Compositional Learning

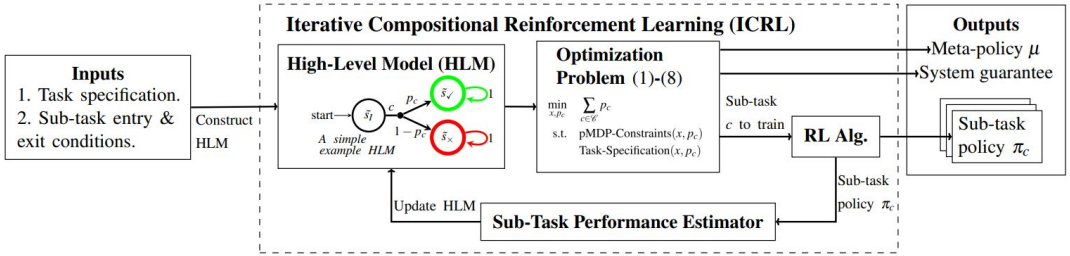
Venkata Sai Santosh Siripurapu
Aryaman Singh Samyal

Introduction

- How do we build complex engineering systems we can trust?
  - Decomposing system-level requirements into component-level ones?
    - Each component may be developed and tested independently
- This is what compositional learning is about!
  - Decide which subsystem to execute to sequentially accomplish specific subtasks.
    - Each sub-system could employ algorithms to learn policies accomplishing its sub-task.
- So, who decides the sequence of execution of these low-level subsystems?
  - A high-level model is plans and analyzes compositions and collection of subsystems
- How can we guarantee that the overall task will be accomplished?
  - If a subtask cannot be satisfied by the corresponding policy, update the high-level model
    - This refinement helps iteratively compute subtask specifications and train subsystems to achieve those subtasks.

Current Work

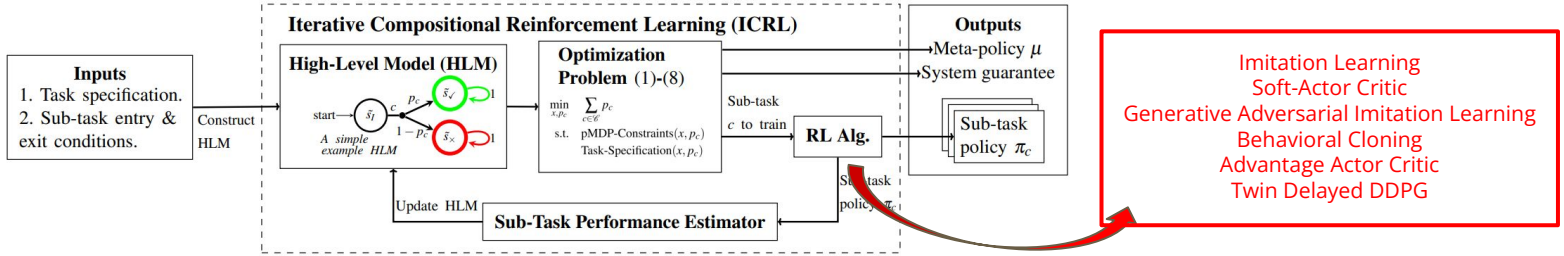
- Uses a parametric Markov decision process (pMDP) high-level model, to plan and to analyze subsystems
- Each low-level subsystem employs RL algorithms trained using proximal policy optimization to navigate different portions of a labyrinth environment



Our contribution

We intend to compare the performance of various algorithms at the low-level of the framework, namely,

- Imitation Learning
- Soft-Actor Critic
- Generative Adversarial Imitation Learning
- Behavioral Cloning
- Advantage Actor Critic
- Twin Delayed DDPG

