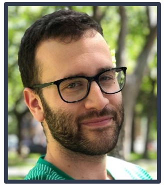
Direct Behavior Specification via Constrained RL

Julien Roy, Roger Girgis, Joshua Romoff, Pierre-Luc Bacon, Christopher Pal

UBISOFT
Mila
POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE
Université de Montréal



Reward hypothesis

- "All of what we mean by goals and purposes can be well thought of as the maximisation of the expected value of the cumulative sum of a received scalar signal (called reward)"

(Sutton & Barto, 2020)

i.e. all tasks can be defined as a scalar function to maximise

RL often leads to unforeseen behaviors

From OpenAI:

source: https://openai.com/blog/faulty-reward-functions/

From Ubisoft LaForge:

RL often leads to unforeseen behaviors

From OpenAI:

source: https://openai.com/blog/faulty-reward-functions/

From Ubisoft LaForge:

The problem

The more complex the task becomes, the more components need to be incorporated in the reward function

$$R(s,a) = A$$

The problem

The more complex the task becomes, the more components need to be incorporated in the reward function

$$R(s,a) = A + B$$

The problem

The more complex the task becomes, the more components need to be incorporated in the reward function

$$R(s,a) = A + B + C$$

The problem

The more complex the task becomes, the more components need to be incorporated in the reward function

$$R(s,a) = A + B + C + D$$

The problem

The more complex the task becomes, the more components need to be incorporated in the reward function

$$R(s,a) = A + B + C + D + E$$

The problem

The more complex the task becomes, the more components need to be incorporated in the reward function

$$R(s,a) = A + \mathbf{B} + c + D + E$$

The problem

The more complex the task becomes, the more components need to be incorporated in the reward function

$$R(s,a) = A + B + C + D + E$$

The problem

The more complex the task becomes, the more components need to be incorporated in the reward function

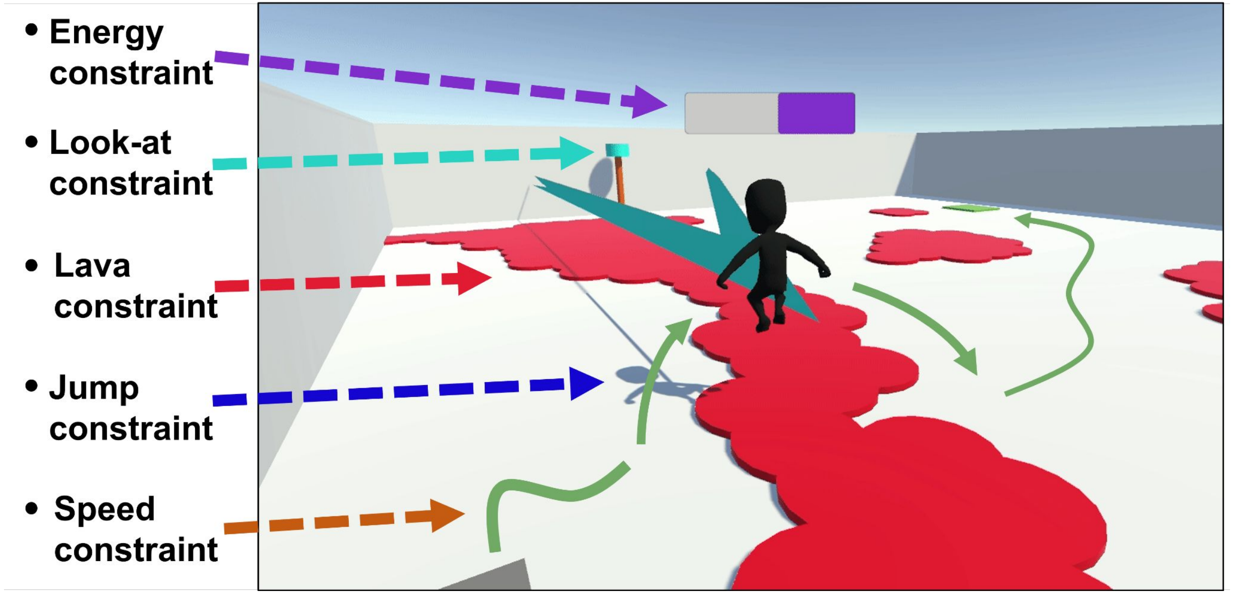
$$R(s,a) = \mathbf{A} + \mathbf{B} + c + D + \mathbf{E}$$

The problem

The more complex the task becomes, the more components need to be incorporated in the reward function

$$R(s,a) = \mathbf{A} + \mathbf{B} + C + D + \mathbf{E}$$

Experimental Setup: Arena env



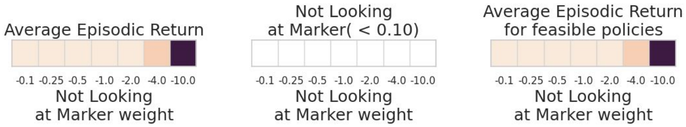
Reward Engineering

$$
\begin{aligned}
R'(s,a) = R(s,a) &- \mathbf{1} * w_{\text{not-looking}} \\
&- \mathbf{1} * w_{\text{in-lava}} \\
&- \mathbf{1} * w_{\text{no-energy}}
\end{aligned}
$$

Reward Engineering

$$R'(s,a) = R(s,a) - \mathbf{1} * w_{\text{not-looking}} - \mathbf{1} * w_{\text{in-lava}} - \mathbf{1} * w_{\text{no-energy}}$$

