Addressing Long-Horizon Tasks by Integrating Program Synthesis and State Machines

NeurIPS 2023 Workshop on Generalization in Planning

Yu-An Lin*
Chen-Tao Lee*
Guan-Ting Liu*
Pu-Jen Cheng
Shao-Hua Sun

Program Machine Policy

Execute

Environment

Reward

Transit. Prob.
high
low

Yu-An Lin*
Chen-Tao Lee*
Guan-Ting Liu*
Pu-Jen Cheng
Shao-Hua Sun

National Taiwan University
Deep Reinforcement Learning

Deep Neural Network

Inference

Environment

Reward
Deep Reinforcement Learning - Issues

Generalization

Simple task \rightarrow \text{X} \rightarrow \text{Complex task}

Interpretability

*Trust, Safety, and Contestability*

Deep neural network policy
Programmatic Reinforcement Learning

Inference

Programmatic Policy

Environment

Reward
Programmatic Reinforcement Learning - Program Synthesis

- Structured in formal languages
- Human interpretable and machine executable

DEF run() m(
  WHILE c(markerPresent c) w(
    WHILE c(markerPresent c) w(
      pickMarker
      move w)
      turnRight
      move
      turnLeft
      WHILE c(markerPresent c) w(
        pickMarker
        move w)
      turnLeft
      move
      turnRight w) m)

Environment

Program

Execute

Reward

LEAPS: Learning Embeddings for Interpretable and Generalizable Program Synthesis

Model → Synthesize → Program → Execute → Environment

Program Synthesizer

```
DEF run() m{
  WHILE c{ markerPresent c } w{
    WHILE c{ markerPresent c } w{
      pickMarker
      move w)
      turnRight
      move
      turnLeft
    }w{ markerPresent c } w{
      pickMarker
      move w)
      turnLeft
      move
      turnRight w} m)
```

LEAPS - Learning a Program Embedding Space

Goal: Learn the grammar and the environment dynamics

LEAPS - Latent Program Search

Search for a task-solving program using the cross-entropy method (CEM)

**Decoded Program**

```
DEF run()
IF frontIsClear
move
ELSE
turnRight
REPEAT(2)
turnRight
putMarker
```
Karel Tasks

StairClimber

Maze

FourCorners

TopOff

Harvester

CleanHouse
Interactive Debugging Interface

Input Program:
```
DEF n; m;
WHILE (n) rotatedMarkersPresent c w;
turnRight
move
w; pushMarker
move
WHILE (n) rotatedMarkersPresent c w;
turnRight
move
w; pushMarker
move
WHILE (n) rotatedMarkersPresent c w;
turnRight
move
w; pushMarker
move
```

Reset Code (Make a mistake?)

Issue with Code?
None

Submit Code (Fuse your program)

New Reward: 0.8000000074980386
Org Reward: 0.89303834010315
Best Reward: 0.89303834010315

Performance Improvement

<table>
<thead>
<tr>
<th></th>
<th>Unedited (LEAPS)</th>
<th>Up to 3 Edits</th>
<th>Up to 5 Edits</th>
</tr>
</thead>
<tbody>
<tr>
<td>TopOff</td>
<td>1.0</td>
<td>0.9</td>
<td>0.8</td>
</tr>
<tr>
<td>FourCorner</td>
<td>0.8</td>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td>Harvester</td>
<td>0.8</td>
<td>0.6</td>
<td>0.5</td>
</tr>
</tbody>
</table>
Programmatic Reinforcement Learning - State Machine

- Inductively generalize to longer-horizon tasks

**Inductive Generalization**

**Task:** Retrieve a car from tight parking spots

**Training Tasks**

(a) Train 1  
(b) Train 2  
(c) Train 3

**Learned State Machine Policy**

Symbolic Modes (i.e., actions) + Transition Function

**Testing Task**

(d) Test
This Work: Integrating Program Synthesis and State Machine

Program

```java
DEF run() m(
    WHILE c(markerPresent c) w(
        WHILE c(markerPresent c) w(
            pickMarker
            move w)
        turnRight
        move
        turnLeft
    WHILE c(markerPresent c) w(
        pickMarker
        move w)
    turnLeft
    move turnRight w) m)
```

