Asynchronous Zeroth-Order Distributed Optimization with Residual Feedback

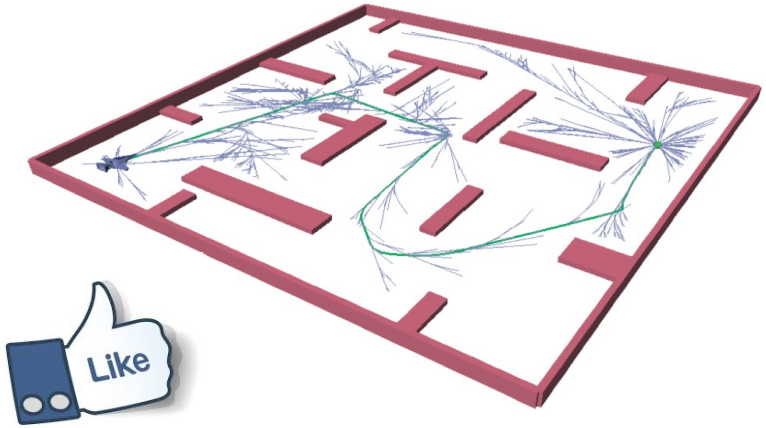
Yi Shen
Mechanical Engineering & Materials Science
Duke University

Joint work with: Yan Zhang, Scott Nivison, Zachary I. Bell and Michael M. Zavlanos

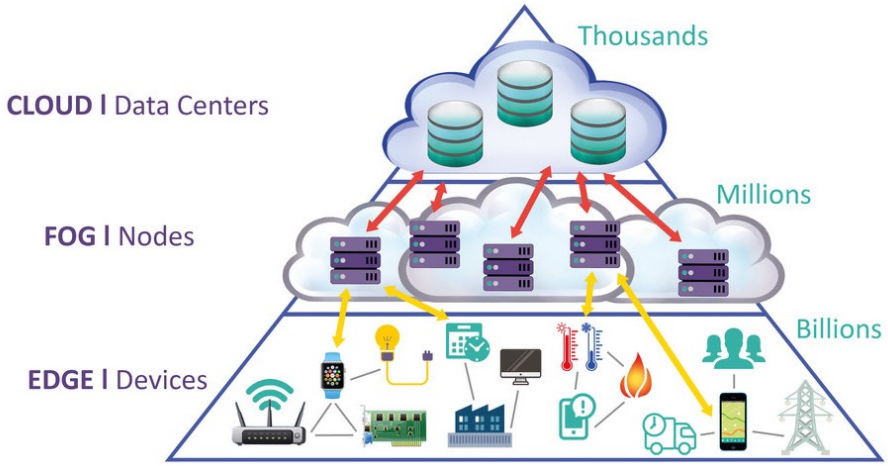
Assured Autonomy in Contest Environments (AACE)
Fall 2021 Review
Nov 9th, 2021

Duke
UNIVERSITY

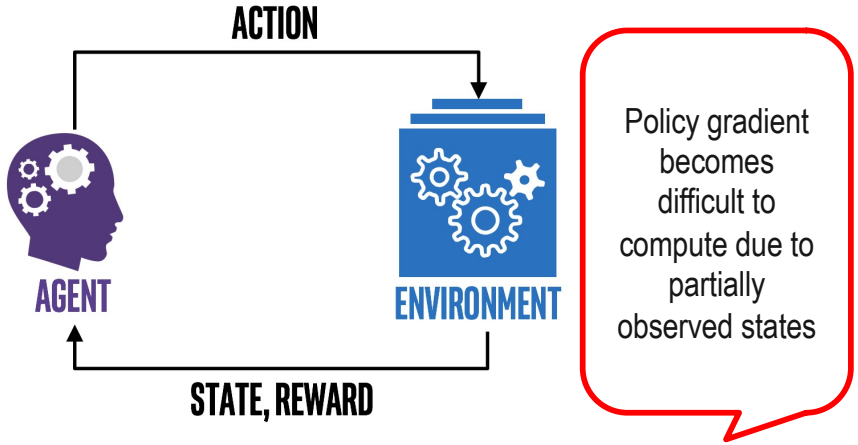
Optimization with Complex or Unknown Models



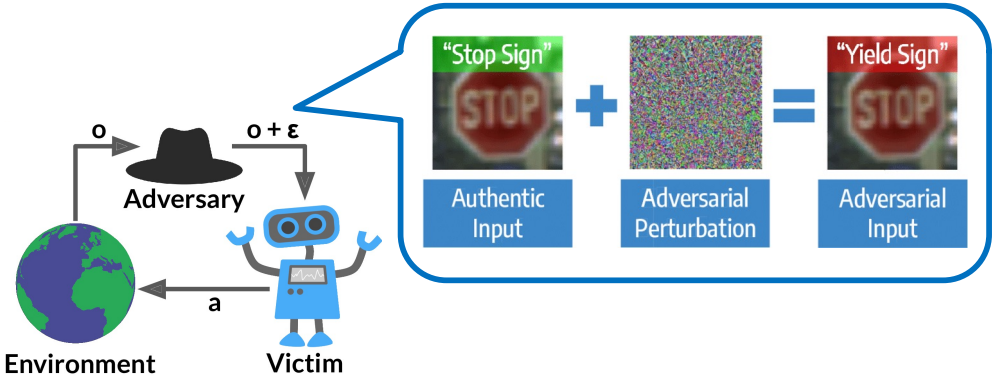
Human-in-the-loop robot planning



Computation management in IoT systems



Policy search in RL $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



Adversarial attacks and defense in DRL

Zeroth-Order (Derivative-Free) Optimization

Optimization problem: $\min_{x \in \mathbb{R}^n} f(x)$

Gradient is unavailable, incomputable, private

Zeroth-order gradient estimators:

The one-point estimator $G_\delta(x_t) = \dfrac{f(x_t + \delta u_t)}{\delta} u_t$ is subject to large variance and, therefore, slow convergence rate.

The two-point estimator $G_\delta(x_t) = \dfrac{f(x_t + \delta u_t) - f(x_t)}{\delta} u_t$ has less variance but requires the objective function to be time-invariance.

