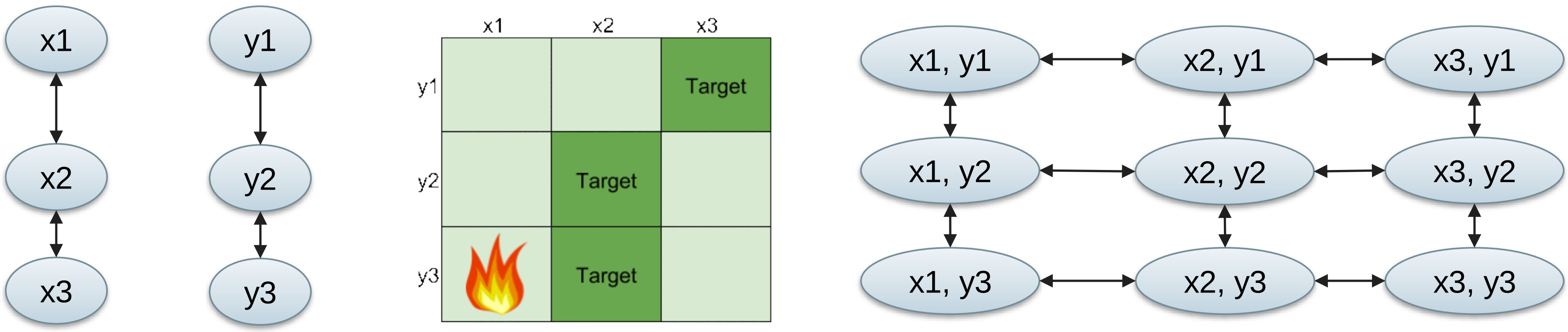
  Graph $\xrightarrow{\text{Graph Neural Network}}$ State embedding

- **Policy Learning:**

  State Embedding $\xrightarrow{\text{Neural Network}}$ Action Embedding $\longrightarrow$ Policy

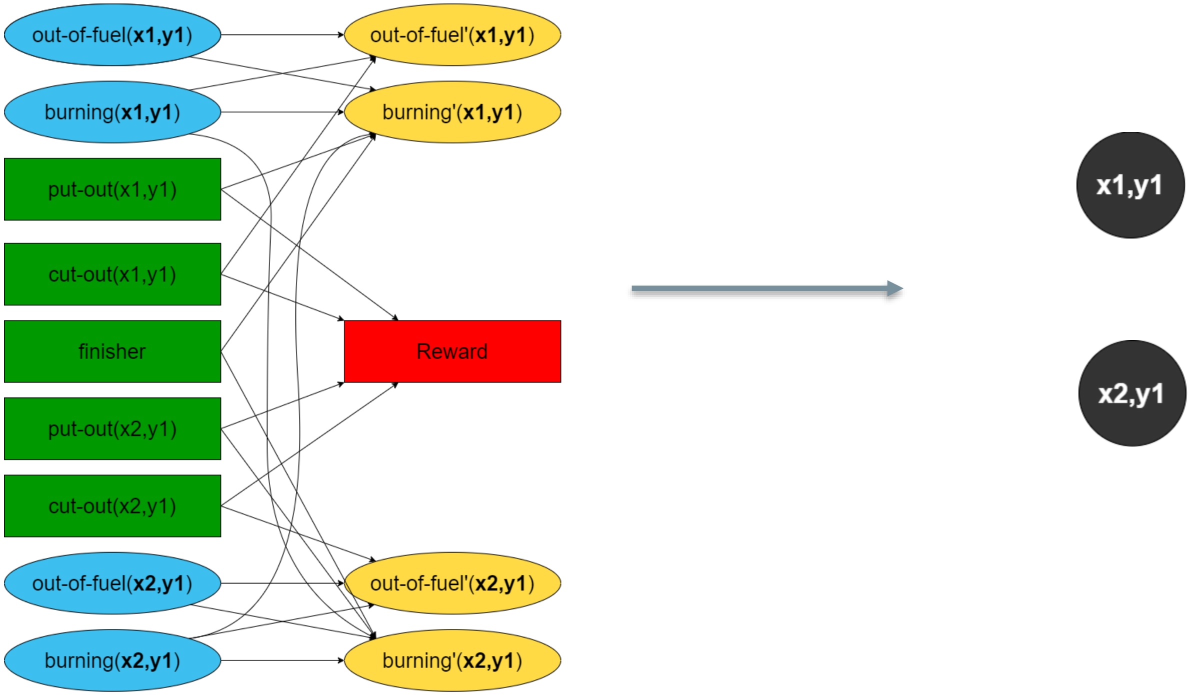
Challenge 1: Instance Graph Construction

- Do we choose objects as nodes?
- If we choose object as node, then which objects?
- How do we add edges to the graph?