<u>1 additional requirement</u>

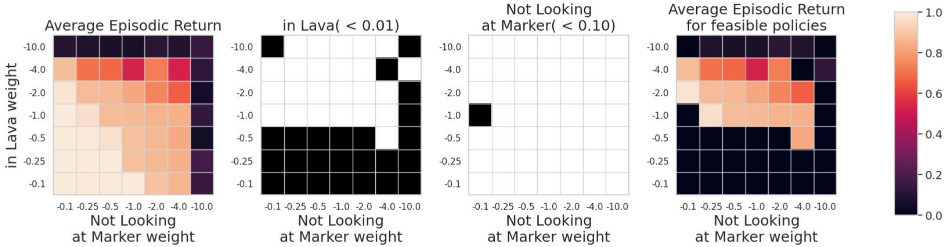


Reward Engineering

$$R'(s,a) = R(s,a) - \mathbf{1} * w_{\text{not-looking}} - \mathbf{1} * w_{\text{in-lava}} - \mathbf{1} * w_{\text{no-energy}}$$

1 additional requirement

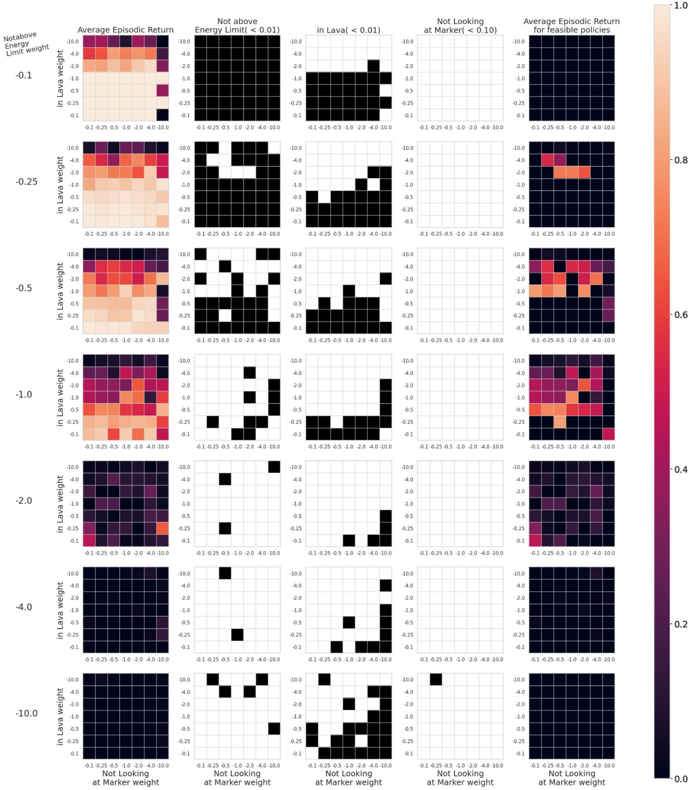
2 additional requirements



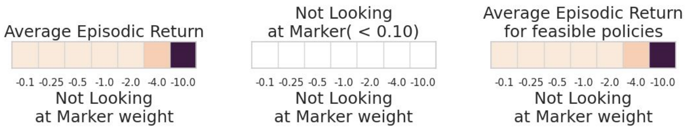
Reward Engineering

$$R'(s,a) = R(s,a) - 1 * w_{\text{not-looking}} - 1 * w_{\text{in-lava}} - 1 * w_{\text{no-energy}}$$

