Autonomous Task Sequencing for Customized Curriculum Design in Reinforcement Learning

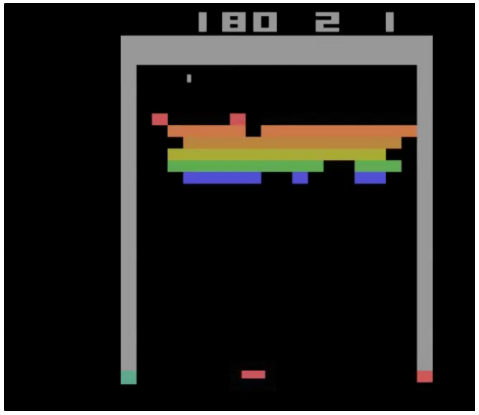
Sanmit Narvekar, Jivko Sinapov, and Peter Stone

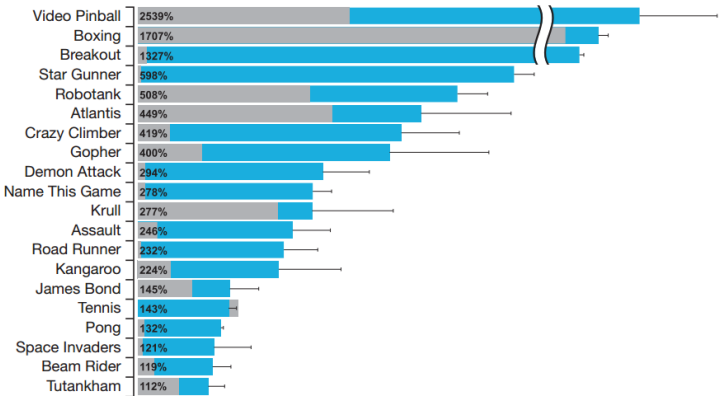
Department of Computer Science

University of Texas at Austin