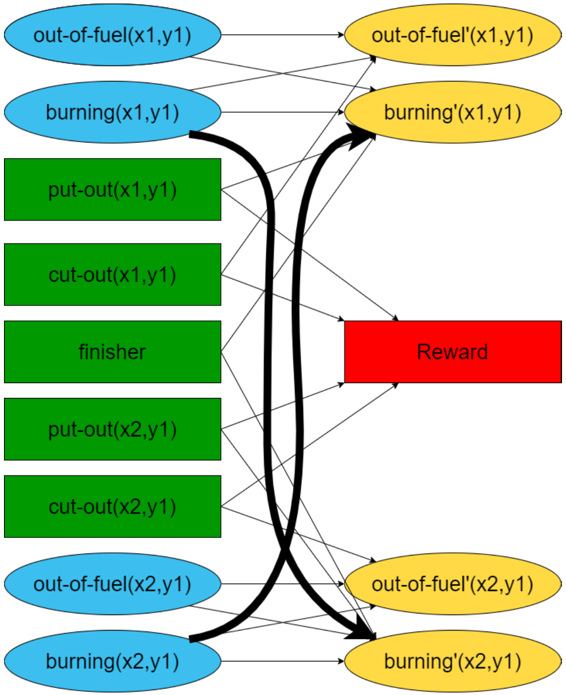
Solution 1: Dynamic Bayes Network (DBN)

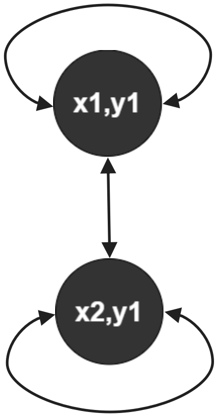
- Every instance of domain compiles to ground DBN
- State and action variables parameterized over sequence of objects as nodes



Solution 1: Dynamic Bayes Network (DBN)

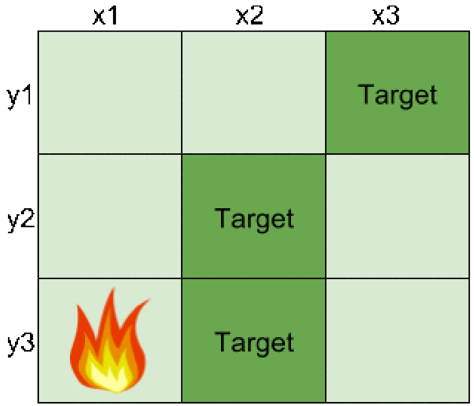
- Edge between two nodes such that they inter-influence in DBN.