Pros
- Interpretable

Cons
- Often fail to generalize to longer-horizon tasks

State Machine

### Pros
- Inductively generalizable

### Cons
- Difficult to interpret
PrOgram Machine Policy (POMP)

Modes: A set of programs that can be executed

Transition function: Alter between a set of program modes

- Mapping: current environment state \( x \) current mode \( \rightarrow \) next mode
Step 1 - Learning a Program Embedding Space

Goal: Learn the grammar and the environment dynamics
Step 2 - Retrieving Effective, Diverse, Compatible Programs

Goal: Retrieve a set of programs as state machine modes

**Effectiveness**: Retrieved programs should (partially) solve the task

→ Task reward: \( \sum_{t=0}^{T} \gamma^t E(s_t, a_t) \sim \text{EXEC}(\rho_z)[r_t] \)

**Diversity**: Retrieved programs should induce non-overlapping, diverse behaviors

→ Diversity bonus: \(- \max_{z_i \in Z} \frac{z \cdot z_i}{\|z\| \|z_i\|}\)

**Compatibility**: Composing retrieved programs in some order should yield good performance

→ Randomly execute retrieved programs before/after executing the current program candidate

---

Program Embedding Space

**Mode 1**

```
DEF run()
  IF NOT frontIsClear
    IF markerPresent
      pickMarker
    ELSE
      turnRight
  ELSE
    move
```

**Mode 2**

```
DEF run()
  IF frontIsClear
    move
    turnLeft
    \(\text{REPEAT}(2)\)
    turnRight
    putMarker
```

**Mode 3**

```
DEF run()
  WHILE frontIsClear
    move
    IF markerPresent
      pickMarker
    ELSE
      turnRight
```
Step 3 - Learning Transition Function

Goal: Learn a transition function, i.e., current environment state $x$ current mode $\rightarrow$ next mode

Modes (Retrieved Programs)

1. DEF run()
   IF NOT frontIsClear
   IF markerPresent
     pickMarker
   ELSE
     turnRight
   ELSE
     move

2. DEF run()
   WHILE frontIsClear
   move
   IF markerPresent
     pickMarker
   ELSE
     turnRight
   ELSE
     move

3. DEF run()
   IF frontIsClear
   move
   turnLeft
   REPEAT(2)
   turnRight
   putMarker

4. DEF run()
   IF NOT frontIsClear
   IF markerPresent
     pickMarker
   ELSE
     turnRight
   ELSE
     move

Learned using RL
Learned Program Machine Policy

Task: Inf-DoorKey

Retrieved Programs

Extracted State Machine
Ablation Study

Diversity & Compatibility

<table>
<thead>
<tr>
<th>Method</th>
<th>SEESEAW</th>
<th>UP-N-DOWN</th>
<th>FARMER</th>
<th>INF-DOORKEY</th>
<th>INF-HARVESTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEM ×</td>
<td>M</td>
<td></td>
<td>0.31 ± 0.18</td>
<td>0.88 ± 0.12</td>
<td>0.22 ± 0.00</td>
</tr>
<tr>
<td>CEM+diversity top k, k =</td>
<td>M</td>
<td></td>
<td>0.09 ± 0.11</td>
<td>0.72 ± 0.36</td>
<td>0.23 ± 0.00</td>
</tr>
<tr>
<td>CEM+diversity ×</td>
<td>M</td>
<td></td>
<td>0.47 ± 0.39</td>
<td>0.76 ± 0.31</td>
<td>0.24 ± 0.03</td>
</tr>
<tr>
<td>POMP (Ours)</td>
<td><strong>0.90 ± 0.02</strong></td>
<td><strong>0.97 ± 0.00</strong></td>
<td><strong>0.88 ± 0.01</strong></td>
<td>0.91 ± 0.01</td>
<td>0.67 ± 0.03</td>
</tr>
</tbody>
</table>

CEM Search Trajectories with Diversity Bonus
Takeaway

Program Synthesis  ×  State Machine  =

Interpretable and Inductively Generalizable Policies