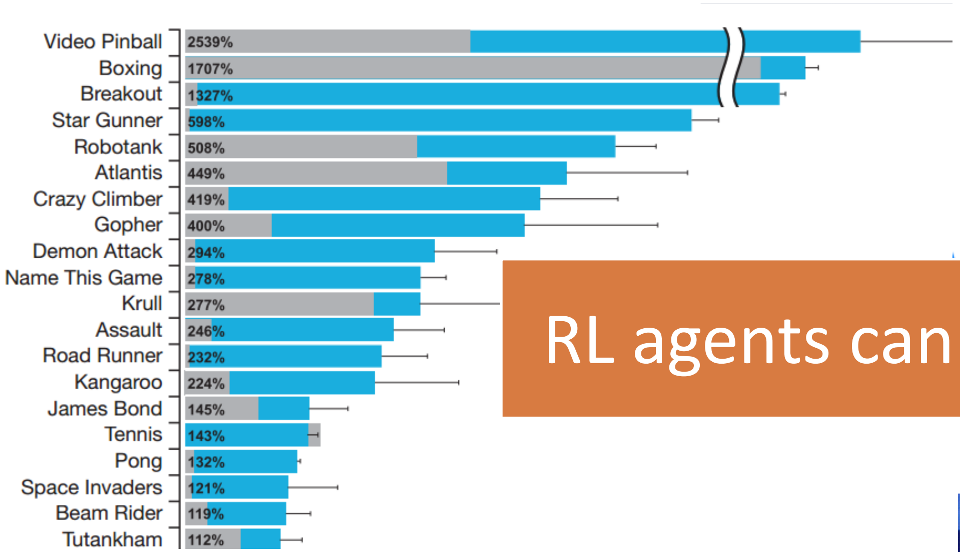
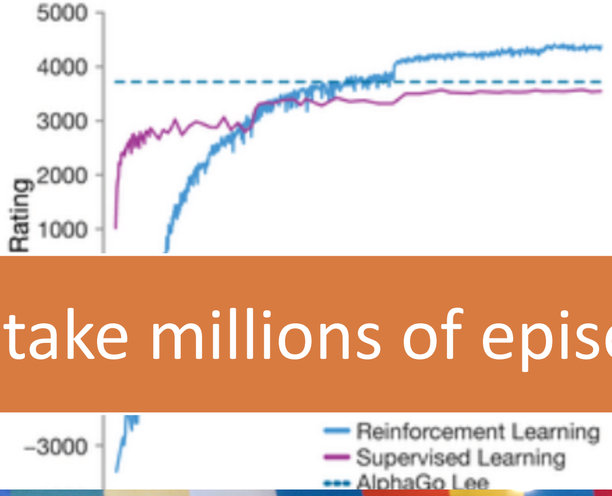
Curriculum Learning in Reinforcement Learning

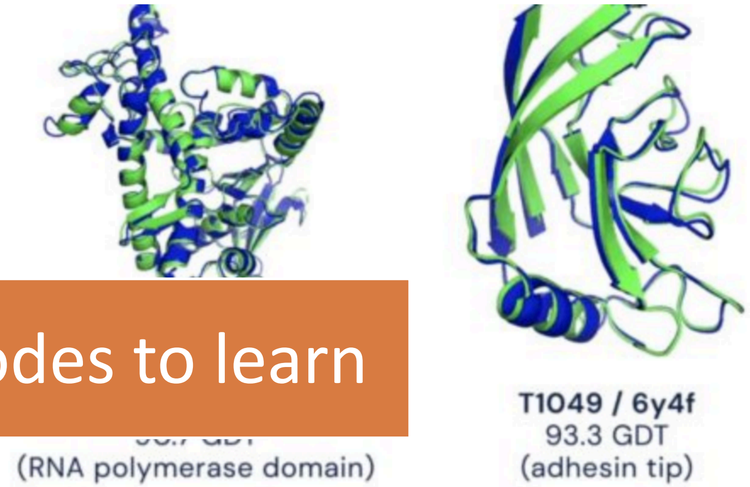
PhD Defense

Sanmit Narvekar
Department of Computer Science
University of Texas at Austin
sanmit@cs.utexas.edu

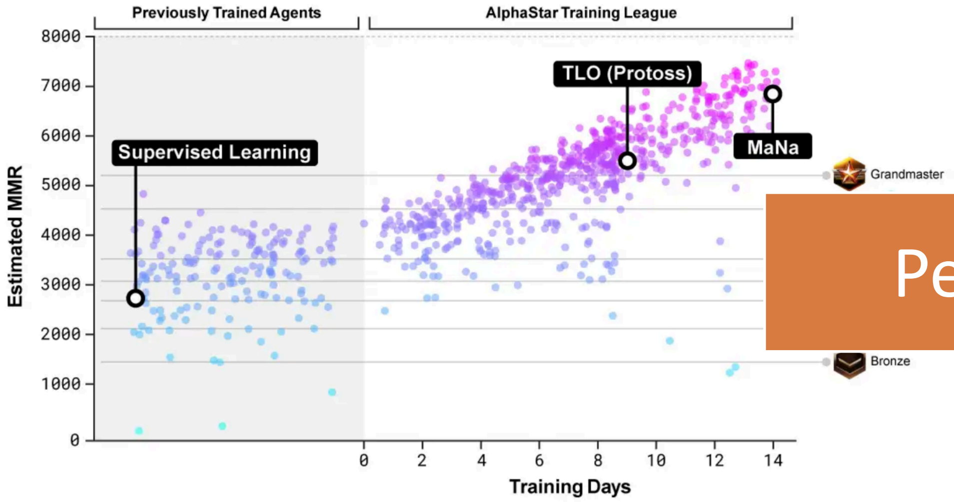
Successes of Reinforcement Learning



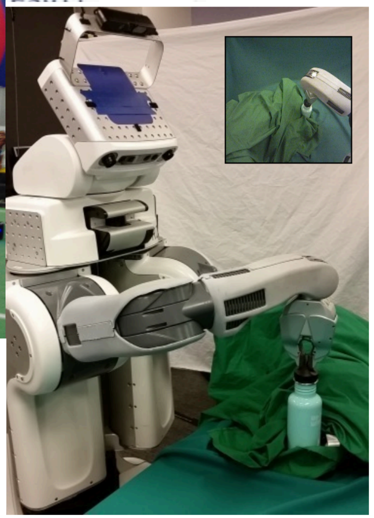




RL agents can take millions of episodes to learn

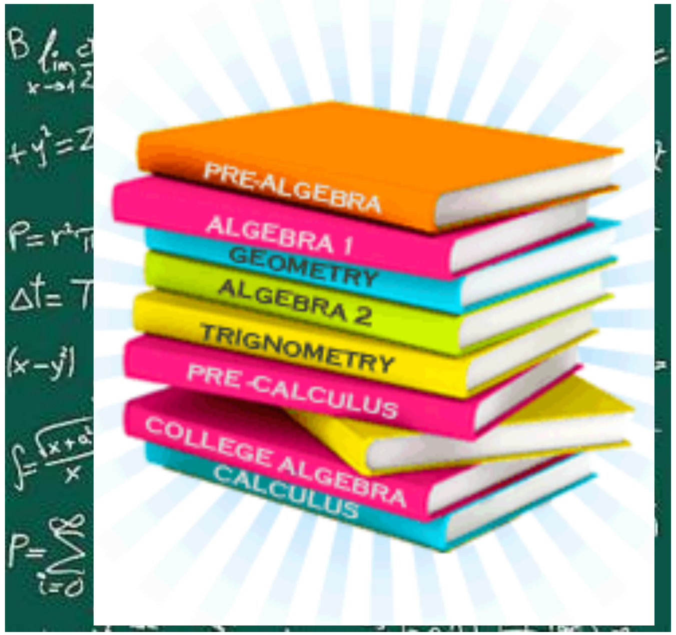


People learn much faster



By 2050?

People Learn via Curricula

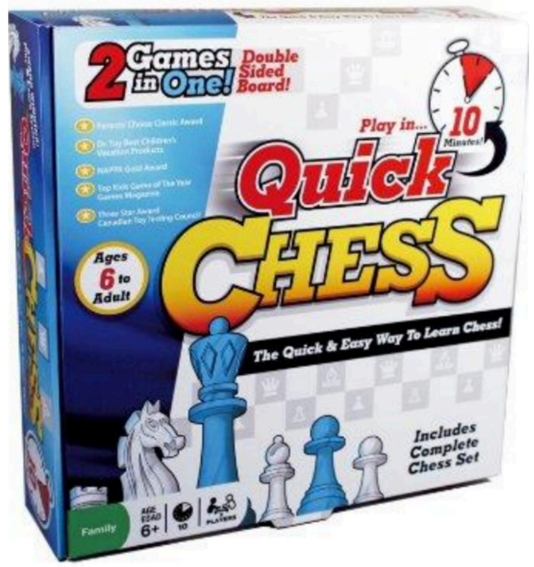


People are able to learn a lot of complex tasks very efficiently

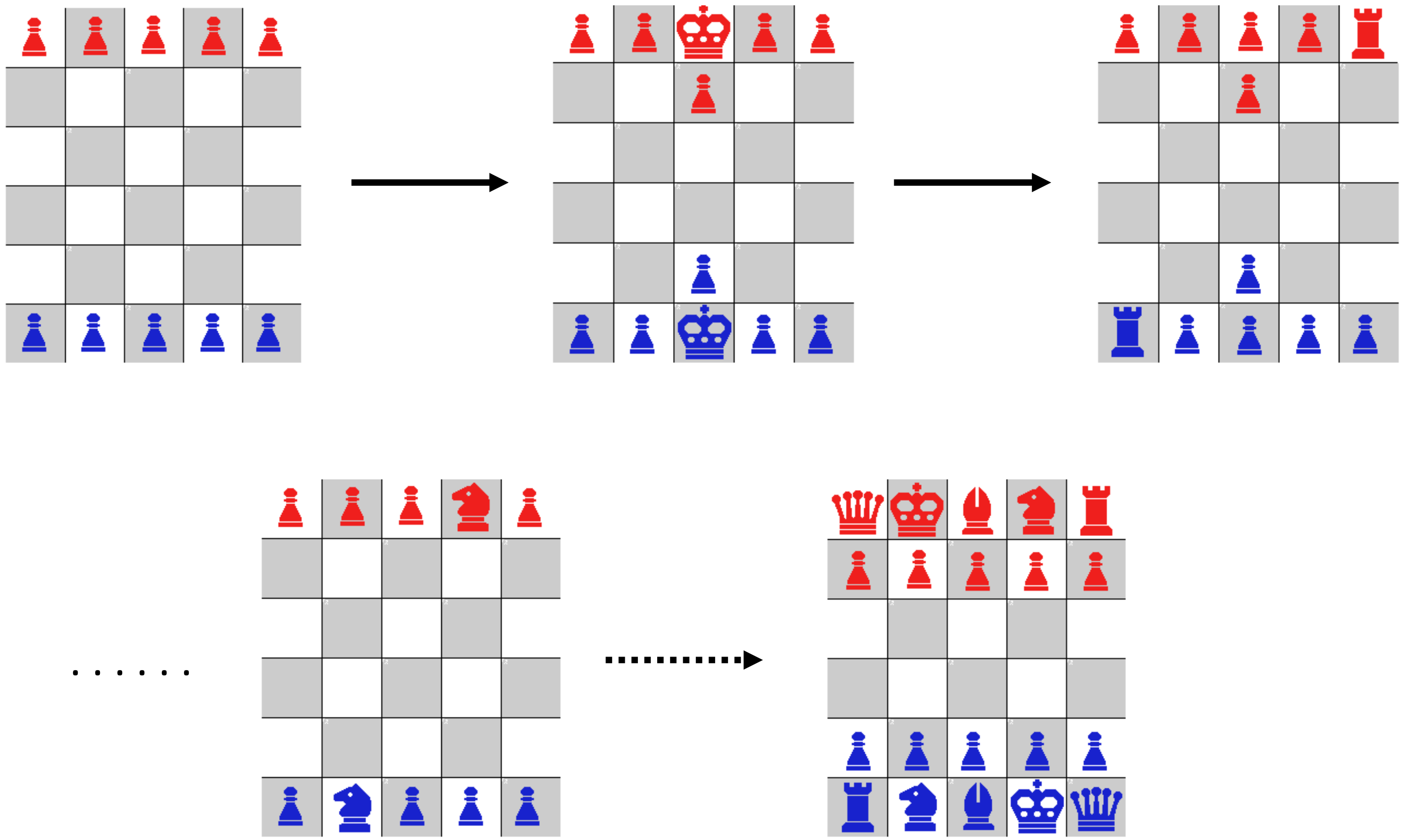
Example: Quick Chess

- Quickly learn the fundamentals of chess

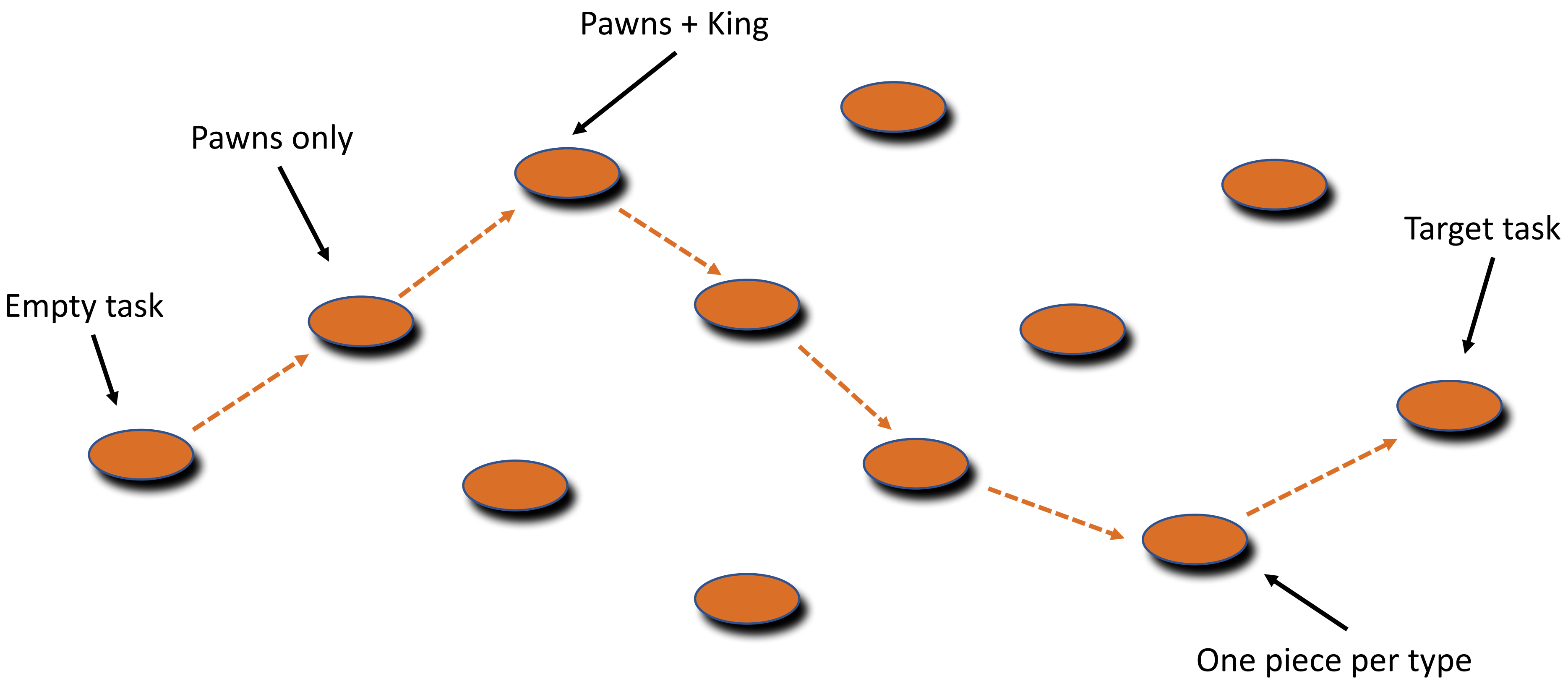
- 5 x 6 board
- Fewer pieces per type
- No castling
- No en-passant



Example: Quick Chess



Task Space



- Quick Chess is a curriculum designed for people
- We want to do something similar automatically for autonomous agents

Thesis Question(s)

- Can reinforcement learning agents benefit from learning via a curriculum?
- How can we automatically design one tailored to both the learning agent and task in question?

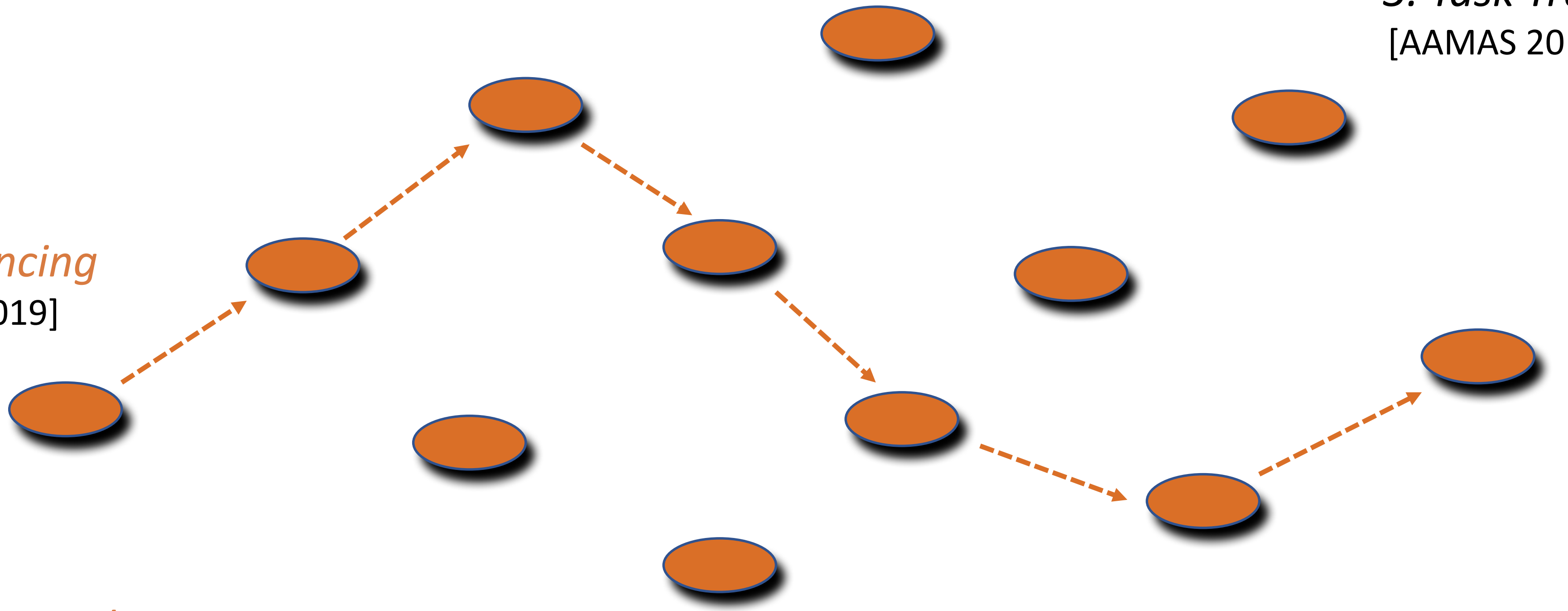
Contributions

1. Problem Formalization
[JMLR 2020] (Chapter 3)

2. Task Generation
[AAMAS 2016] (Chapter 4)

3. Task Transferability
[AAMAS 2015] (Chapter 5)

4. Automatic Sequencing
[IJCAI 2017, AAMAS 2019]
(Chapters 6 & 7)

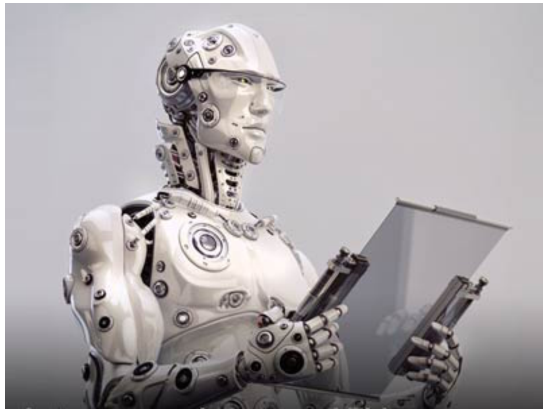


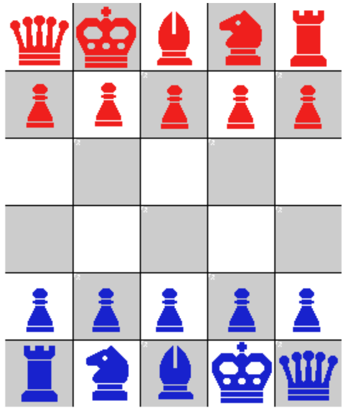
5. Curriculum Adaptation
[ICML WS 2020] (Chapter 8)

6. Taxonomy of CL
[JMLR 2020] (Chapter 9)

7. Empirical Evaluation

Background





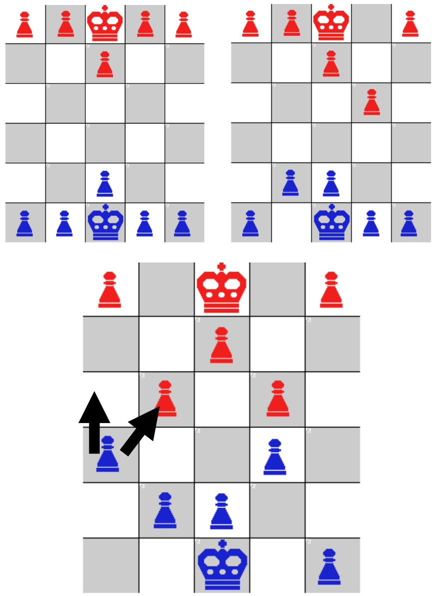
- Reinforcement Learning
- Transfer Learning

Markov Decision Processes (MDPs)

- Model agent's interaction with a task as an episodic MDP

$$M = (\mathcal{S}, \mathcal{A}, p, r, \Delta s_0, \mathcal{S}_f)$$

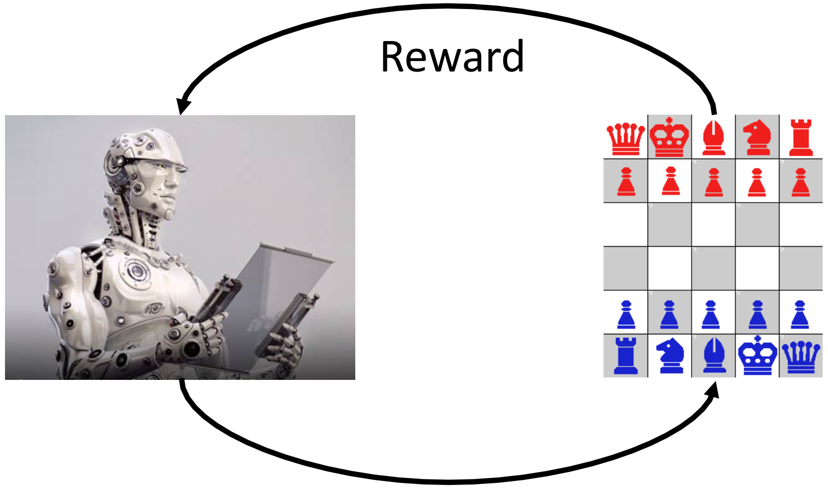
- S: set of states
- A: set of actions
- p: transition function
- r: reward function



State

Reward

Agent



Environment

Action

Markov Decision Processes (MDPs)

- Goal is to learn an optimal policy π*: S → A that maximizes sum of rewards
- Learn the optimal action-value function

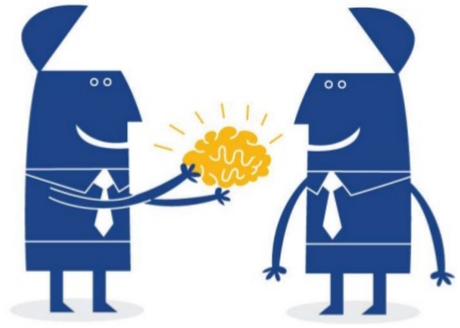
$$q_*(s,a) = r(s,a) + \sum_{s'} p(s'|s,a) \max_{a'} q_*(s',a')$$

- Gives the expected return of taking action a in state s, and following π* after
- Can be learned using methods such as SARSA

$$q(s,a) \leftarrow q(s,a) + \underset{\text{learning rate}}{\alpha}\underbrace{[r(s,a) + q(s',a') - q(s,a)]}_{\text{TD error}}$$

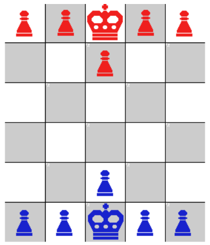
- Act greedily with respect to Q

Transfer Learning

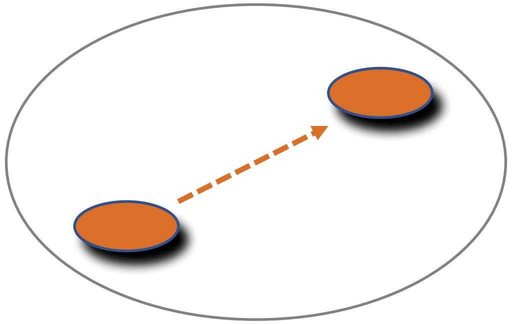


- Key Idea:

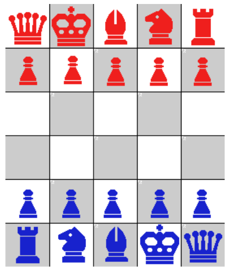
  Instead of learning tabula rasa on target task, transfer knowledge from a related source task



Source task



Target task

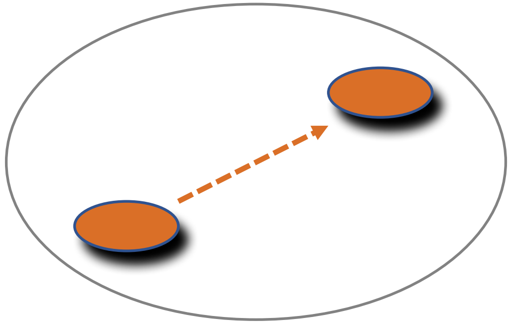


- ***Given*** a good source and target task, ***how*** to transfer knowledge
- Many ways to do this

Value Function Transfer

- Initialize Q function in target task using values learned in a source task

$q_{source}(s,a)$



- Assumptions:
  - Tasks have overlapping state and action spaces
  - OR an inter-task mapping is provided
    - Existing related work on learning mappings

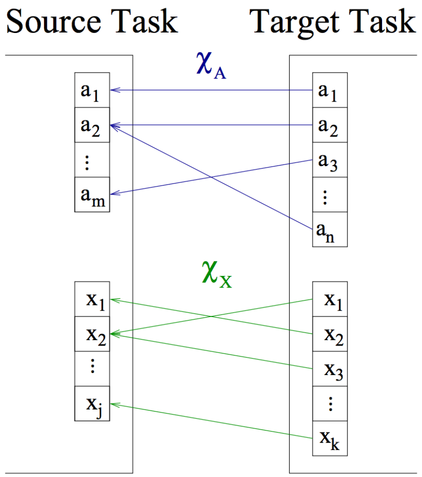


Image credit: Taylor and Stone, JMLR 2009

Reward Shaping Transfer

- Reward function in target task augmented with a shaping reward *f*:

$$\underbrace{r'(s,a,s')}_{\text{New Reward}} = \underbrace{r(s,a,s')}_{\text{Old Reward}} + \underbrace{f(s,a,s')}_{\text{Shaping Reward}}$$

- Potential-based advice restricts f to be difference of potential functions:

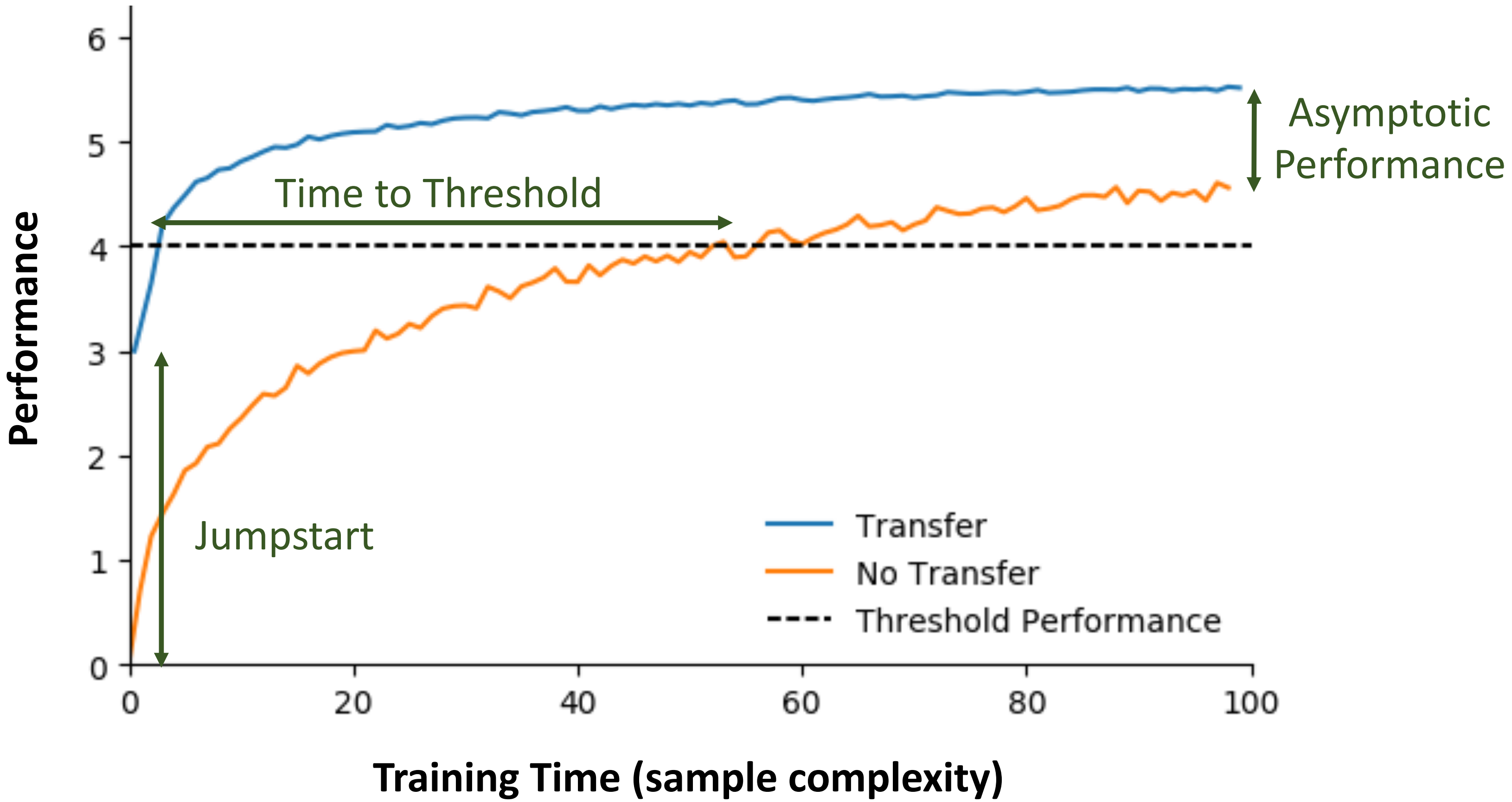
$$f(s,a,s') = \Phi(s',\pi(s')) - \Phi(s,a)$$

- Use the value function of the source as the potential function:

$$\Phi(s,a) = Q_{source}(s,a)$$

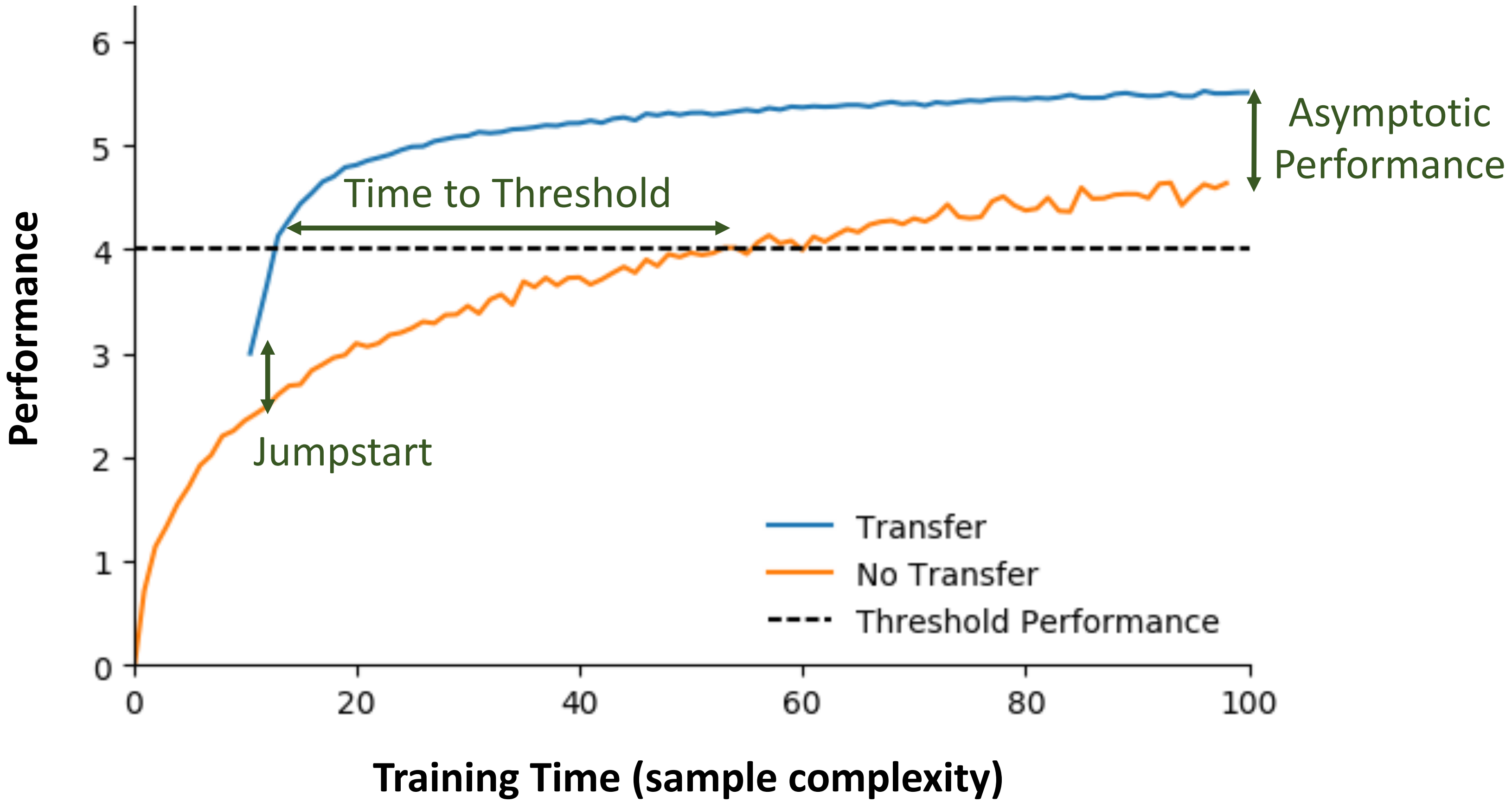
Quantifying Utility of Transfer

- Strong vs weak transfer



Quantifying Utility of Transfer

- Strong vs weak transfer



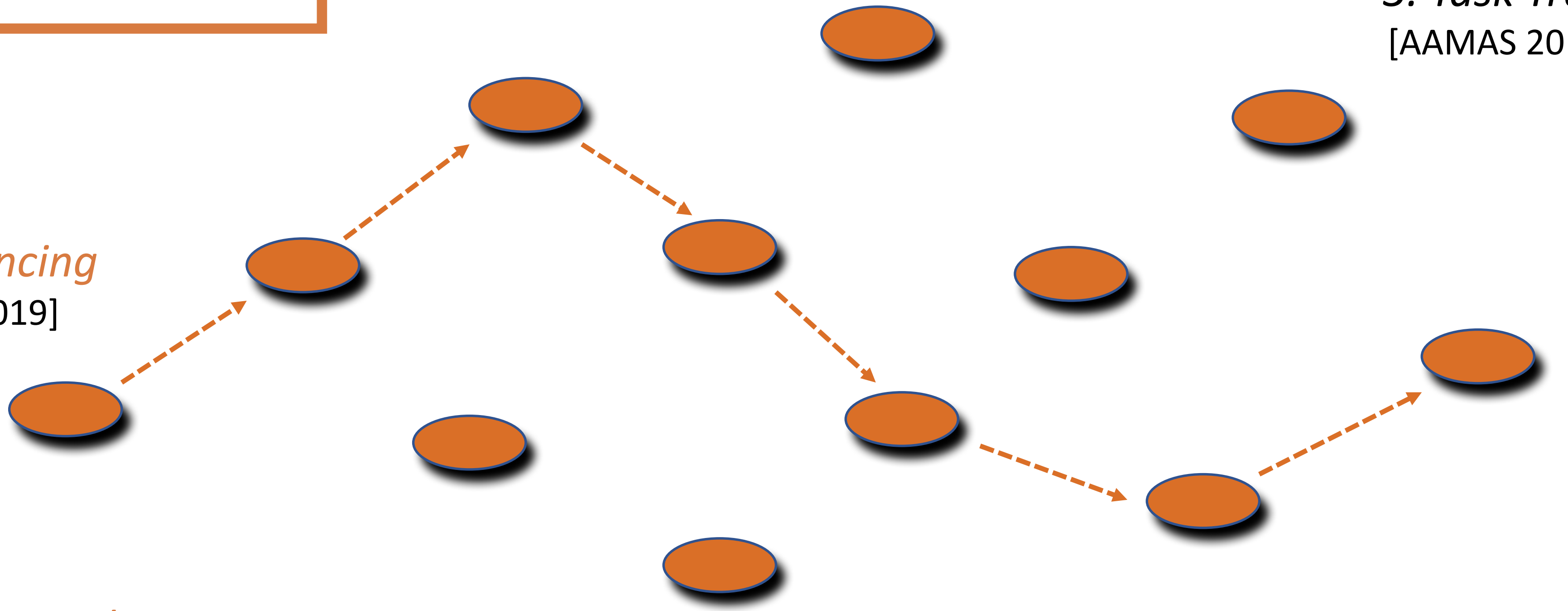
Contributions

1. Problem Formalization
[JMLR 2020] (Chapter 3)

2. Task Generation
[AAMAS 2016] (Chapter 4)

3. Task Transferability
[AAMAS 2015] (Chapter 5)

4. Automatic Sequencing
[IJCAI 2017, AAMAS 2019]
(Chapters 6 & 7)

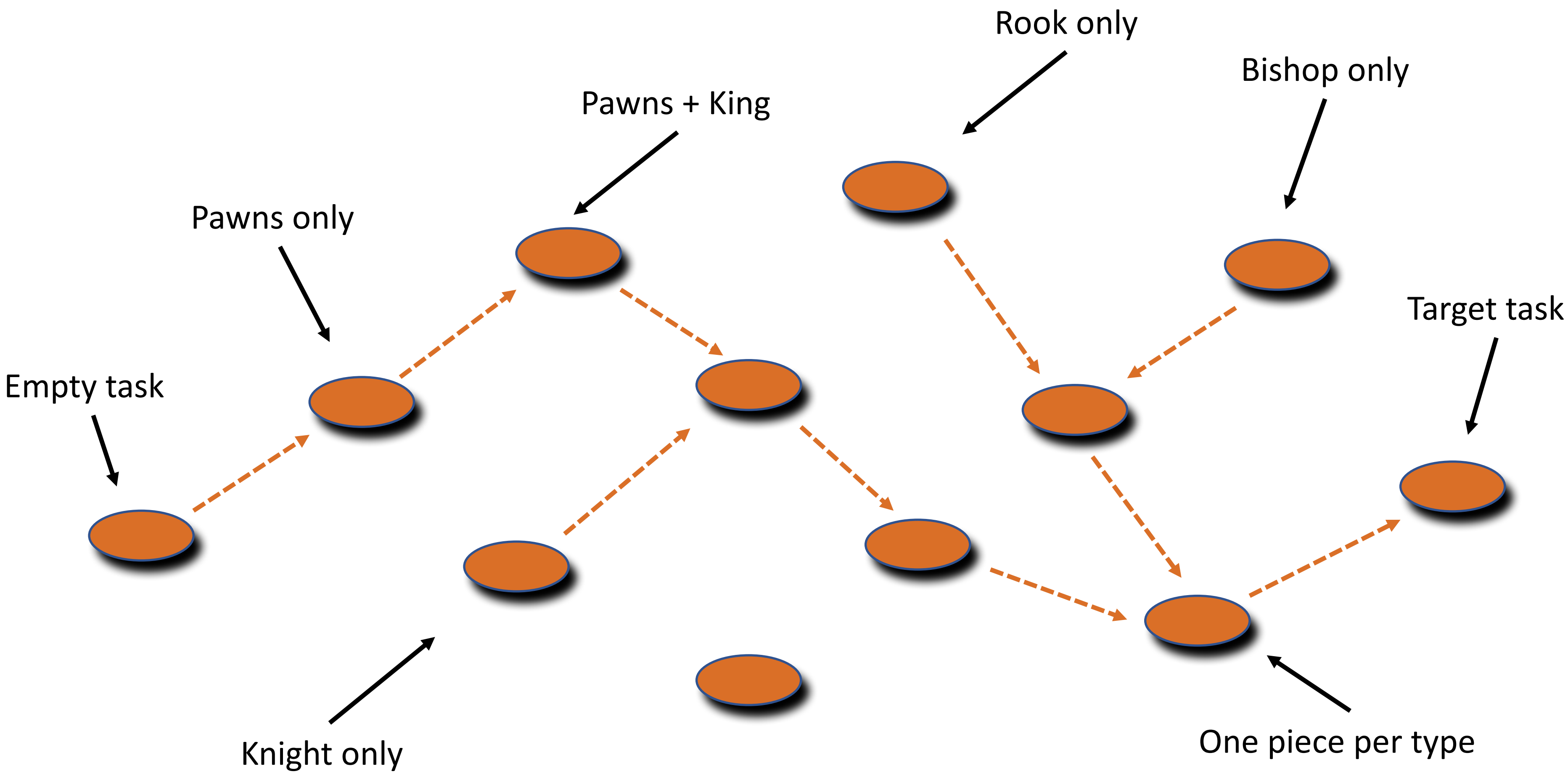


5. Curriculum Adaptation
[ICML WS 2020] (Chapter 8)

6. Taxonomy of CL
[JMLR 2020] (Chapter 9)

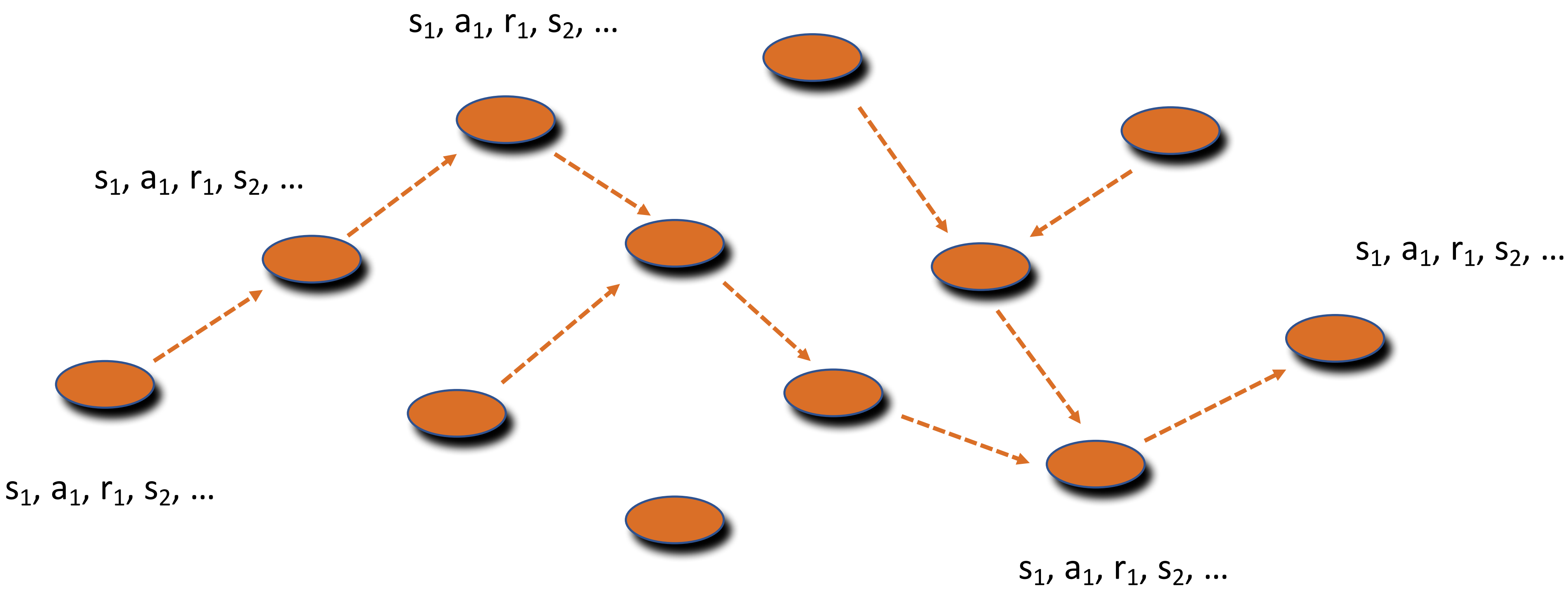
7. Empirical Evaluation

What is a Curriculum?



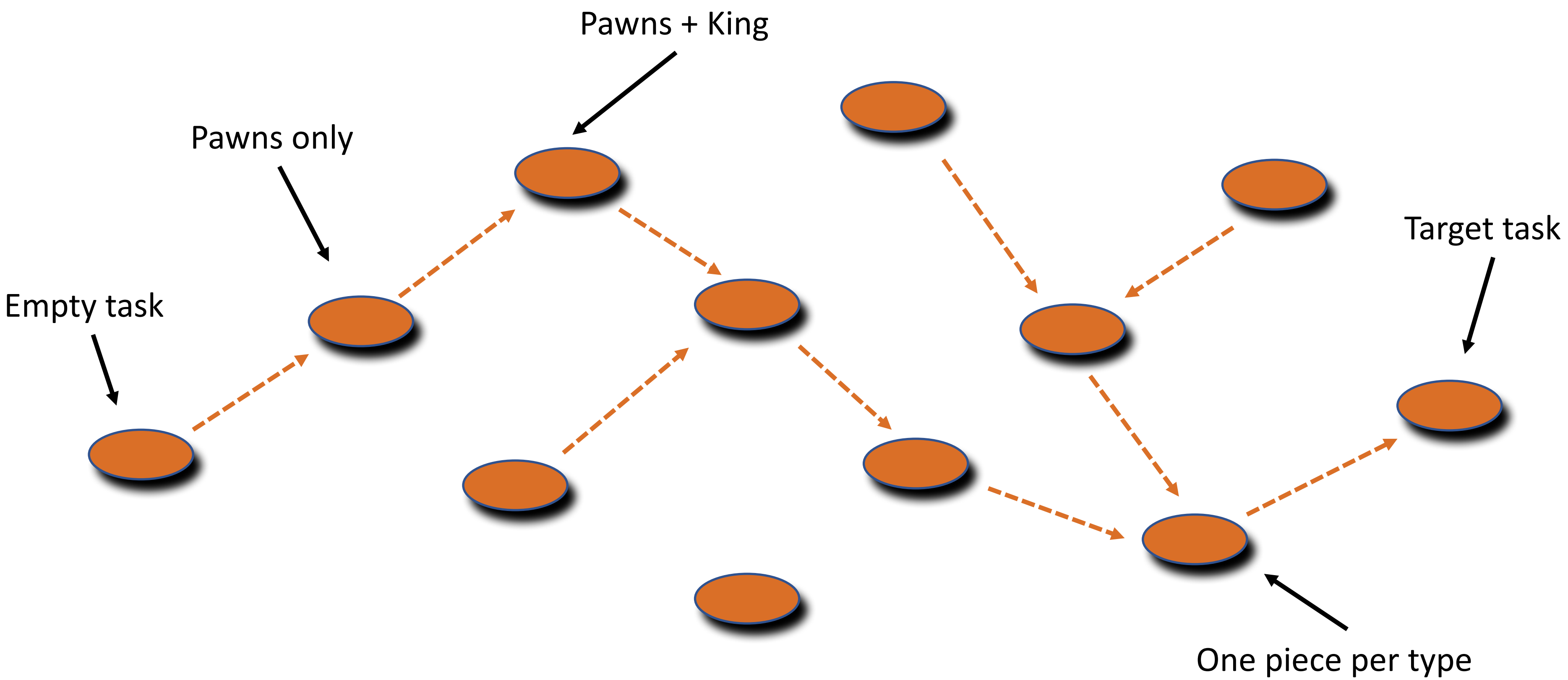
- RL agents don't need to train sequentially
- Learn skills simultaneously, then combine

What is a Curriculum?



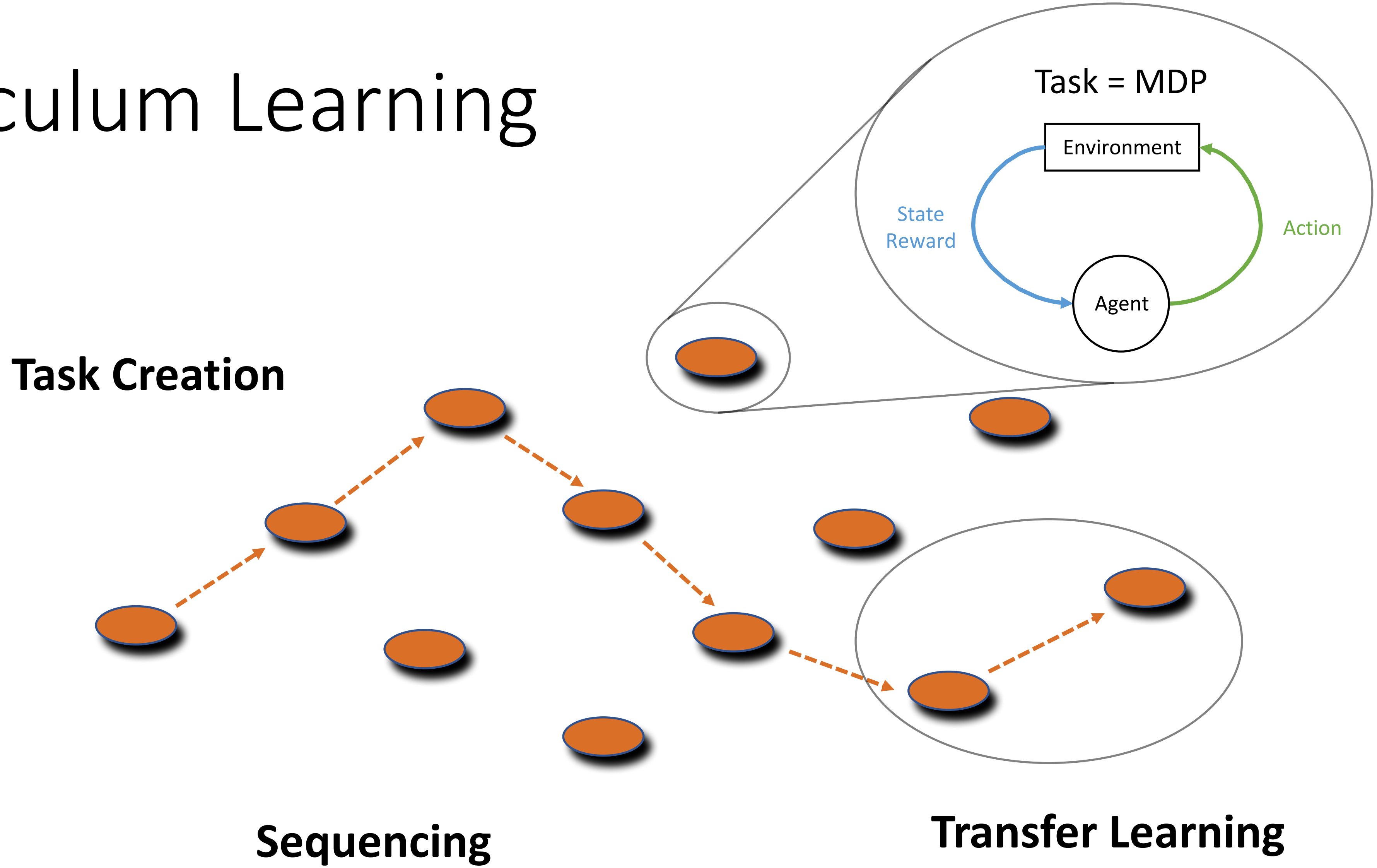
- More abstractly, each node is associated with a set of samples derived from the set of tasks
- These samples at nodes may be associated with exactly one task, but this is not necessary

What is a Curriculum?