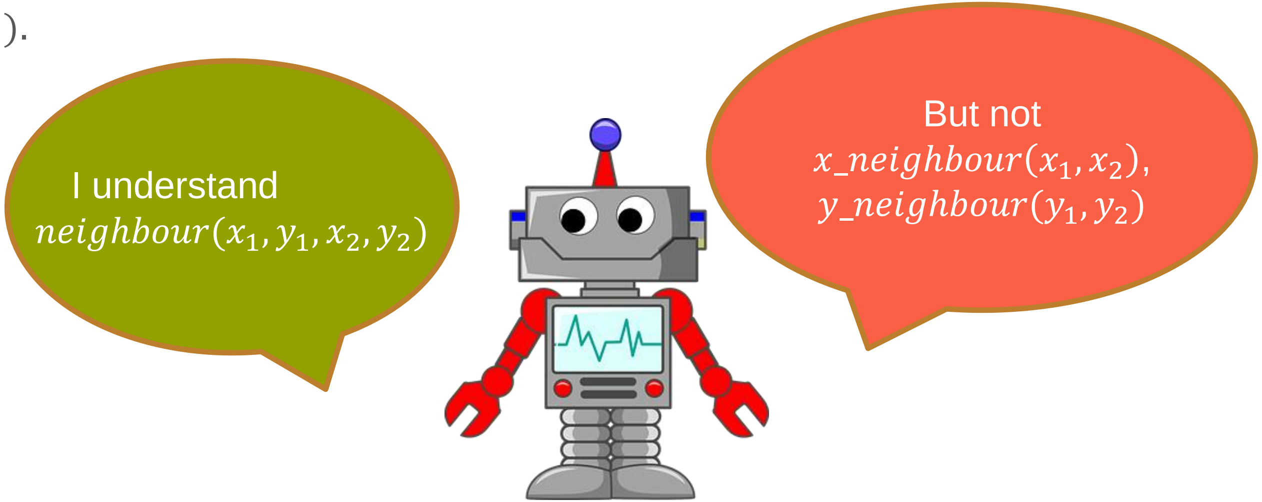




Challenge 2: Multiple RDDL Representations

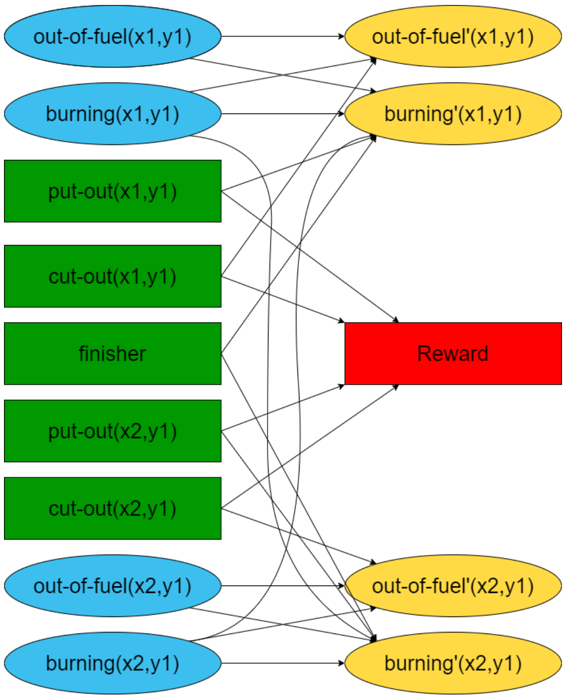
- Multiple RDDL representations of a domain make it hard to design a model
- E.g. Connection between points (x_1, y_1) and (x_2, y_2) can be represented as:
    - $x\_neighbour(x_1, x_2)$ and $y\_neighbour(y_1, y_2)$
    - $neighbour(x_1, y_1, x_2, y_2)$.





Solution 2: Dynamic Bayes Network (Again!!)

- DBN specifies dynamics of domain → hence RDDL representation independent



Overview of SymNet

- **Problem Representation:**

  Instance $\longrightarrow$ DBN $\longrightarrow$ Graph

- **Representation Learning:**

  Graph $\xrightarrow{\text{Graph Neural Network}}$ State embedding

- **Policy Learning:**

  State Embedding $\xrightarrow{\text{Neural Network}}$ Action Embedding $\longrightarrow$ Policy

Overview of SymNet

- **Problem Representation:**

  Domain $\longrightarrow$ DBN $\longrightarrow$ Graph

- **Representation Learning:**

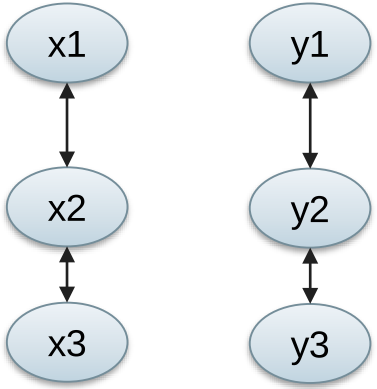
  Graph $\xrightarrow[\text{[Veličković, et. al. 2018]}]{\text{Graph Attention Networks}}$ State embedding