Duke UNIVERSITY

Asynchronous Distributed Optimization

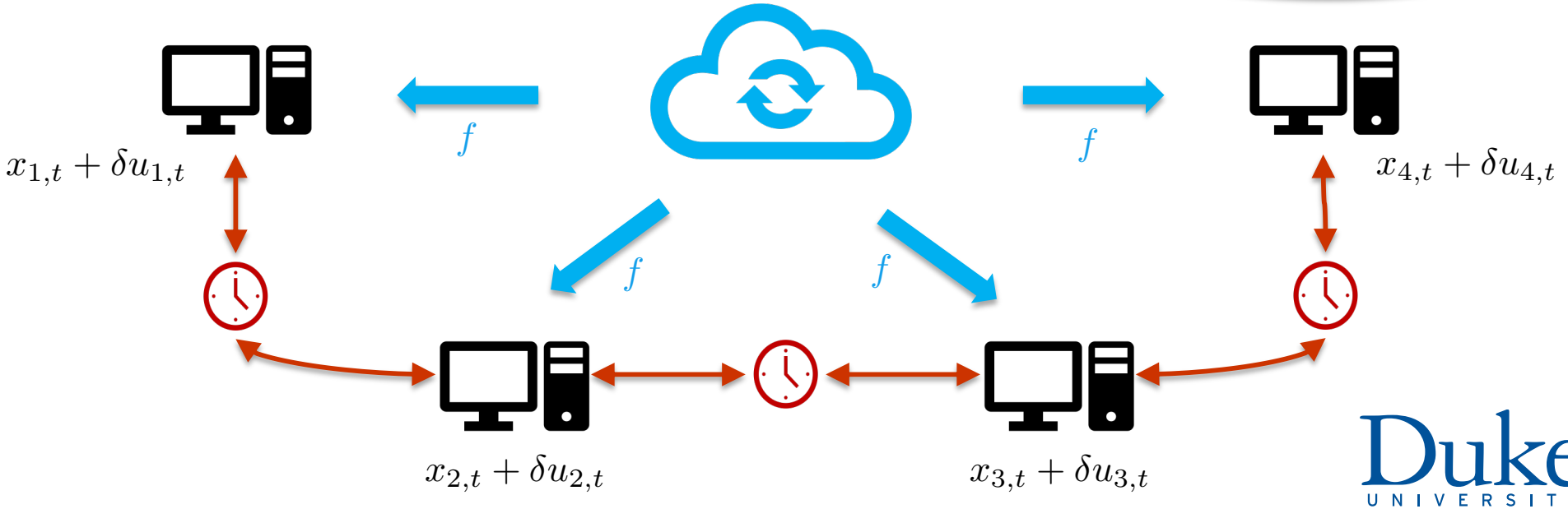
Distributed Optimization Problem: $\min_x f(x)$

where the decision vector $x = [x_1^T, x_2^T, \ldots, x_N^T]^T$ concatenates local decision variables.

Synchronous Gradient Estimator:

$$G_\delta(x_t) = \frac{f(x_{1,t} + \delta u_{1,t}, ..., x_{1,N} + \delta u_{1,N}) - f(x_{1,t}, ..., x_{1,N})}{\delta} u_t$$

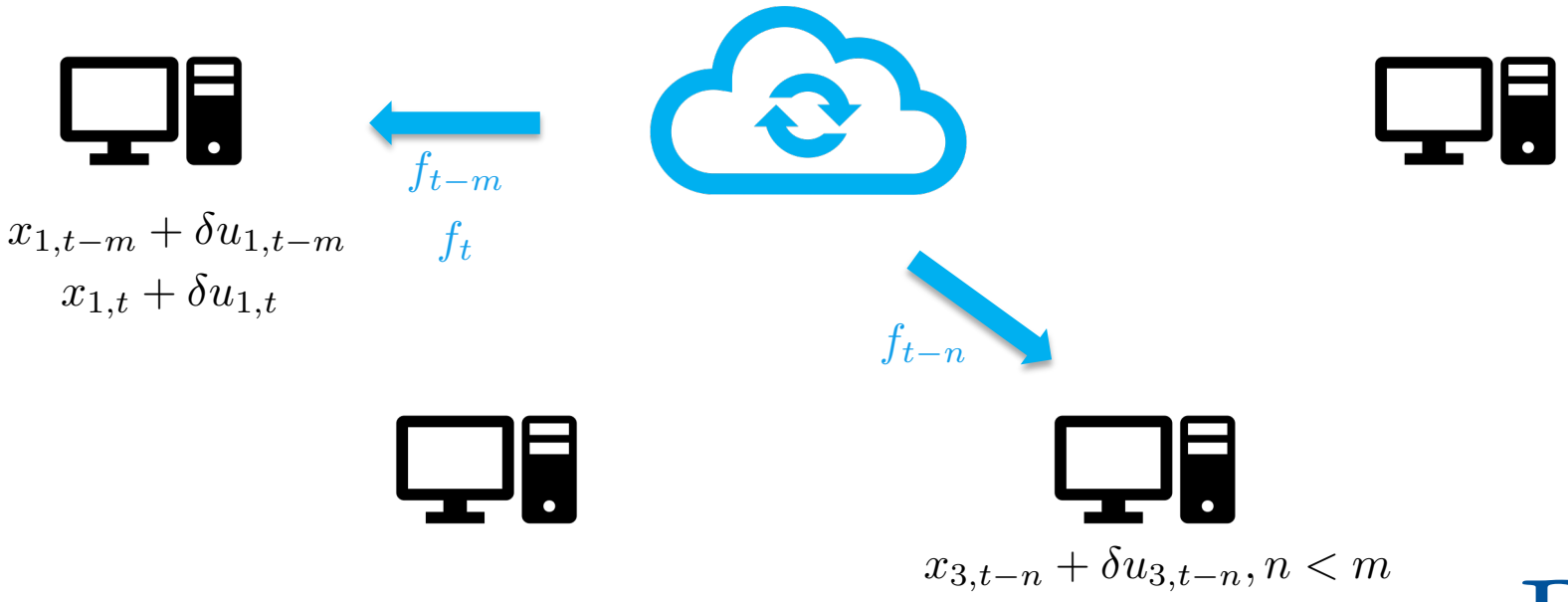
Require synchronous perturbation and updates



Duke UNIVERSITY

Asynchrony Model

Definition: (Asynchrony Model) At each time step, one agent is independently and randomly selected according to a fixed distribution $P = [p_1, p_2, \ldots, p_N]$. The selected agent i can query the value of the cost function once and update its local decision variable, while the decisions of the other agents $\{x_j\}_{j \neq i}$ are fixed.