- A curriculum is a directed acyclic graph over sets of samples
- This definition encompasses all known CL work
- This thesis will use the most common sequence of tasks representation

Curriculum Learning



- Curriculum learning is a methodology that ties task creation, sequencing, and transfer learning
- Focus on task creation and sequencing, leveraging existing work on transfer learning

Taxonomy of CL Methods + Related Work

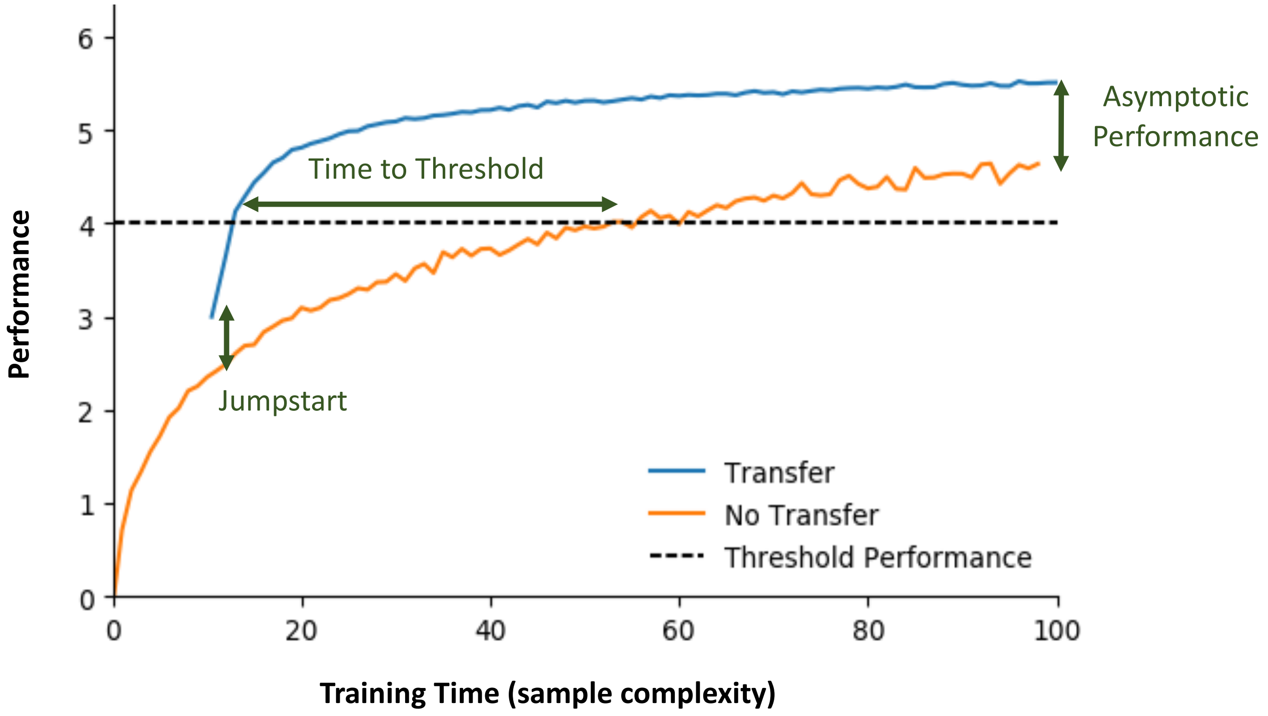
- Primary assumptions of curriculum learning:
  - Environment can be configured to create subtasks
  - Agent discovers on its own reusable pieces of knowledge
- Organized methods by the degree to which source tasks can differ

Sequencing Methods of this Thesis

Sample Sequencing	Co-learning	Reward and Initial/Terminal State	No Restrictions
PER (Schaul et al. 2016)	Asymetric Self-Play (Sukhbaatar et al. 2018)	SAGG-RIAC (Baranes and Oudeyer 2013)	TSCL (POMDPs) (Matiisen et al. 2017)
HER (Andrychowicz et al. 2017)	AlphaStar (Vinyals et al. 2019)	RCG (Florensa et al. 2017)	Curriculum Graphs (Svetlik et al. 2017)
CHER (Fang et al. 2019)	Emergent Curricula (Baker et al. 2020)	SAC-X (Riedmiller et al. 2018)	Combinatorial Search (Foglino et al. 2019)

Quantifying Utility of a Curriculum

- Offset for time spent in source tasks in the curriculum
- Time spent creating curriculum?
  - Most work does not, allows comparison of the quality of the curriculum itself
  - Can compare with human generated curricula



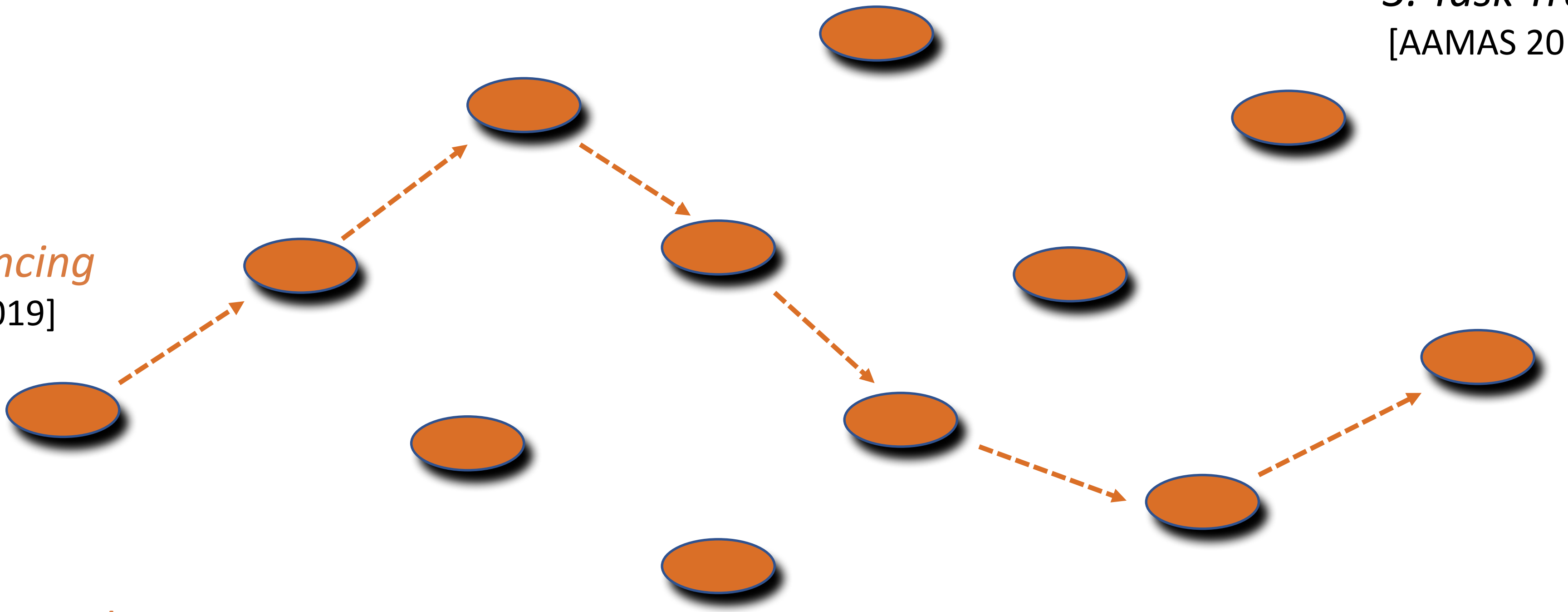
Contributions

2. Task Generation
[AAMAS 2016] (Chapter 4)

1. Problem Formalization
[JMLR 2020] (Chapter 3)

3. Task Transferability
[AAMAS 2015] (Chapter 5)

4. Automatic Sequencing
[IJCAI 2017, AAMAS 2019]
(Chapters 6 & 7)



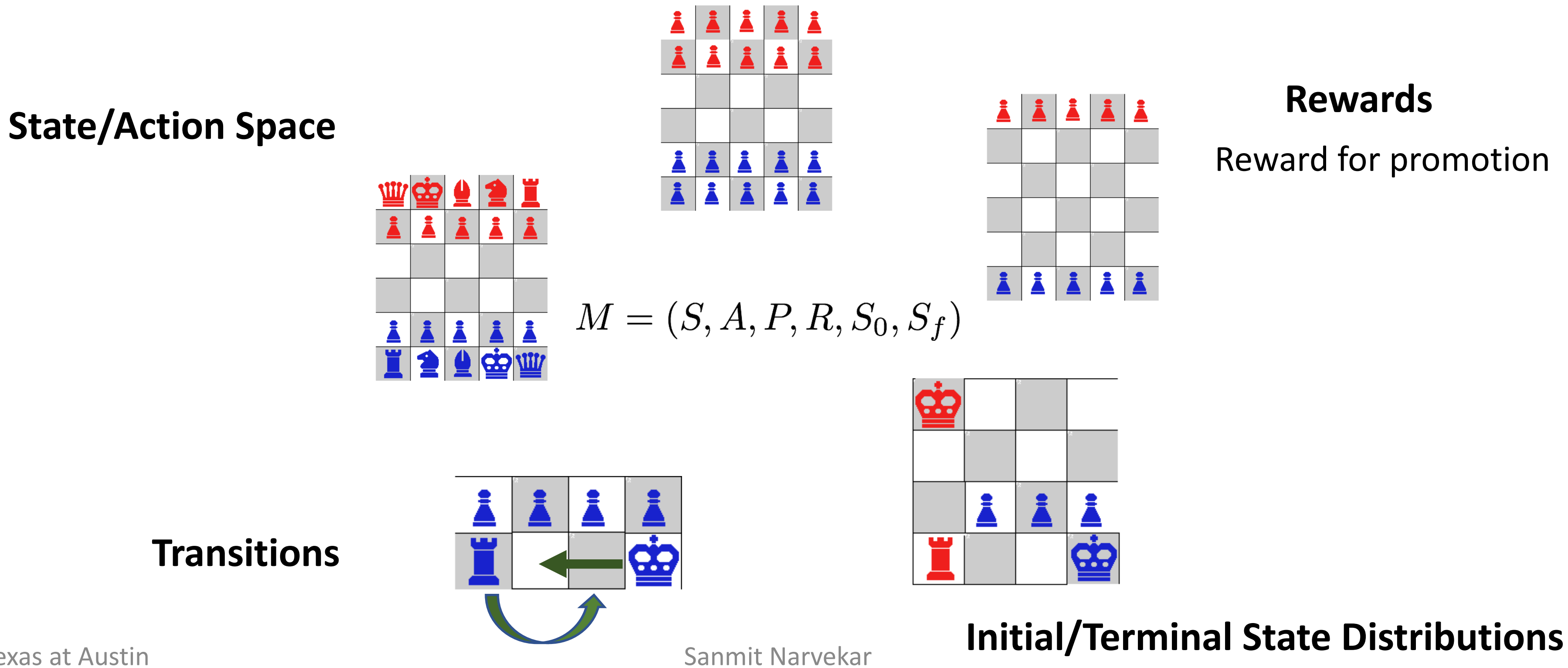
5. Curriculum Adaptation
[ICML WS 2020] (Chapter 8)

6. Taxonomy of CL
[JMLR 2020] (Chapter 9)

7. Empirical Evaluation

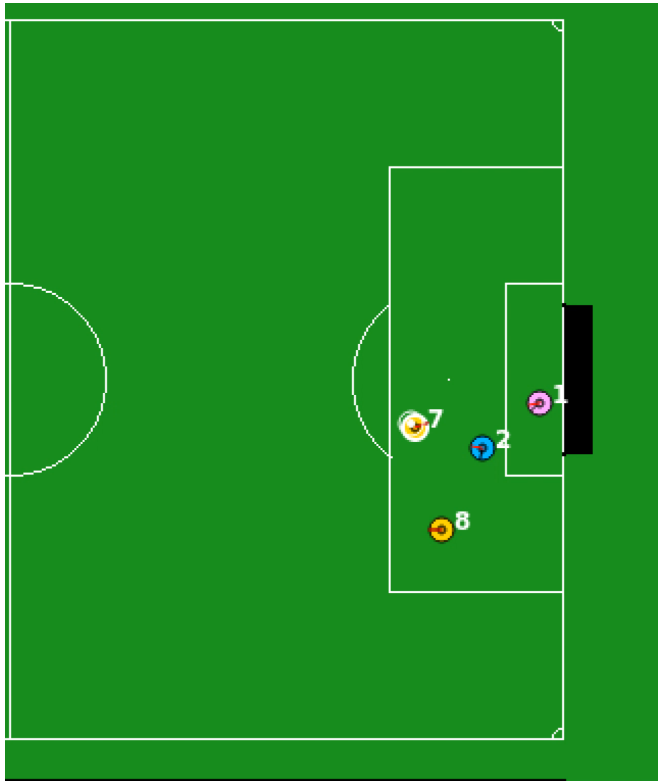
Task Generation

- Proposed a set of 7 heuristic functions $f : M_t \times X \mapsto M_s$
- Use parameterized model of the domain and observations of the agent performing the target task to create source tasks

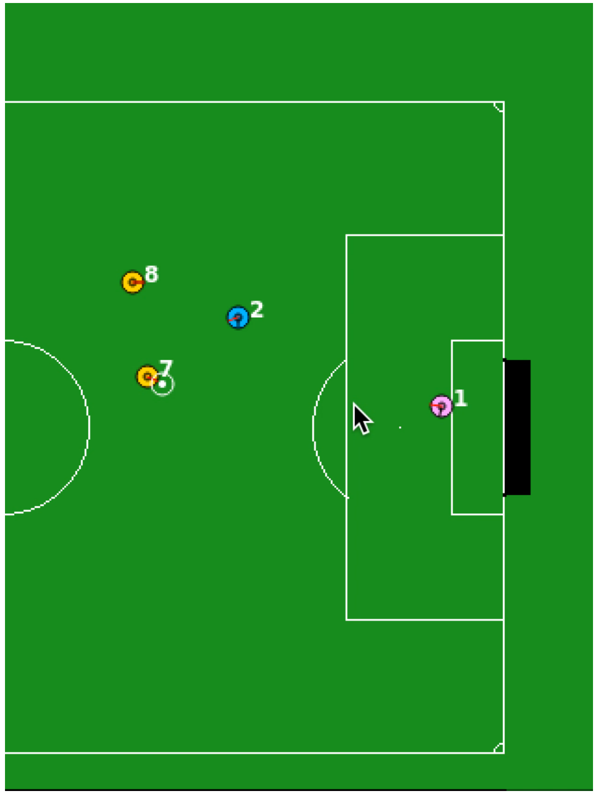


Generated Tasks in 2D Simulated Soccer

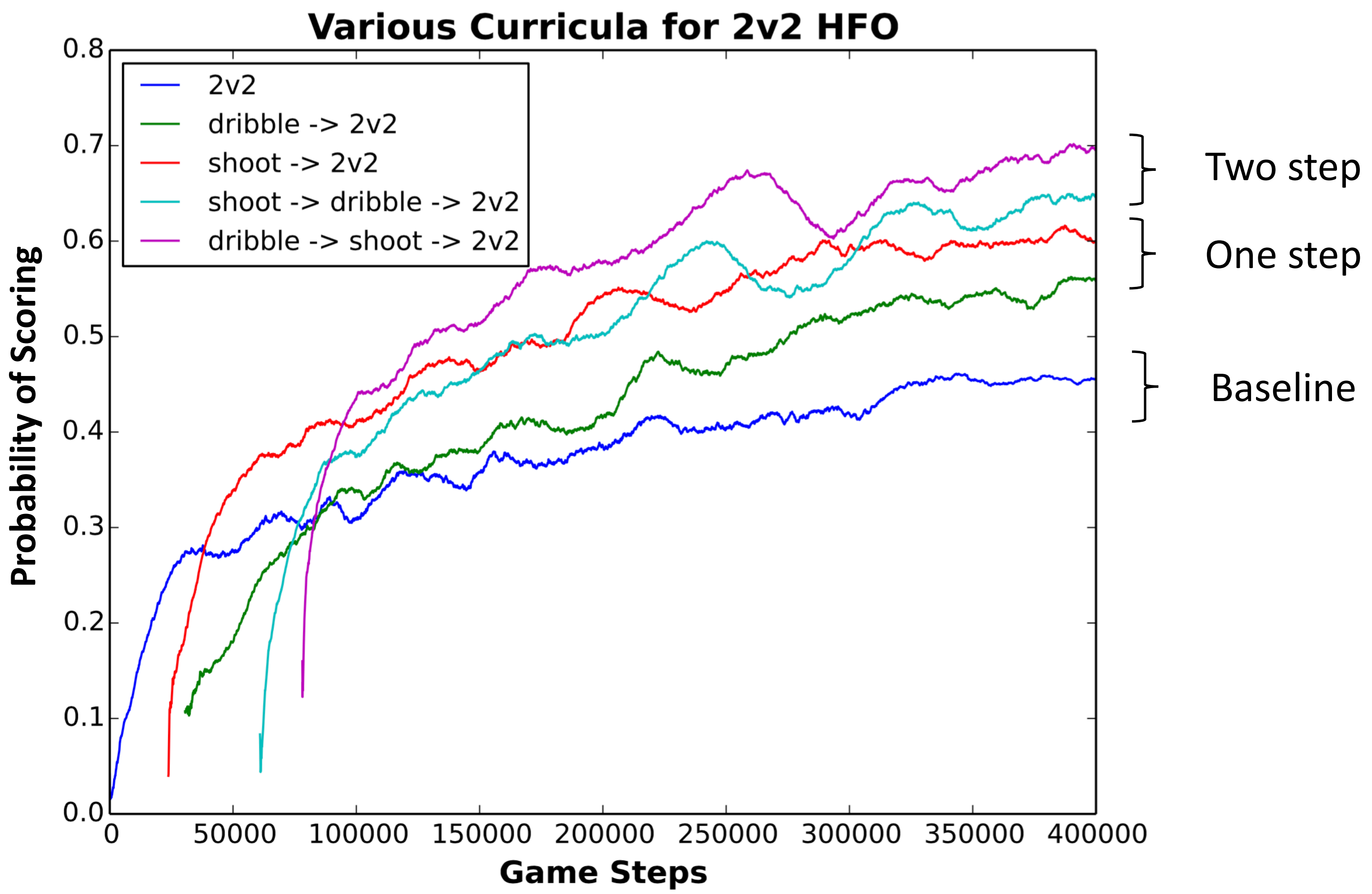
Shoot Task



Dribble Task

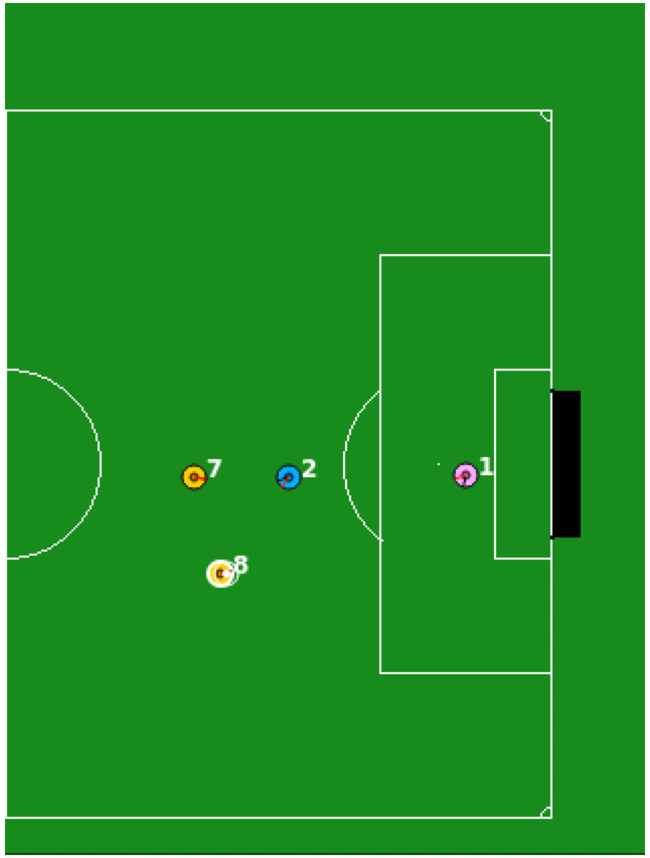


2v2 HFO Results

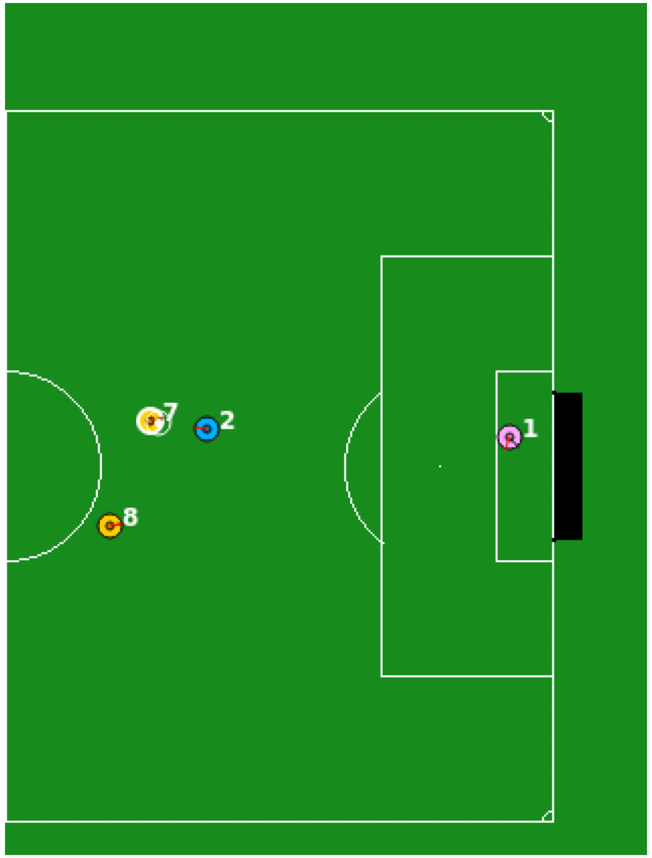


2v2 HFO Sample Policies

Baseline



2 step curricula

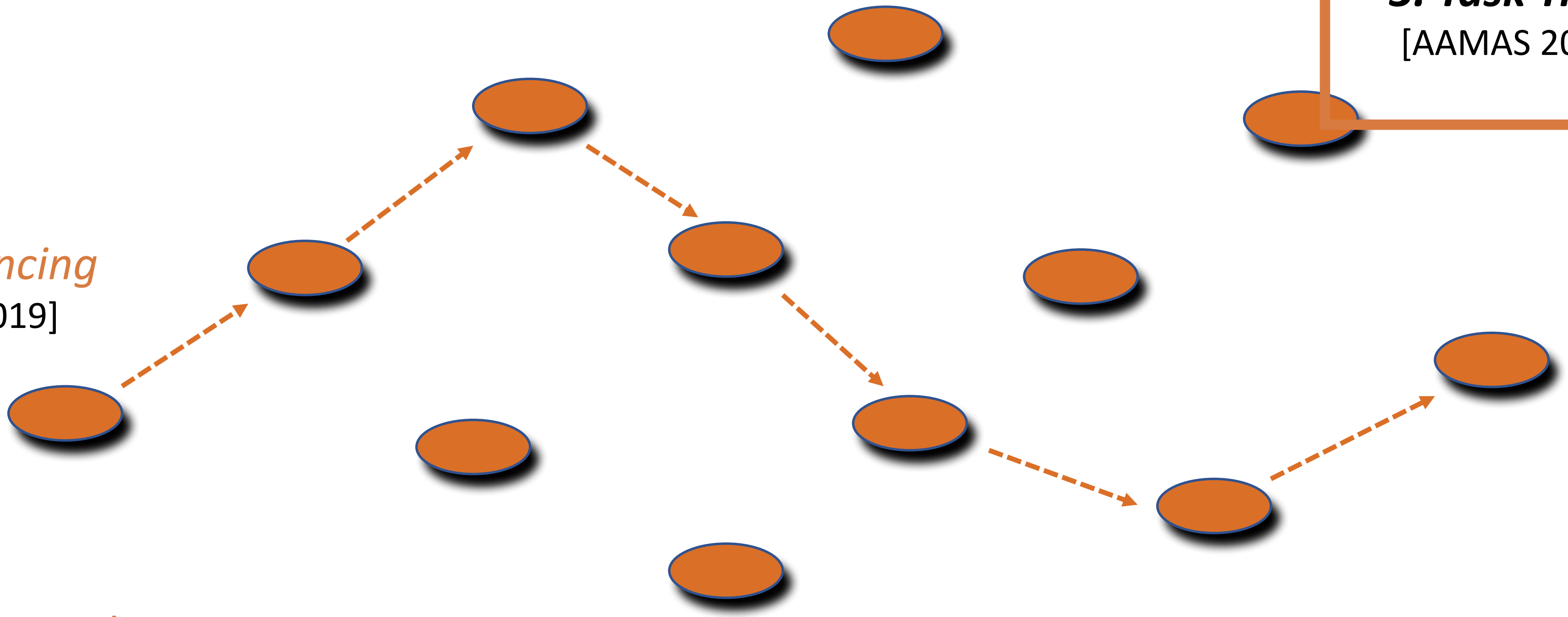


Contributions

1. Problem Formalization
[JMLR 2020] (Chapter 3)

2. Task Generation
[AAMAS 2016] (Chapter 4)

3. Task Transferability
[AAMAS 2015] (Chapter 5)



4. Automatic Sequencing
[IJCAI 2017, AAMAS 2019]
(Chapters 6 & 7)

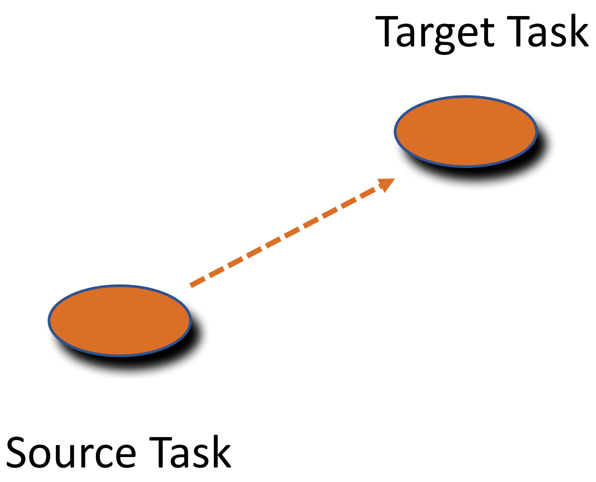
5. Curriculum Adaptation
[ICML WS 2020] (Chapter 8)

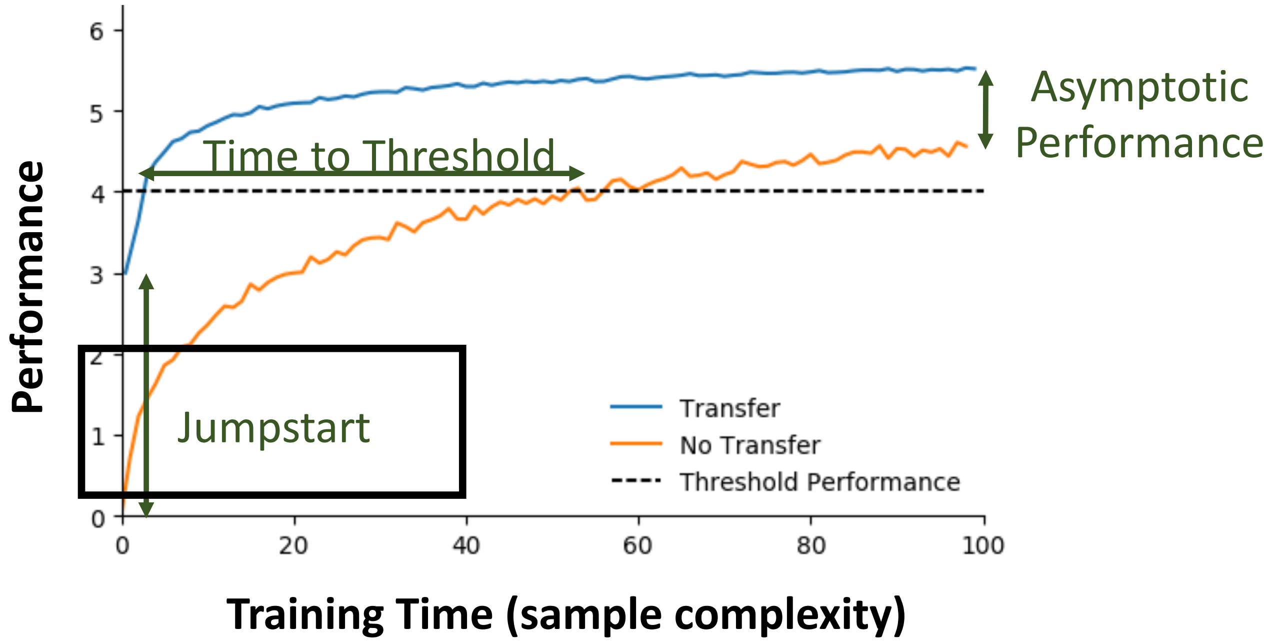
6. Taxonomy of CL
[JMLR 2020] (Chapter 9)

7. Empirical Evaluation

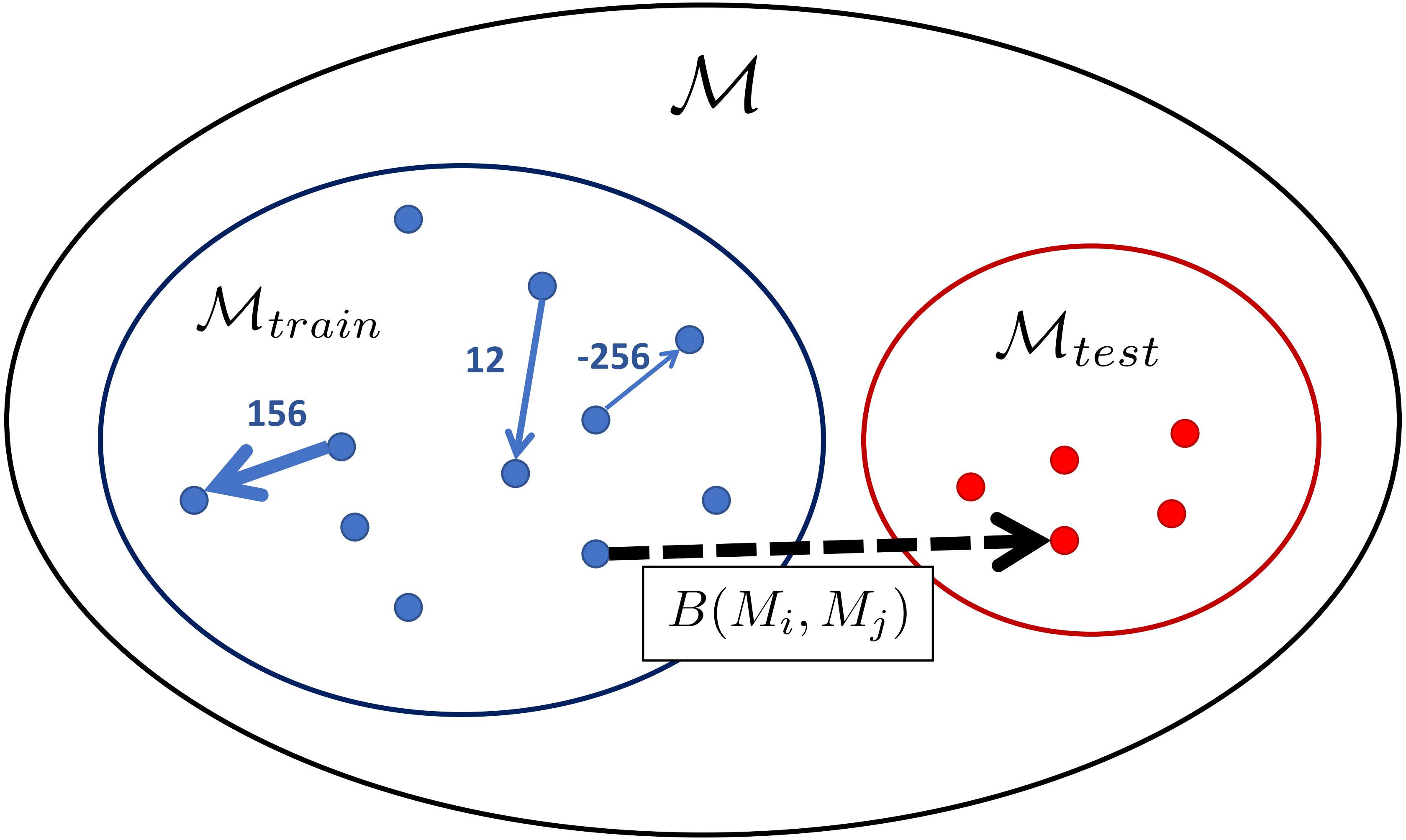
Task Transferability

- Given a source task and target task, estimate the expected benefit of transfer
- Represent tasks by a feature descriptor $\boldsymbol{f}_i \in \mathbb{R}^n$ and train regression model
- Can be used for source task selection