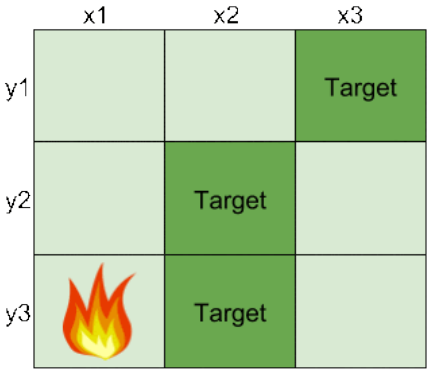
- **Policy Learning:**

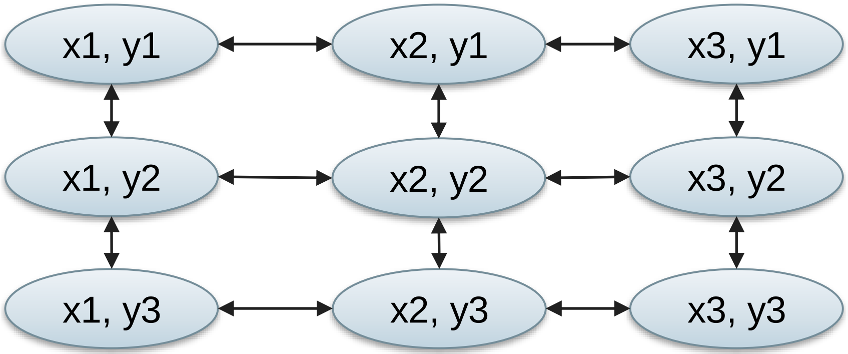
  State Embedding $\xrightarrow{\text{Neural Network}}$ Action Embedding $\longrightarrow$ Policy

Challenge 3: Action Template Parameterization

- What should be parameters of action template?
- Action can span object sequence not appearing in graph.
- E.g. *Finisher*

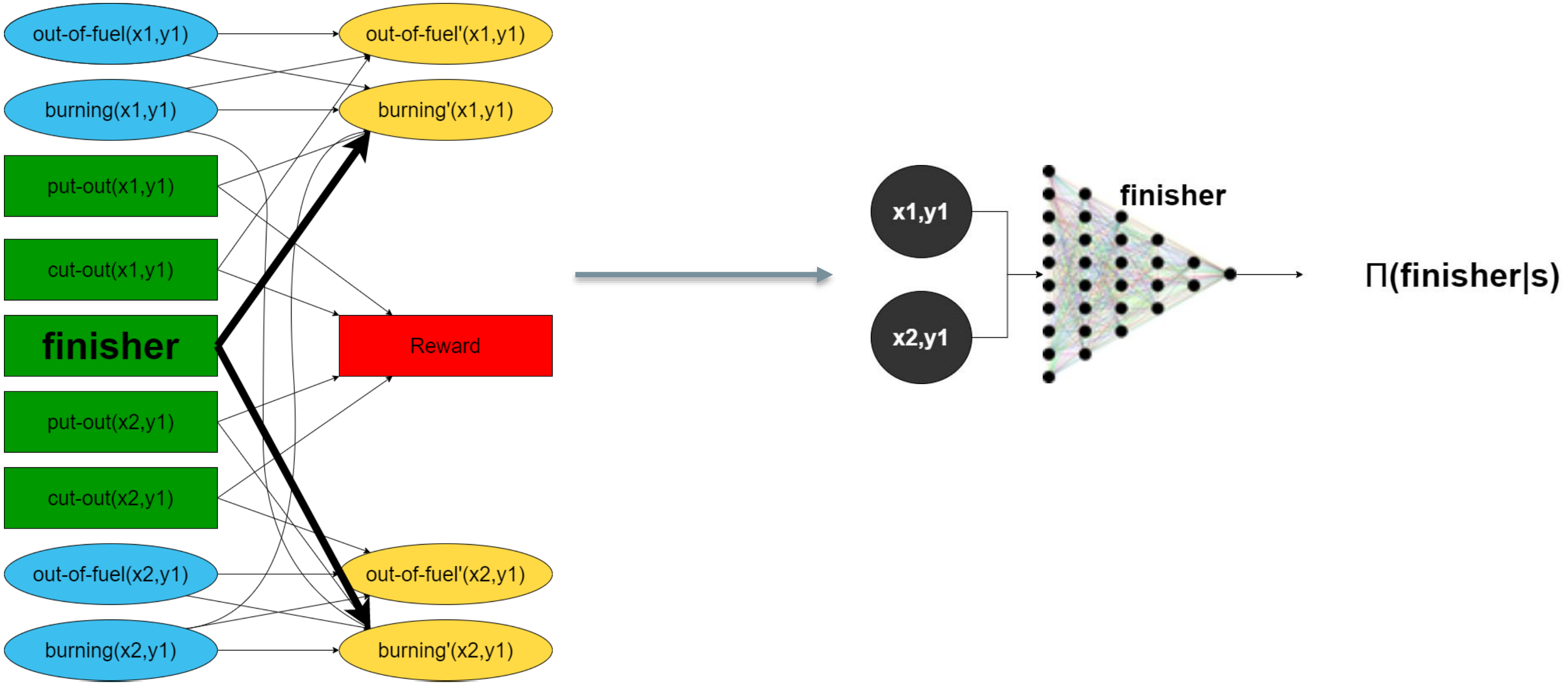






Solution 3: Dynamic Bayes Network (Yet Again!!)

