Learning Transferable Graph Exploration

Authors: Hanjun Dai et al.

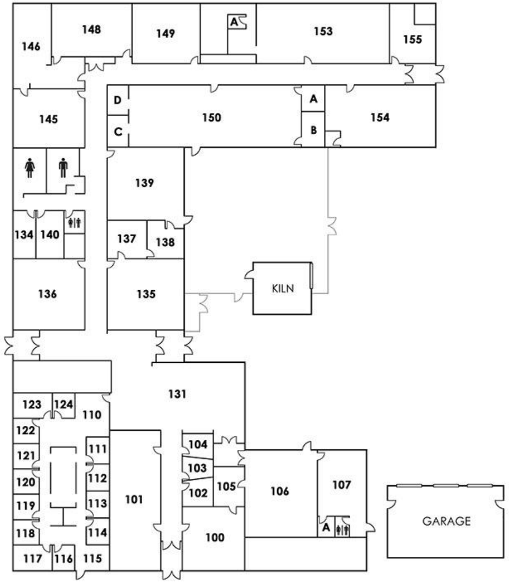
Presented by Qinghai Zhou

ILLINOIS

Outline

- Background and Problem Formulation
- Methodology
- Evaluation

Background



State-space Coverage Problem

Goal: given an environment, efficiently reach as distinct/unique states as possible

Applications:

- model checking: design test inputs to expose as many potential errors as possible
- active map building: construct a map of unknown environment efficiently
- exploration in reinforcement learning in general

Common Approaches: Undirected Exploration

Key idea: randomly choose states to visit / actions to take

Examples:

1. Random walk based method on graph
   - Cover time depends on graph structure ($O(n\log n)$, $O(n^3)$)
2. Epsilon-greedy selection
   - selecting random action with probability epsilon
   - preventing being locked on sub-optimal action

Directed Exploration

Key idea: optimize a certain objective that encourages exploration/coverage

Examples:

1. Upper Confidence Bound (UCB) for Bandit Problems:
   - in addition to maximizing the reward, encourage exploring unselected actions
2. Reinforcement Learning (RL)
   - pseudo-count (similar to UCB): rewards change in state density estimates
   - information gain: take actions from which you learn about the environment (reduces entropy)
   - predictive error: encourage actions that lead to unpredictable outcome (for instance, unseen states)

Graph Exploration

Goal: an efficient exploration strategy to reach as many vertices as possible

- effectiveness of random walk greatly depends on the graph structure

Question to be answered:

Given the distribution of graphs in training time, can the algorithm learn an efficient covering/exploration strategy?

Formulation

Several key components in RL:

- Environment: for agent to interact with
- Action: state to cover/explore
- Reward: feedback from the environment

Formulation

Environment: Graph-structured state-space

- at time t, the agent observes a graph $G_{t-1} = \{E_{t-1}, V_{t-1}\}$, and a coverage mask $c_{t-1} : V_{t-1} \to \{0, 1\}$ indicating the nodes' exploration status so far
- the agent takes an action a_t and receives a new graph G_t
- Budget: number of steps/actions allowed to take

Same dist. as training

Goal:

learn exploration strategy such that given an unseen environment, the agent can efficiently visit as many unique nodes as possible

Reward (maximize the number of visited nodes)

Coverage mask

Cumulative rewards: $$\max_{\{a_1,a_2\ldots a_t\}} \sum_{v \in V_t} \frac{\boxed{c_t}(v)}{|\mathcal{V}|}$$

Per-step reward: $$r_t = \sum_{v \in V_t} \frac{c_t(v)}{|\mathcal{V}|} - \sum_{v \in V_{t-1}} \frac{c_{t-1}(v)}{|\mathcal{V}|}$$

They are equivalent $$\sum_{v \in V_0} c_0(v) = 0$$

Objective

$$\max_{\{\theta_1,\theta_2\ldots\theta_t\}} \mathbb{E}_{\mathcal{G}\sim\mathcal{D}}\left[\sum_{t=1}^{T}\mathbb{E}_{a_t^{\mathcal{G}}\sim\pi(a|h_t^{\mathcal{G}},\theta_t)}\left[r_t^{\mathcal{G}}\right]\right]$$