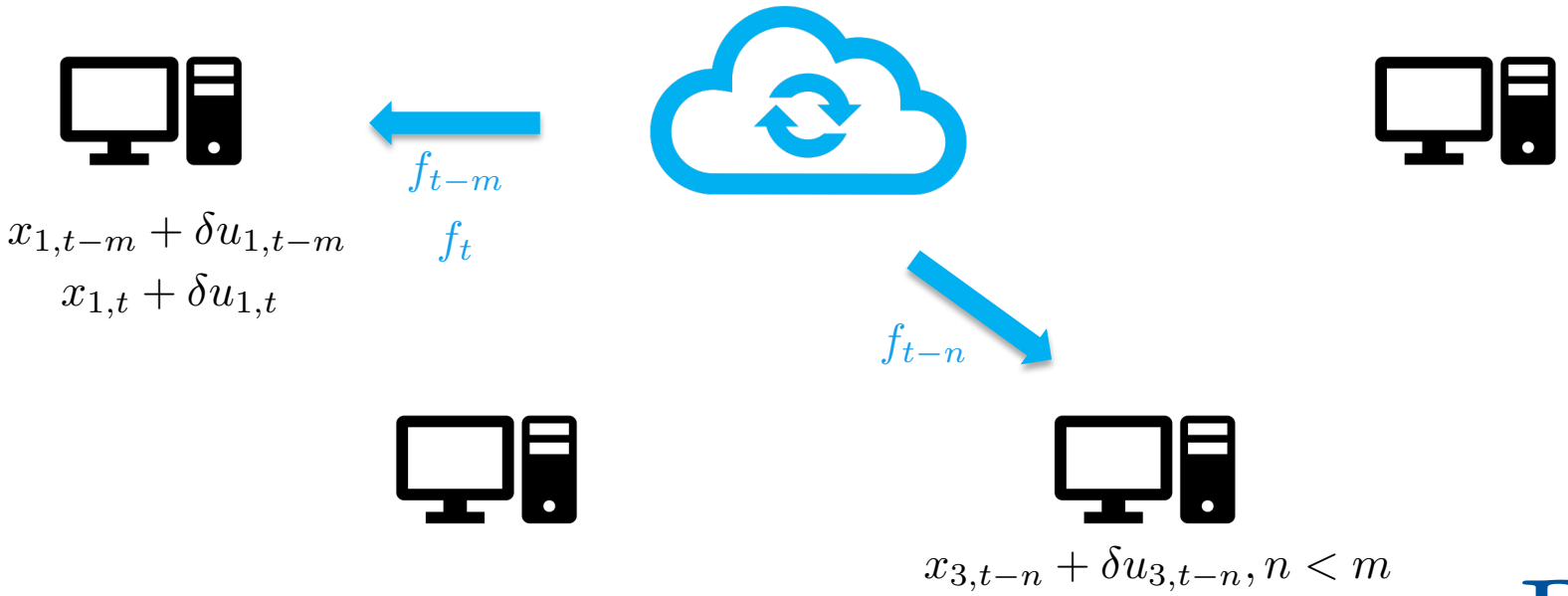
Duke UNIVERSITY

Asynchrony Residual-Feedback Gradient Estimator

Asynchronous Gradient Estimator:

$$G_{i,\delta}(x_t) = \frac{u_{i,t}}{\delta}\left(f(x_t + \delta u_{i,t}) - f(x_{t-m} + \delta u_{i,t-m})\right)$$

In the asynchronous setting, $u_{i,t}$ only perturbs the local decision $x_{i,t}$ of agent i, who is activated at time t. The other decision variables in x_t can be different from those in x_{t-m}.



Duke UNIVERSITY

Second Moment of the Asynchronous Gradient Estimator

Consider the asynchronous update at agent i, who is activated at time t:

$$x_{i,t+1} = x_{i,t} + \alpha G_{i,\delta}(x_t)$$

where $G_{i,\delta}(x_t) = \frac{u_{i,t}}{\delta}\left(f(x_t + \delta u_{i,t}) - f(x_{t-m} + \delta u_{i,t-m})\right)$

Lemma Under the Asynchrony Model, and define by

$$\mathbb{E}[\|G_\delta(x_t)\|^2] := \mathbb{E}_{i_t}\left[\mathbb{E}[\|\tilde{\nabla}_i f(x_t)\|^2 | i_t = i]\right].$$

Then, running the asynchronous updates, we have that

$$\mathbb{E}[\|G_\delta(x_t)\|^2] \leq \frac{2\bar{n}L_0^2\alpha^2 t}{\mu^2}\sum_{m=0}^{t-1}(1-p_{\min})^m \mathbb{E}[\|G_\delta(x_{t-m-1})\|^2] + 4L_0^2\left((4+\bar{n})^2 + \bar{n}^2\right)$$

The second moment at time t depends on the whole history of the algorithm.

Duke UNIVERSITY

Convergence Analysis

Theorem Under the Asynchrony Model, and run the asynchronous updates for T iterations. Let $\tilde{x}$ be uniformly randomly selected from T iterations. Selecting the step size $\alpha = \sqrt{p_{\min}}/T^{\frac{2}{3}}$ and the smoothing parameter $\delta = 2L_0\sqrt{n}/T^{\frac{1}{6}}$, we have that

$$\mathbb{E}[\|\nabla f(\tilde{x})\|^2] \leq \mathcal{O}(\bar{n}^2 T^{-\frac{1}{3}}).$$

Deterministic asynchronous algorithm achieves the same convergence rate as the stochastic synchronous algorithm. The asynchrony can be thought as one source of function evaluation noise at local agents.