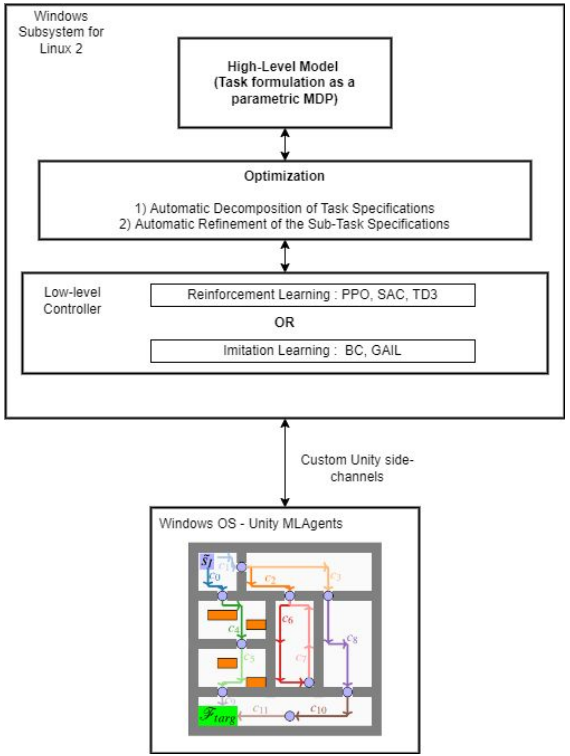


Motivation

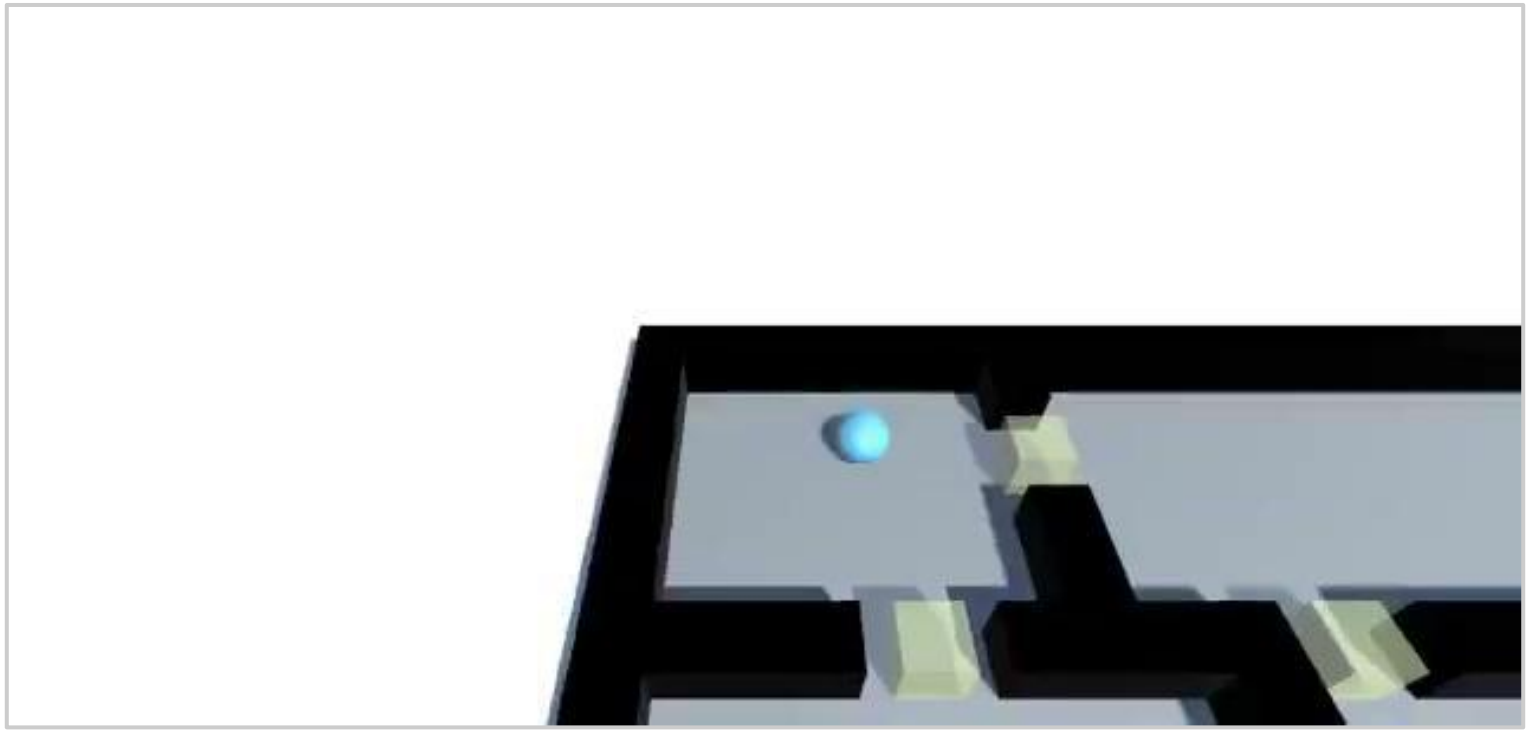
- Imitation Learning
  - RL models need to a reward mechanism, which can be challenging to define.
  - IL requires fewer training episodes than RL because of the ‘dense’ nature of feedback
- Generative Adversarial Imitation Learning
  - Has obtained a substantial performance boost over existing methods in imitating complex behaviors
- Soft Actor Critic
  - Seeks to also the entropy of the policy, ensuring exploration.
- Advantage Actor Critic
- Behavioral Cloning
  - Low complexity and efficiency

Components

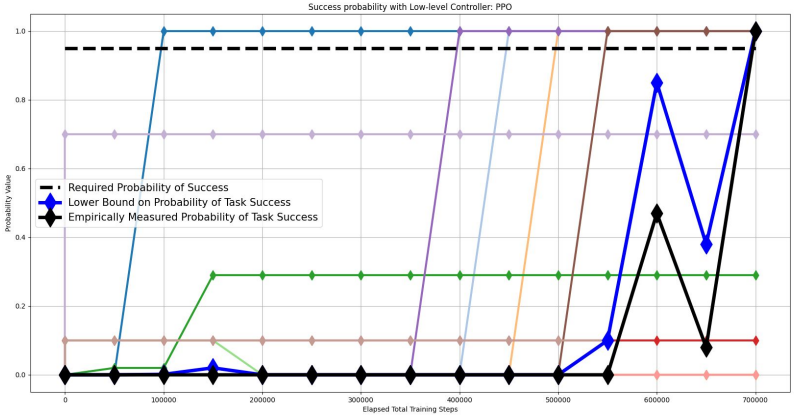
- Use of WSL2 to resolve dependencies
  - Interface between the low-level policies and the Unity Labyrinth environment.