- $h_t = \{(a_i, G_i, c_i)\}$ (i=1,...,t) is the exploration history
- $\pi(a|h_t, \theta_t)$ is the action policy at time t
- D is the environment distribution

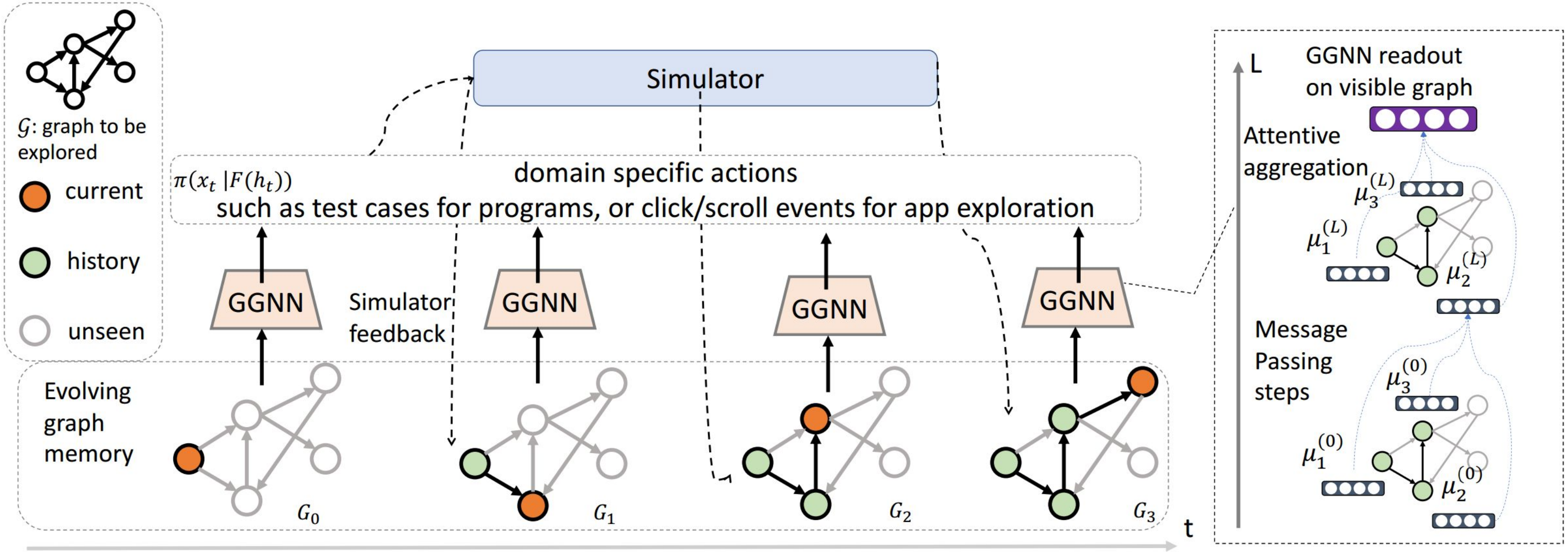
Representing Exploration History

Graph representation:

- use graph neural networks to learn a representation (one-bit information c_t)
- starting from node $\mu_v^{(0)}$, update representation via message passing: $\mu_v^{(l+1)} = f(\mu_v^{(l)}, \{e_{uv}; \mu_u^{(l)}\})$, ($f(\cdot)$: GGNN model)
- apply attention weighted-aggregation
- learned via unsupervised link prediction

Li Y, Tarlow D, Brockschmidt M, Zemel R. Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493. 2015 Nov 17.