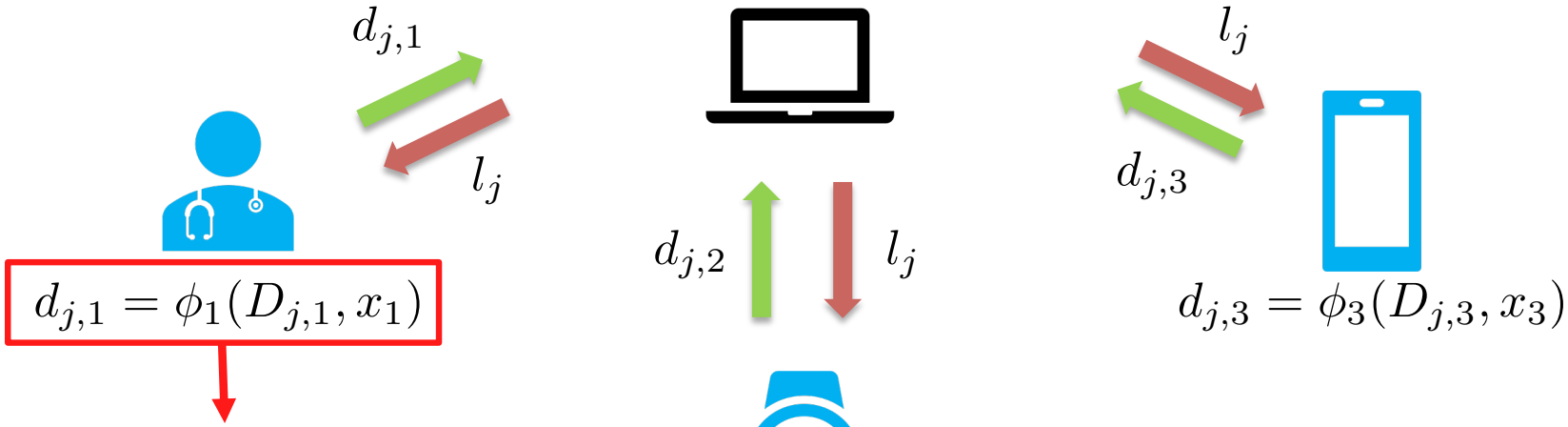
Duke UNIVERSITY

Numerical Experiments

Distributed Feature Learning Problem

For each patient j, different types of data are collected at heterogenous data collectors. These data are used collaboratively to make predictions.

$$\min_{x_1,x_2,x_3} l(\{d_{j,1:3}, y_j\}) := \sum_j \log\left(\sigma(y_i \sum_i W_i d_{j,i})\right)$$



Data collector 1 collects raw bio-signal data $D_{j,1}$ for patient j, extracts low dimensional biomarker feature $d_{j,1}$ through feature extraction function ϕ_1 with parameter x_1.

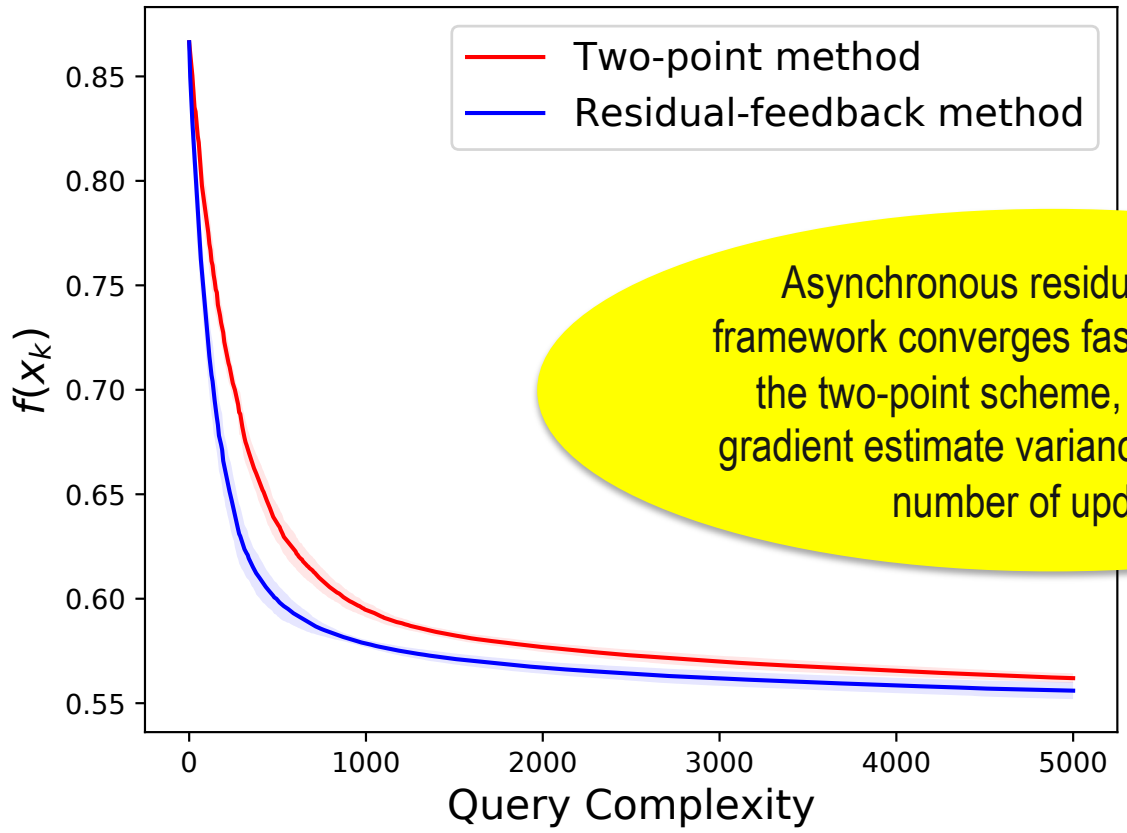
$$d_{j,2} = \phi_2(D_{j,2}, x_2)$$

Duke UNIVERSITY

Numerical Experiments

Comparison to an asynchronous two-point gradient estimator

$$G_{i,\delta}(x_t) = \frac{u_{i,t}}{\delta}\left(f(x_t + \delta u_{i,t}) - f(x_{t-m})\right)$$



Asynchronous residual-feedback framework converges faster compared to the two-point scheme, due to its low gradient estimate variance and twice the number of updates.

Duke UNIVERSITY

Summary

We proposed an asynchronous zeroth-order gradient estimator using residual feedback, proved that the converge rate is the same as stochastic synchronous cases, and validated it in an asynchronous distributed feature learning problem.

Duke
UNIVERSITY

Support

Thank You for Your Attention !

Duke
UNIVERSITY

Transfer Reinforcement Learning in Heterogeneous Action Spaces using Subgoal Mapping

Kavinayan Sivakumar
Electrical and Computer Engineering
Duke University

Joint work with: Yan Zhang, Scott Nivison, Zachary I. Bell and Michael M. Zavlanos