{sanmit, jsinapov, pstone} @cs.utexas.edu

Successes of Reinforcement Learning



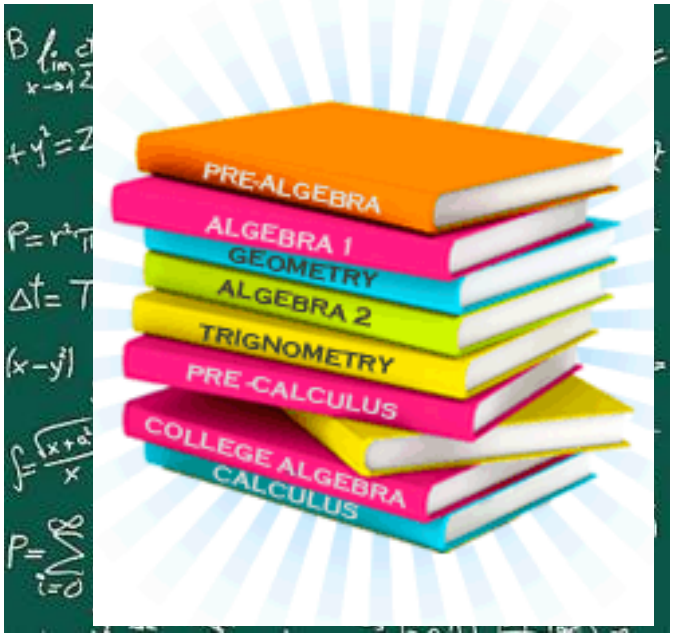


Approaching or passing human level performance

BUT

Can take *millions* of episodes! People learn this MUCH faster

People Learn via Curricula

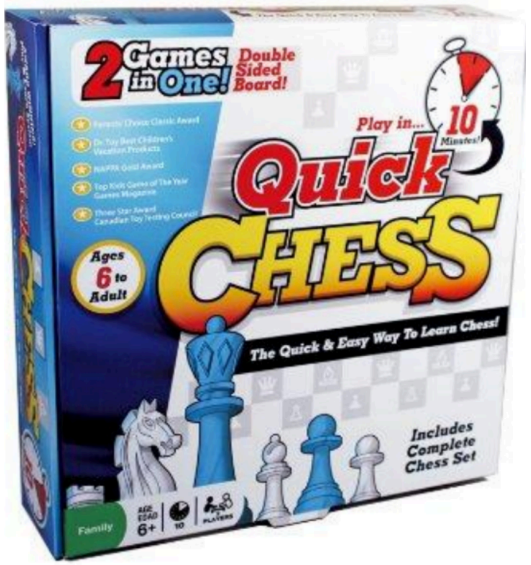


