Learning to Ground Existentially Quantified Goals

Martin Funkquist[1], Simon Ståhlberg[2], Hector Geffner[2]

Linköping University[1], RWTH Aachen University[2]

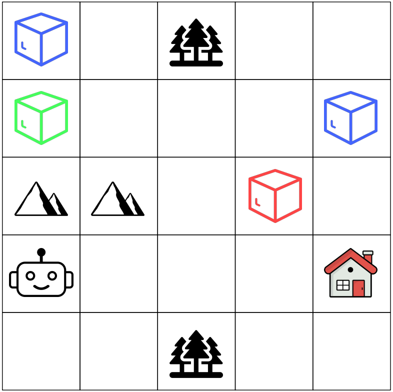
November 7, 2024

Motivation

- We want agents that understand Natural Language (NL) instructions
- In NL we refer to objects with *descriptions*, not by *ids*
- In classical planning, goals are assumed to be grounded e.g. contain ids
- Goals represented with FOL existential quantifiers need no ids

Example

Goal: "Deliver a **blue** package to the house"



$$\exists x(Package(x) \wedge Blue(x) \wedge At(x, h))$$

Classical Planning

- States and goals are represented as sets of propositional atoms $p(c_1, ..., c_n) \in s$.
- A plan is a sequence of actions $a_0, ..., a_n$ that transitions from the initial state s_0 to a state where the goal atoms are true
- Full observability and deterministic actions assumed
- Existentially quantified goals are handled in a way that is *exponential* in the number of variables e.g. $\exists x \varphi(x) = \bigvee_{o \in O} \varphi[x/o]$

Previous Work

- Learning general policies for classical planning using GNNs [Ståhlberg et al, 2022]
- General policies are policies for solving any instance, any ground goal
- Can similar approach be used to handle existentially quantified goals *without having to expand them first*?

Task

- Learn to estimate cost $V^*(s, G)$ of existentially quantified goal G from state s
- The cost is the length shortest plan to reach a state where the quantified goal is true
- A quantified goal is true in a state, if there exists a *substitution* $x \leftarrow c$ for each variable in the goal such that the resulting grounded goal is true in the state

Method

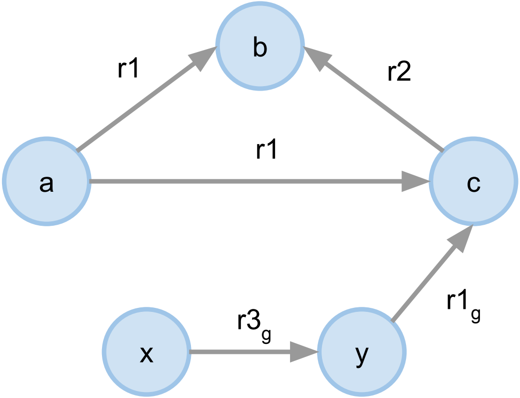
- A Graph Neural Network (GNN) is trained to produce embeddings $f_L(o)$ for each object o in the planning instance
- Train value function $V(s, G) = MLP(\sum_{o \in O} f_L(o))$ to estimate cost for *partially* quantified goals
- Generalization is shown by training on small instances and testing on larger instances and different goals
- Generalization made possible by fixed set of predicates per domain

Graph Neural Networks (GNNs)

- GNNs are Neural Networks that operate on graphs
- GNNs maintain node embeddings $f_i(v) \in \mathbb{R}^k$ for each node $v \in G$ in the graph G.
- Nodes in the graph send messages through the edges
- The embeddings are iteratively updated over L layers to produce $f_L(v)$
- Relational GNNs (R-GNNs) send messages between objects $o_i \in p(o_1, ...o_n)$ in atoms
- An MLP_p is trained for each predicate p in the domain

GNN example

r1(a, b), r1(a, c), r2(c, b), $r3_g$(x, y), $r1_g$(y, c)



Method - Training

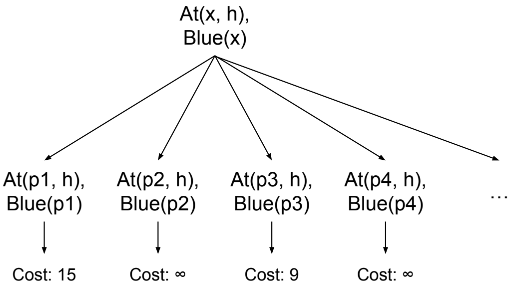
- Learning $V(s, G)$ in supervised manner from small instances where $V^*(s, G)$ available
- Data instances $\langle (s, G), V^*(s, G) \rangle$, where s is a state, G is a *partially* quantified goal and $V^*(s, G)$ is the optimal cost for reaching G
- The model is trained to predict $V^*(s, G)$ and the loss is the mean square error $\mathcal{L}(s, G) = (V(s, G) - V^*(s, G))^2$
- A separate network is trained for each domain using same architecture and hyperparameters

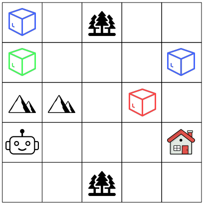
Method - Grounding

- Let $G' = G_{x=c}$ be the set of all partially quantified goals that result from grounding a *single* variable in G to a constant $c \in s$
- A *substitution* is made by replacing x with c such that $\min_{x \in G, c \in s} V(s; G_{x=c})$
- This is done iterativly until G' is *fully grounded*, e.g. there are no variables left in G'
- The *quality* of a grounding is measured by the ratio $V^*(s, G')/V^*(s, G)$
- Exponential blowup is avoided by grounding single variable at a time