Assured Autonomy in Contest Environments (AACE)
Fall 2021 Review
Nov 9th, 2021

Duke
UNIVERSITY

Outline

Transfer Reinforcement Learning

Few-Shot RL in Heterogeneous Action Spaces using Subgoal Mapping

Duke
UNIVERSITY

Motivation for Transfer RL

Duke
UNIVERSITY

Outline

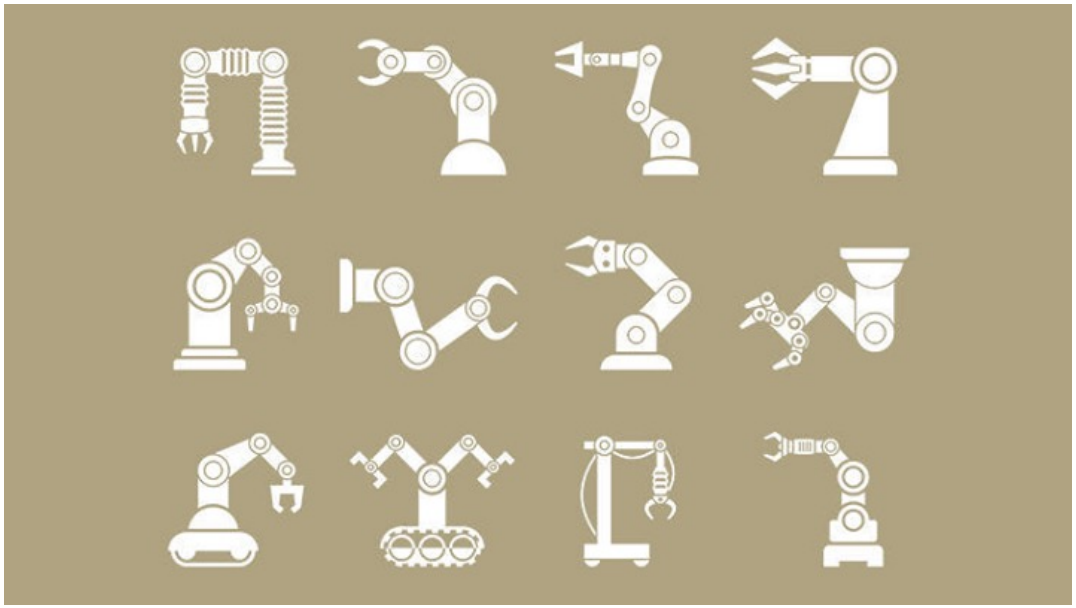
Transfer Reinforcement Learning

Few-Shot RL in Heterogeneous Action Spaces using Subgoal Mapping

- **Problem setting and related method**
- Proposed method
- Numerical experiments

Duke UNIVERSITY

Transfer RL with Heterogeneous Agents



Duke
UNIVERSITY

Problem Formulation

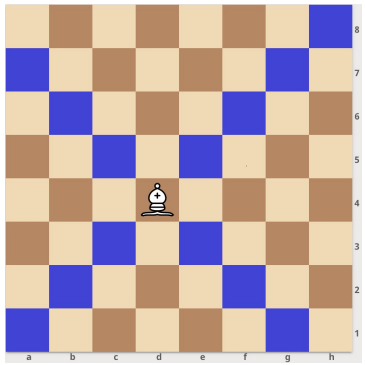
Problem statement:

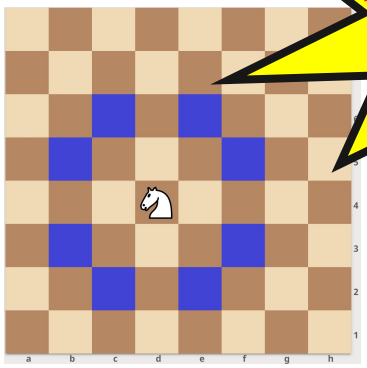
Define a source data set D_s with expert and learner demonstrations for some tasks sampled from distribution D, with expert demonstrations in action space A_s and learner demonstrations in action space A_l. The goal is to transfer knowledge from expert to learner so the learner can quickly solve unseen tasks in D using D_s.

Terminology:

Task sampled from overall distribution $d \sim D(p)$

Trajectory of an agent for a task $\tau(d) = [(s_0, a_0), (s_1, a_1), \ldots$





Knight and bishop have different action spaces but similar tasks on the chessboard

Duke UNIVERSITY

Previous Approaches

Neural Network Architectures: [M. Wan, 2020]

- Uses a fixed teacher policy to share policy parameters with a student policy
- Blending policies can be complex in many environments, especially if action spaces are so different
- If there is little overlap between different agents, then sharing of parameters is not as useful

Modular Approaches: [C. Devin, 2016]

- Uses modular blocks that represent policies to solve tasks to combine with agent blocks
- Requires explicit meshing of blocks
- Has similar limitation to architecture based heterogeneous transfer

Handcoded Mappings: [J. Karttunen, 2019]

- Human expert maps between two action spaces that are different
- Can be subject to bias, accuracy is not guaranteed
- Not scalable
- Mappings are most agnostic to very differing action spaces

Duke UNIVERSITY

Outline

Transfer Reinforcement Learning

Few-Shot RL in Heterogeneous Action Spaces using Subgoal Mapping

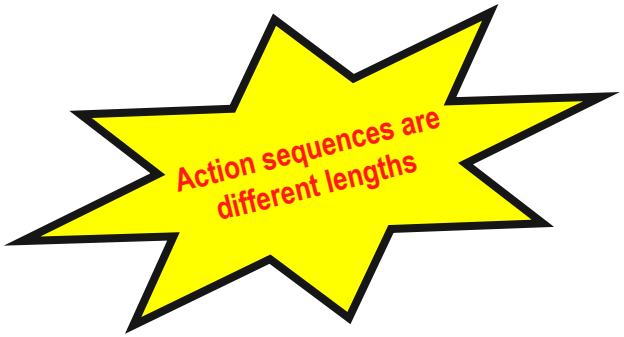
- Problem setting and related method
- **Proposed method**
- Numerical experiments

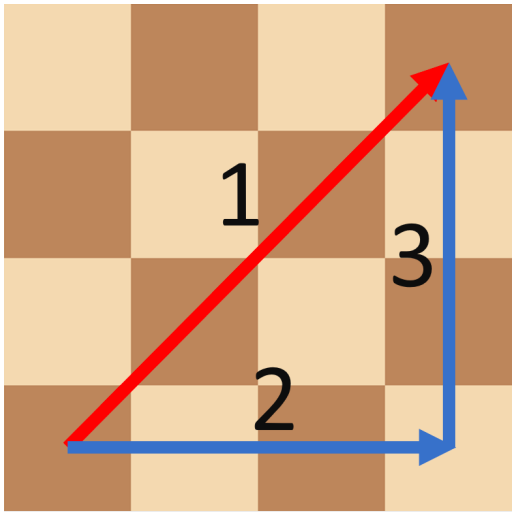
Duke UNIVERSITY

Intuition Behind Algorithm

Varying primitive action sequences

Red agent: Moves diagonally
Blue agent: Moves vertically and horizontally





Solution: use trajectories of subgoals instead of primitive actions to break policy hierarchically

Duke UNIVERSITY

Hierarchical Algorithm Design

Hierarchical Policy Functions

Mapping Function

$$\pi_\mu : (\{g_{\text{expert}}\}_{t=0}^{T}, p) \rightarrow \{g_{\text{learner}}\}_{t=0}^{T}$$