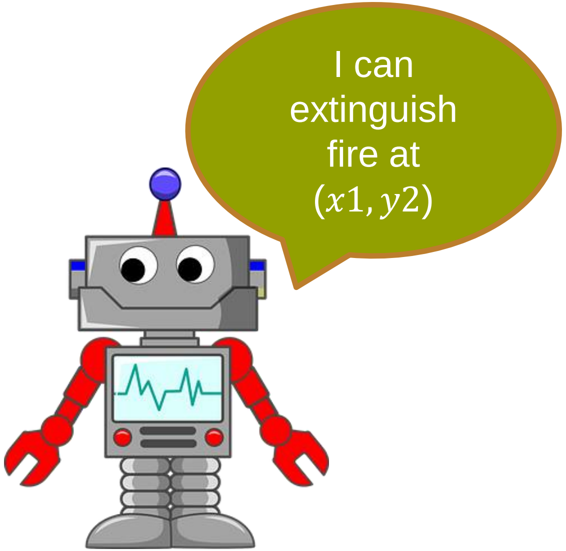
- DBN also represents state variables influenced by actions.
- Nodes influenced by actions will be parameters to the action module.

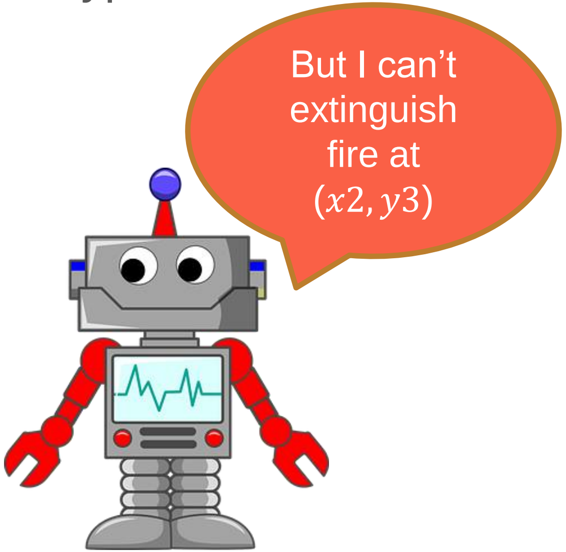


Challenge 4: Size Invariance

- Standard RL models every ground action explicitly, which makes it difficult to learn new action.

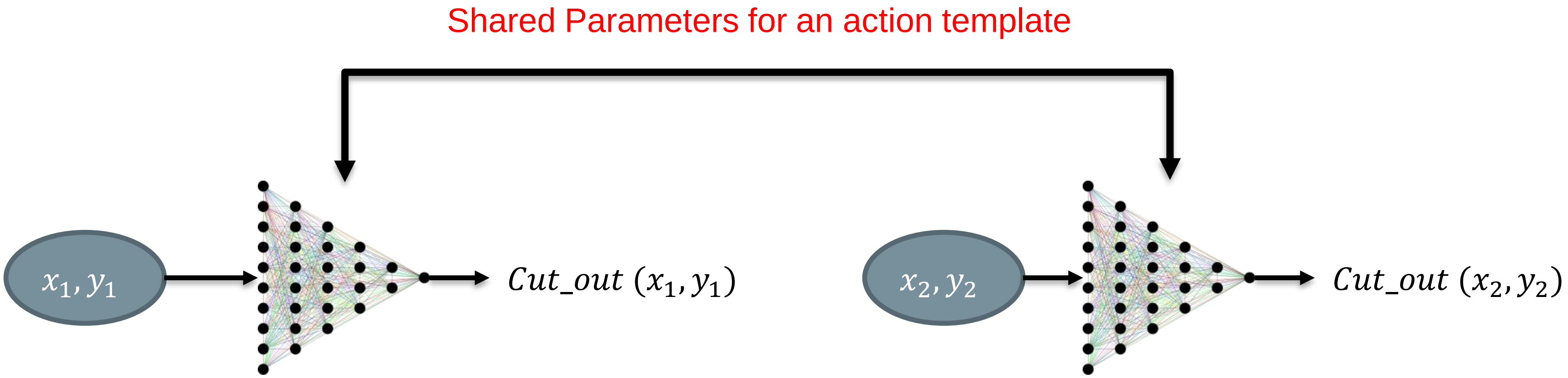
- Does not utilize the similarity between the same type of actions





Solution 4: Modeling action template

- To achieve size independency, we learn function action templates which parameterize on objects instead of modelling ground actions independently.



