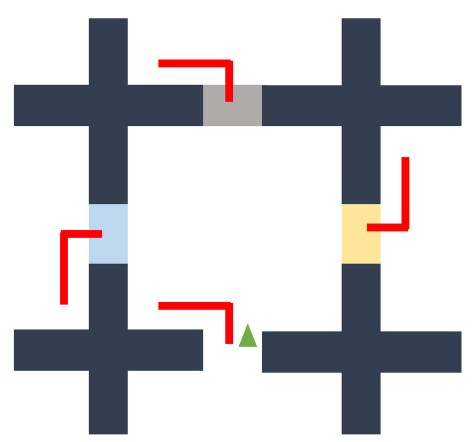
Robust Subtask Learning for Compositional Generalization

Kishor Jothimurugan, Steve Hsu, Osbert Bastani and Rajeev Alur

Penn University of Pennsylvania



Subtasks (Reach Exit)

left

right

up

Robot enters a similar room upon exiting

OBJECTIVE

Learn **one policy per subtask**
During test time **user gives a task**–i.e., a sequence of subtasks

EXAMPLE TASK

τ = left → right → up → left

PROBLEM STATEMENT

Given a set of subtasks Σ, learn one policy per subtask $\Pi = \{\pi_\sigma \mid \sigma \in \Sigma\}$ to maximize the **worst-case expected reward w.r.t. the choice of tasks**

$$J(\Pi) = \inf_{\tau \in \mathcal{T}} \mathbb{E}_{\rho \sim \mathcal{D}_\tau^\Pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{\tau[i_t]}(s_t, \pi_{\tau[i_t]}(s_t))\right]$$

Paper

My Website

CHALLENGES

1. ***Initial state distribution*** used during training may not match with initial state distribution encountered during testing

2. Subtask policy might lead the robot to ***an unrecoverable state*** from which next subtask is impossible

3. The subtask policies must work for ***all tasks***

OUR APPROACH

STEP I: REDUCE TO TWO-PLAYER GAME

STATES. State of the game is a **pair** $(\boldsymbol{s}, \boldsymbol{\sigma})$ where s is **environment state** and σ is **current subtask**

PLAYER 1. The agent learning the subtask policies – one agent represents **all the subtask policies** $\{\pi_\sigma \mid \sigma \text{ is a subtask}\}$

PLAYER 1's POLICY. Equivalent to one policy per subtask

$$\boldsymbol{\pi_1(s, \sigma) = \pi_\sigma(s)}$$

PLAYER 2. The adversary that **selects the next subtask** upon completion of current subtask – **only acts in exit states**

STEP II: SOLVE THE GAME

- Two **value iteration** algorithms to compute V*
- A Q-learning algorithm that **converges in the limit**
- An SAC based **algorithm for infinite state/action spaces –** modified to **train subtask policies** (instead of one policy) – solves the **two-player game** (instead of an MDP)
- An asynchronous algorithm for **learning options in parallel**

ROBUST OPTION SAC (ROSAC)

Use **Soft Actor-Critic** for Player 1 with a **separate actor and critic network for each subtask**

- **Target value** for training critic Q_σ on transition $(s, \sigma) \to_a (s', \sigma)$ is

$$R_\sigma(s, a, s') + \gamma \cdot \min_{\boldsymbol{\sigma}'} \boldsymbol{V(s', \sigma')}$$

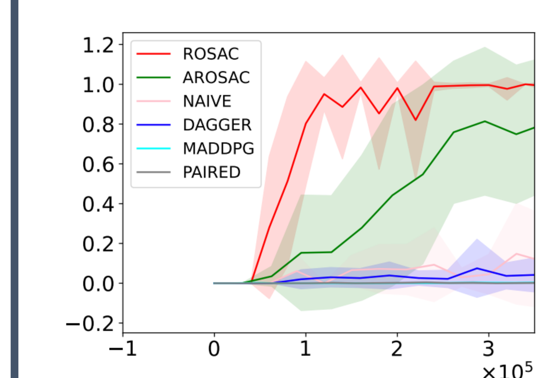
if s' is an **exit state for subtask** $\boldsymbol{\sigma}$

- The action of **Player 2** at any exit state s' is

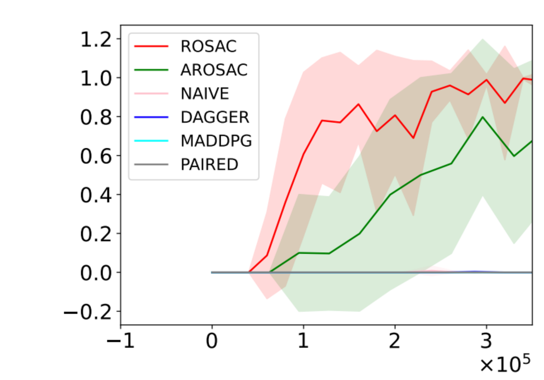
$$\operatorname*{\mathbf{argmin}}_{\boldsymbol{\sigma}'} \boldsymbol{V(s', \sigma')}$$