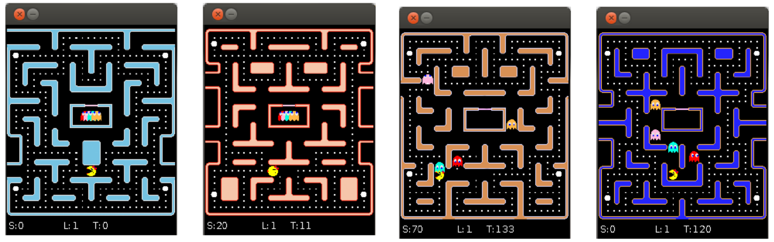




Modeling Task Transferability



Source Task Selection Loss

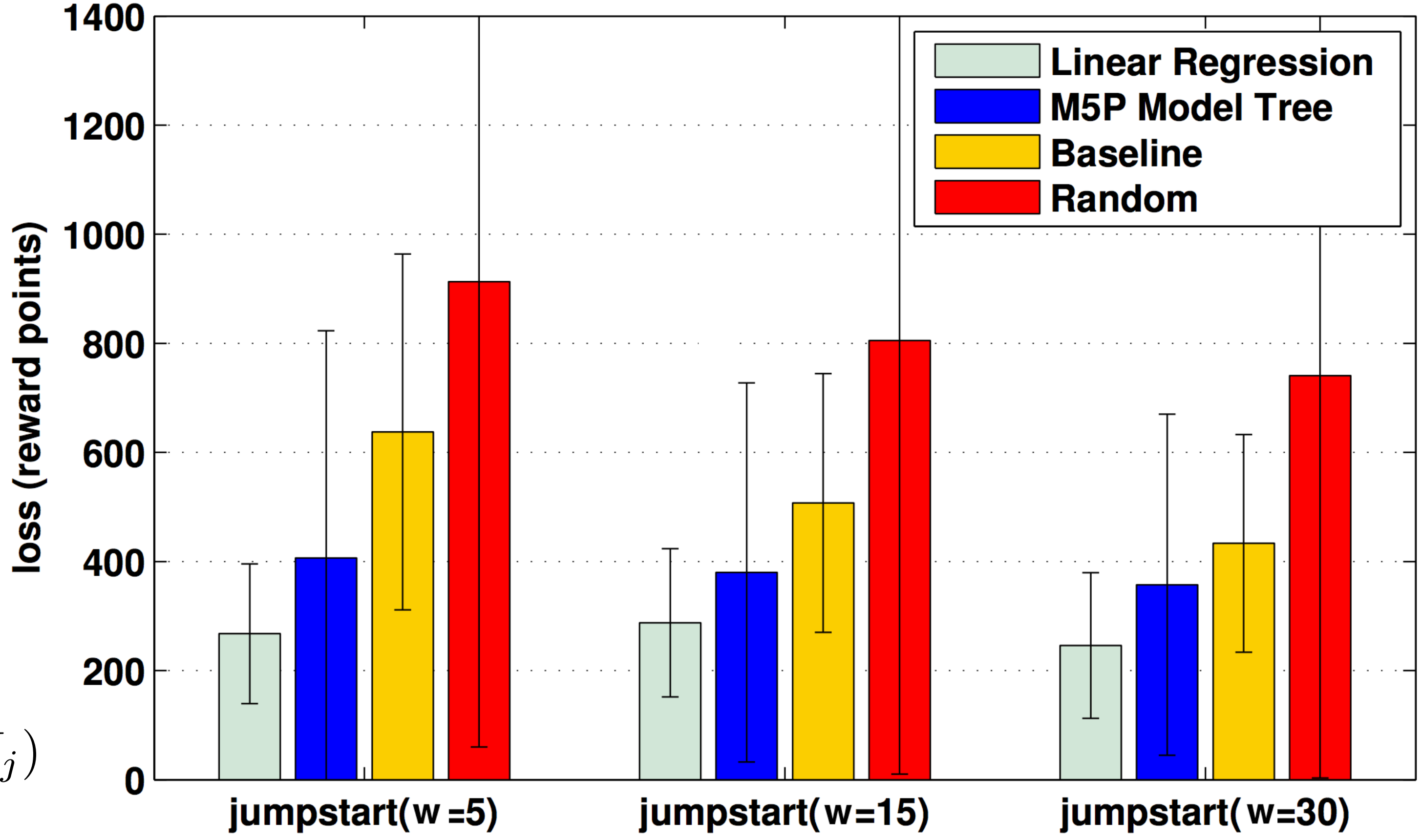


- Trained 2 types of regression models

- Baseline: choose task with closest feature vector by squared distance

- Loss:

$$loss(M_i) = B(M^*, M_j) - B(M_i, M_j)$$



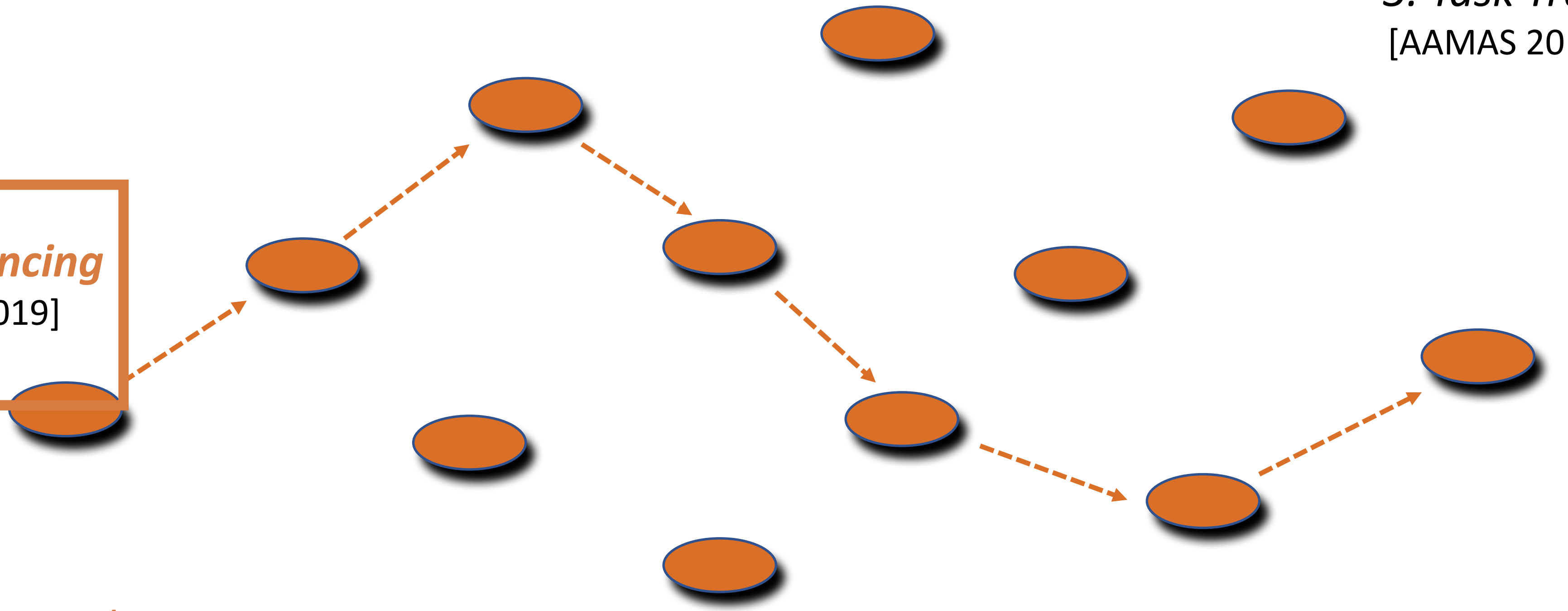
Contributions

1. Problem Formalization
[JMLR 2020] (Chapter 3)

2. Task Generation
[AAMAS 2016] (Chapter 4)

3. Task Transferability
[AAMAS 2015] (Chapter 5)

4. Automatic Sequencing
[**IJCAI 2017**, AAMAS 2019]
(**Chapters 6** & 7)



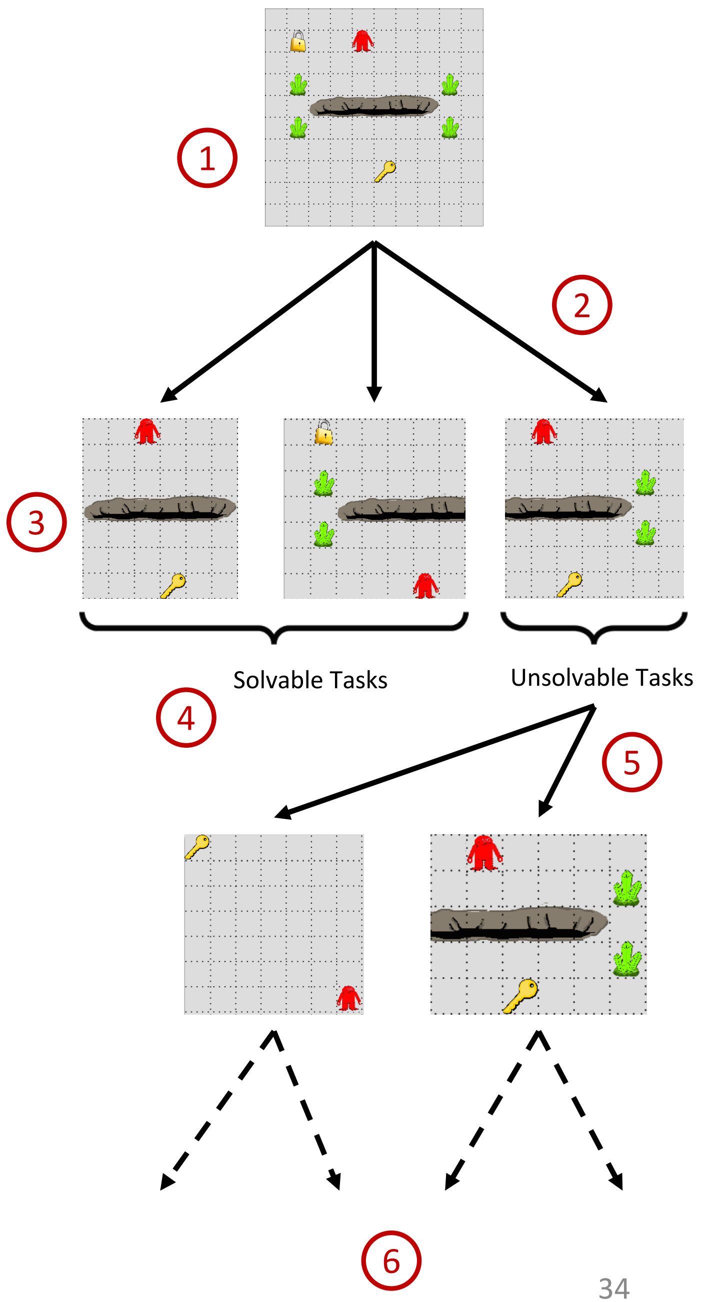
5. Curriculum Adaptation
[ICML WS 2020] (Chapter 8)

6. Taxonomy of CL
[JMLR 2020] (Chapter 9)

7. Empirical Evaluation

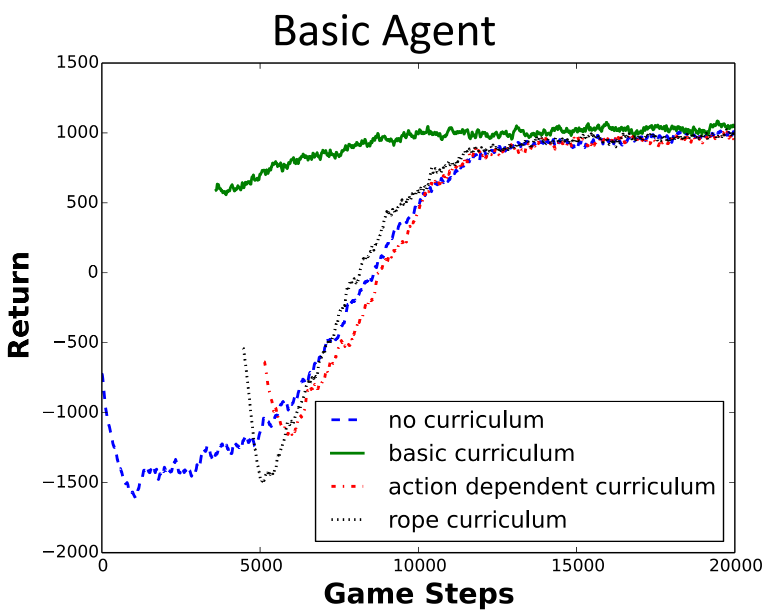
Automatic Heuristic Sequencing

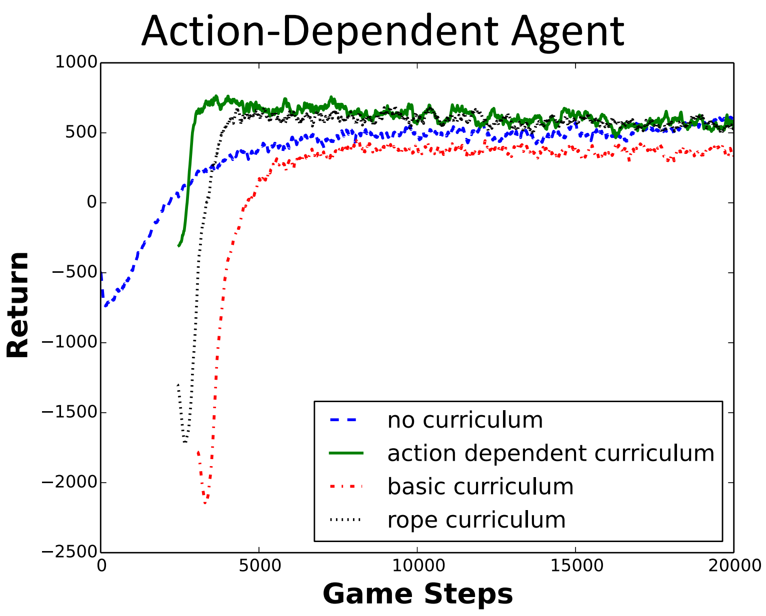
- Recursive algorithm
- Collect experience samples in target task
- Create source tasks and attempt to solve
- Heuristic: select task that updates the policy the most on collected samples. Assumes no negative transfer
- Learning a task updates the agent's policy, leading to new samples in target task
- Terminates when performance on target task greater than desired performance threshold

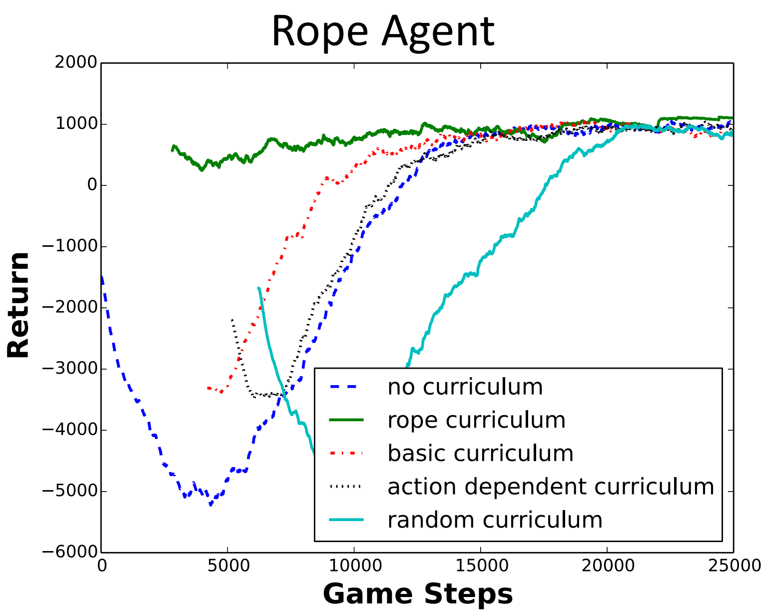


Experimental Results

- Created curricula for 3 different agents with different sensing/action abilities
- Curriculum tailored for agent in green
- In all cases, tailored curriculum is better than no curriculum and other agent curricula







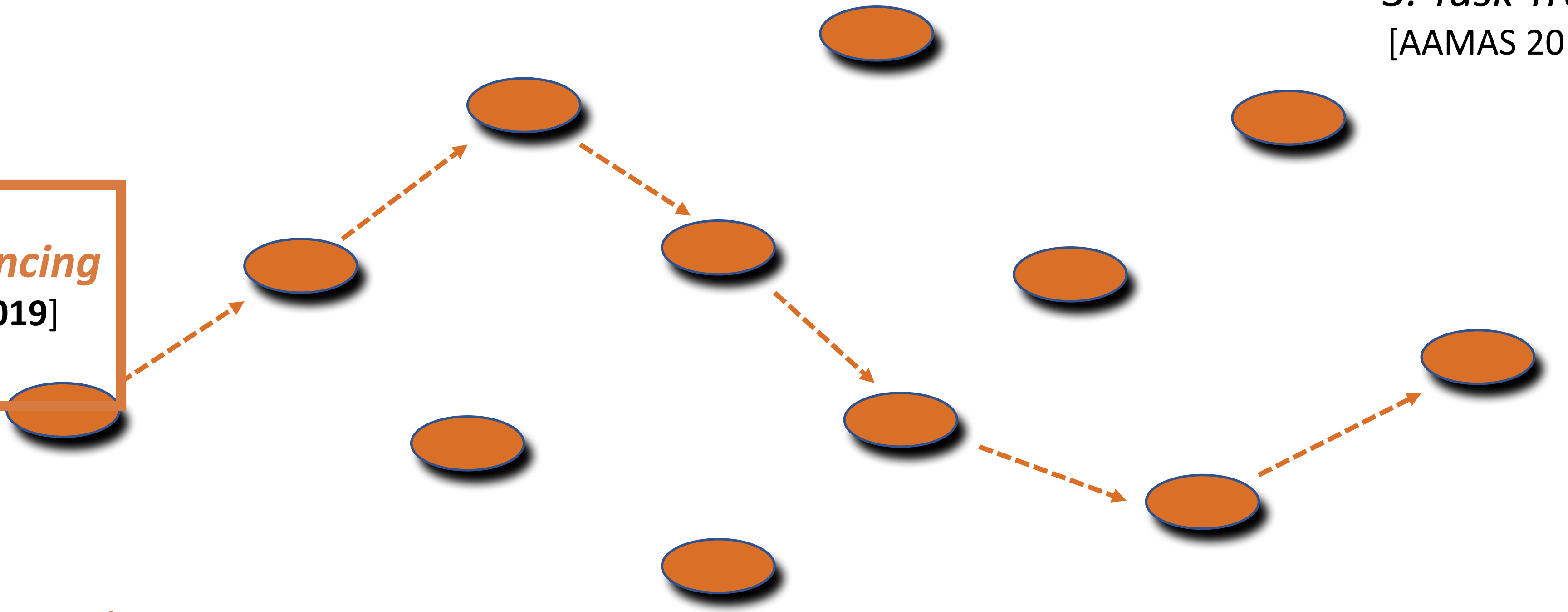
Contributions

1. Problem Formalization
[JMLR 2020] (Chapter 3)

2. Task Generation
[AAMAS 2016] (Chapter 4)

3. Task Transferability
[AAMAS 2015] (Chapter 5)

4. Automatic Sequencing
[IJCAI 2017, **AAMAS 2019**]
(Chapters 6 & **7**)



5. Curriculum Adaptation
[ICML WS 2020] (Chapter 8)

6. Taxonomy of CL
[JMLR 2020] (Chapter 9)

7. Empirical Evaluation

Why Sequencing is Hard

- Possible sequences grows combinatorially with number of source tasks

A	1!	ϕ
AB, BA	2! = 2	ϕ, A, B
ABC, ACB, BAC, BCA, CAB, CBA	3! = 6	ϕ, A, B, C, AB, AC, BA, BC, CA, CB
ABCD, ABDC, ACBD, ACDB, ADBC, ADCB, …	4! = 24	….
…	10! = 3.6M	….

- Even more if size of curriculum not fixed in advance or tasks can repeat
- Learning in a task is stochastic (environment + exploration)
- Learning in a task affects how the agent learns in the next task
- Evaluating a curriculum is expensive

Sequencing using Learning

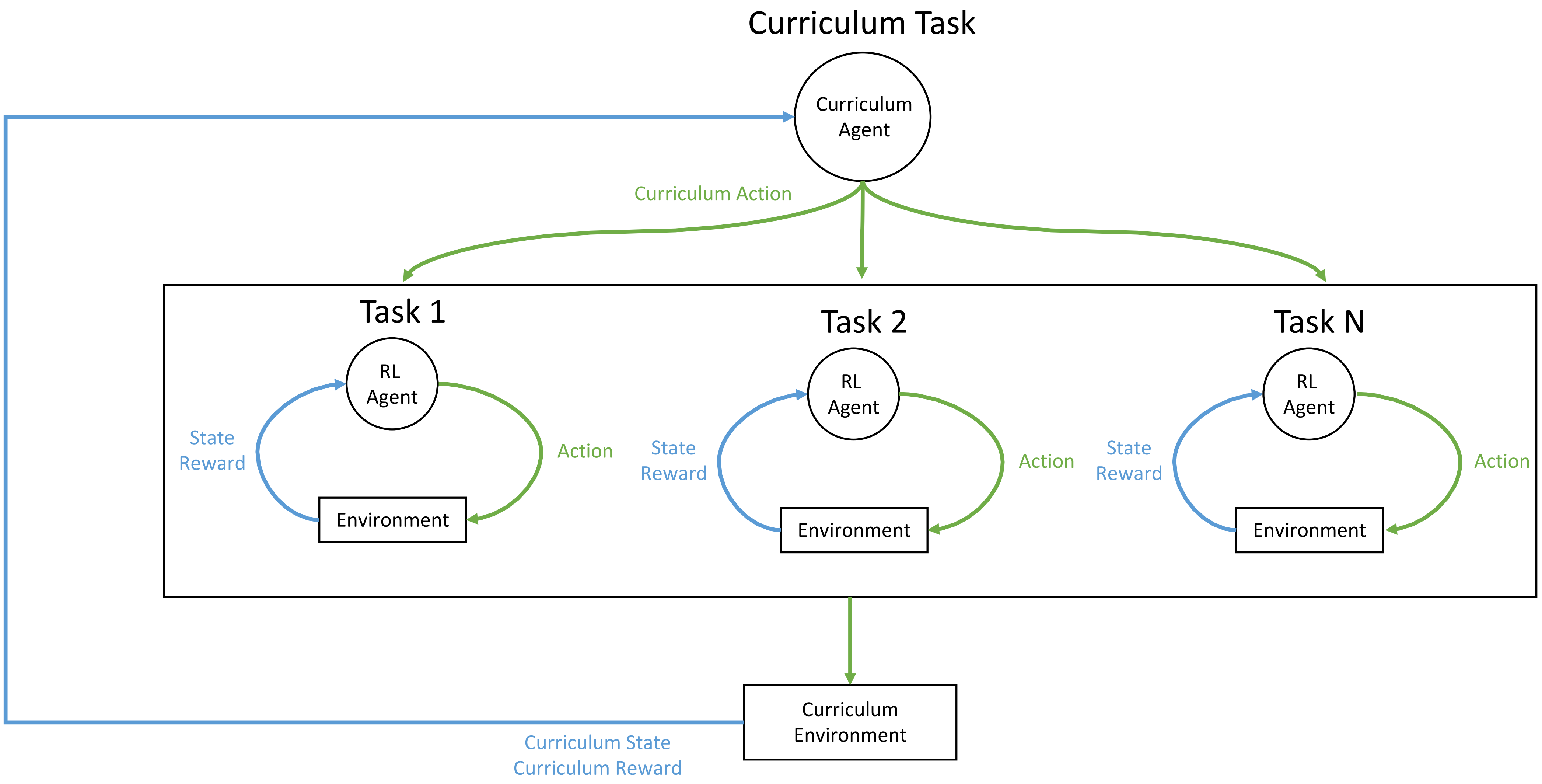
Previous Method

- Used a heuristic for sequencing
- Assumed generated source tasks were relevant to target task
- I.e. no negative transfer
- Fast, but more sensitive

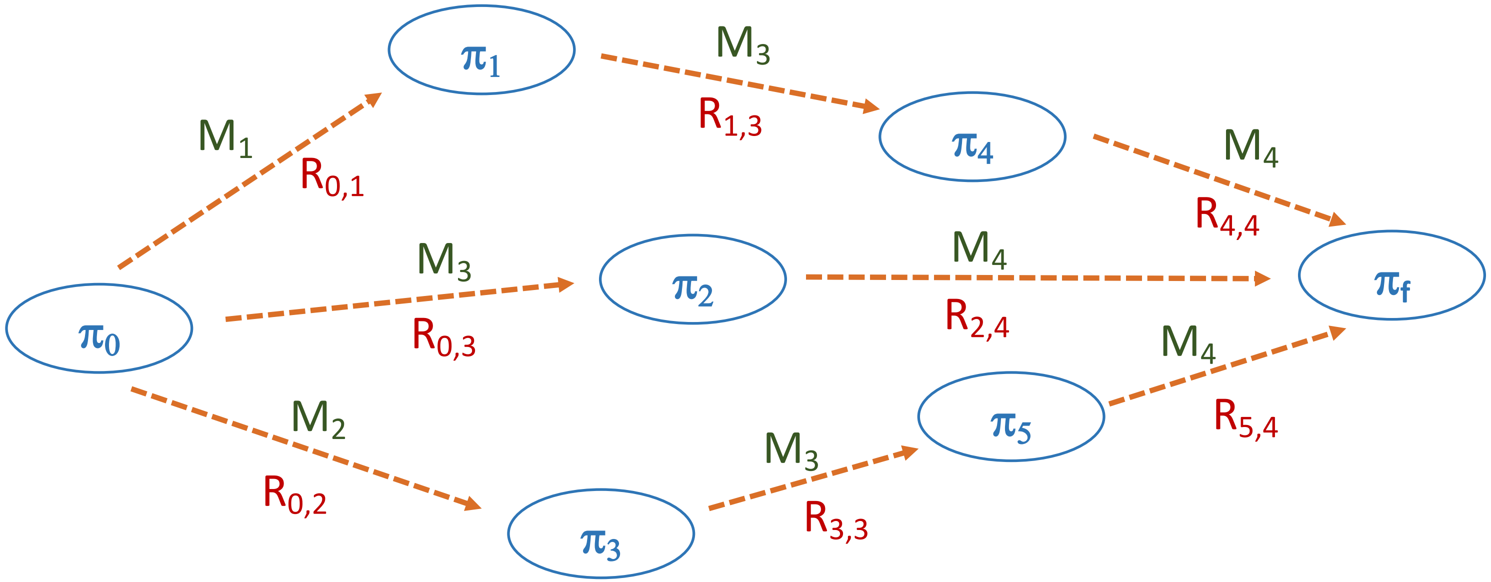
This Method

- Uses data to learn how to sequence
- Trajectories of curricula
- No assumptions on quality of source tasks
- Slower, but more robust