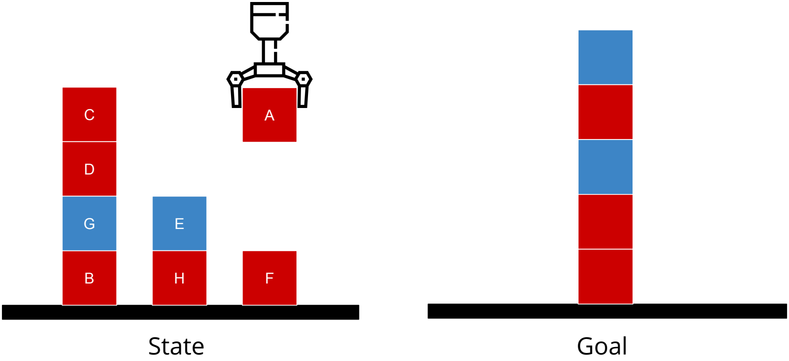
Grounding - Example

$$\exists x(Package(x) \wedge Blue(x) \wedge At(x, h))$$



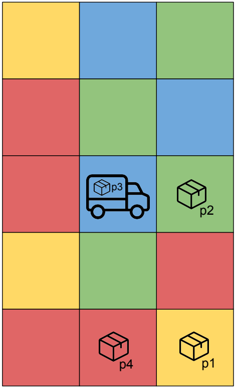


Domain - Blocks

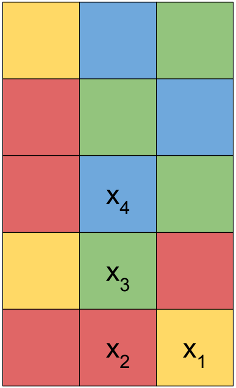


$$\exists x_1, \ldots x_5 : \text{Blue}(x_1) \land \text{Red}(x_2) \land \text{Blue}(x_3) \land \text{Red}(x_4) \land \text{Red}(x_5) \land \text{On}(x_1, x_2) \land \text{On}(x_2, x_3) \land \text{On}(x_3, x_4) \land \text{On}(x_4, x_5)$$

Domain - Delivery



State



Goal

$\exists x_1, \ldots, x_4 : \text{At}(p_1, x_3) \wedge \text{At}(p_3, x_2) \wedge \text{At}(p_4, x_1) \wedge \text{At}(t, x_4) \wedge \text{Yellow}(x_1) \wedge \text{Red}(x_2) \wedge \text{Green}(x_3) \wedge \text{Blue}(x_4)$

Results

Domain	No $x_i \neq x_j$ Cov.	No $x_i \neq x_j$ V/V^*	With $x_i \neq x_j$ Cov.	With $x_i \neq x_j$ V/V^*	Random groundings All Cov.	Random groundings Val Cov.	Random groundings V/V^*
Blocks	99.8 %	1.211	99.8 %	1.211	3.8 %	76.2 %	1.391
Blocks-C	99.8 %	1.103	100 %	1.019	6.8 %	100 %	1.797
Gripper	100 %	1.298	99.4 %	1.189	3 %	86.6 %	1.499
Delivery	99.8 %	1.238	100 %	1.177	10.8 %	100 %	1.522
Visitall	100 %	1.181	99.2 %	1.080	2.6 %	100 %	1.445

- No $x_i \neq x_j$: no contraints on the variables
- With $x_i \neq x_j$: bindings are not valid if any two variables are bound to the same constant
- Random groundings: each variable is randomly bound to a constant

Results - LAMA

Domain	Cov. (%)	LAMA Cov. (%)	V/V^L	Speedup
Blocks	**100 %**	99.4 %	1.245	**19.837**
Blocks-C	100 %	100 %	1.179	**1.359**
Gripper	**97.6 %**	96.2 %	1.263	**104.335**
Delivery	**99 %**	98.6 %	1.459	**50.495**
Visitall	**100 %**	98 %	1.480	**33.219**

bold means top performing

- Instances here are larger than for V^*
- Large speedup compared to LAMA

Expressivity Limitations

- Tight relationship between GNNs, Weisfeiler-Leman and two-variable first-order logic with counting quantifiers (C_2) [Grohe, 2021]
- Example of C_2 logic: $\exists^{\geq 4} x, \exists y[E(x, y)]$ e.g. there exists a node y which has at least 4 neighbors
- C_2 provides an *upper bound* of the expressivity of our GNN model

Expressivity Limitations - Example

- Important to know if object and variable have same color
- $SameColor(o, x) = \exists c : HasColor(o, c) \wedge HasColor(x, c)$, likely not in C_2 as it uses *three* variables o, x, c
- Fixed set of colors $C_1, ..., C_n$, then $SameColor(o, x) = [C_1(o) \wedge C_1(x)] \vee ... \vee [C_n(o) \wedge C_n(x)]$, only need *two* variables o, x

Conclusions

- NL instruction following involves grounding abstract references to concrete objects
- Learning groundings by learning general value function $V(s, G)$ that estimates cost of quantified goal G from state s
- GNN-approach generalizes to any domain state and family of quantified goals
- Limitation: C2 restriction narrows the scope of the approach

Paper