Subgoal Function (High Level)

$$\pi_h : \{g_{\text{learner}}\}_{t=0}^{T} \rightarrow g_{\text{learner}}$$

Primitive Action Function (Low Level)

$$\pi_\lambda : (s_{\text{learner}}, g_{\text{learner}}) \rightarrow \{a_{\text{learner}}\}_{t=0}^{T}$$

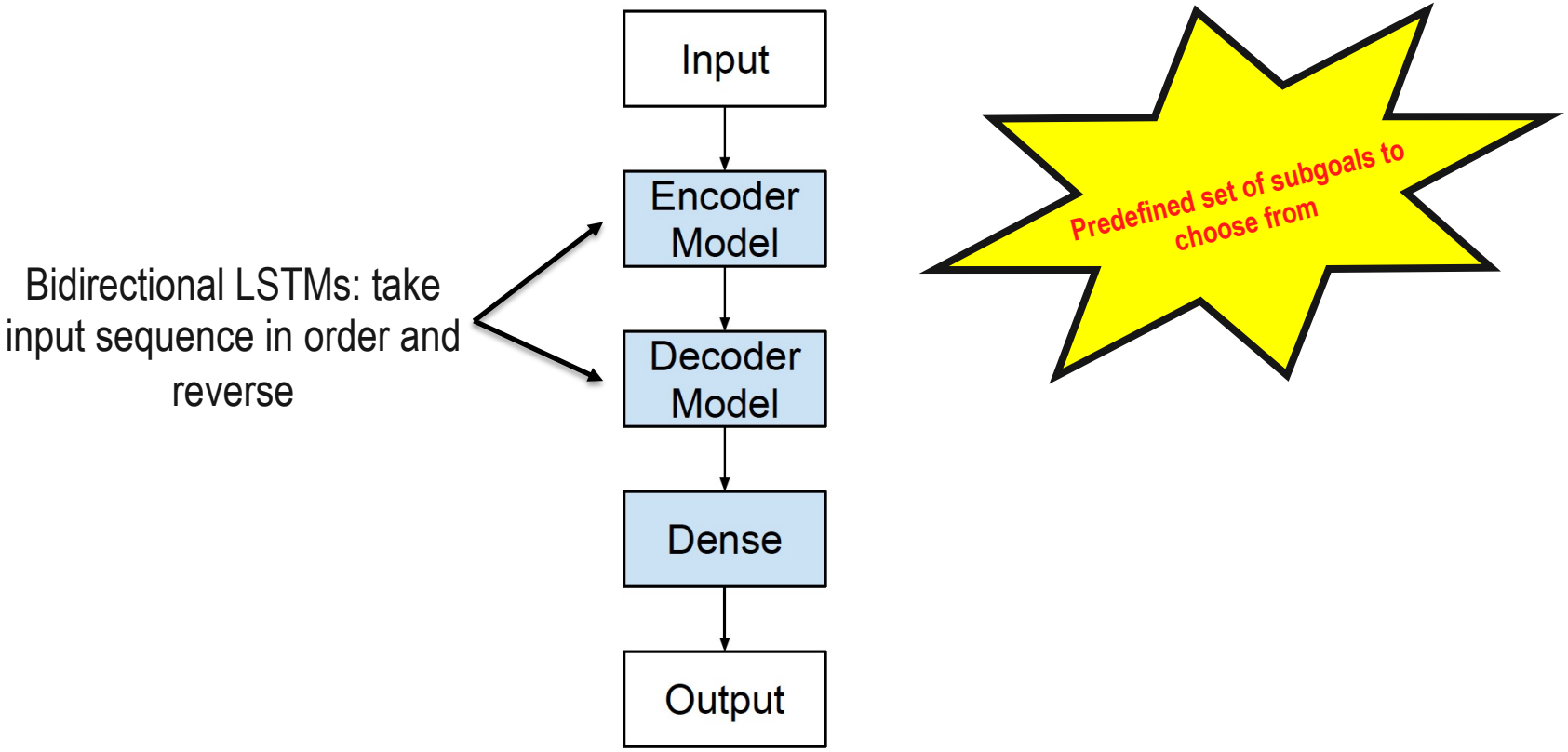
Duke UNIVERSITY

Subgoal Sequence as a LSTM

LSTM Neural Network Model and Training

$$\pi_\mu : (\{g_{\text{expert}}\}_{t=0}^{T}, p) \rightarrow \{g_{\text{learner}}\}_{t=0}^{T}$$

Sequence to Sequence



Duke UNIVERSITY

Algorithm Design

Warm Initializing High Level Learner Policy

- LSTM output for an unseen task is a sequence of predicted subgoals for the learner agent
- However, LSTM output might not always be accurate: how can we extract useful subgoal information if there are any errors?
- Solution: warm initialize the high level learner policy with supervised learning, as the goal is to bias the weights initially towards the LSTM model's subgoal predictions

Learning Optimal Policy from Warm Initialized High Level Policy

- Once the high level learner policy is warm initialized, we use a standard reinforcement learning algorithm to train the high level and low level policies for that unseen task
- If LSTM output is correct, then the training time to converge onto the optimal policy must be faster than training the policy without any transfer

Duke
UNIVERSITY

Outline

Transfer Reinforcement Learning

Few-Shot RL in Heterogeneous Action Spaces using Subgoal Mapping

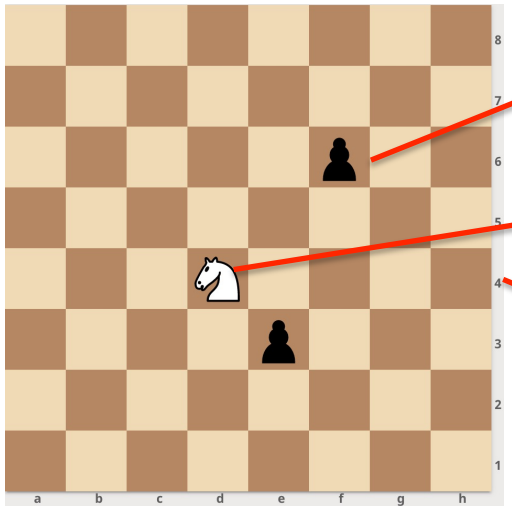
- Problem setting and related method
- Proposed method
- **Numerical experiments**

Duke UNIVERSITY

Numerical Experiments

Example of a task within overall set

Goal is to take both pawns within episode (10 timesteps). -1 for an empty square, +10 for a pawn