[[1] Garg et. al., ICAPS 2019]

Overview of SymNet

- **Problem Representation:**

  Instance $\longrightarrow$ DBN $\longrightarrow$ Graph

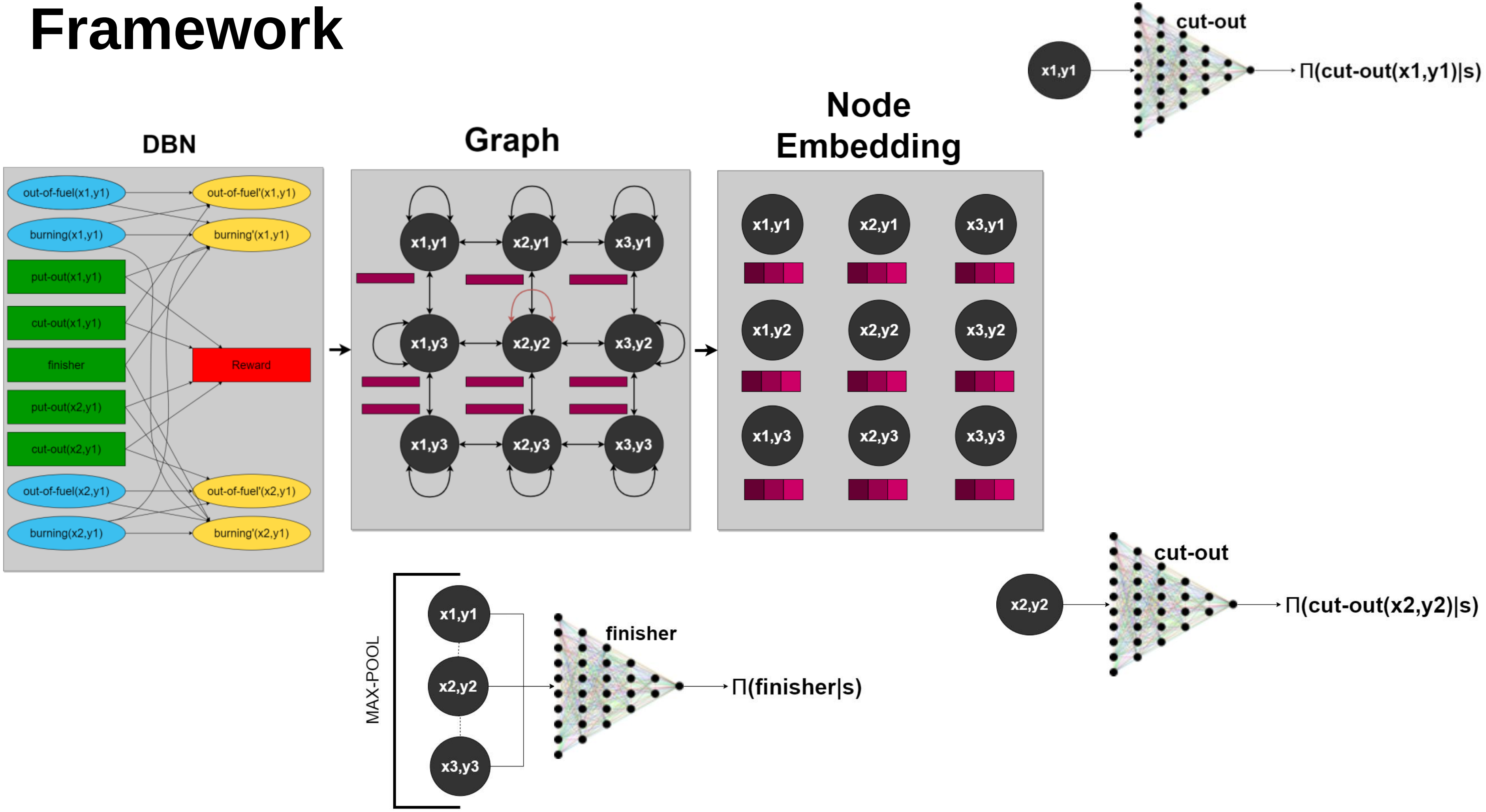
- **Representation Learning:**

  Graph $\xrightarrow{\text{Graph Attention Network}}$ State embedding

- **Policy Learning:**

  State Embedding $\xrightarrow{\text{Neural Network}}$ Action Embedding $\longrightarrow$ Policy