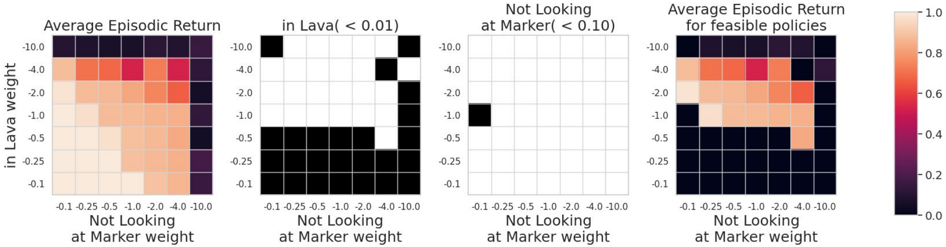
3 additional requirements



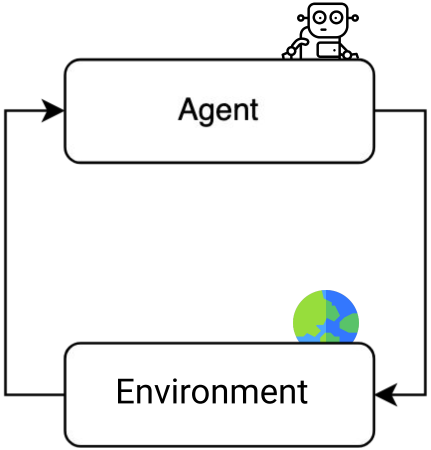
1 additional requirement



2 additional requirements



MDPs



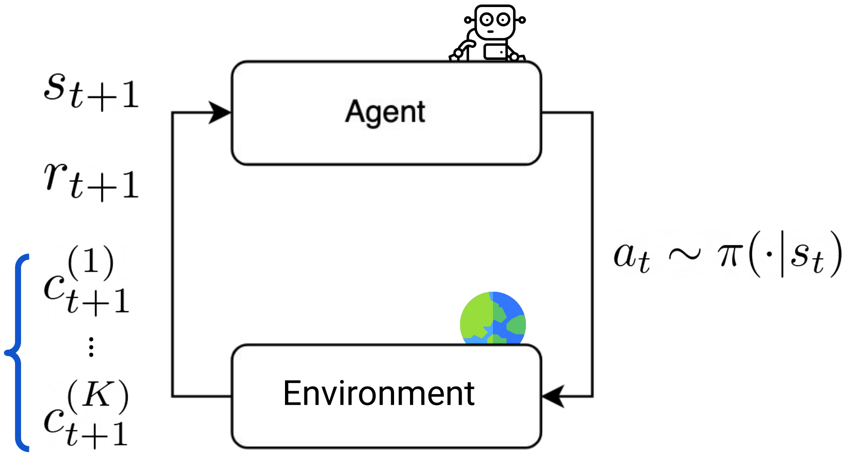
s_{t+1}

r_{t+1}

$a_t \sim \pi(\cdot|s_t)$

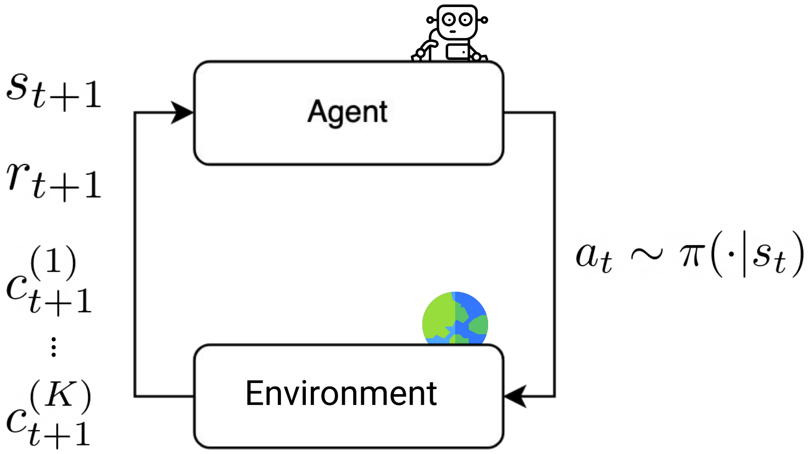
$$\pi^* = \text{argmax}_{\pi \in \Pi} \; J_R(\pi)$$

Constrained MDPs



$$\pi^* = \operatorname{argmax}_{\pi \in \Pi} J_R(\pi)$$

Constrained MDPs



$$\pi^* = \mathrm{argmax}_{\pi \in \Pi} \; J_R(\pi),$$
$$\text{s.t.} \quad J_{C_k}(\pi) \leq d_k \;, \quad k = 1, \ldots, K$$

Lagrangian methods

- Popular method to solve constrained optimisation problems

$$\max_{\pi} \min_{\lambda \geq 0} \mathcal{L}(\pi, \lambda)$$

$$\mathcal{L}(\pi, \lambda) = J_R(\pi) - \sum_{k=1}^{K} \lambda_k (J_{C_k}(\pi) - d_k)$$

Lagrangian methods

➢ Can be solved with gradient-based optimisation

Policy update:

$$\nabla_\pi \mathcal{L}(\pi, \lambda) = \nabla_\pi J_L(\pi),$$

$$L(s,a) = R(s,a) - \sum_{k=1}^{K} \lambda_k C_k(s,a)$$

Multipliers update:

$$\nabla_{\lambda_k} \mathcal{L}(\pi, \lambda) = -(J_{C_k}(\pi) - d_k)$$

Proposed approach

- ➢ Use a special family of CMDPs to ease the behavior specification task
- ➢ Use modified Lagrangian method to handle the many constraints case

In particular:

1. C_k as indicator cost functions
2. Normalized multipliers
3. Bootstrap constraint

Proposed approach

1. C_k **as indicator cost functions** $C_k(s,a) = I(\text{behavior } k \text{ is met in } (s,a))$

Proposed approach

1. C_k **as indicator cost functions** $$C_k(s,a) = I(\text{behavior } k \text{ is met in } (s,a))$$

By using an indicator cost-function, the expected discounted sum admits an intuitive interpretation:

$$\begin{aligned} J_{C_k}(\pi) &= \mathbb{E}_{\tau\sim p_\pi}\left[\sum_{t=0}^{T} \gamma^t C_k(s_t,a_t)\right] \\ &= Z(\gamma,T)\mathbb{E}_{(s,a)\sim x_\pi(s,a)}[C_k(s,a)] \\ &= Z(\gamma,T)\mathbb{E}_{(s,a)\sim x_\pi(s,a)}[I(\text{behavior } k \text{ met in } (s,a))] \\ &= Z(\gamma,T)Pr\big(\text{behavior } k \text{ met in } (s,a)\big),\ (s,a)\sim x_\pi \end{aligned}$$

Proposed approach

1. C_k **as indicator cost functions** $C_k(s,a) = I(\text{behavior } k \text{ is met in } (s,a))$

By using an indicator cost-function, the expected discounted sum admits an intuitive interpretation:

$$J_{C_k}(\pi) = \mathbb{E}_{\tau \sim p_\pi}\left[\sum_{t=0}^{T} \gamma^t C_k(s_t, a_t)\right]$$
$$= Z(\gamma, T)\mathbb{E}_{(s,a)\sim x_\pi(s,a)}[C_k(s,a)]$$
$$= Z(\gamma, T)\mathbb{E}_{(s,a)\sim x_\pi(s,a)}[I(\text{behavior } k \text{ met in } (s,a))]$$
$$= Z(\gamma, T)Pr\big(\text{behavior } k \text{ met in } (s,a)\big),\ (s,a) \sim x_\pi$$

This design choice allows us to easily specify the corresponding thresholds: $\tilde{d}_k \in [0,1]$

Proposed approach

2. Normalized multipliers

$$\lambda_k = \frac{\exp(z_k)}{\exp(a_0) + \sum_{k'=1}^{K} \exp(z_{k'})}, \quad k = 1, \ldots, K$$

$$\max_{\pi} \min_{z_{1:K} \geq 0} \mathcal{L}(\pi, \lambda)$$

$$\mathcal{L}(\pi, \lambda) = \lambda_0 J_R(\pi) - \sum_{k=1}^{K} \lambda_k (J_{C_k}(\pi) - d_k)$$