People are able to learn a lot of complex tasks very efficiently

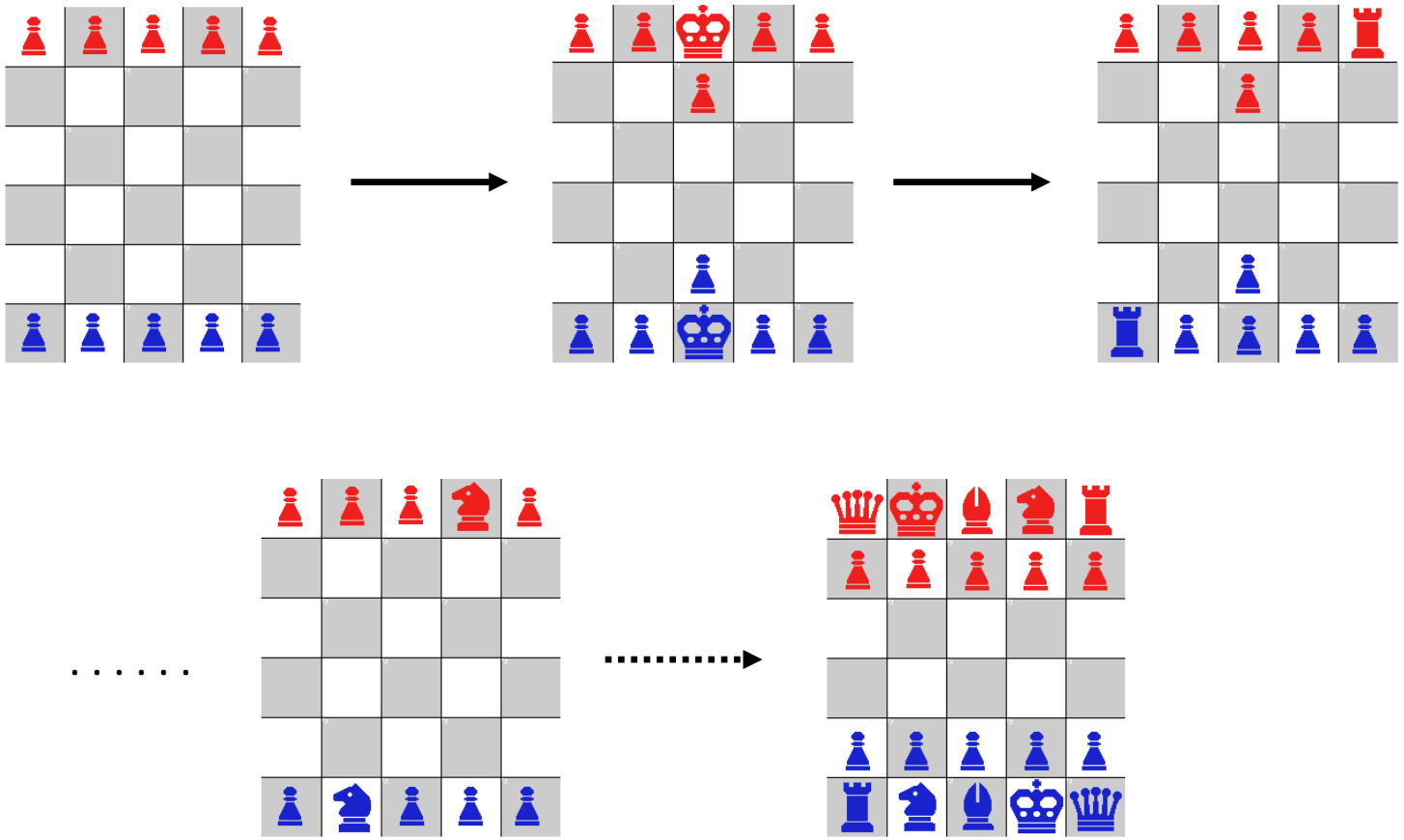
Example: Quick Chess

- Quickly learn the fundamentals of chess

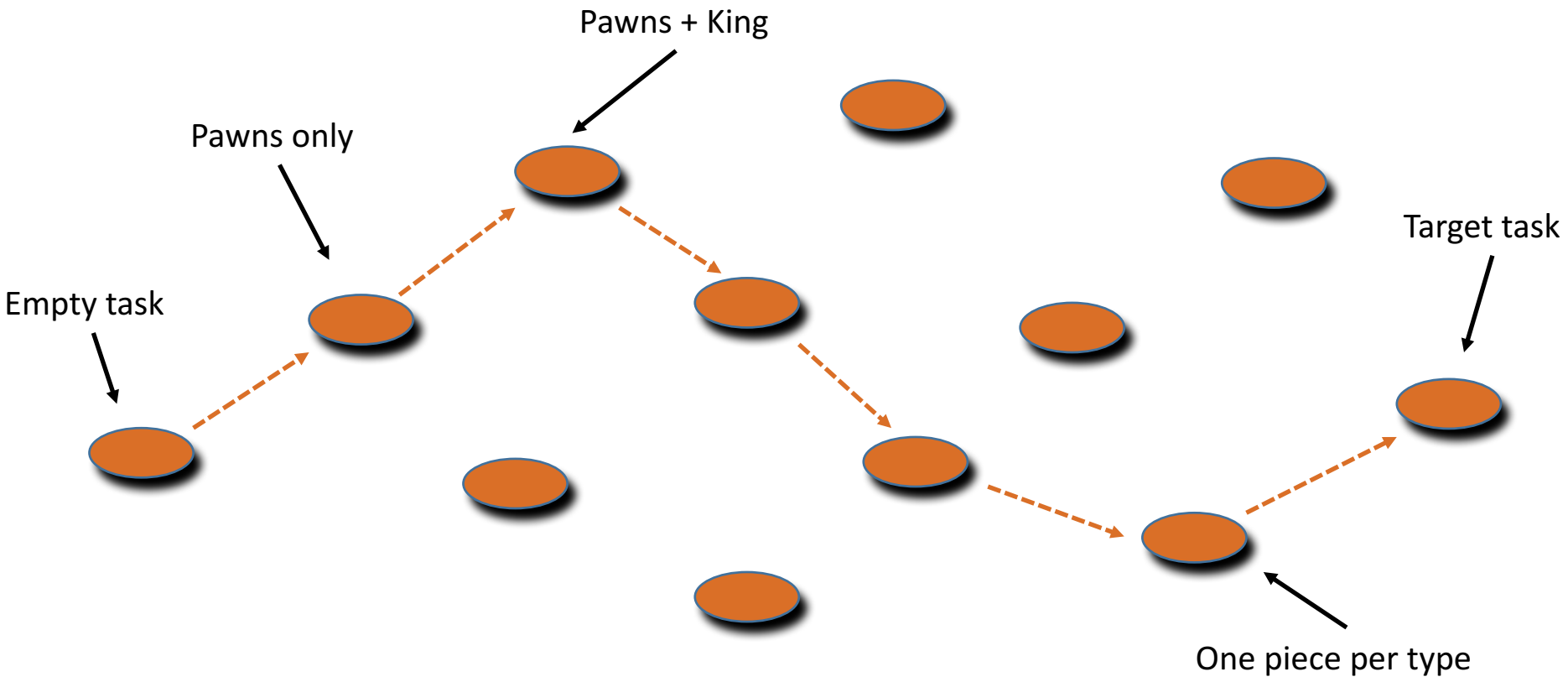
- 5 x 6 board
- Fewer pieces per type
- No castling
- No en-passant