Framework



DBN
Graph
Node Embedding
out-of-fuel(x1,y1)
out-of-fuel'(x1,y1)
burning(x1,y1)
burning'(x1,y1)
put-out(x1,y1)
cut-out(x1,y1)
finisher
Reward
put-out(x2,y1)
cut-out(x2,y1)
out-of-fuel(x2,y1)
out-of-fuel'(x2,y1)
burning(x2,y1)
burning'(x2,y1)
x1,y1
x2,y1
x3,y1
x1,y3
x2,y2
x3,y2
x1,y3
x2,y3
x3,y3
x1,y1
x2,y1
x3,y1
x1,y2
x2,y2
x3,y2
x1,y3
x2,y3
x3,y3
x1,y1
cut-out
Π(cut-out(x1,y1)|s)
x2,y2
cut-out
Π(cut-out(x2,y2)|s)
MAX-POOL
x1,y1
x2,y2
x3,y3
finisher
Π(finisher|s)

Experimental Settings

- Test domains - Academic Advising (AA), Crossing Traffic (CT), Game of Life (GOL), Navigation (NAV), Skill Teaching, (ST), Sysadmin (Sys), Tamarisk (Tam), Traffic (Tra), and Wildfire (Wild).
- We train the policy on problem instances 1, 2, 3.
- We test the policy on domain instances from 5 to 10.
- We compare our method SymNet trained on small instances to ToRPIDo, TraPSNet and SymNet trained from scratch on larger instance

Metrics

To measure generalization power we report:

$$\alpha_{symnet}(0) = \frac{V_{symnet}(0) - V_{random}}{V_{max} - V_{random}}$$

Where V_{max} and V_{random} are the maximum and minimum (random) reward obtained by any algorithm at any time. [α closer to 1 is better.]

For comparison to other algorithms we report:

$$\beta_{algo} = \frac{\alpha_{symnet}(0)}{\alpha_{algo}(t)}$$

where t is the training time of algorithm [$t = 4hrs$].

Results for testing in instance 10

Domain	$\alpha_{symnet}(0)$	Training State Space	Testing State Space
Academic Advising	$\mathbf{0.91 \pm 0.05}$	2^{30}	2^{60}
Crossing Traffic	$\mathbf{1.00 \pm 0.05}$	2^{24}	2^{84}
Game of Life	0.64 ± 0.08	2^{9}	2^{30}
Navigation	$\mathbf{1.00 \pm 0.02}$	2^{20}	2^{100}
Skill Teaching	0.89 ± 0.03	2^{24}	2^{48}
Sysadmin	$\mathbf{0.96 \pm 0.03}$	2^{20}	2^{50}
Tamarisk	$\mathbf{0.95 \pm 0.06}$	2^{20}	2^{48}
Traffic	0.87 ± 0.13	2^{44}	2^{80}
Wildfire	$\mathbf{1.00 \pm 0.01}$	2^{32}	2^{72}

Comparison with other baseline on instance 10

Domain	$\beta_{symnet-scratch}$	$\beta_{torpido}$ [1]
Academic Advising	**1.32**	0.93
Crossing Traffic	**1.22**	**4.99**
Game of Life	**1.25**	0.68
Navigation	**INF**	**INF**
Skill Teaching	**1.30**	0.95
Sysadmin	**1.18**	**1.50**
Tamarisk	**2.35**	**7.99**
Traffic	**1.53**	**1.86**
Wildfire	**34.80**	**11.19**

[[1] Bajpai et. al., NeurIPS 2018]

Conclusion

- We present first neural approach to learn generalized policy of RMDP in RDDL
- Our method can solve any RMDP problem out of the box.
- We obtained good results without any training on the large problems.
- There is still room for improvement as better policies exist.

Check out our code on https://github.com/dair-iitd/symnet

Thank You