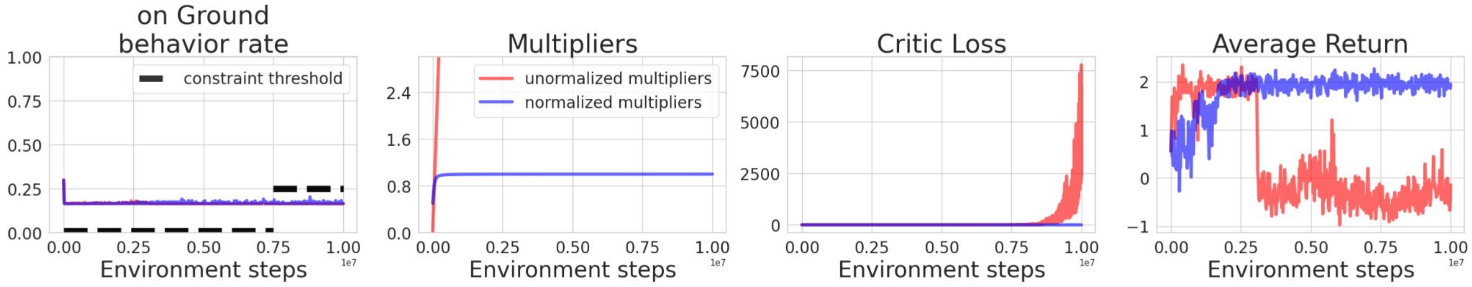
Proposed approach

2. Normalized multipliers

$$\lambda_k = \frac{\exp(z_k)}{\exp(a_0) + \sum_{k'=1}^{K} \exp(z_{k'})}, \quad k = 1, \ldots, K$$

$$\max_{\pi} \min_{z_{1:K} \geq 0} \mathcal{L}(\pi, \lambda)$$

$$\mathcal{L}(\pi, \lambda) = \lambda_0 J_R(\pi) - \sum_{k=1}^{K} \lambda_k (J_{C_k}(\pi) - d_k)$$



Proposed approach

3. **Bootstrap constraint**

Idea: Granting to our reward function the same powers our constraints have

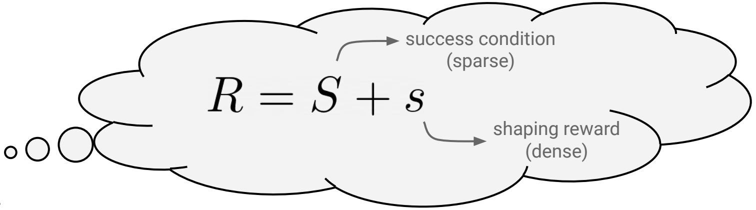
But: Want to preserve a maximisation problem