Sequencing as an MDP

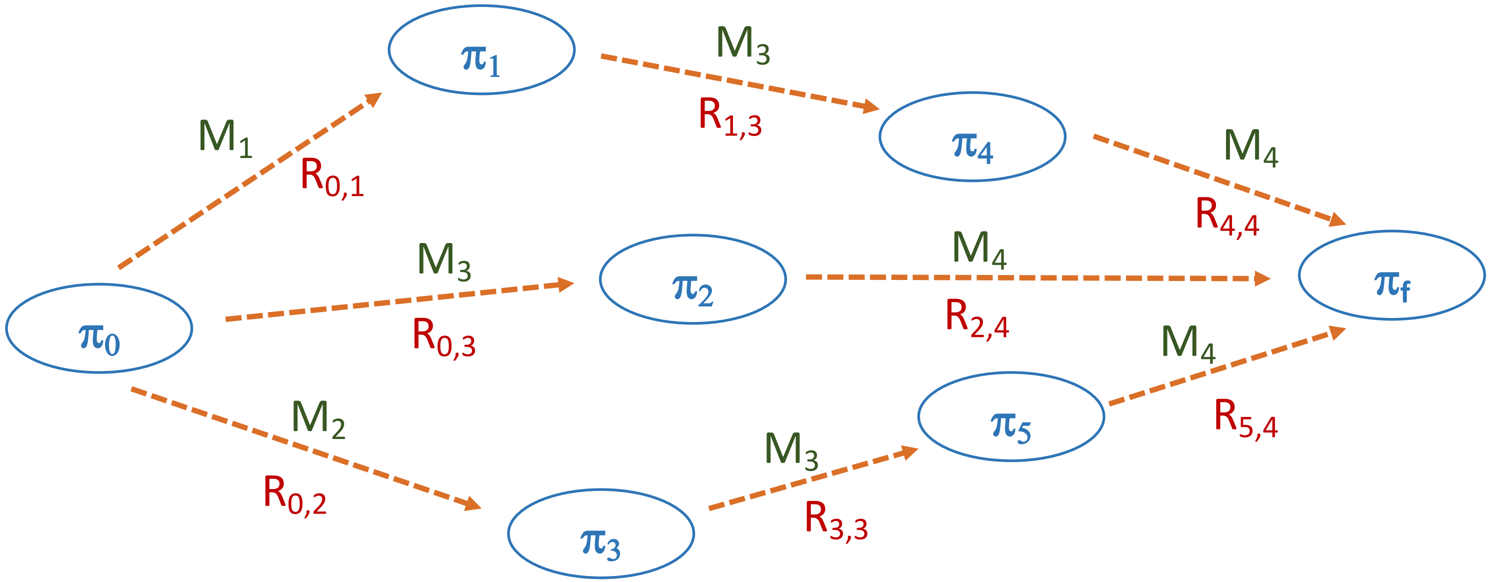


Sequencing as an MDP



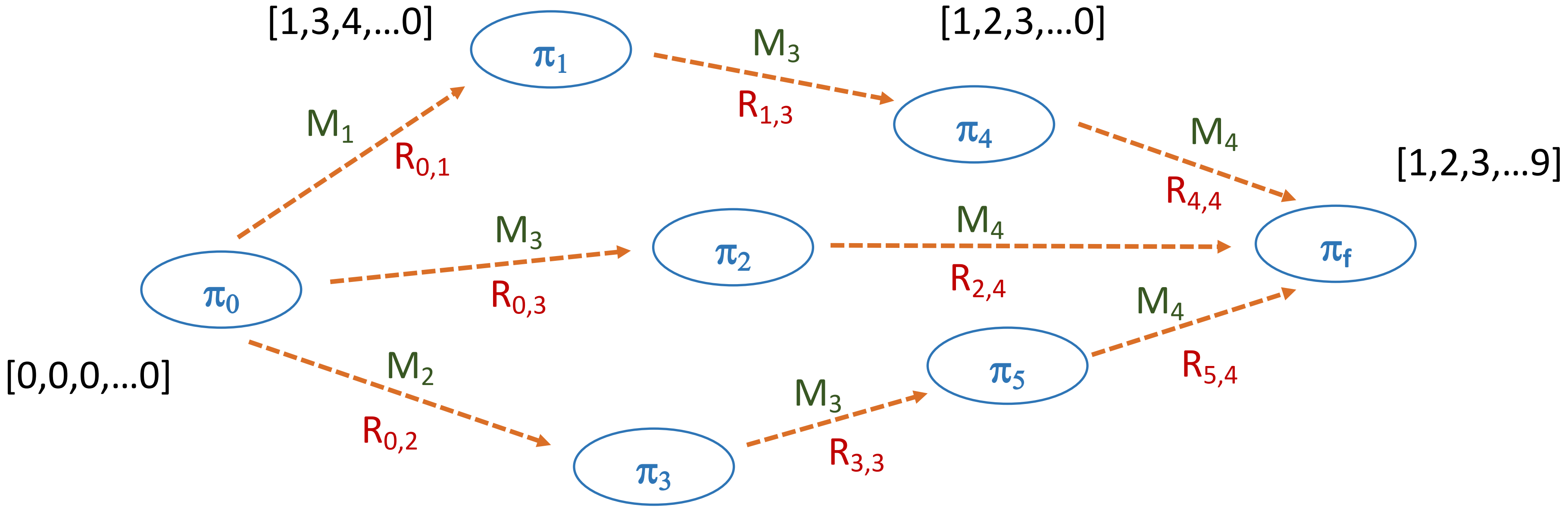
- **State space S^C**: All policies π_i an agent can represent
- **Action space A^C**: Different tasks M_j an agent can train on (e.g. to convergence)
- **Transition function $p^C(s^C,a^C)$**: Learning task a^C transforms an agent's policy s^C
- **Reward function $r^C(s^C,a^C)$**: Cost in time steps to learn task a^C given policy s^C

Sequencing as an MDP



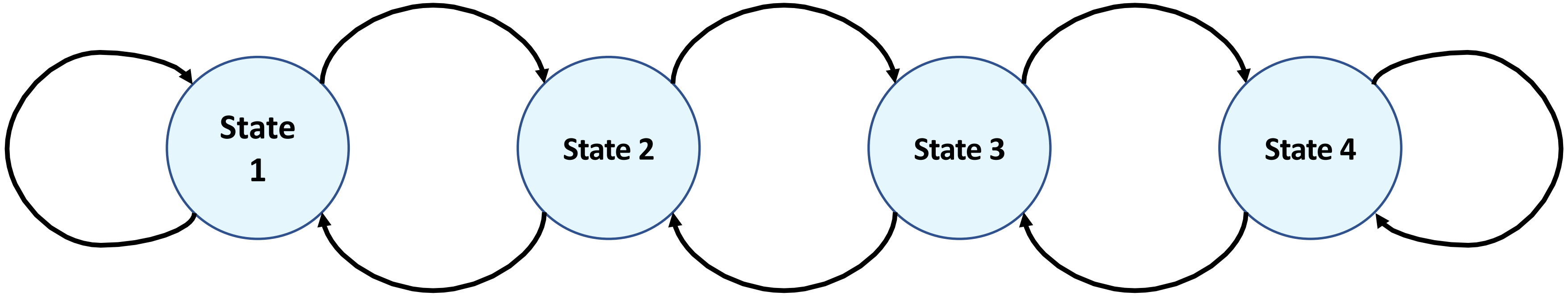
- A policy π^C: $S^C \rightarrow A^C$ on this curriculum MDP (CMDP) specifies which task to train on given learning agent policy π_i
- Essentially training a teacher
- How to learn a curriculum policy over this CMDP?
- How does CMDP change when transfer method or evaluation metric changes?

Learning in Curriculum MDPs



- Express raw CMDP state using the weights of base agent's VF/policy
- Extract features so that similar policies (CMDP states) are "close" in feature space

Example: Discrete Representations



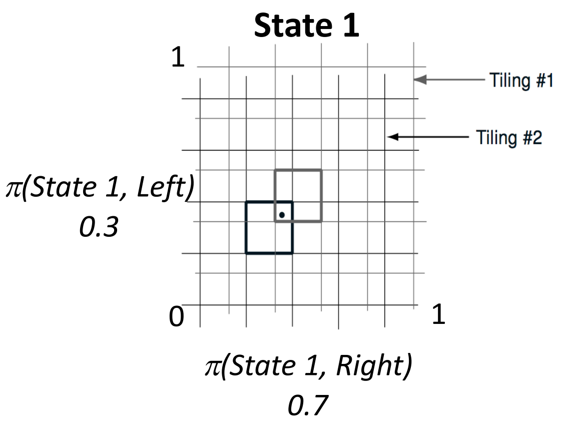
CMDP State 1			
	Left	Right	Policy
State 1	0.3	0.7	→
State 2	0.1	0.9	→
State 3	0.4	0.6	→
State 4	0.0	1.0	→

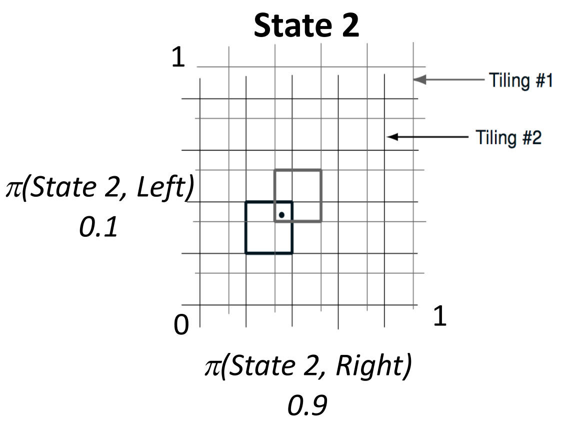
CMDP State 2			
	Left	Right	Policy
State 1	0.2	0.8	→
State 2	0.2	0.8	→
State 3	0.2	0.8	→
State 4	0.3	0.7	→

CMDP State 3			
	Left	Right	Policy
State 1	0.7	0.3	←
State 2	0.9	0.1	←
State 3	0.6	0.4	←
State 4	0.0	1.0	→

- CMDP states 1 and 2 encode very similar policies, and should be close in CMDP representation space
- Then they will have similar action values/probabilities

Example: Discrete Representations





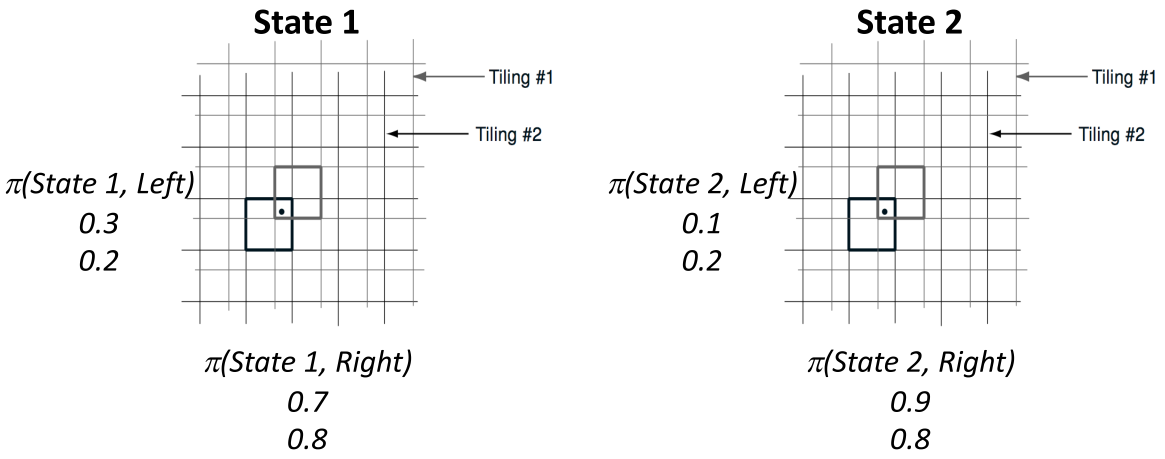
....

CMDP State 1			
	Left	Right	Policy
State 1	0.3	0.7	→
State 2	0.1	0.9	→
State 3	0.4	0.6	→
State 4	0.0	1.0	→

- One approach: use tile coding
- Lays a grid of overlapping tilings over subsets of state variables
- Each tile in tiling associated with a weight
- Activated tiles in each tiling contribute equally to the output

- Create a separate tiling on a state-by-state level

Example: Discrete Representations



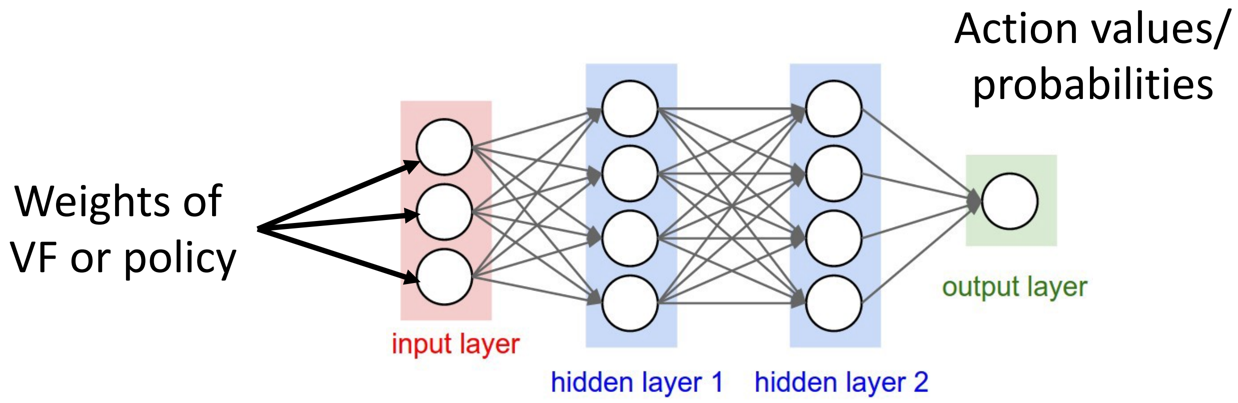
CMDP State 1			
	Left	Right	Policy
State 1	0.3	0.7	→
State 2	0.1	0.9	→
State 3	0.4	0.6	→
State 4	0.0	1.0	→

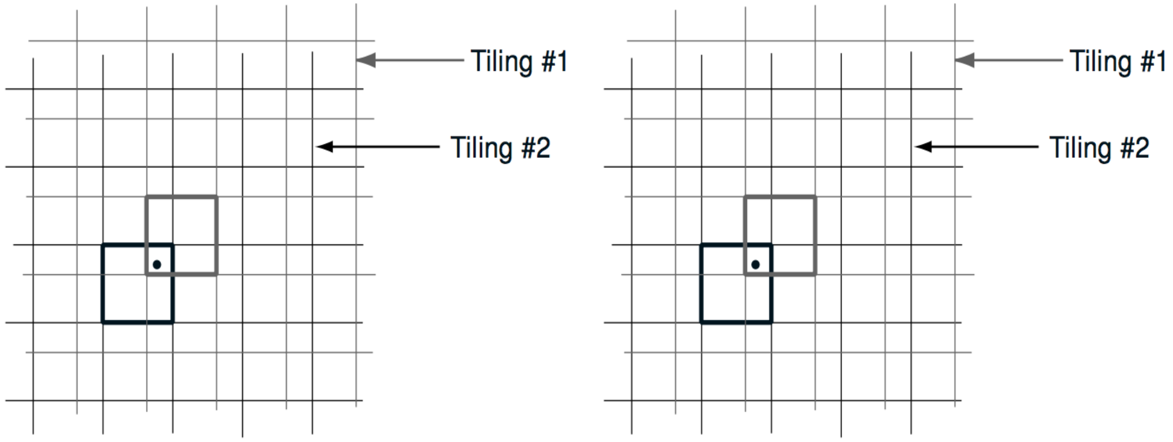
CMDP State 2			
	Left	Right	Policy
State 1	0.2	0.8	→
State 2	0.2	0.8	→
State 3	0.2	0.8	→
State 4	0.3	0.7	→

- The more similar the policies are in a primitive state, the more common tiles will be activated
- The more primitive states that are common, the more similar the output action value/probability will be

Continuous CMDP Representations

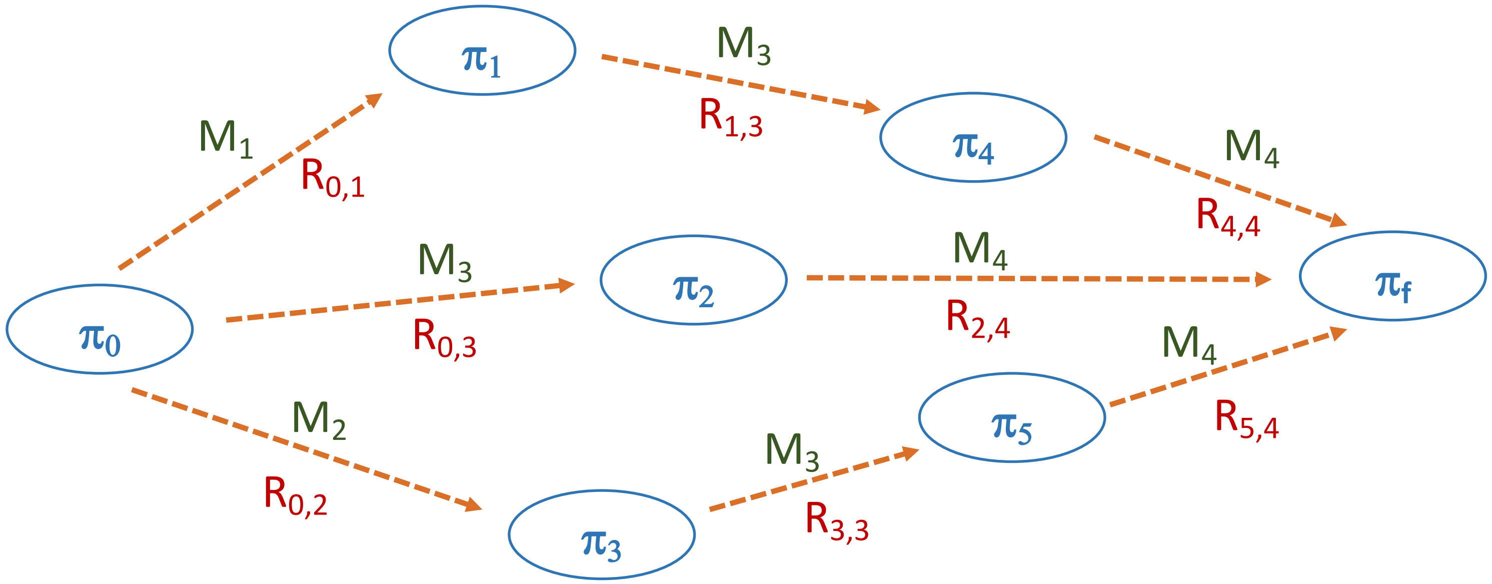
- In continuous domains, weights are not local to a state
- Needs to be done separately for each domain
  - Neural networks
  - Tile coding
  - Etc…
- If the base agent uses a linear function approximator, one can use tile coding as before, creating a separate tiling for each weight variable





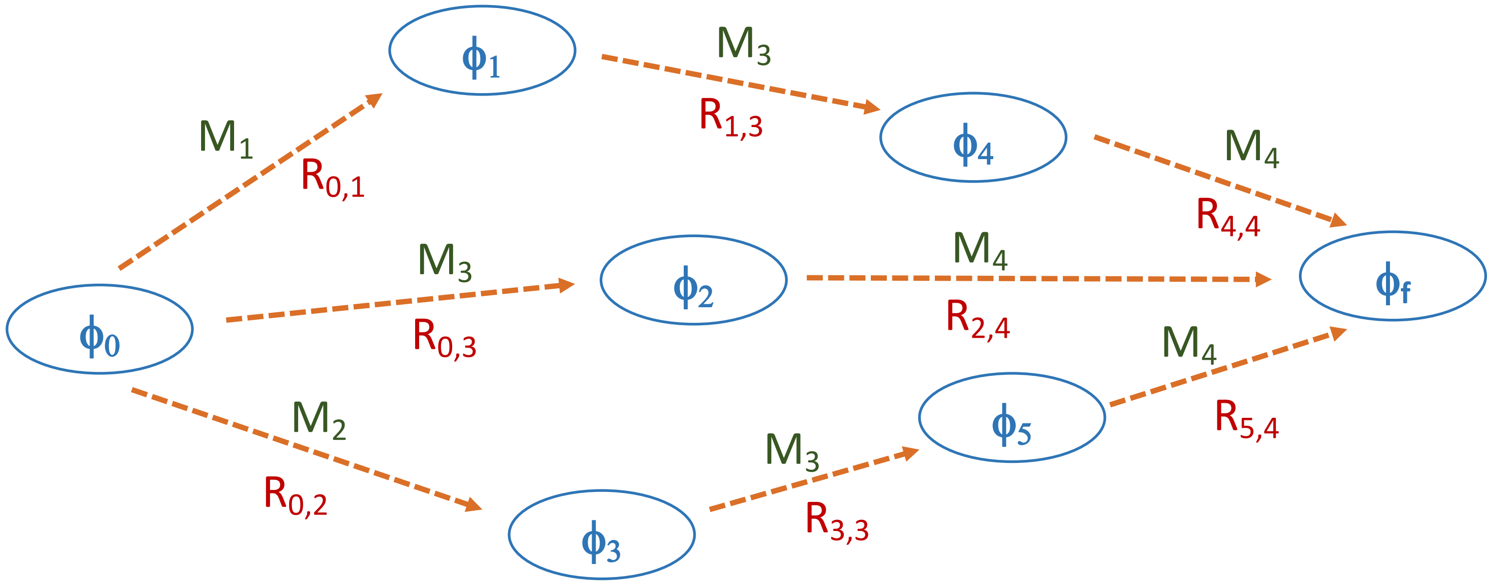
Multiple tilings over different subsets of weights

Changes in Transfer Algorithm



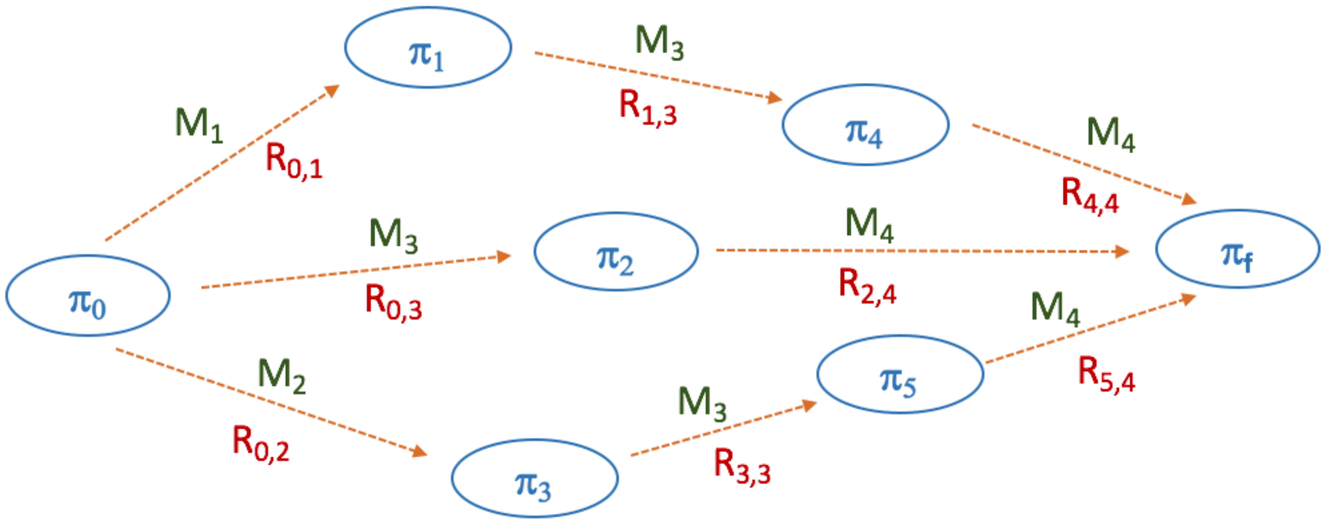
- Transfer method directly affects CMDP state representation and transition function
- CMDP states represent "states of knowledge"
- Knowledge can be represented in terms of the student's policies/value functions, but also in the reward function by transferring a shaping reward

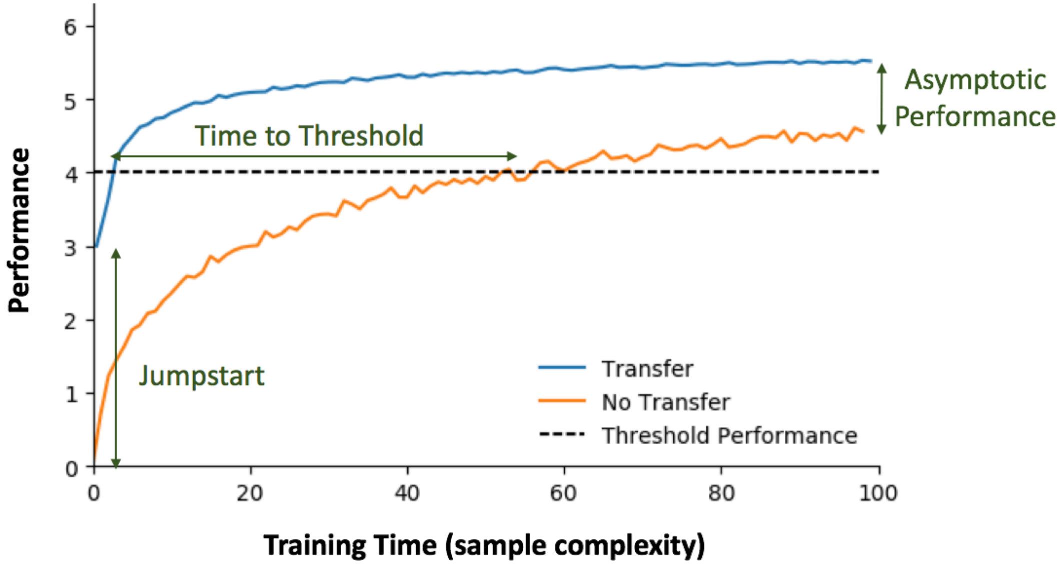
Changes in Transfer Algorithm



- Transfer shaping reward by:
  - Use value function learned in sources to create potential functions
  - Potential function used to generate shaping reward in next task
  - Potentials are accumulated over the course of curriculum
- Similar process as before since potentials are parameterizable

Optimizing Different Metrics





Change reward function $r^C(s^C, a^C)$ based on metric to optimize:

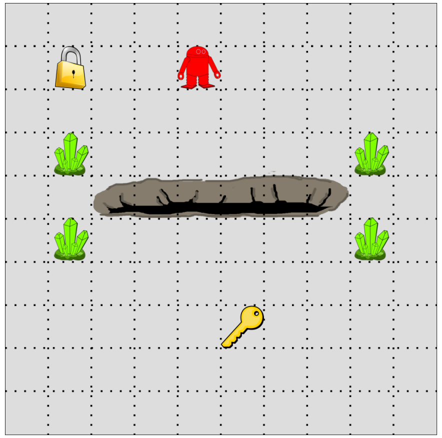
- Time to threshold: Cost in time steps to learn task a^C given policy s^C
- Asymptotic performance: Reward transitions to terminal states by final performance
- Jumpstart: Reward transitions to terminal states by increase in performance

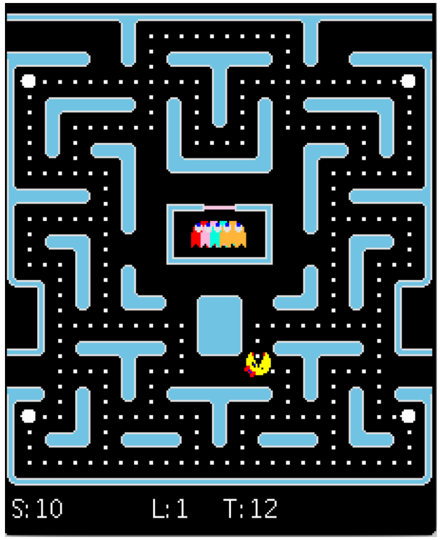
Experimental Results

- Evaluate whether curriculum policies can be learned

- Grid world
  - Multiple base agents
  - Multiple CMDP state representations

- Pacman
  - Multiple transfer learning algorithms
  - How long to train on sources?