EXPERIMENTS

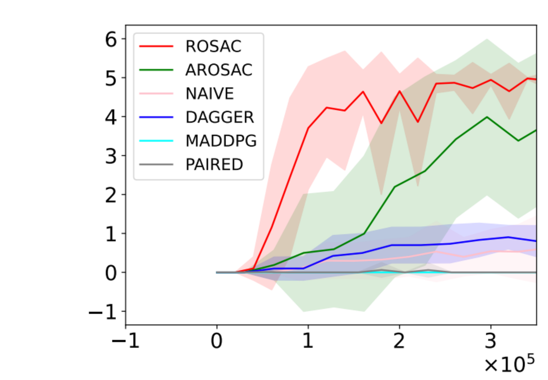
Rooms Environment



(a) Success probability against random adversary

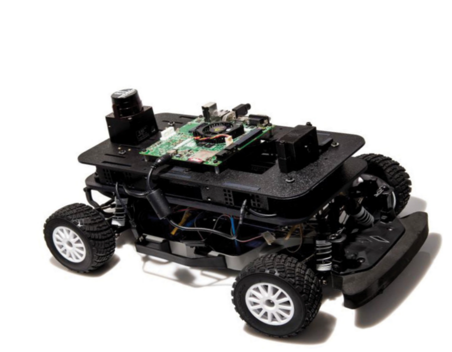


(b) Success probability against MCTS adversary

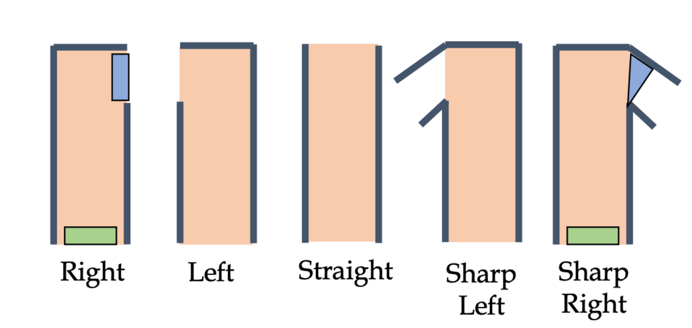


(c) Number of subtasks completed against MCTS adversary

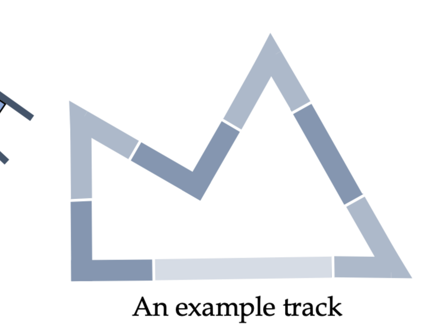
F1/10th Environment



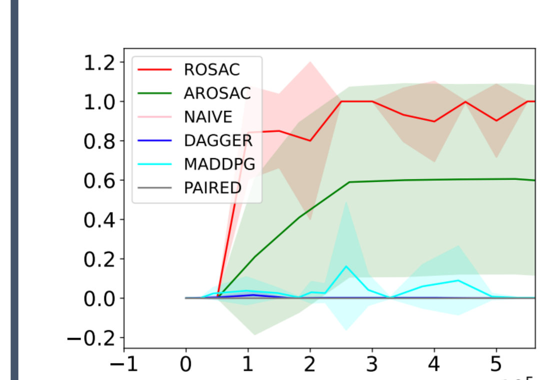
(a) F1/10th Car



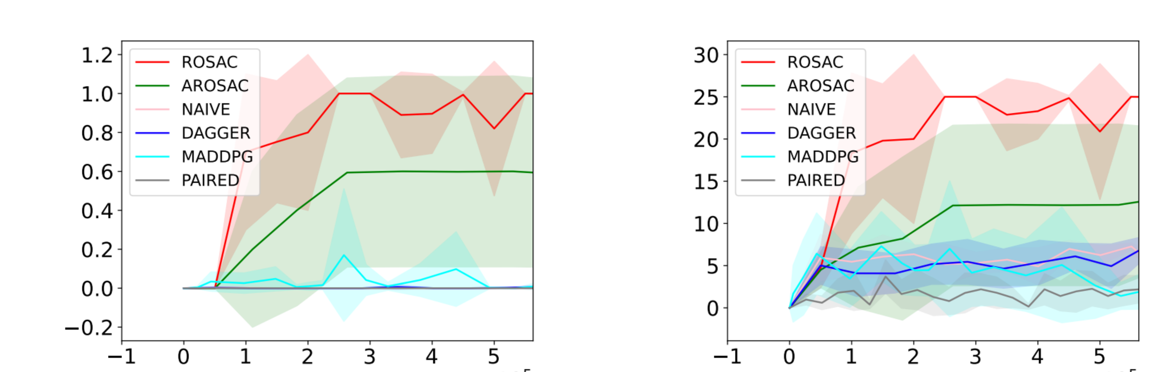
(b) Segment Types



(a) Success probability against random adversary



(b) Success probability against MCTS adversary



(c) Number of subtasks completed against MCTS adversary