Proposed approach

3. Bootstrap constraint

Idea: Granting to our reward function the same powers our constraints have

But: Want to preserve a maximisation problem

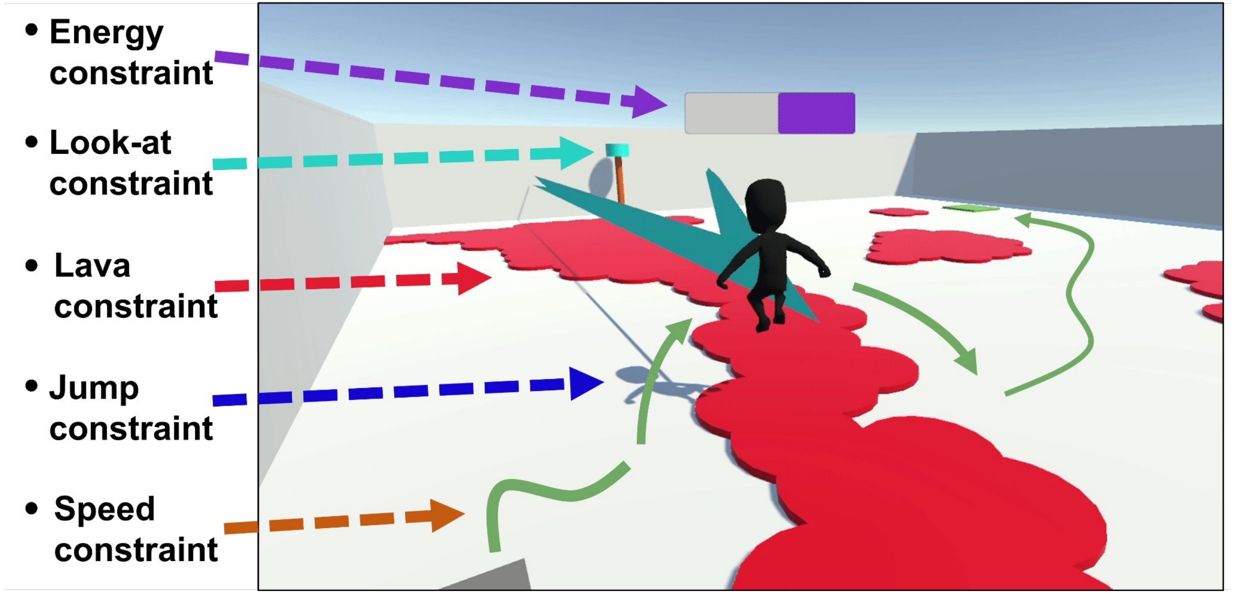


1. We add a success constraint S_{K+1}

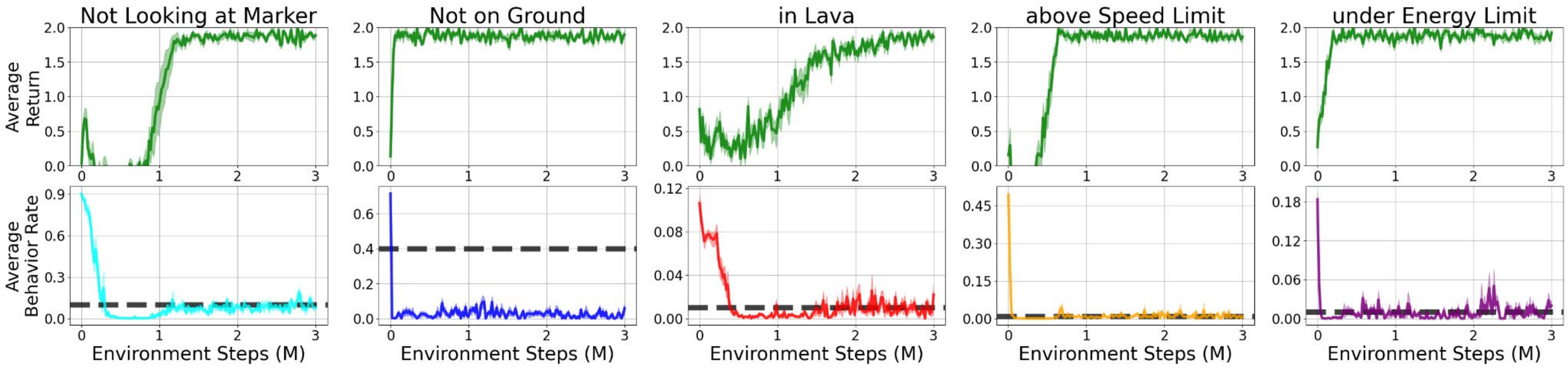
2. We lend its multiplier λ_{K+1} to the main reward function R when constraints are unsatisfied, and use λ_0 otherwise

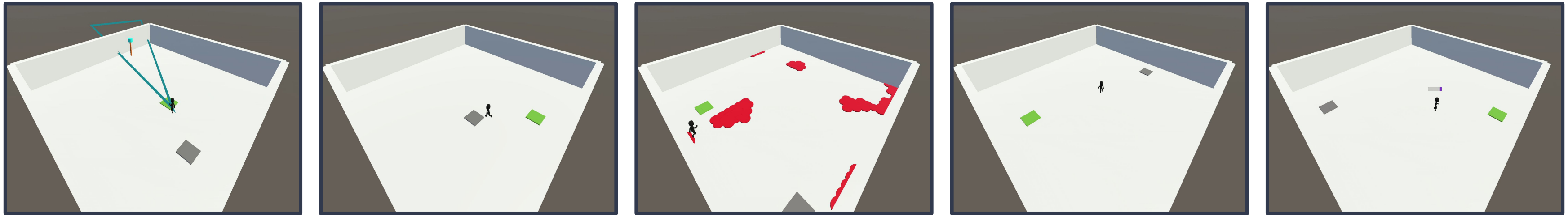
$$\tilde{\lambda}_0 = max(\lambda_0, \lambda_{K+1})$$