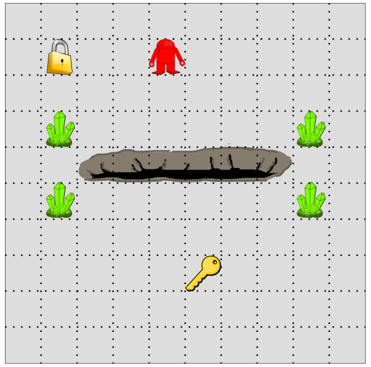




Grid world Setup

Agent Types

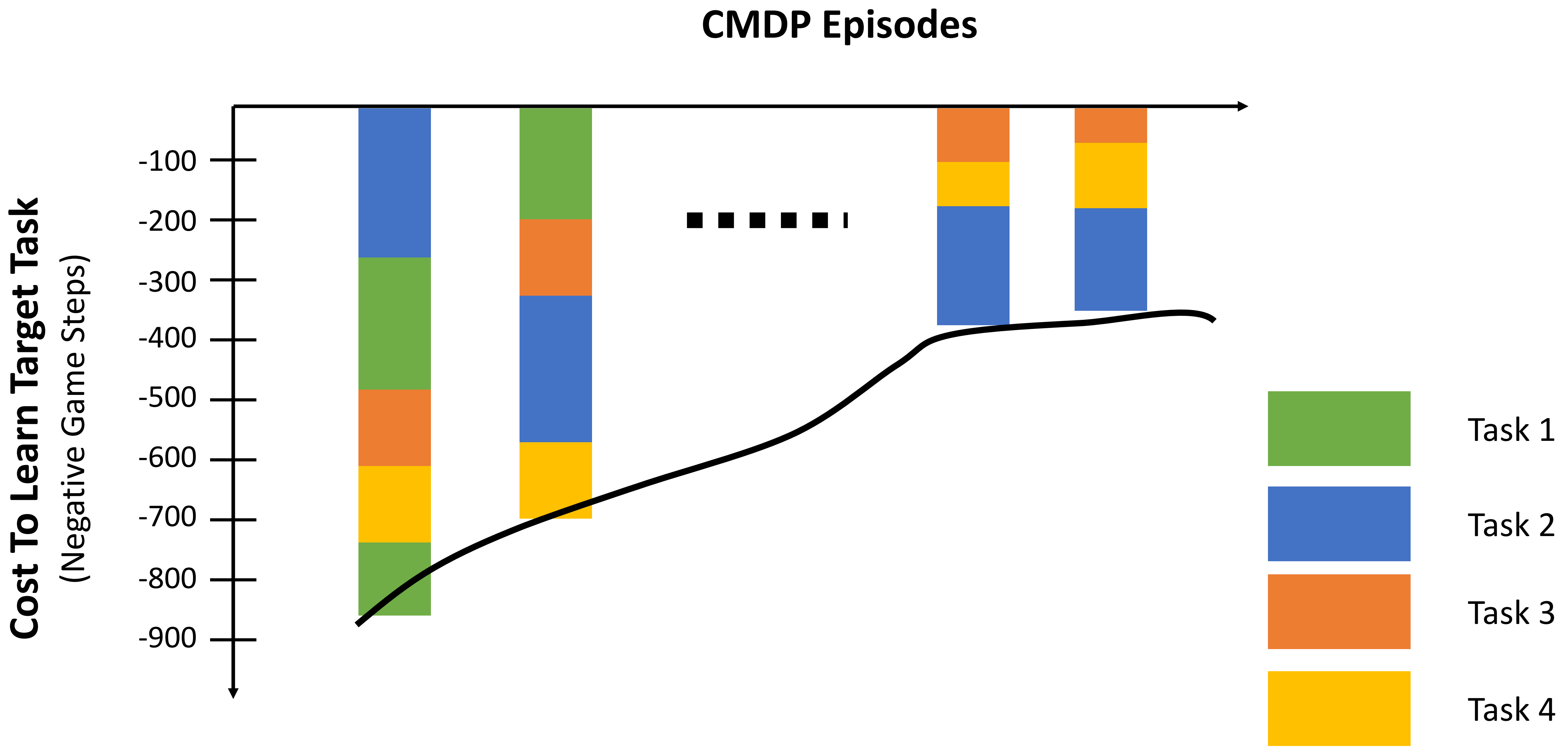
- Basic Agent
  - State: Sensors on 4 sides that measure distance to keys, locks, etc.
  - Actions: Move in 4 directions, pickup key, unlock lock
- Action-dependent Agent
  - State difference: weights on features are shared over 4 directions
- Rope Agent
  - Action difference: Like basic, but can use rope action to negate a pit



CMDP Representations

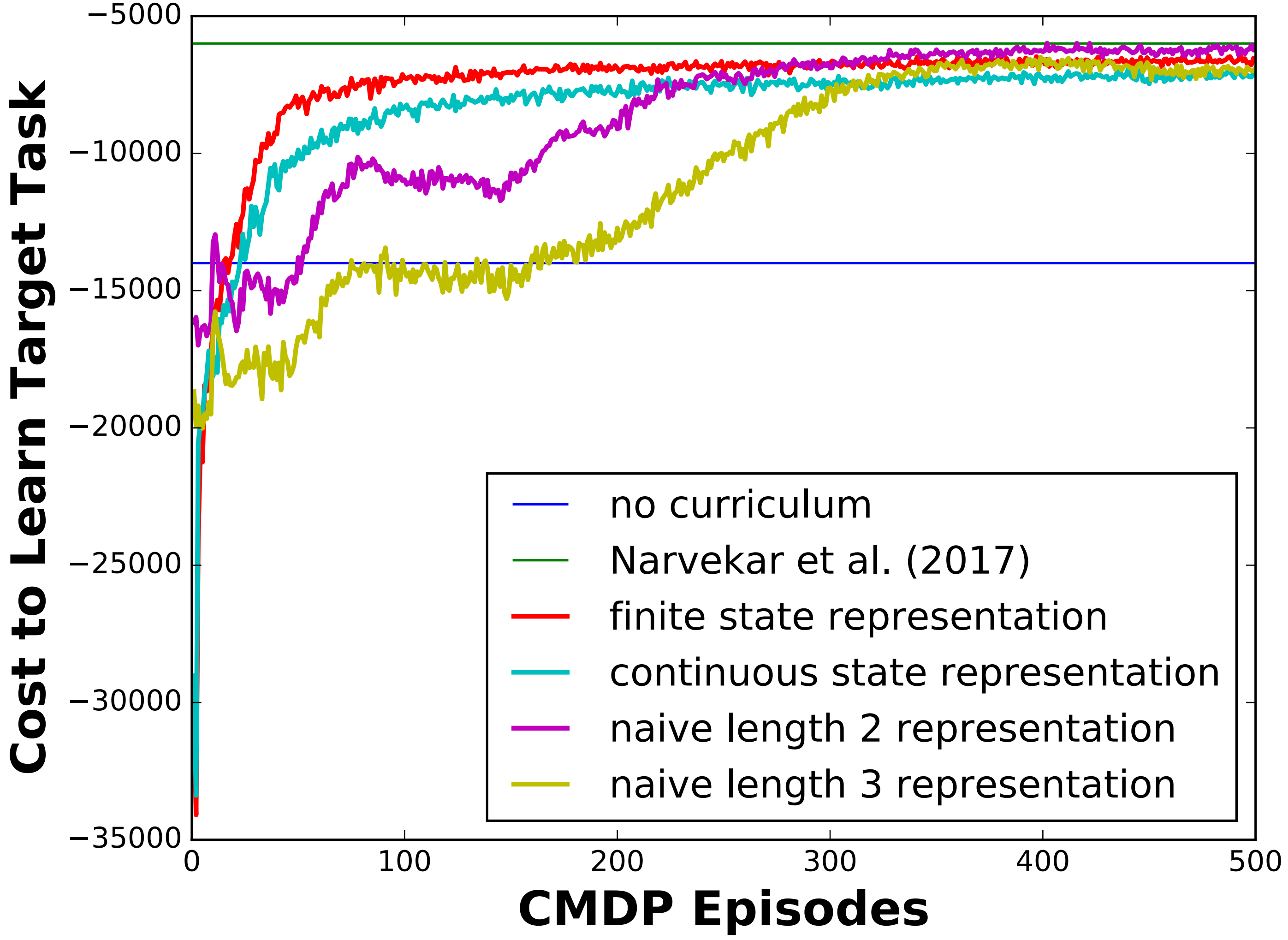
- Finite State Representation
  - For discrete domains, groups and normalizes raw weights state-by-state to form CMDP features
- Continuous State Representation
  - Directly uses raw weights of learning agent as features for CMDP agent

CMDP Curves

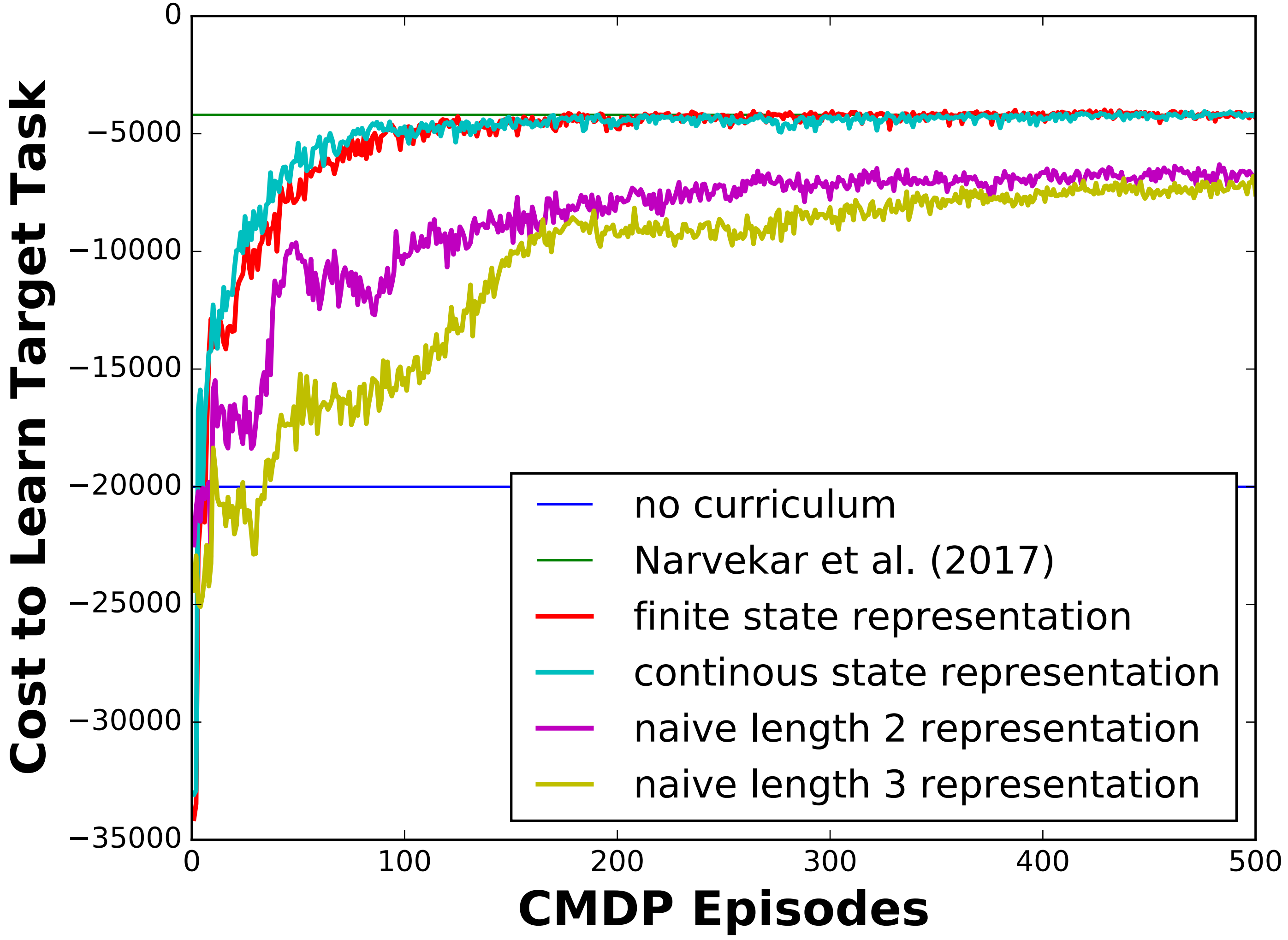


CMDP Episodes
Cost To Learn Target Task
(Negative Game Steps)
-100
-200
-300
-400
-500
-600
-700
-800
-900
Task 1
Task 2
Task 3
Task 4

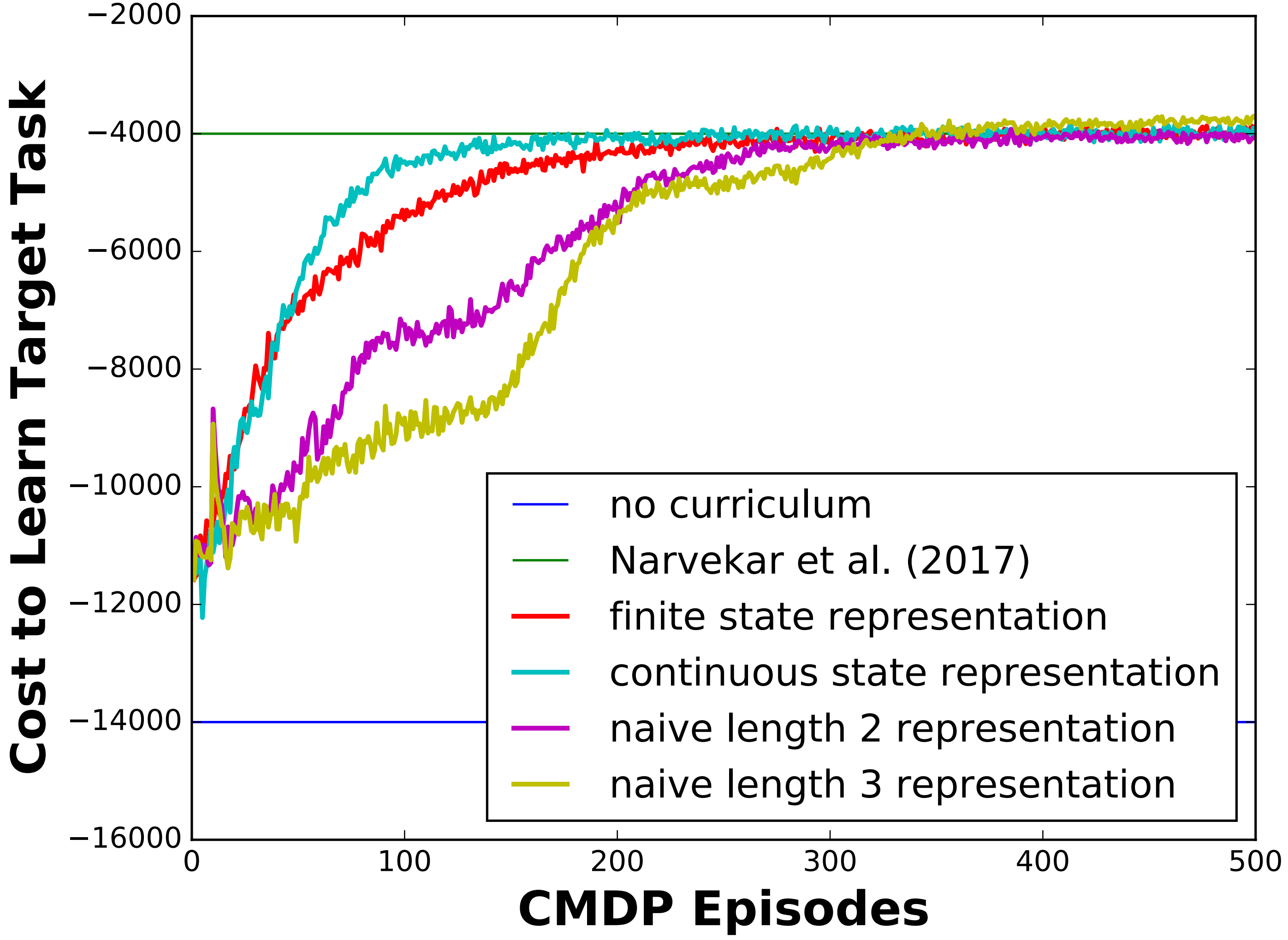
Basic Agent Results



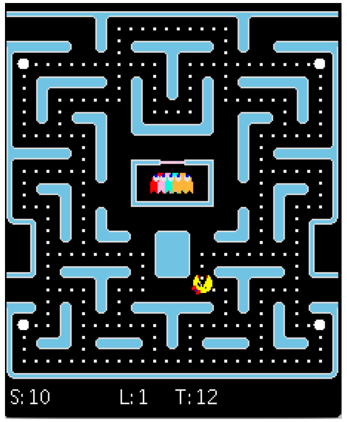
Action-Dependent Agent Results



Rope Agent Results



Pacman Setup



Agent Representation

- Action-dependent egocentric features

CMDP Representation

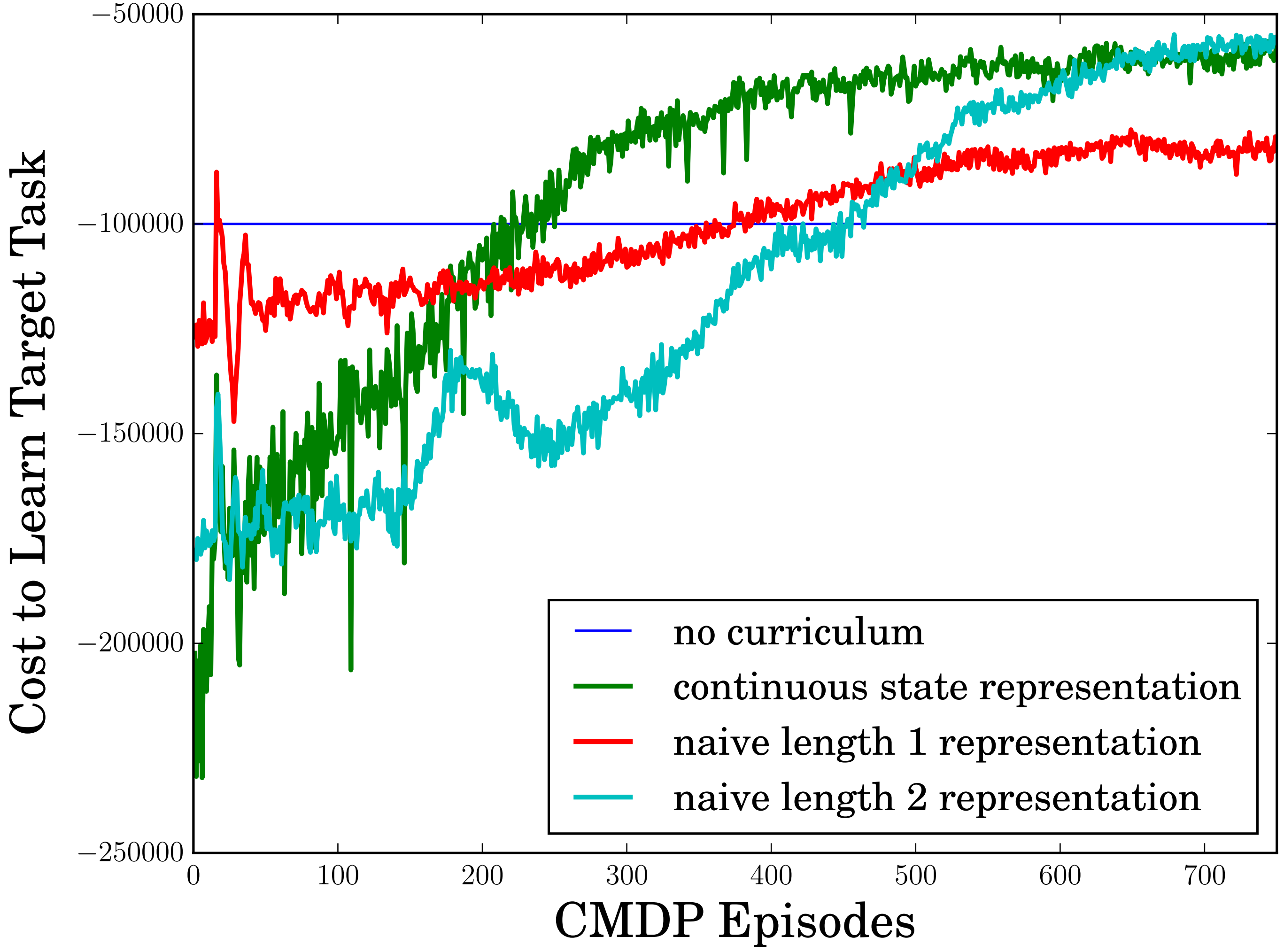
- Continuous State Representation
  - Directly uses raw weights of learning agent as features for CMDP agent

Transfer Methods

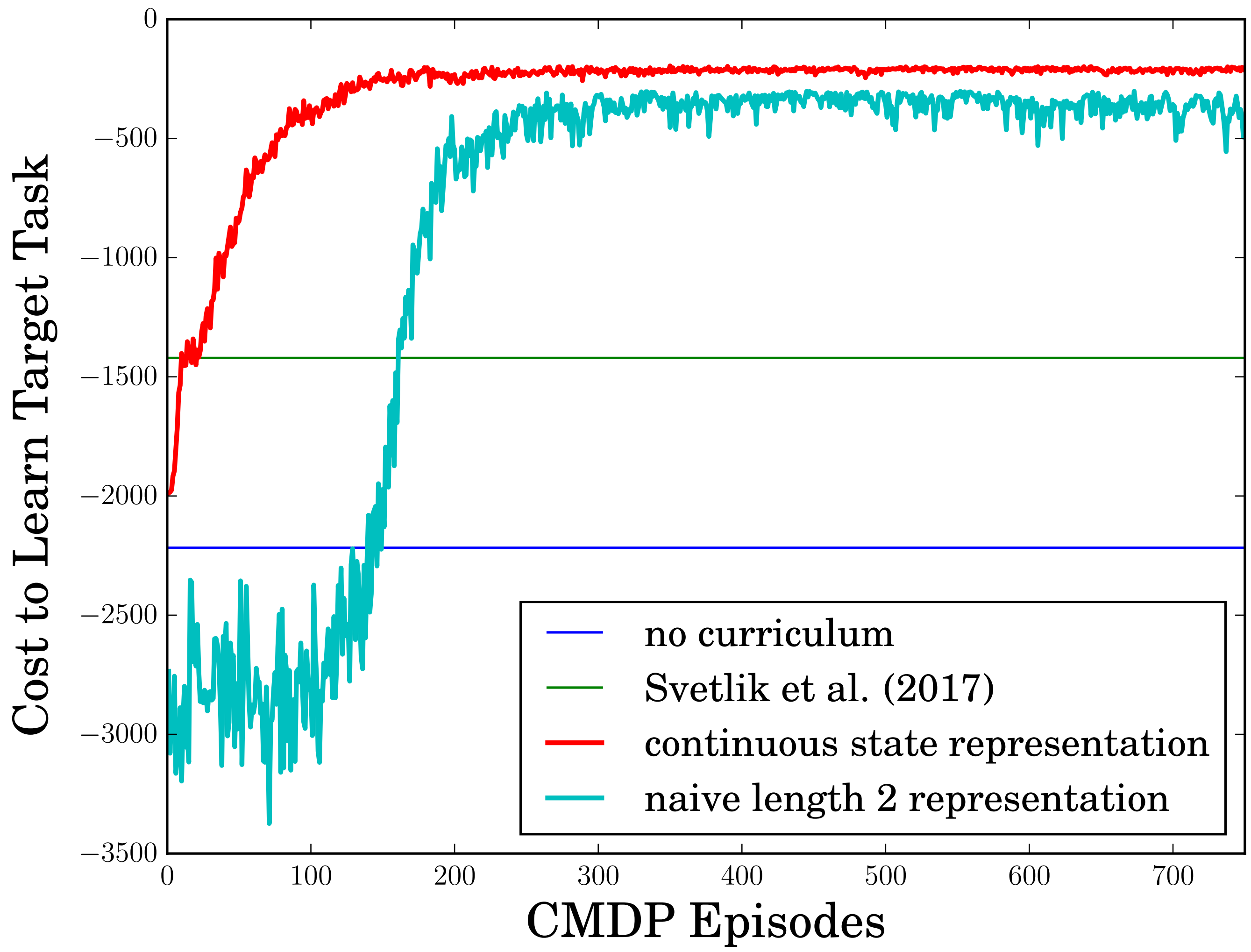
- Value Function Transfer
- Reward Shaping Transfer

How long to train on a source task?

Pacman Value Function Transfer

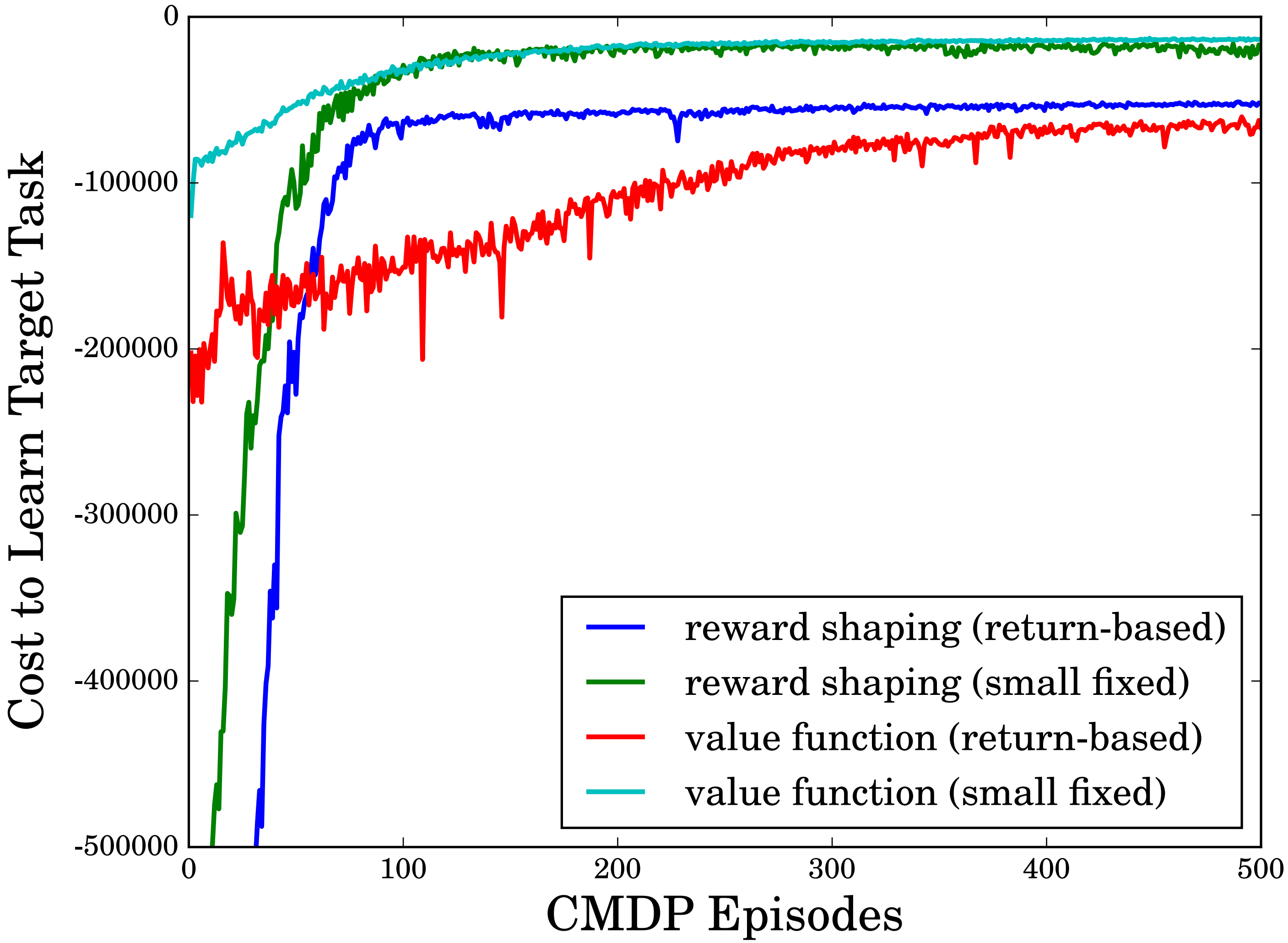


Pacman Reward Shaping Transfer



How long to train?