Representing Exploration History

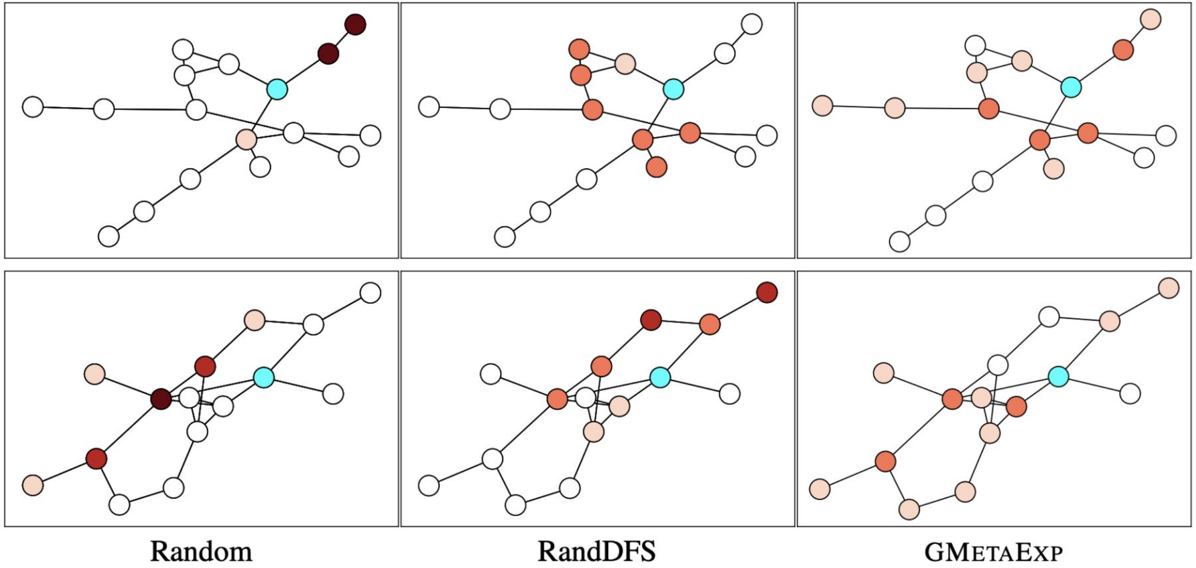


History representation: summarize representation up to the current step

Auto-regressive $$\boxed{F}(h_t) = \text{LSTM}(F(h_{t-1}, \boxed{g}(G_t, c_t)))$$ encoder