Pawns always on dark squares to accommodate dark square bishop

Assume same fixed position for both agents

Global coordinates for each square go from left to right, bottom to top

Duke UNIVERSITY

LSTM Training and Output

Training the LSTM Model

Compute bishop and knight expert trajectories using Djikstra's algorithm

↓

One-hot encode both trajectories

↓

Input through LSTM Neural Network Model and train with categorical cross entropy loss function

LSTM Output on Unseen Task

Padding built in to allow for variable output lengths

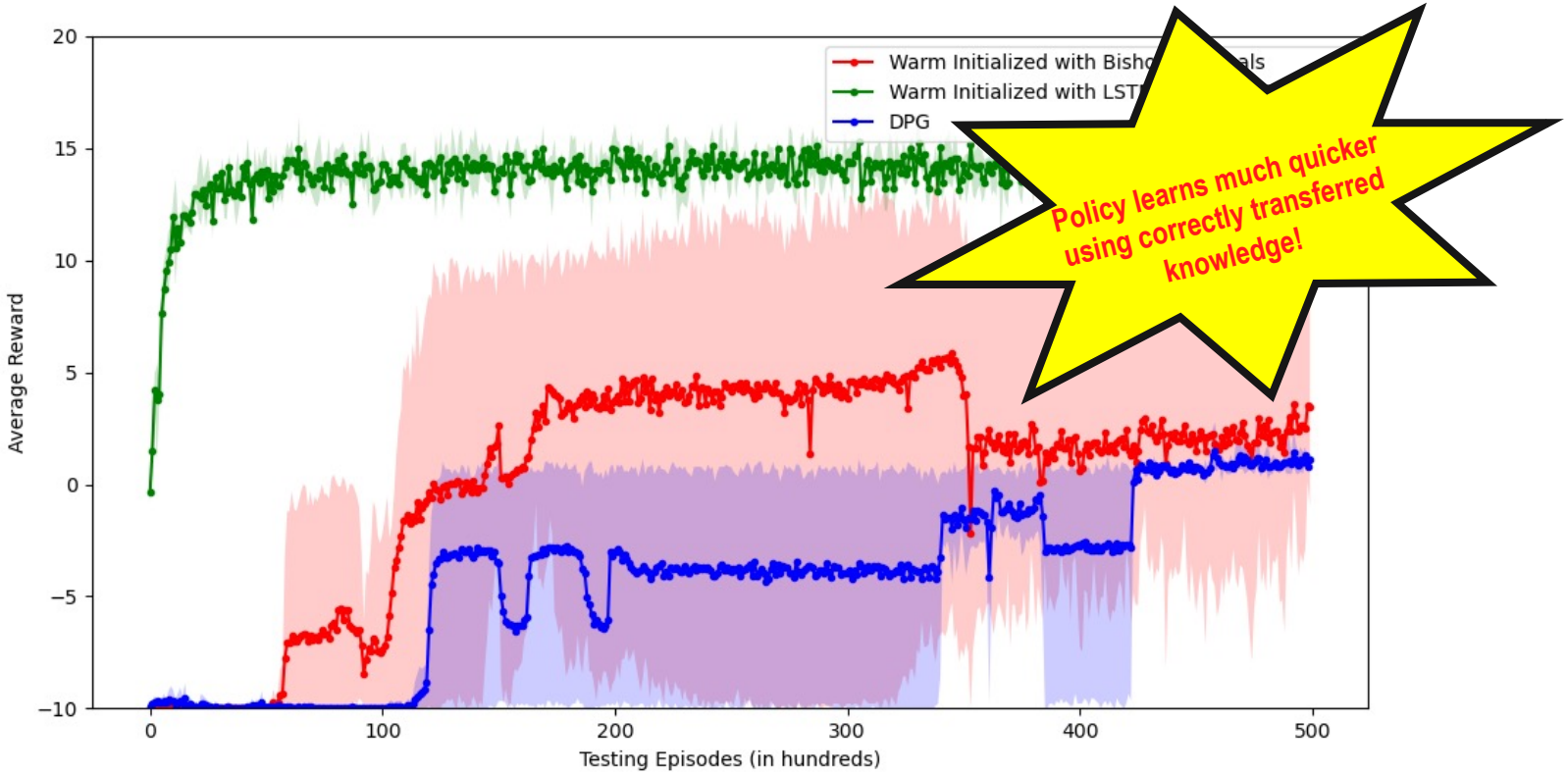
```
=>          37 43 28 43  (expect          37 43 28 34 )

=>          26 32 26 9  (expect          42 32 26 9 )
```

Duke UNIVERSITY

Numerical Experiments

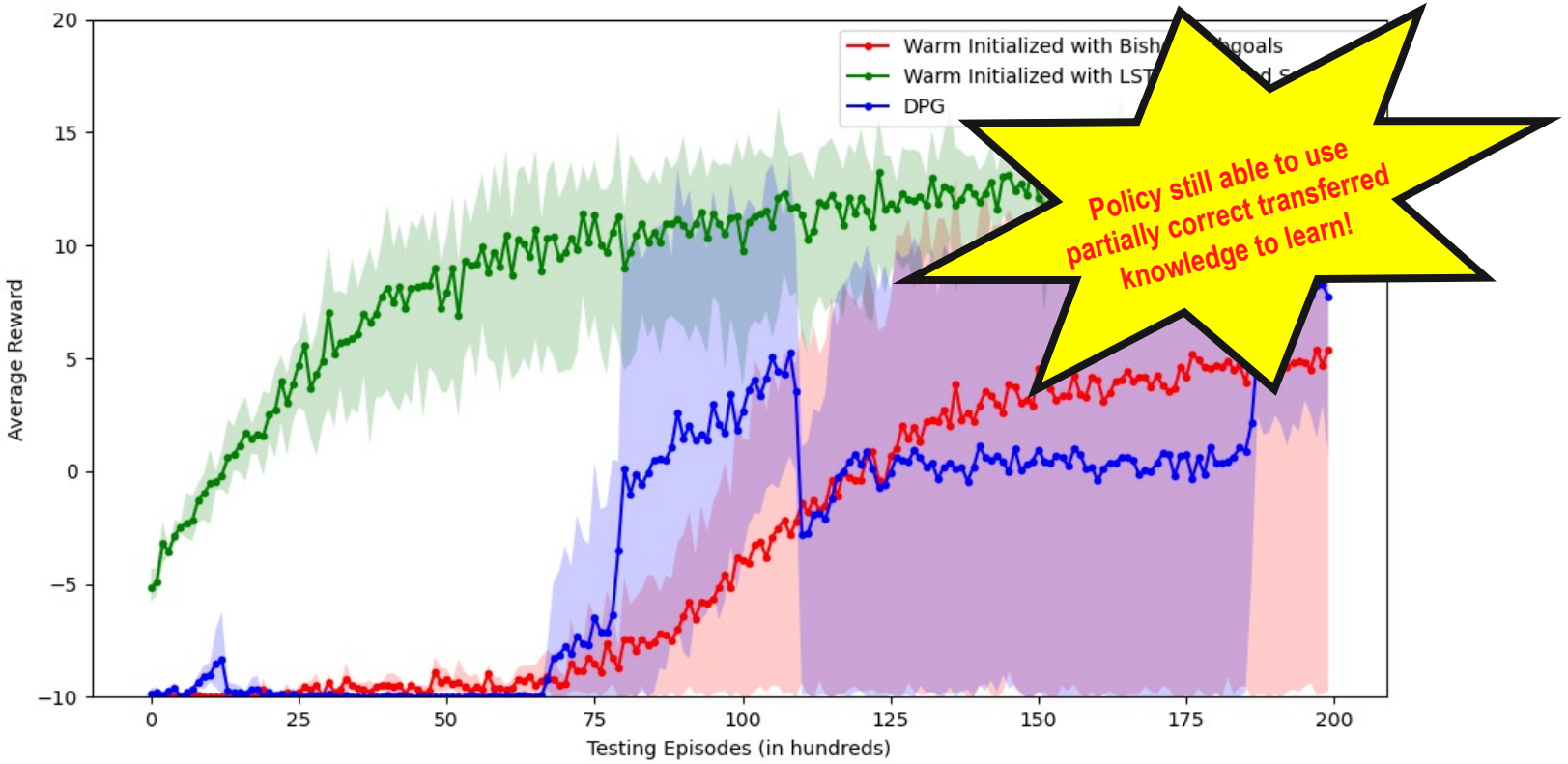
LSTM Correctly Predicts Learner Sequence of Subgoals