- Return-based
  - Train until convergence to a specified return
- Small-fixed
  - Train 5 episodes at a time
- Upshot: curriculum policy learns how long to spend on each task



CMDP Results Key Takeaways

1. Curriculum policy learns a curriculum that improves over time

2. This curriculum learns at least as fast/good or better than several baseline methods

3. Robust to CMDP state representation and transfer method

4. Learns how long to spend on source tasks

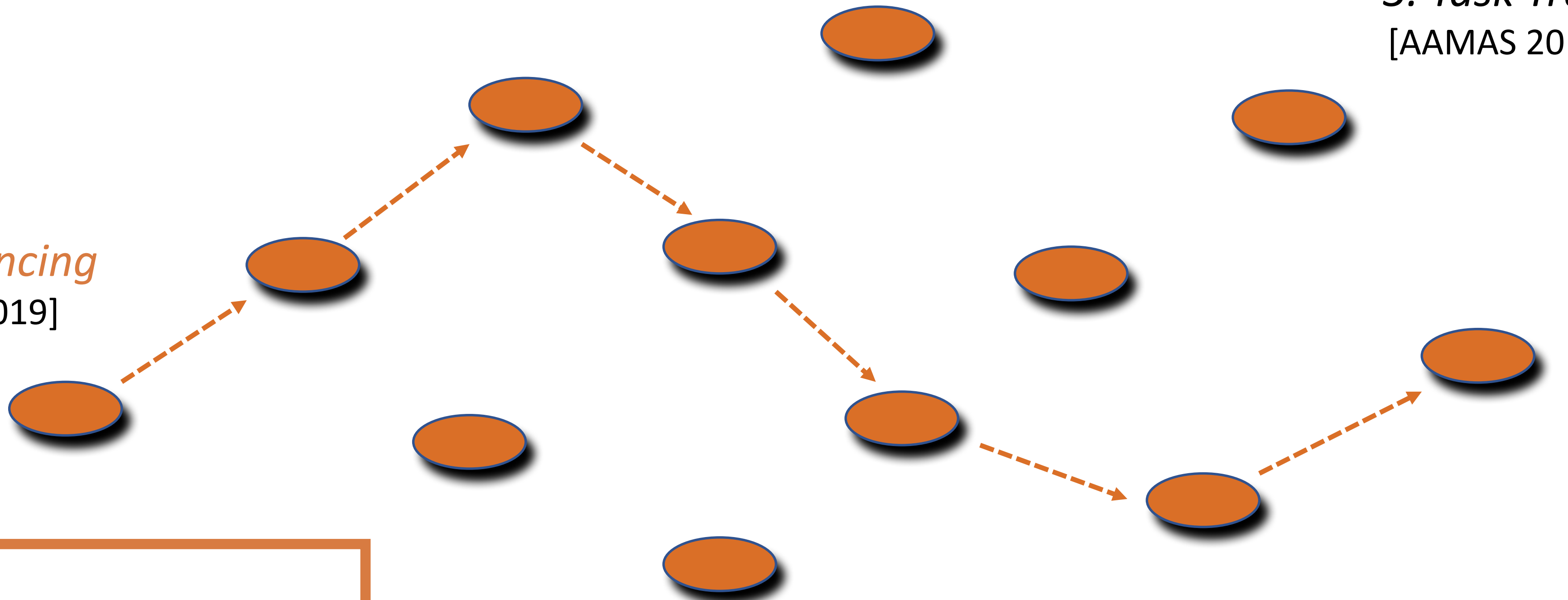
Contributions

1. Problem Formalization
[JMLR 2020] (Chapter 3)

2. Task Generation
[AAMAS 2016] (Chapter 4)

3. Task Transferability
[AAMAS 2015] (Chapter 5)

4. Automatic Sequencing
[IJCAI 2017, AAMAS 2019]
(Chapters 6 & 7)

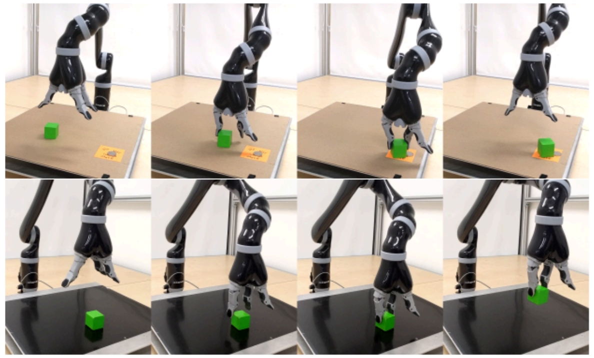


5. Curriculum Adaptation
[ICML WS 2020] (Chapter 8)

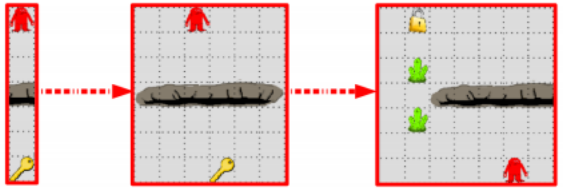
6. Taxonomy of CL
[JMLR 2020] (Chapter 9)

7. Empirical Evaluation

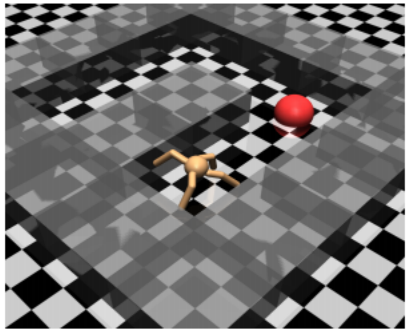
Curricula in RL



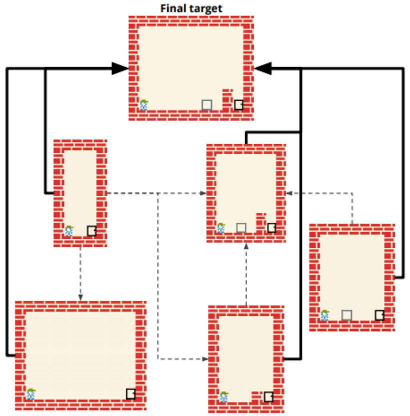
Riedmiller et al. (2018)



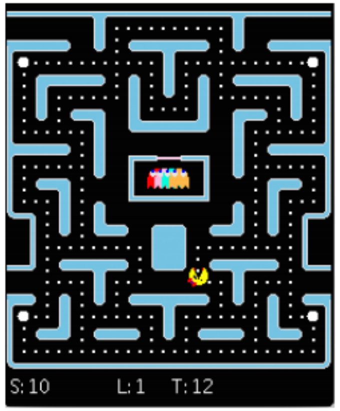
Narvekar et al. (2017)



Florensa et al. (2018)



Svetlik et al. (2017)



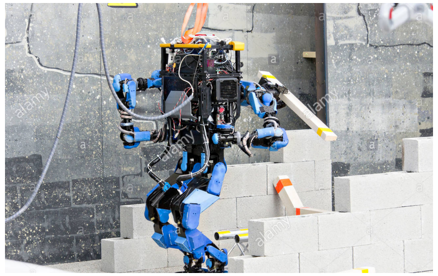
Narvekar & Stone (2019)

- Curricula must be recreated from scratch for each new task or agent
- Generating curricula independently for each agent can be expensive

Curricula in Human Learning

- Curricula are used to teach many people, many different tasks

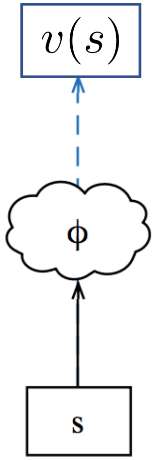
- Can we use knowledge gained about learning a curriculum for one task to speed up learning of a curriculum for a new task?



Combining CMDPs with UVFAs

- Value Function (undiscounted settings)

$$v_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} r(s_t, a_t, s_{t+1}) | s_0 = s\right]$$



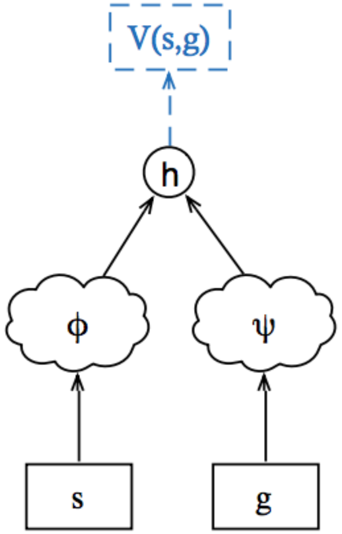
Function Approximator

(linear, neural net, etc.)

- Learn value function that generalizes over states

- Universal Value Function Approximators (UVFAs) generalize over states *s* and goals *g*

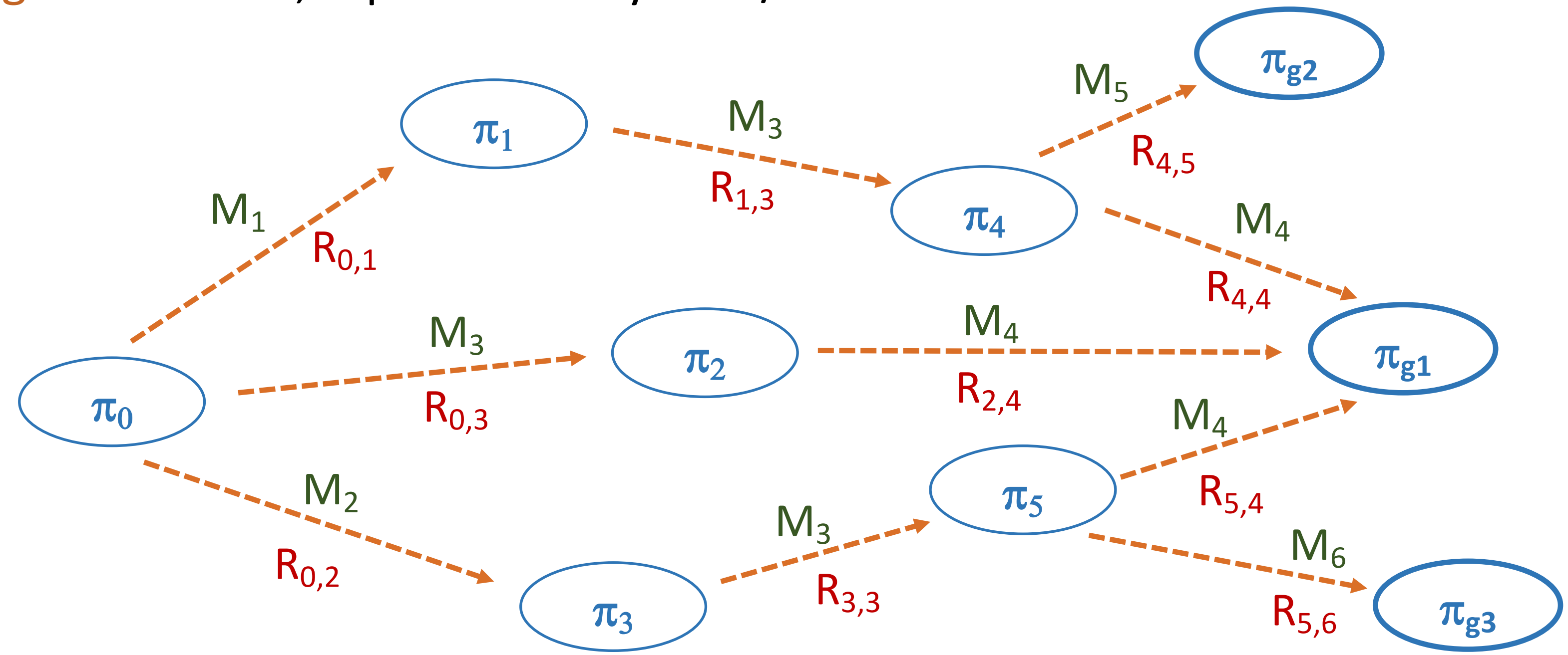
$$v_\pi(s, g) = \mathbb{E}\left[\sum_{t=0}^{\infty} r_g(s_t, a_t, s_{t+1}) | s_0 = s\right]$$



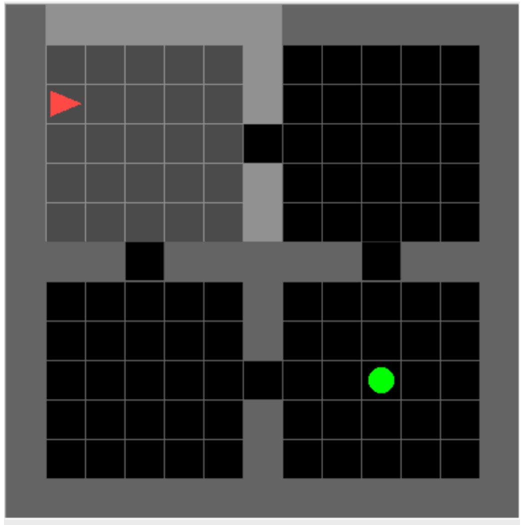
Schaul et al. (2015)

Combining CMDPs with UVFAs

- What is a goal?
  - A waypoint, or more simply a terminal state
- What does this mean in a CMDP?
- Represent goals by a parameterized representation of their task
  - Navigational tasks, represented by start/end coordinates



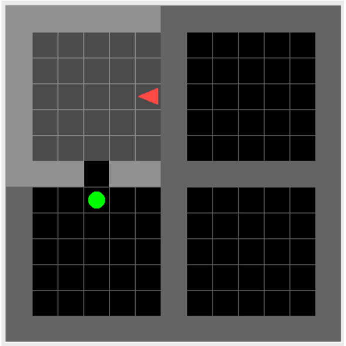
Experimental Results

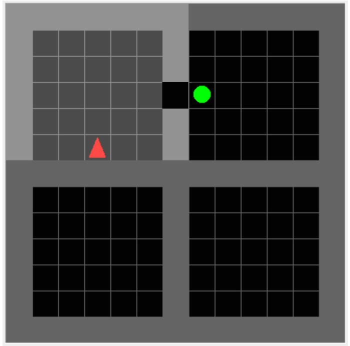


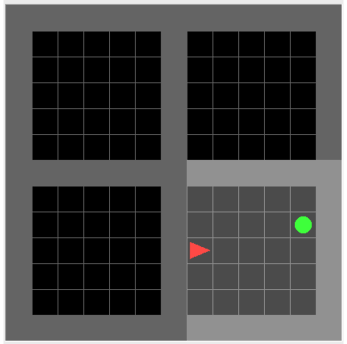
- Evaluate whether curriculum policies learned for one set of tasks can generalize to a novel set of unseen tasks

- Domain where easy to create many task variations

- Navigational tasks
  - Start x
  - Start y
  - End x
  - End y





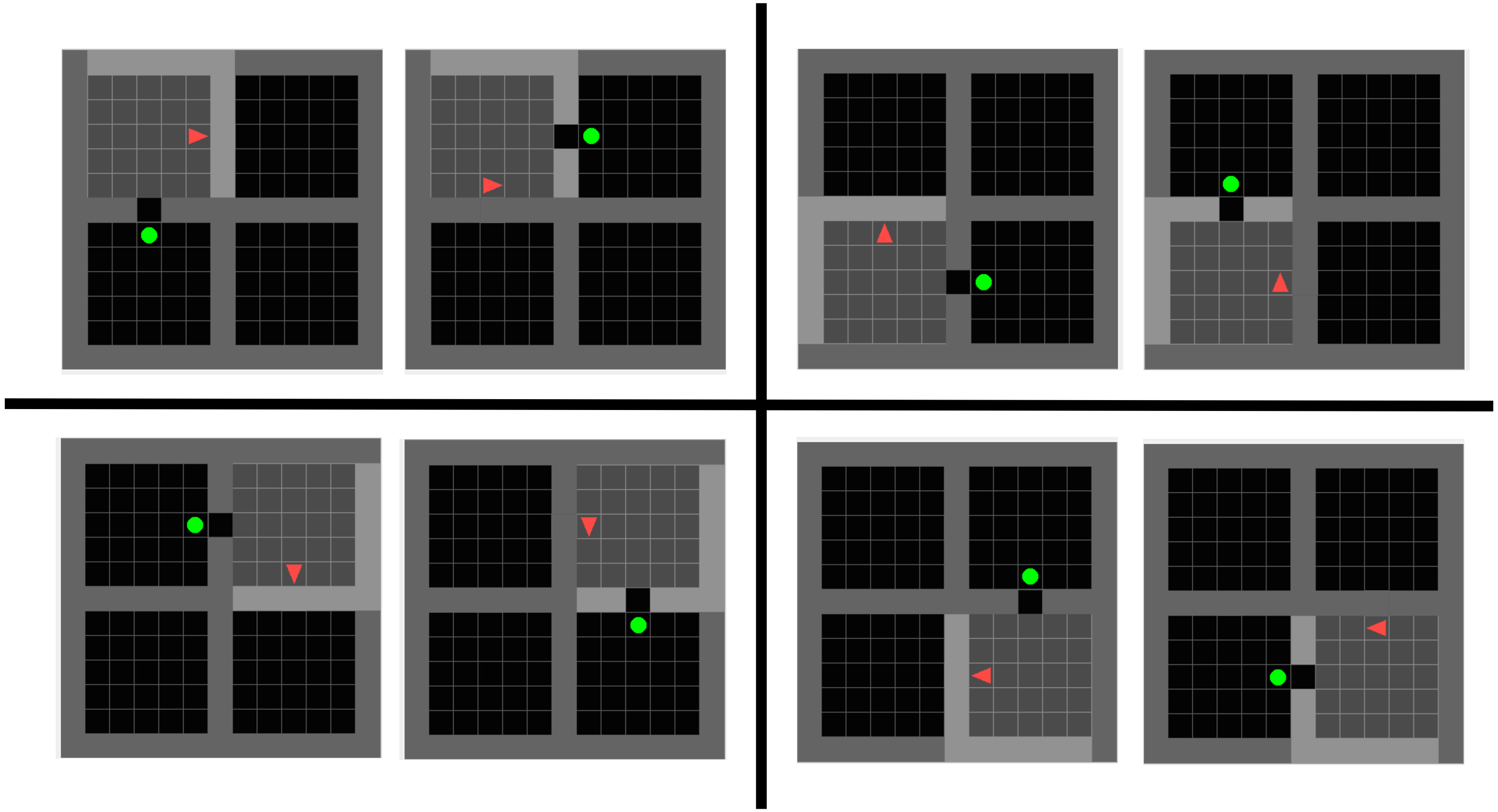


- 9900 distinct possible tasks

Source Tasks

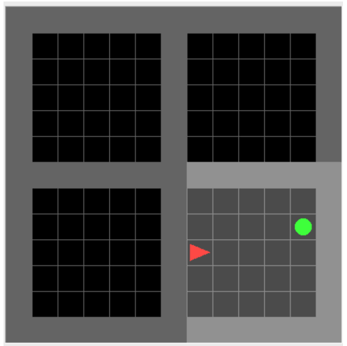
8 Static

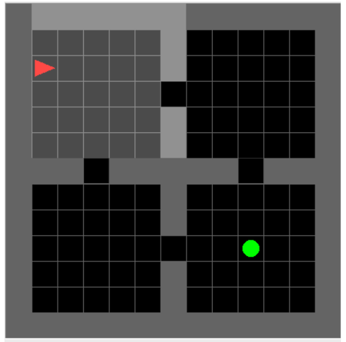
Navigate to adjacent room



1 Dynamic

Navigate to goal in room

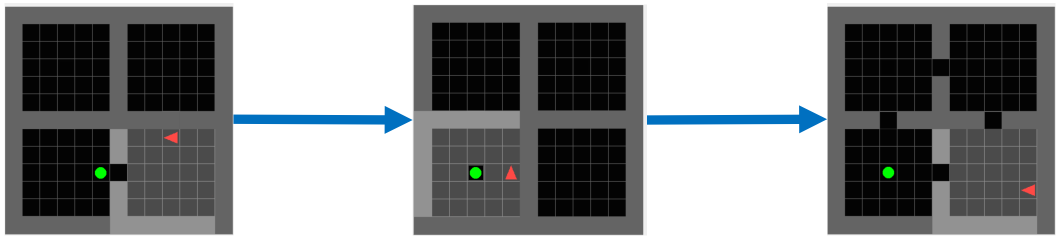




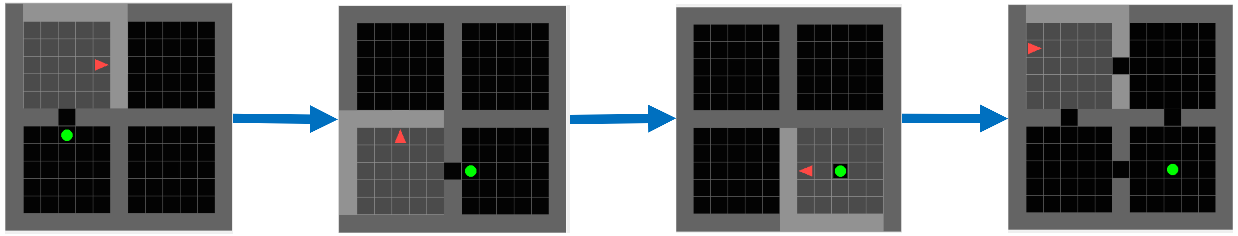
Target Task

A Natural Curriculum

- Navigate to correct room using static tasks
- Navigate to goal using dynamic task
- Combine into target task

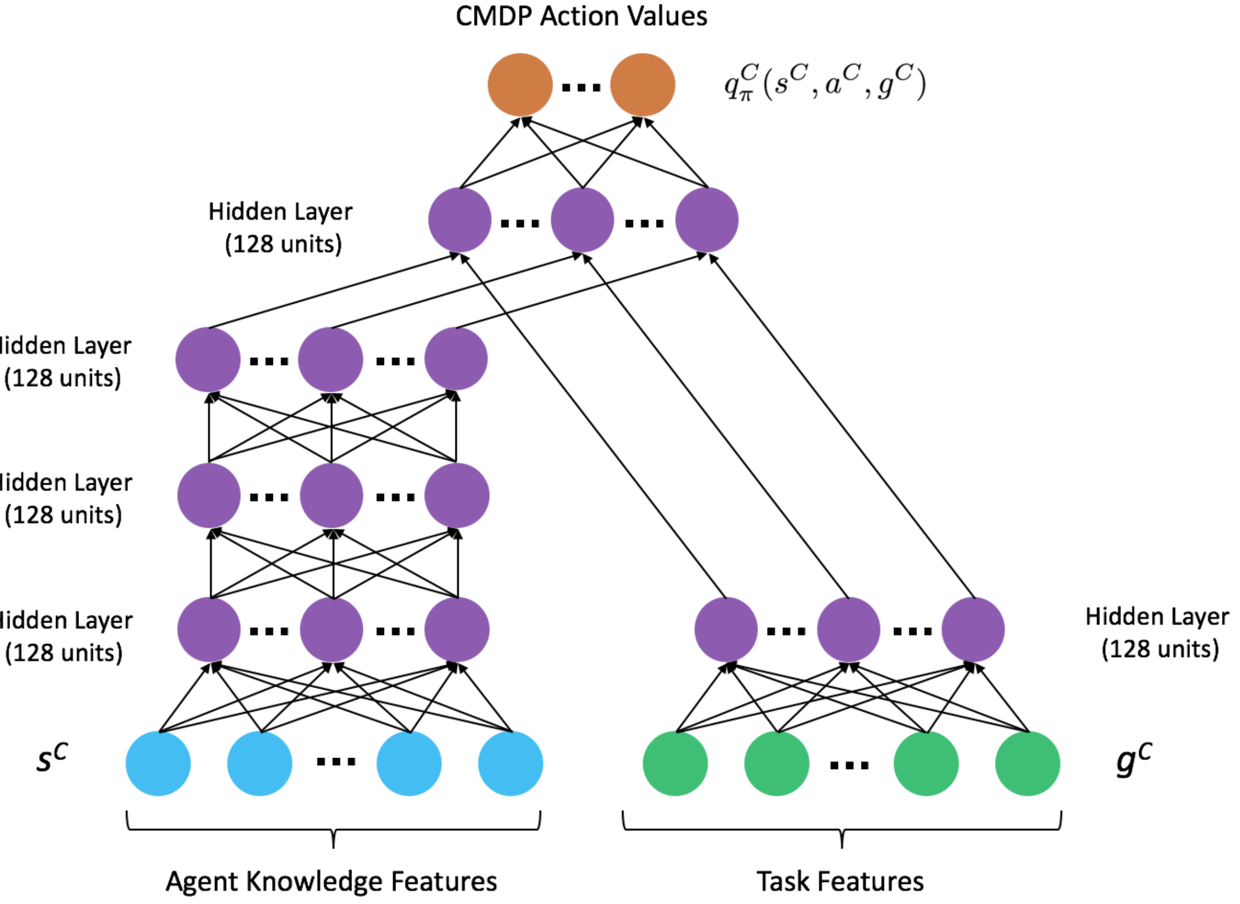


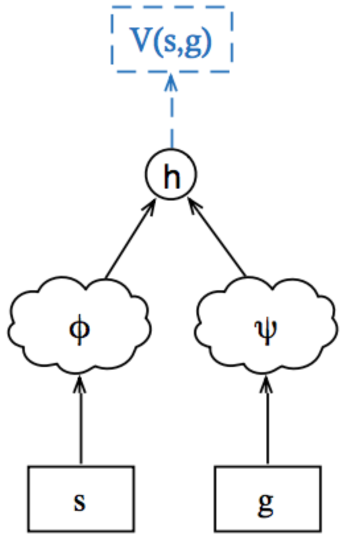
Target Task



Target Task

Network Architecture





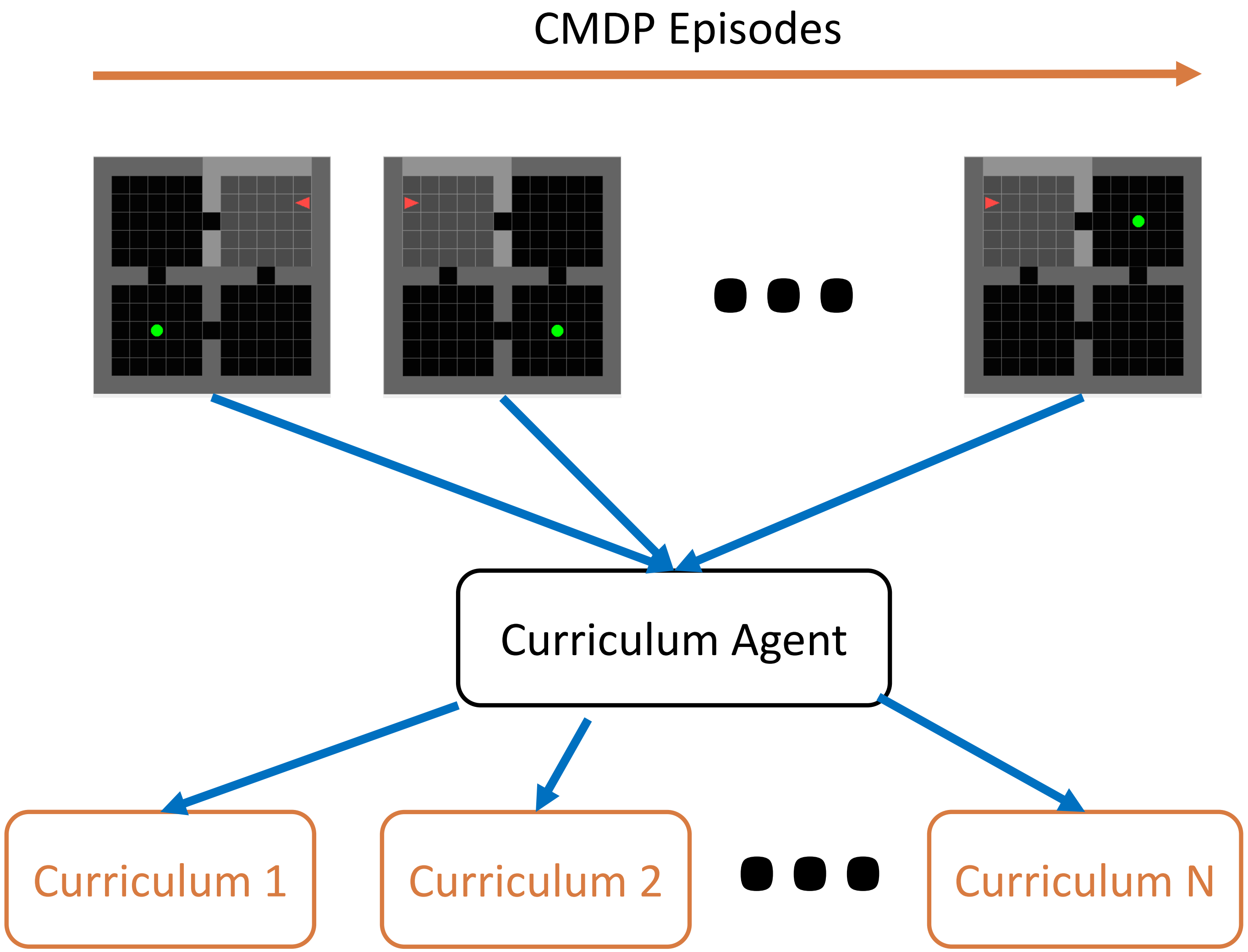
Schaul et al. (2015)

Weights of student RL agent

[Start_x, Start_y, End_x, End_y]