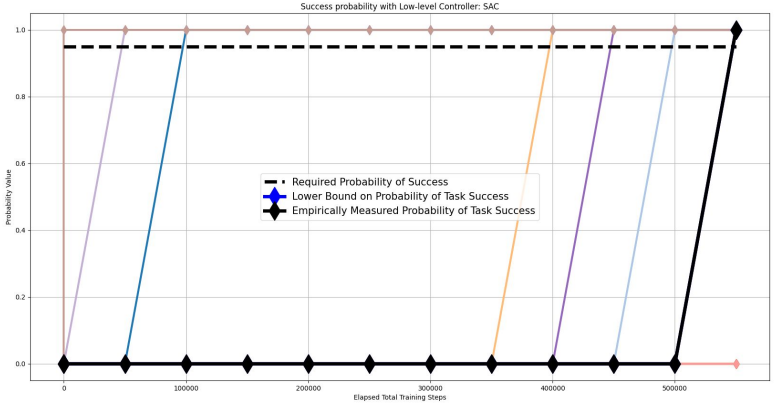
Demo



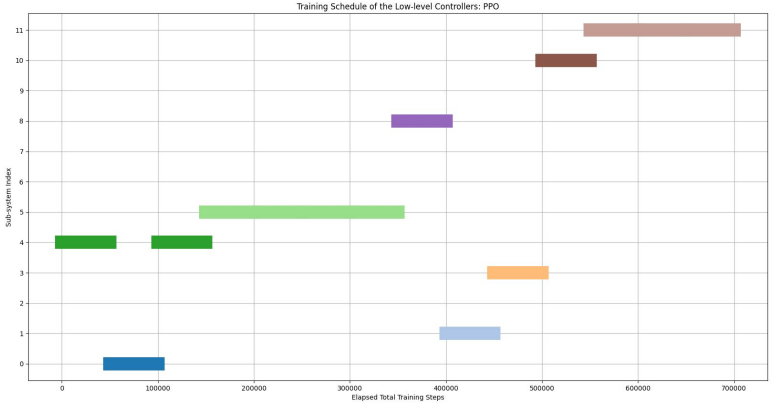
Results (1)


Success probability with Low-level Controller: PPO
Required Probability of Success
Lower Bound on Probability of Task Success
Empirically Measured Probability of Task Success
Probability Value
Elapsed Total Training Steps
0.0
0.2
0.4
0.6
0.8
1.0
0
100000
200000
300000
400000
500000
600000
700000

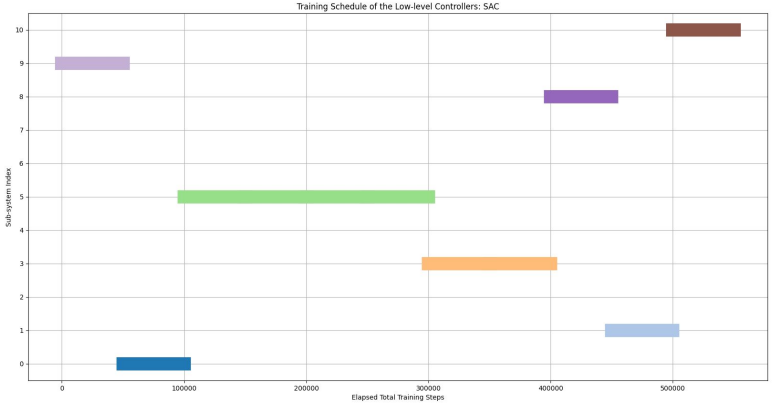
Results (2)


Success probability with Low-level Controller: SAC
Probability Value
1.0
0.8
0.6
0.4
0.2
0.0
0
100000
200000
300000
400000
500000
Elapsed Total Training Steps
Required Probability of Success
Lower Bound on Probability of Task Success
Empirically Measured Probability of Task Success

Results (3)


Training Schedule of the Low-level Controllers: PPO
Sub-system Index
Elapsed Total Training Steps
0
100000
200000
300000
400000
500000
600000
700000
0
1
2
3
4
5
6
7
8
9
10
11

Results (4)


Training Schedule of the Low-level Controllers: SAC
Sub-system Index
Elapsed Total Training Steps
0
100000
200000
300000
400000
500000
0
1
2
3
4
5
6
7
8
9
10