Example: Quick Chess

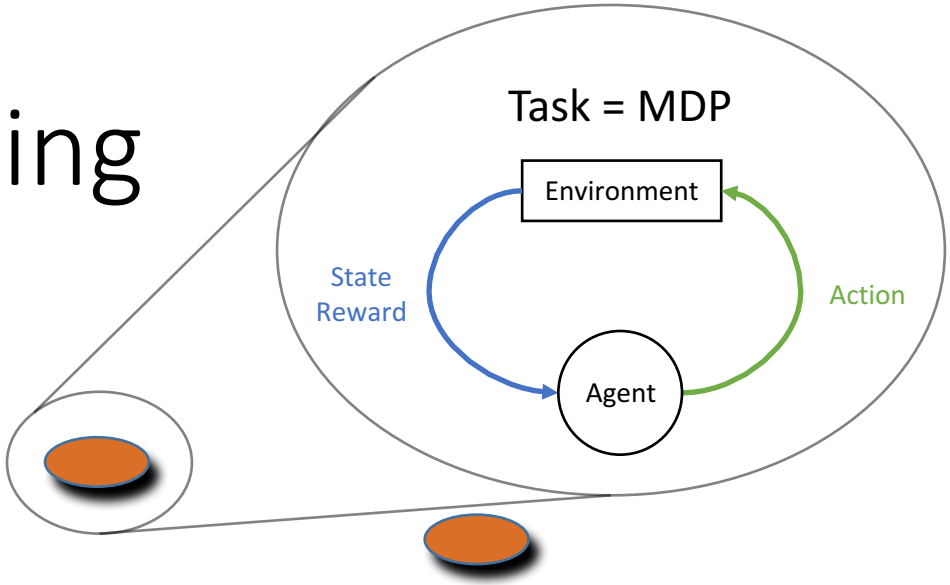


Task Space



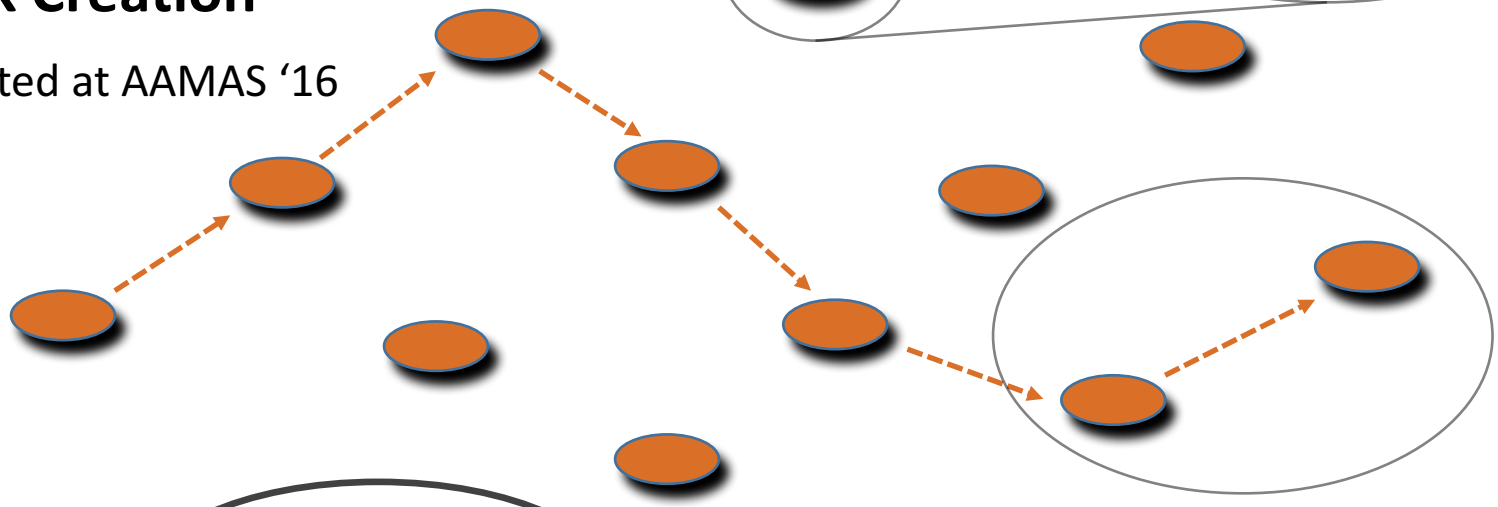
- Quick Chess is a curriculum designed for people
- We want to do something similar automatically for autonomous agents