Duke UNIVERSITY

Numerical Experiments

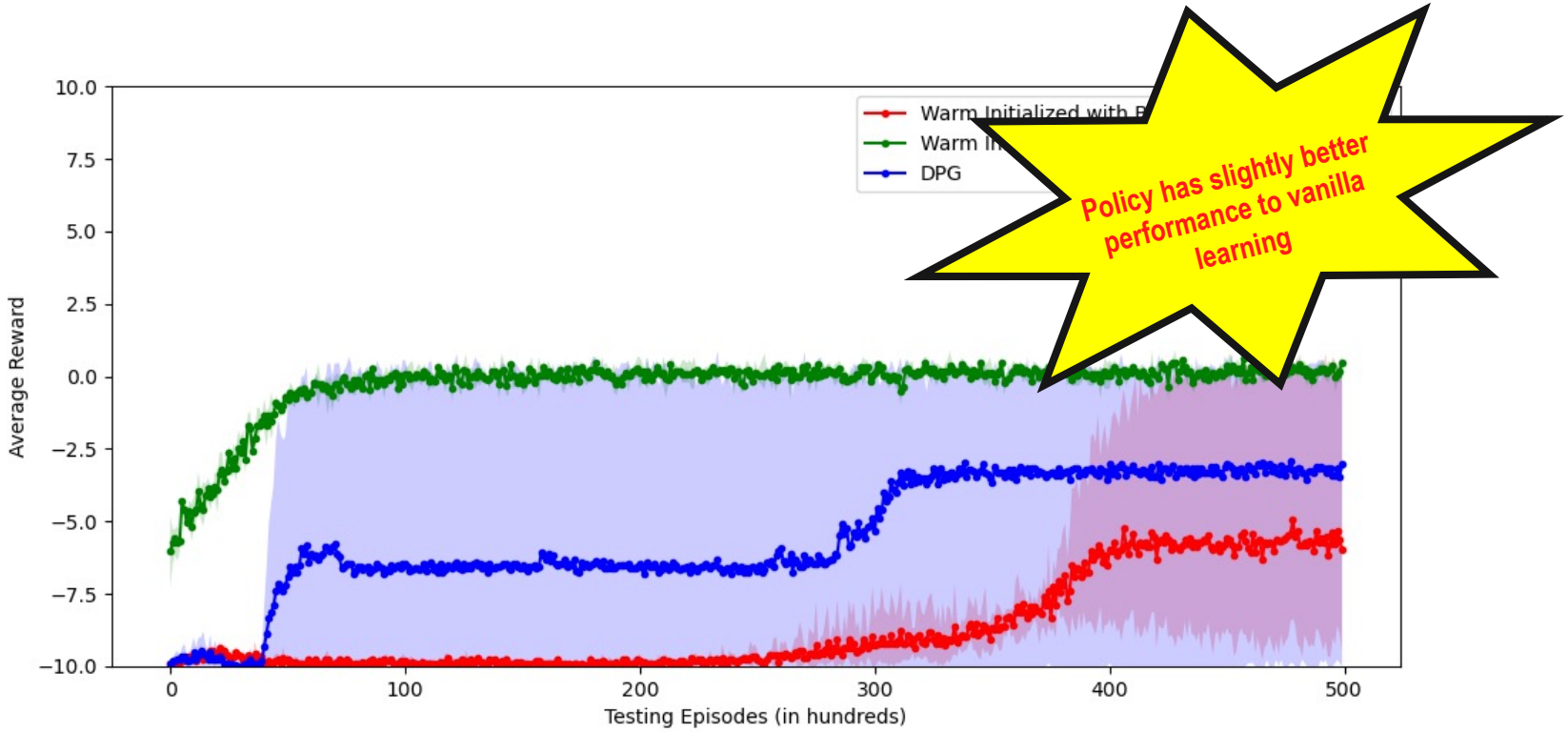
LSTM Makes 1 Error in Prediction of Learner Sequence of Subgoals



Duke UNIVERSITY

Numerical Experiments

LSTM Makes 2+ Errors in Prediction of Learner Sequence of Subgoals



Duke UNIVERSITY

Summary

- Transferring across heterogeneous agents with different action spaces is difficult because there is no explicit mapping that is known between the expert and learner agent
- Our method seeks to learn this mapping and use its predictions on unseen tasks to give the learner agent an initial biased high level policy which can better direct the agent's exploration towards an optimal policy
- Our experimental results in a discrete space show that our method is viable in transferring across agents with heterogeneous action spaces
- Future works include extending this work to a continuous case and discovering subgoals rather than predefining them

Duke
UNIVERSITY

Support

Thank You for Your Attention !

Duke
UNIVERSITY