Experimental Setup: Arena env

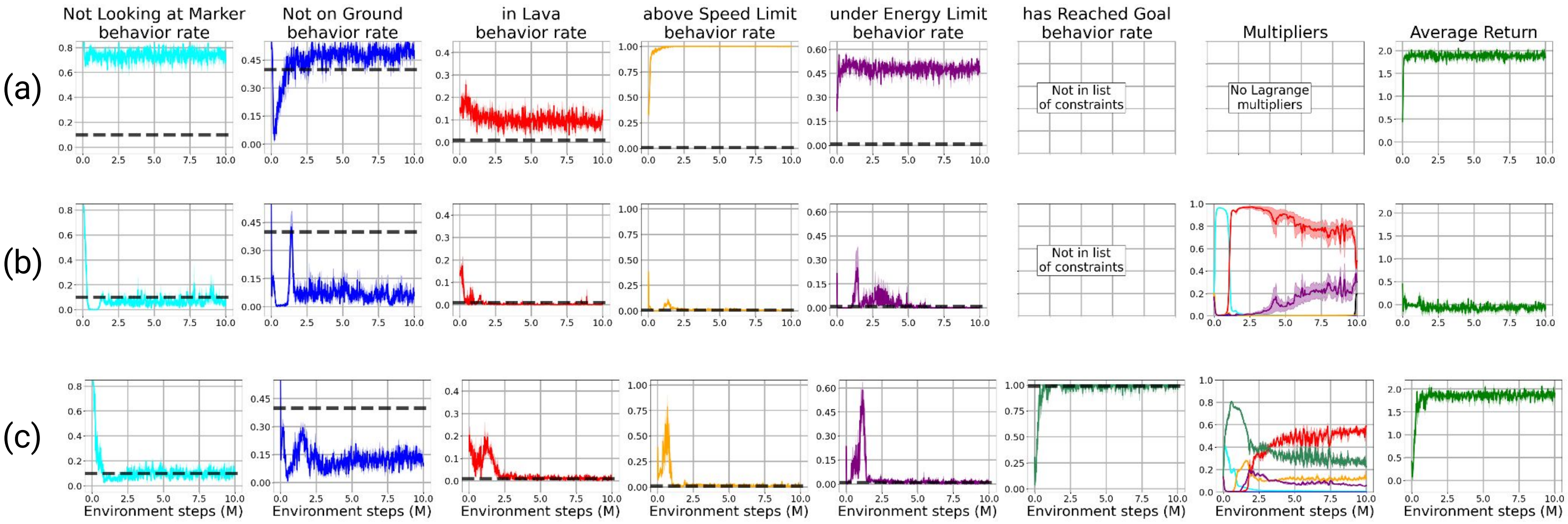


Results: single constraint case



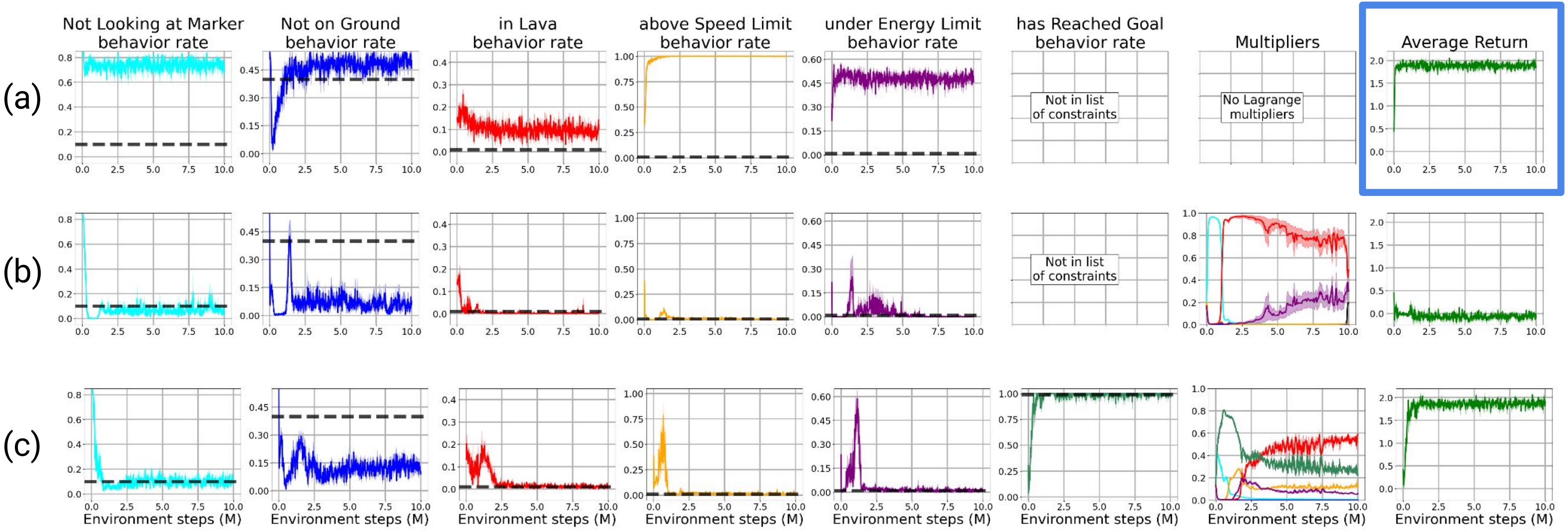


Results: SAC-Unconstrained



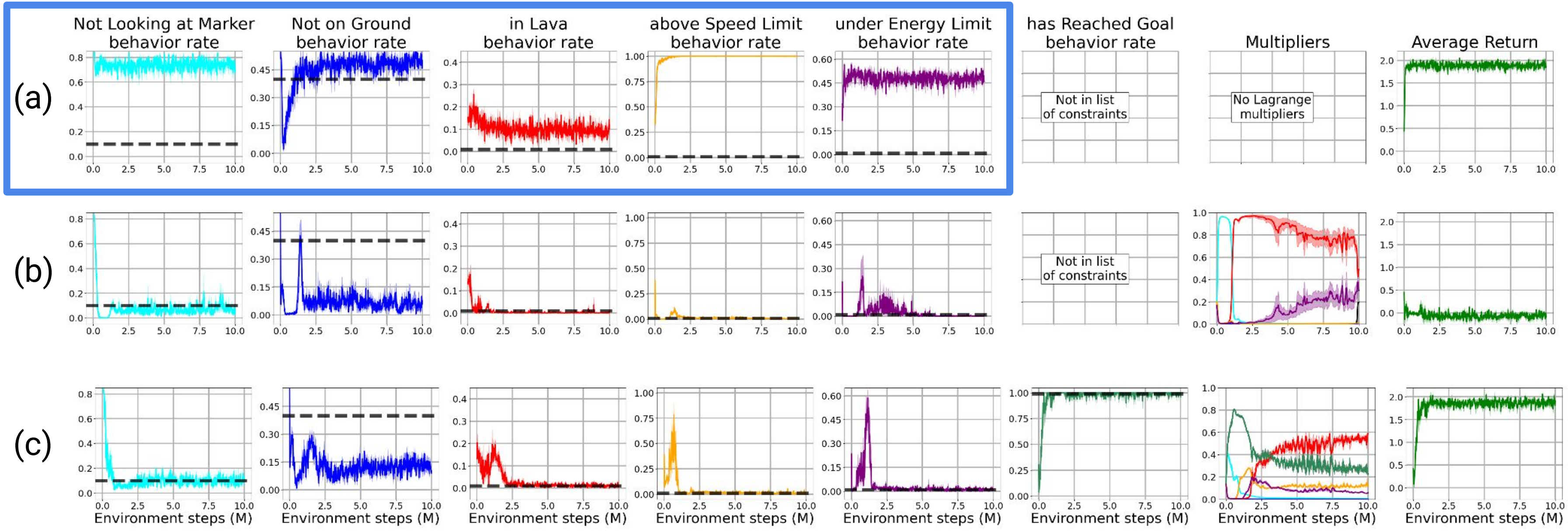
(a): SAC (unconstrained)
(b): SAC-Lagrangian
(c): SAC-Lagrangian + Bootstrap constraint