Curriculum Learning



Task Creation

Presented at AAMAS ‘16



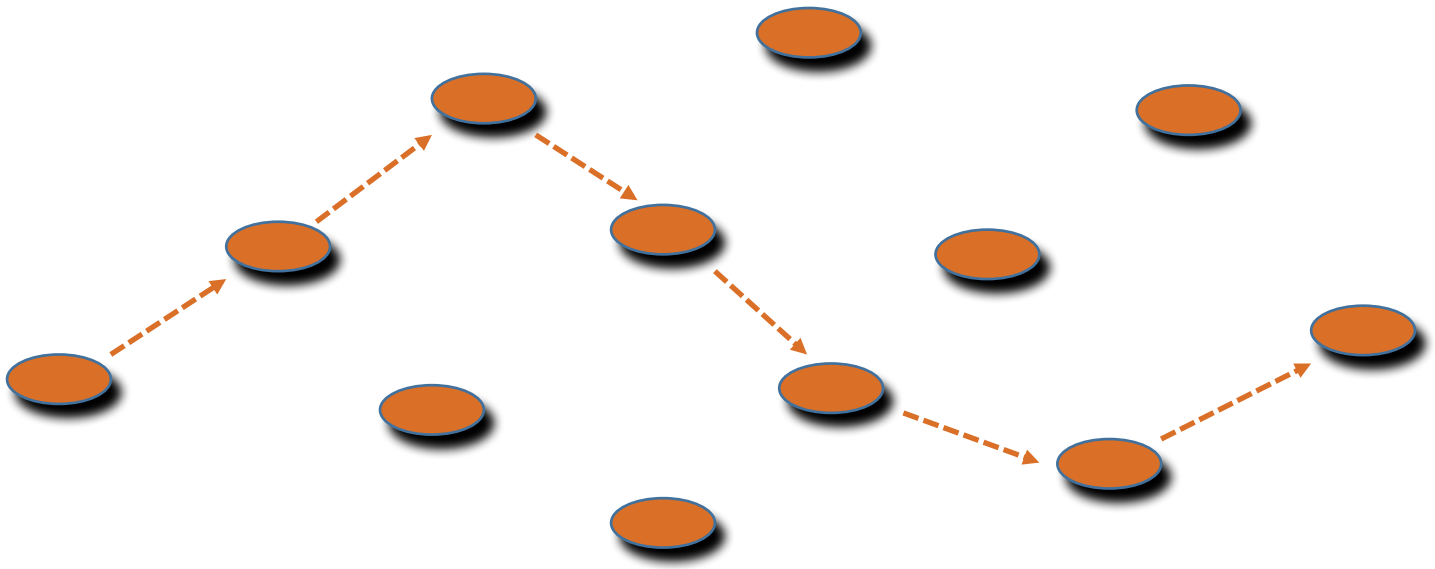
Sequencing

Transfer Learning

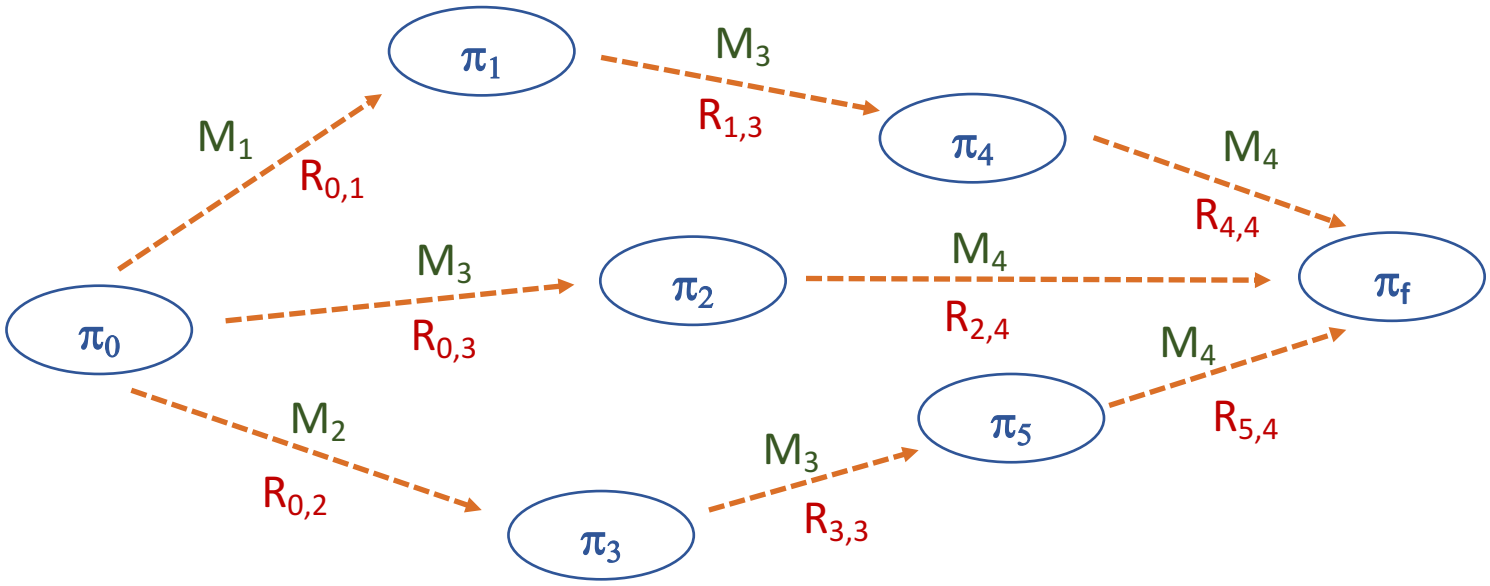
via Value Function Transfer

- Curriculum learning is a complex problem that ties task creation, sequencing, and transfer learning