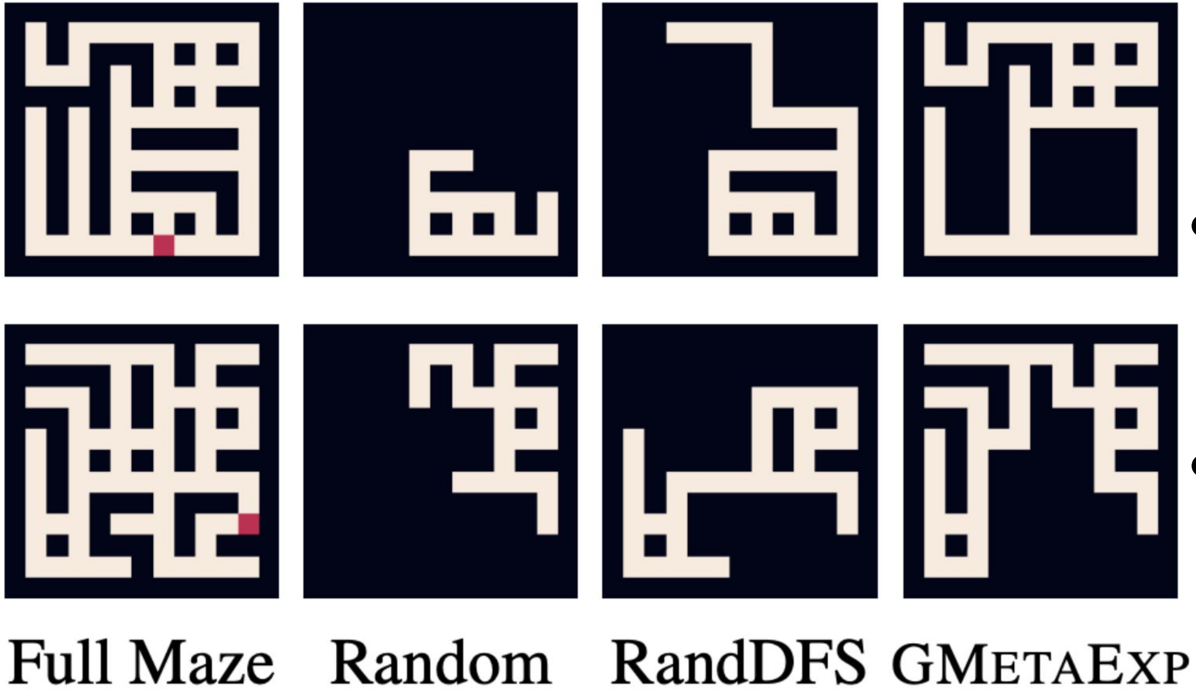
Evaluation

Two settings: (1) unknown graph; (2) known but complex (program testing)



Erdos-Renyi Random Graph:
blue node indicates starting point; darker colors represent more visit counts

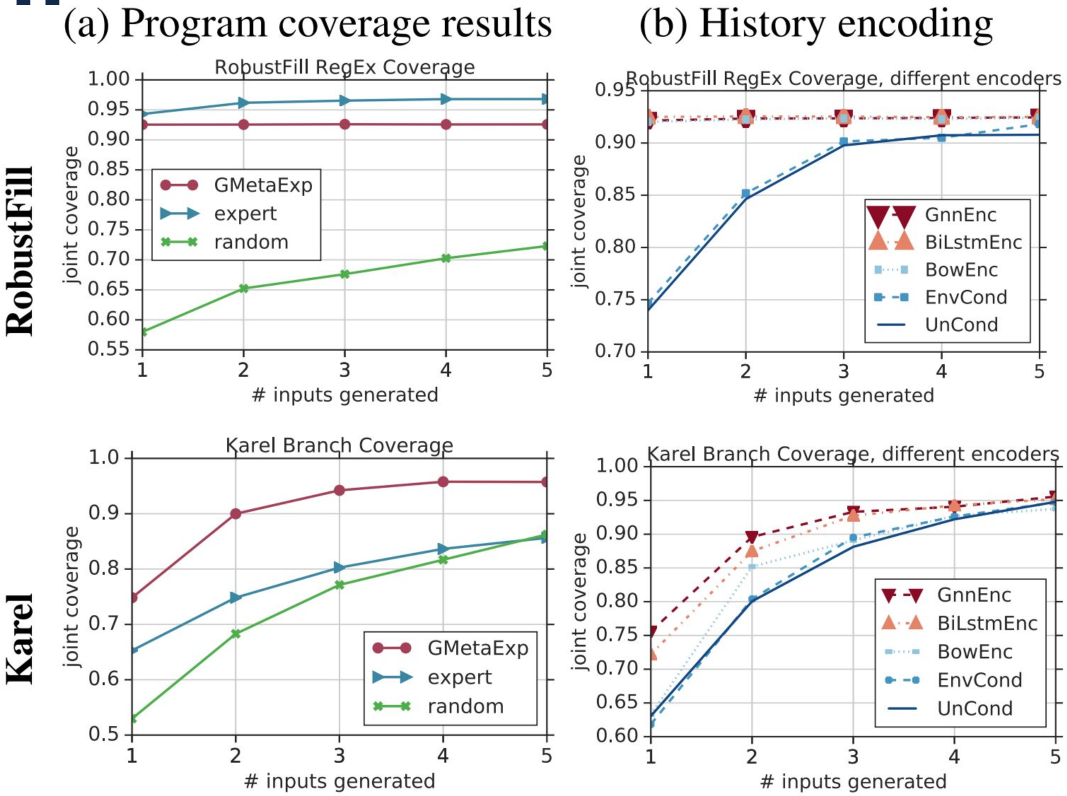
Evaluation



2D Maze

- given fixed budget (T = 36), the agent is trained to traverse the 6x6 maze as much as possible
- test on held-out mazes from the same distribution

Evaluation



Program testing (coverage guided fuzzing):
Propose test cases to cover as many code branches as possible

Limitation:

- requires reasonable large amount of training data
- cannot scale to large programs

Possible extension:

- RL-based dynamic graph representation

Please check the paper for more details about method and experiment.

Thank you!
Q & A