Results: SAC-Unconstrained



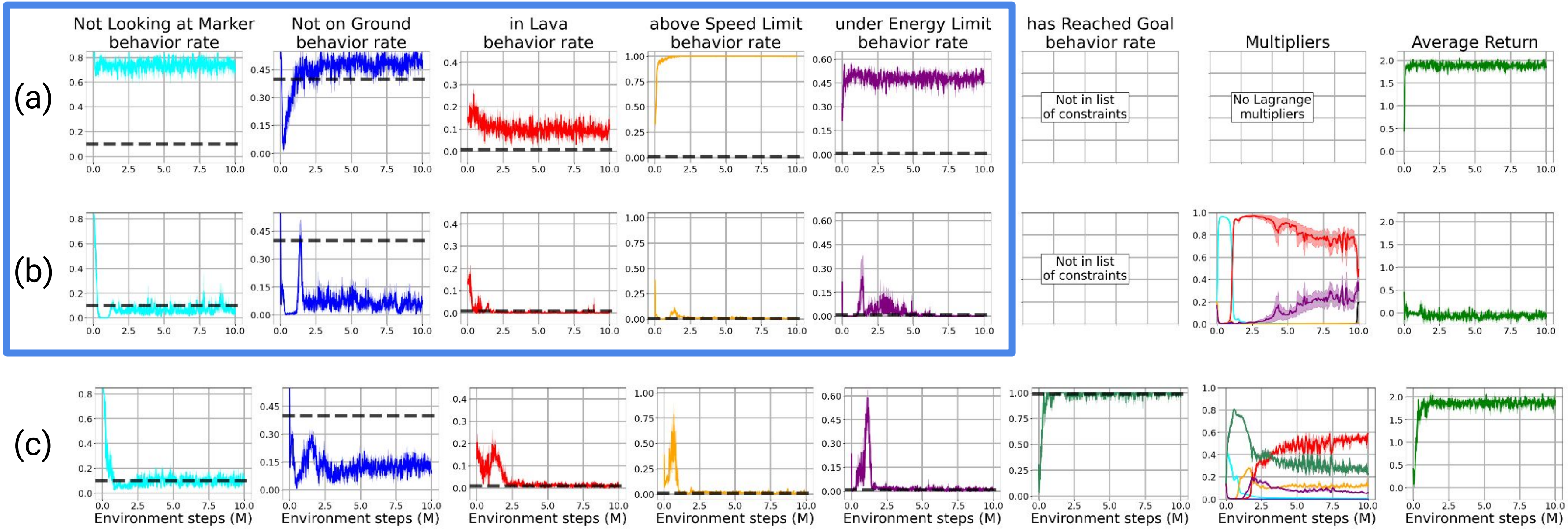
(a): SAC (unconstrained)
(b): SAC-Lagrangian
(c): SAC-Lagrangian + Bootstrap constraint

Results: SAC-Unconstrained



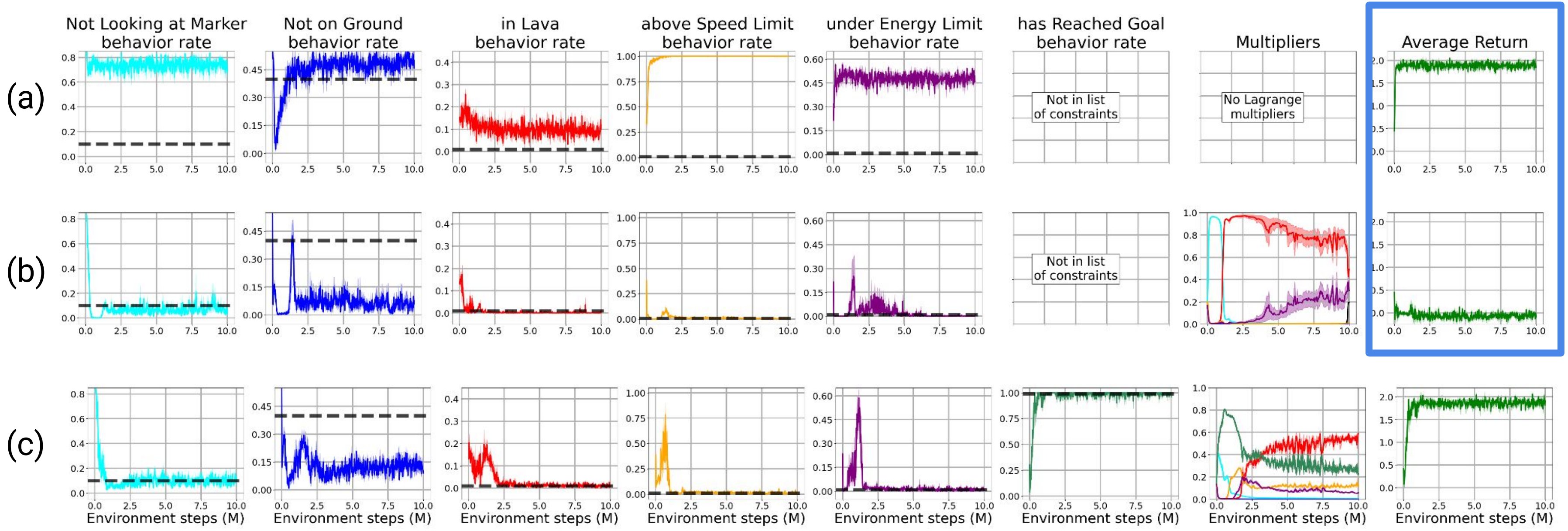
(a): SAC (unconstrained)
(b): SAC-Lagrangian
(c): SAC-Lagrangian + Bootstrap constraint