Autonomous Task Sequencing

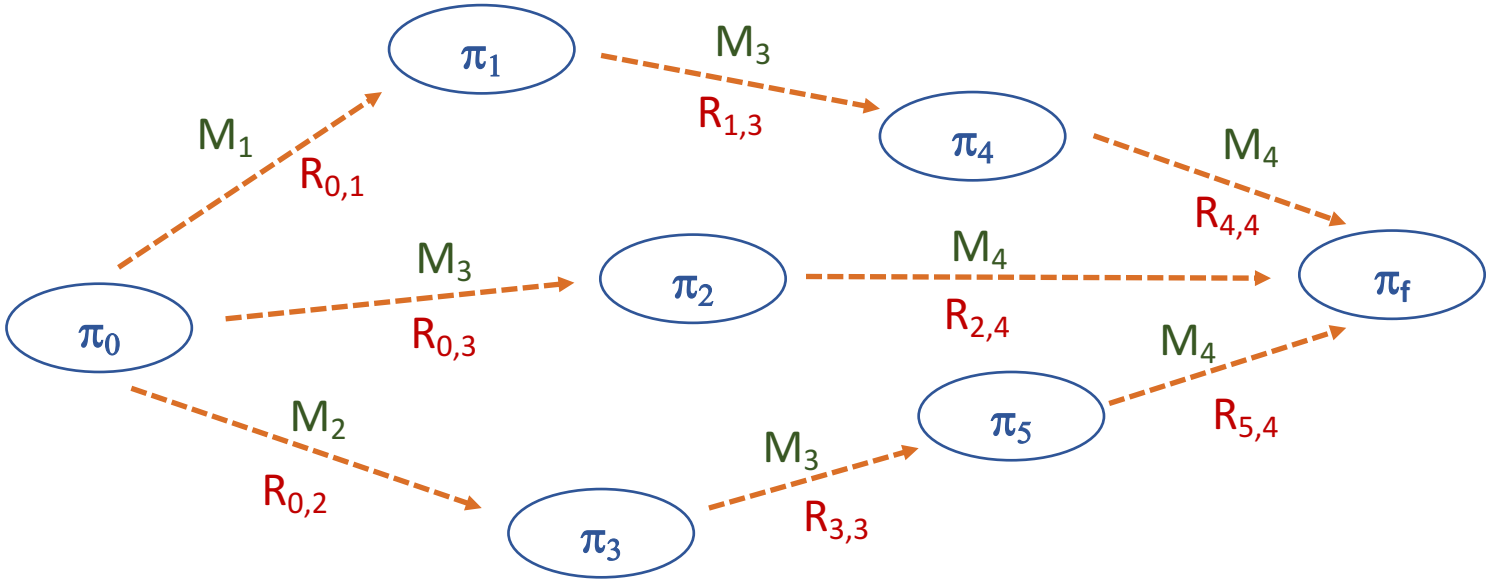


Sequencing as an MDP



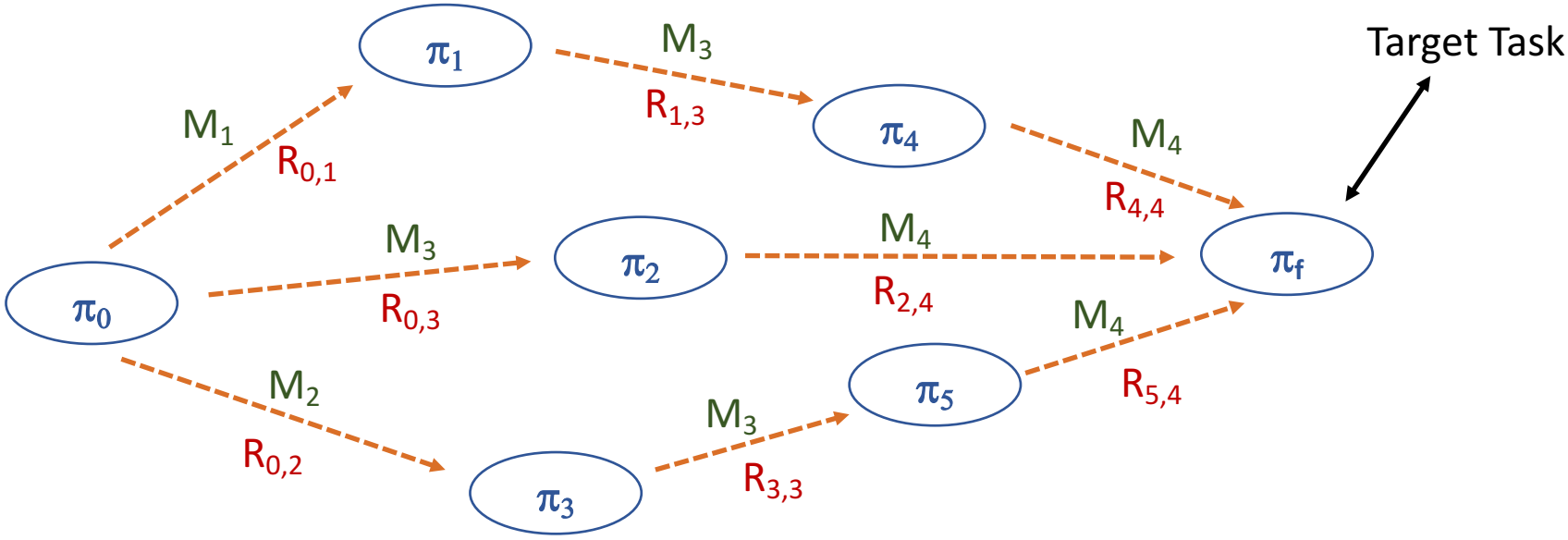
- **State space S^C**: All policies π_i an agent can represent
- **Action space A^C**: Different tasks M_j an agent can train on
- **Transition function $p^C(s^C,a^C)$**: Learning task a^C transforms an agent's policy s^C
- **Reward function $r^C(s^C,a^C)$**: Cost in time steps to learn task a^C given 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
- Learning full policy π^C can be difficult!
  - Taking an action requires solving a full task MDP
  - Transitions are not deterministic