Interpolation Experiments

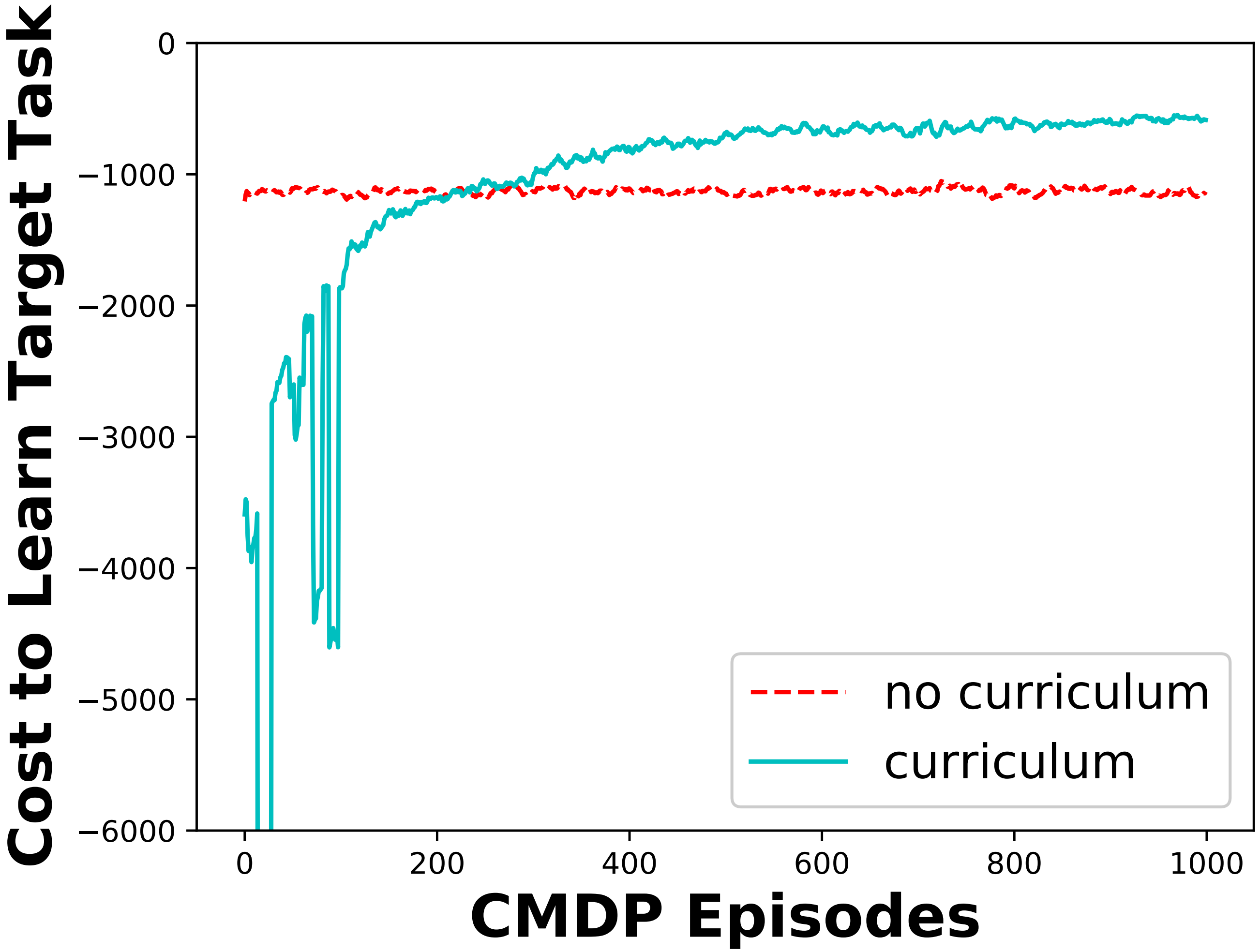
- Randomly shuffle all tasks
- Present tasks one by one
- Each task seen is novel, though similar tasks might have been seen previously
- Over time, learn to produce better curricula for new tasks



Interpolation Results

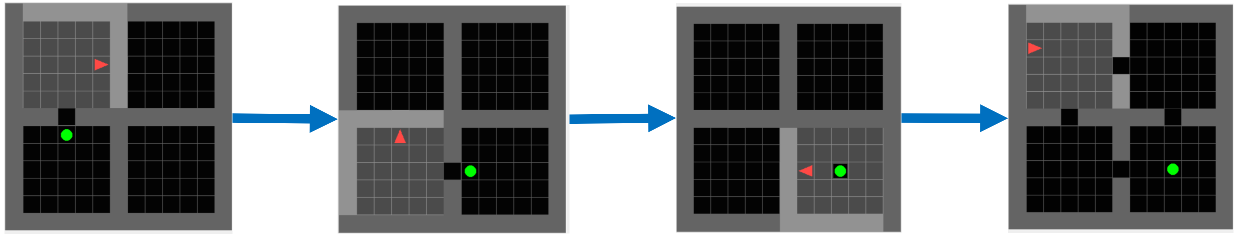
- Learns to interpolate between tasks

- After seeing about 220 tasks, produces curricula that are better than training tabula rasa

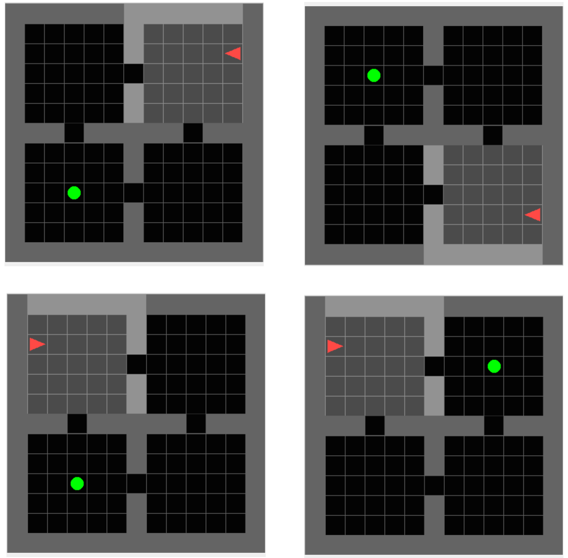


Extrapolation Experiments

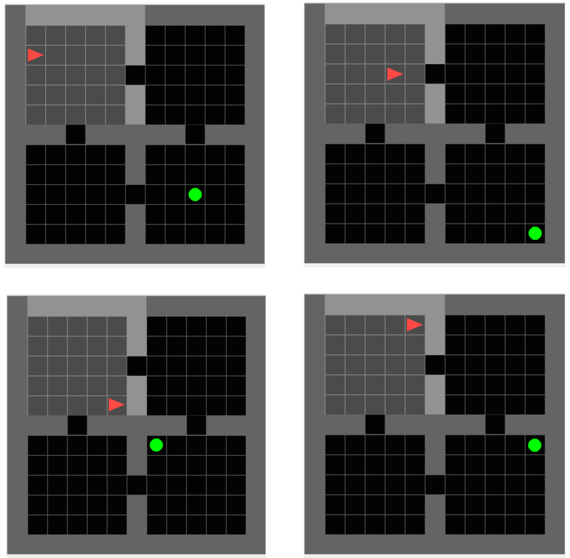
- Evaluate ability to generalize to tasks that need a new curriculum
- Split tasks into train/test set
- Test set tasks start in top left room and end in bottom right
- Optimal test set curricula not seen in training set



Target Task



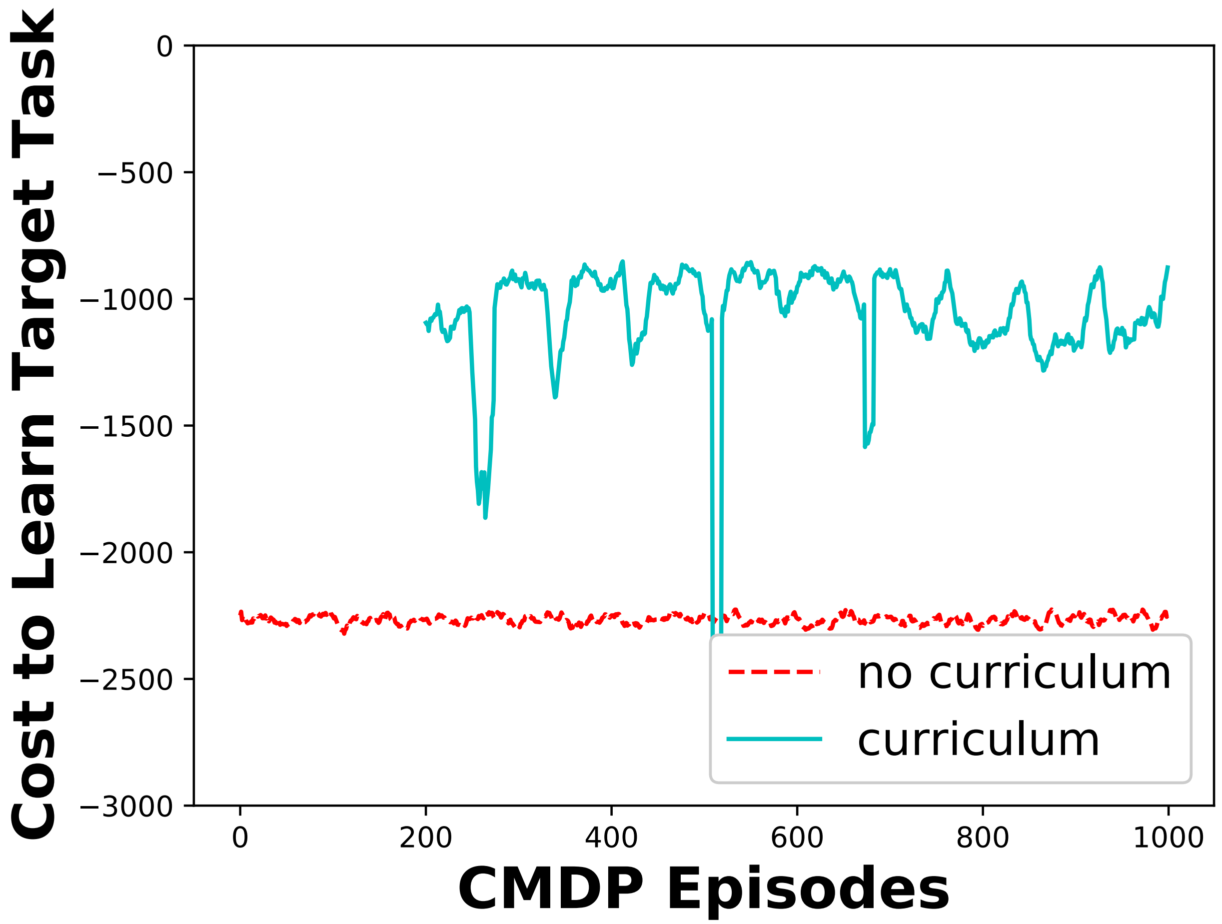
Training Set
Target Tasks



Test Set
Target Tasks

Extrapolation Results

- Train on tasks in training set for 200 episodes
- Evaluate on tasks in test set
- Learns to extrapolate to unseen types of tasks



Can reinforcement learning agents benefit from learning via a curriculum?

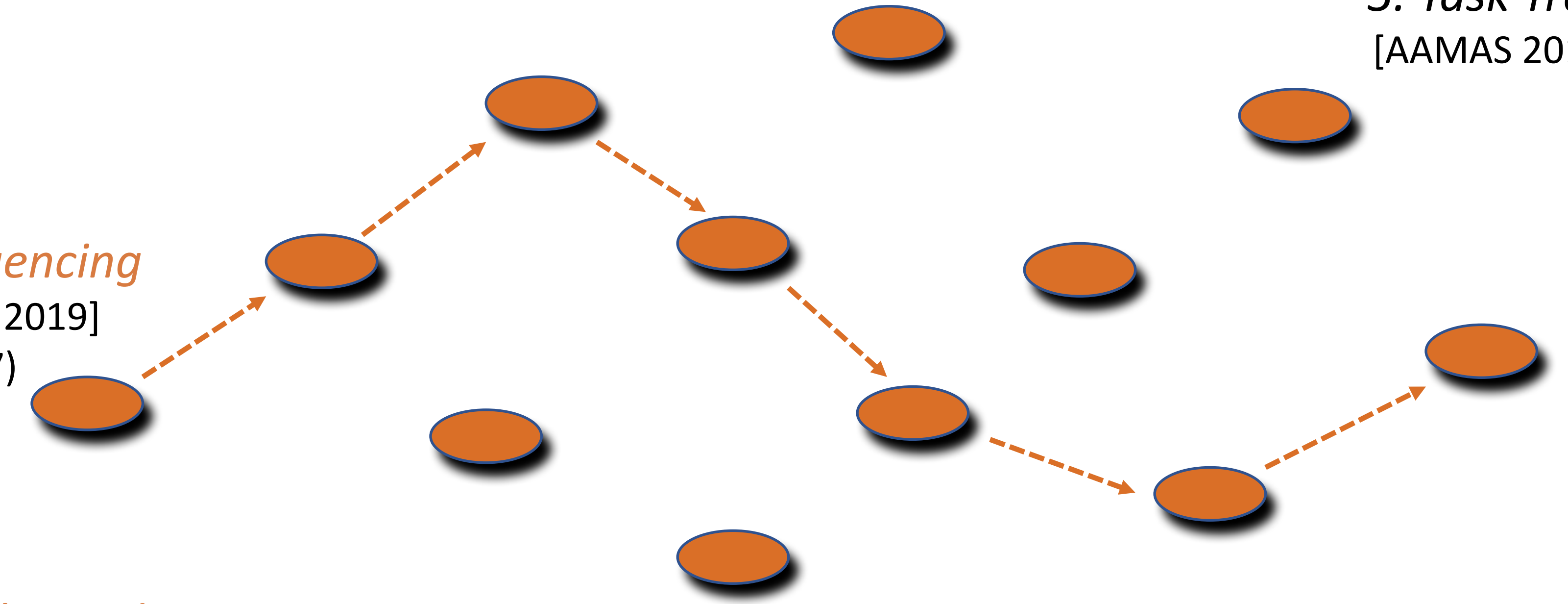
How can we automatically design one tailored to both the learning agent and task in question?

1. Problem Formalization
[JMLR 2020] (Chapter 3)

2. Task Generation
[AAMAS 2016] (Chapter 4)

3. Task Transferability
[AAMAS 2015] (Chapter 5)

4. Automatic Sequencing
[IJCAI 2017, AAMAS 2019]
(Chapters 6 & 7)



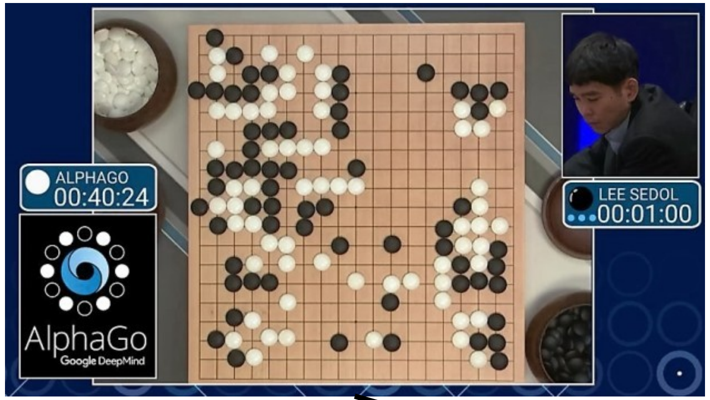
5. Curriculum Adaptation
[ICML WS 2020] (Chapter 8)

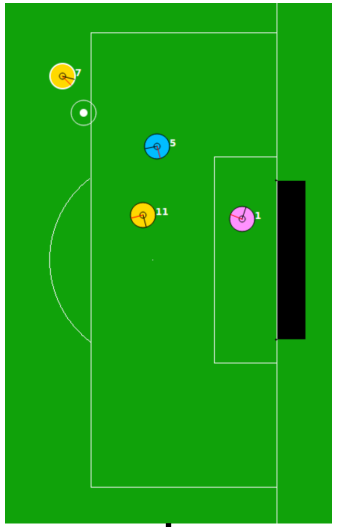
6. Taxonomy of CL
[JMLR 2020] (Chapter 9)

7. Empirical Evaluation

Summary

- Many popular recent RL successes have used CL as a key component
- Training on target task directly is too hard to make progress!



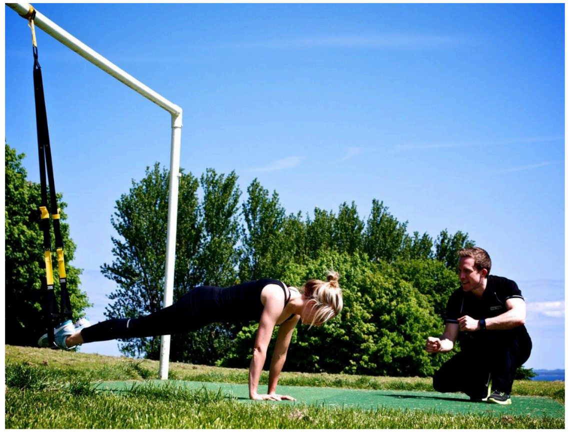


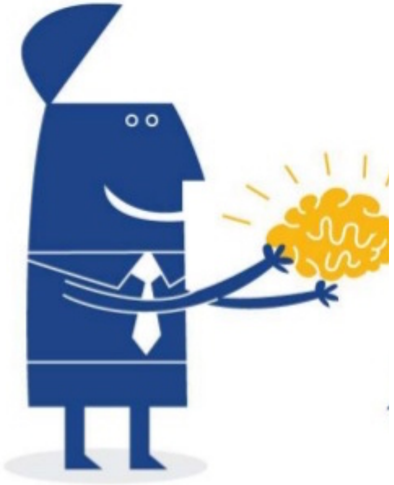
Sample Sequencing | Co-learning | Reward and Initial/Terminal State | No Restrictions

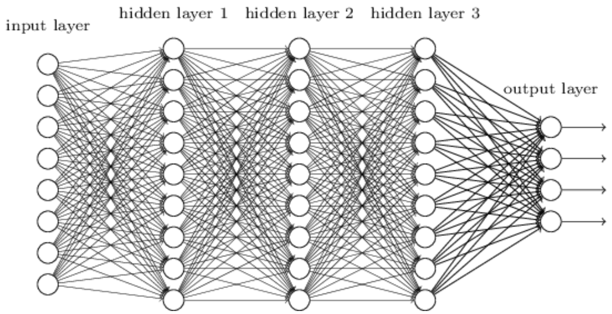
- I expect future RL successes could be a result of research in this area

Future Work

- Human Studies
- CMDP Extensions
- End-to-end Deep CL







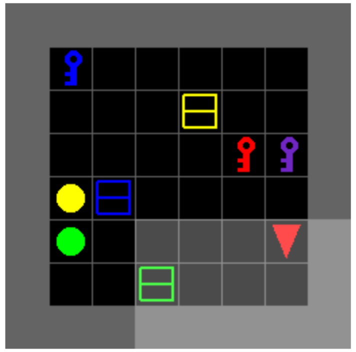
Human Studies

- This thesis: Inspired by human learning, design curricula for artificial agents
- Can we use these ideas to design curricula for humans in motor learning tasks?

- Directly learn a curriculum by replacing RL agent with human student
- Adapt curriculum learned by RL agent to humans

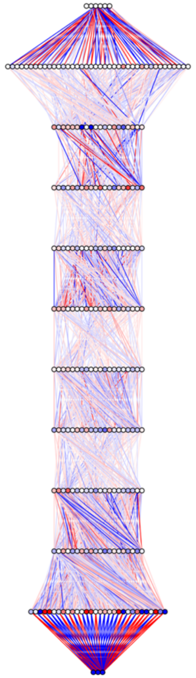
Ghonasgi et al. [IROS WS 2020]

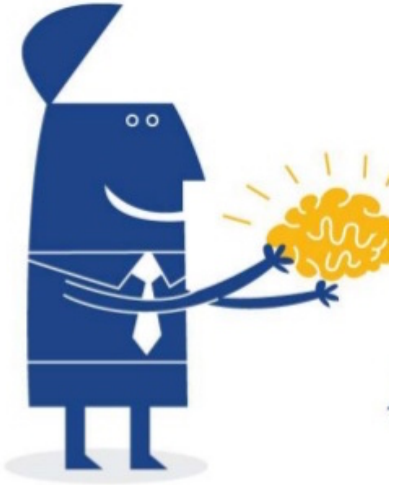
CMDP Extensions



“Pick up the red key”

- Extend to non-navigational tasks, where a more general representation for tasks is needed
  - Language-based interaction tasks

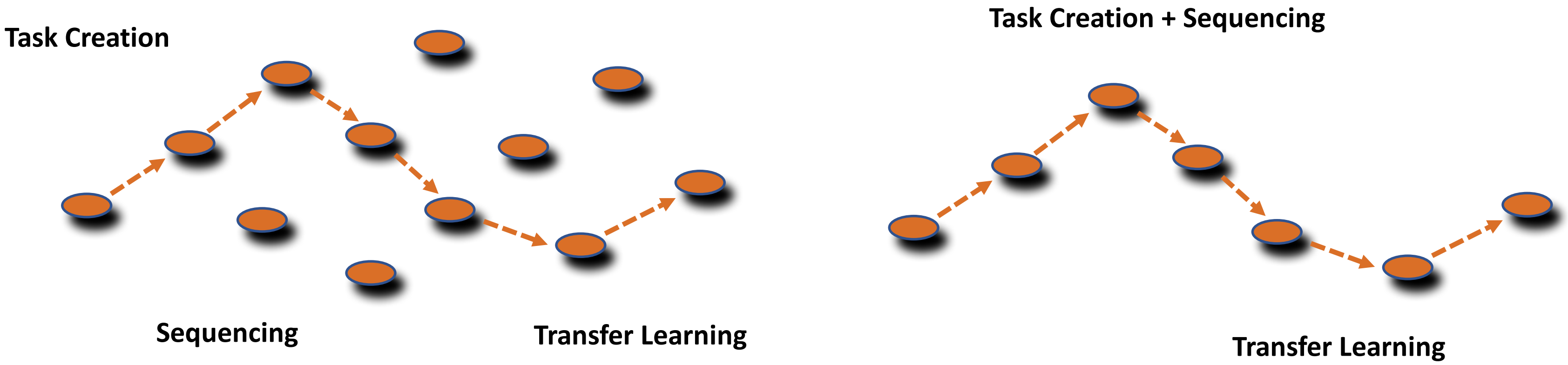




- Extend to settings where it is too expensive or unable to access the agent’s vector of parameters
  - Use a “test” to evaluate agent’s knowledge on a set of important states

Deep Curriculum Design

- Alternative model for curriculum design

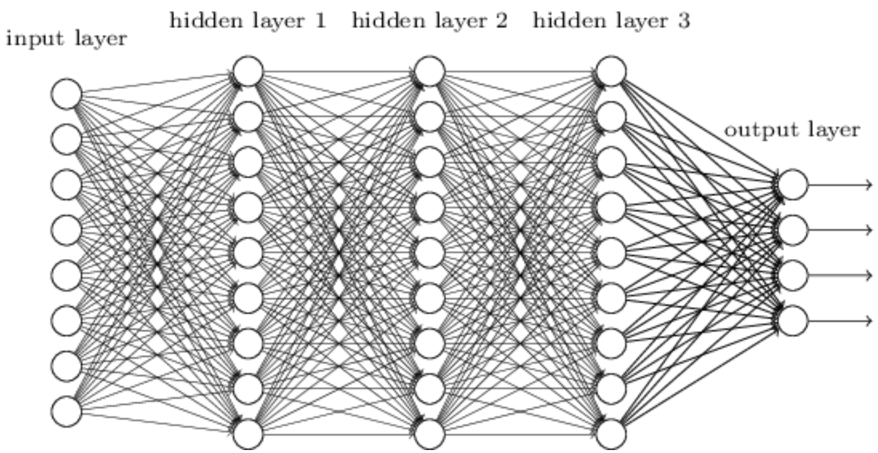


- Directly create next source task in curriculum given target task and agent's policy

Deep Curriculum Design

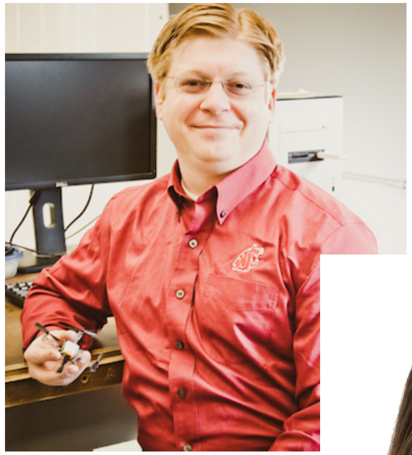
- Generate tasks using Generative Adversarial Networks (GANs)
- Existing work [Held et al. 2017] has shown GANs can create tasks that modify the reward function for intrinsic motivation
- More ambitious in that we want to modify the whole MDP

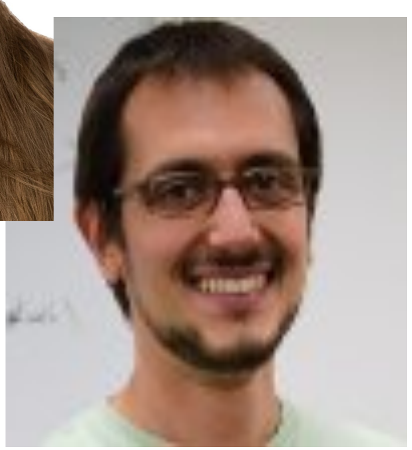
Agent Knowledge
Target Task



Next Source Task

Thank You!





LARG
Learning Agents Research Group
The University of Texas at Austin

Can reinforcement learning agents benefit from learning via a curriculum?

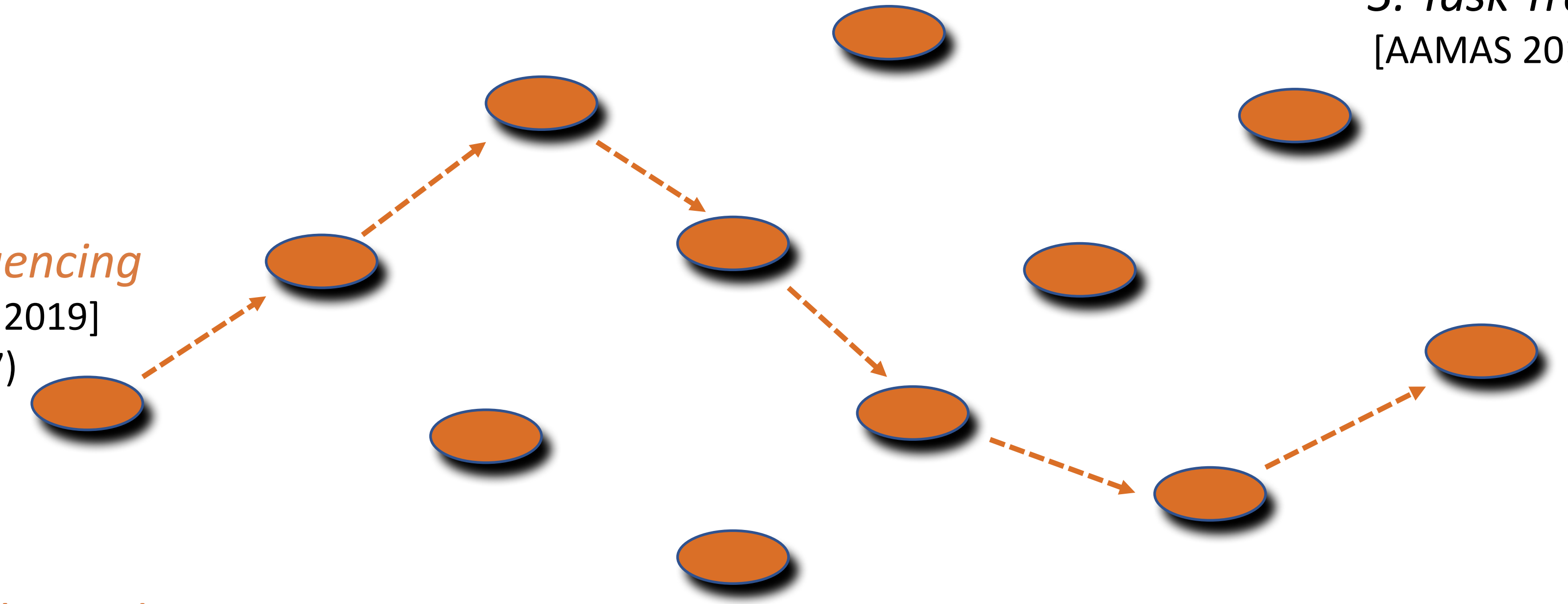
How can we automatically design one tailored to both the learning agent and task in question?

1. Problem Formalization
[JMLR 2020] (Chapter 3)

2. Task Generation
[AAMAS 2016] (Chapter 4)

3. Task Transferability
[AAMAS 2015] (Chapter 5)

4. Automatic Sequencing
[IJCAI 2017, AAMAS 2019]
(Chapters 6 & 7)



5. Curriculum Adaptation
[ICML WS 2020] (Chapter 8)

6. Taxonomy of CL
[JMLR 2020] (Chapter 9)

7. Empirical Evaluation