Results: SAC-Unconstrained



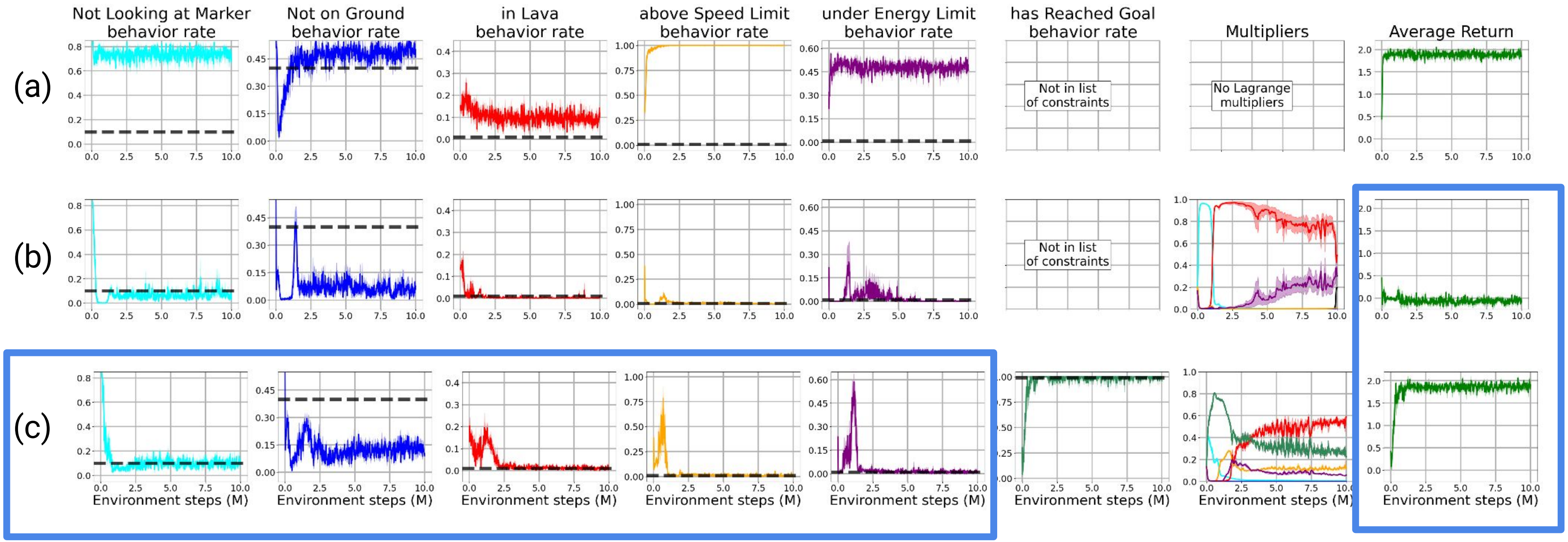
(a): SAC (unconstrained)
(b): SAC-Lagrangian
(c): SAC-Lagrangian + Bootstrap constraint

Results: SAC-Unconstrained



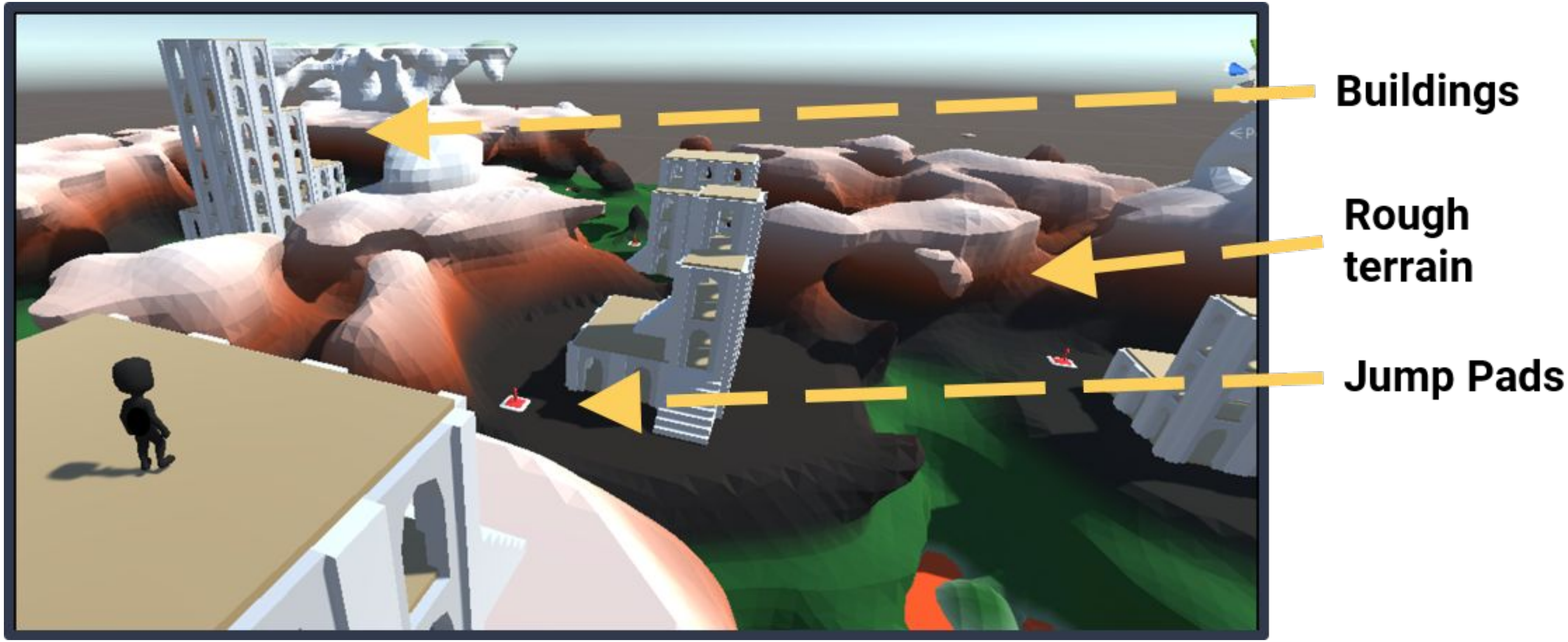
(a): SAC (unconstrained)
(b): SAC-Lagrangian
(c): SAC-Lagrangian + Bootstrap constraint

Results: SAC-Unconstrained



(a): SAC (unconstrained)
(b): SAC-Lagrangian
(c): SAC-Lagrangian + Bootstrap constraint

Experimental Setup: OpenWorld env



Conclusion

Conclusion

1. RL often produces unforeseen behaviors

Conclusion

1. RL often produces unforeseen behaviors

2. Reward engineering does not scale well with the task complexity

Conclusion

1. RL often produces unforeseen behaviors

2. Reward engineering does not scale well with the task complexity

3. A special case of CMDPs offers a viable solution to behavior specification

Thank you!