Sequencing as an MDP



- Instead, find one trace/execution in CMDP of π^{C*}

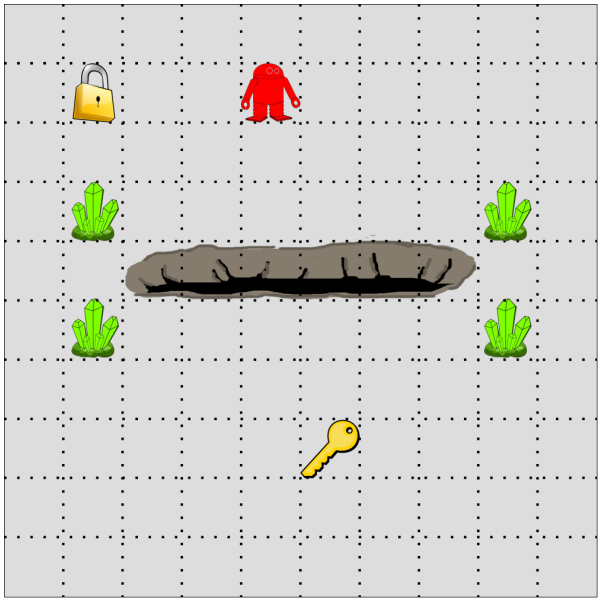
- **Main Idea**: Leverage fact that we know the target task and therefore what is relevant for the final state policy π_f to guide selection of tasks

Autonomous Sequencing

- Grid world domain

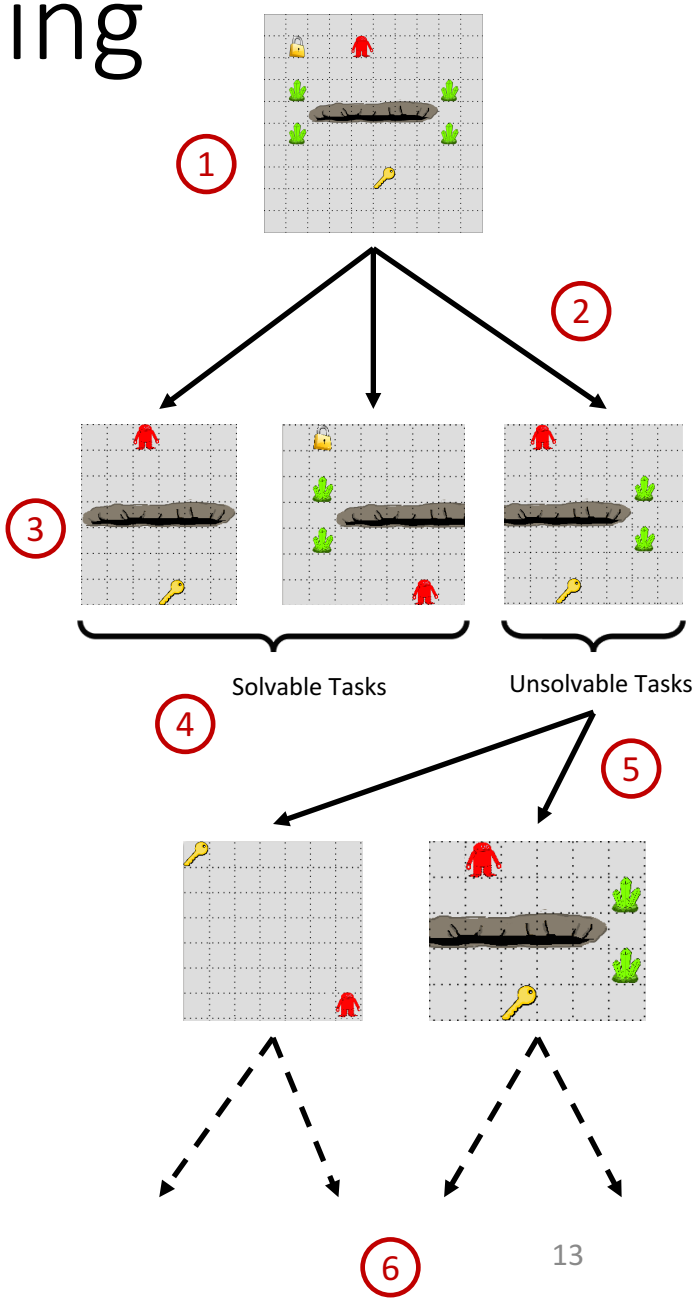
- Objectives
  - Navigate the world
  - Pick up keys
  - Unlock locks
  - Avoid pits

Target Task



Autonomous Sequencing

- Recursive algorithm (6 steps)
- Each iteration adds a source task to the curriculum
- This in turn updates the policy
- Terminates when performance on target task greater than desired performance threshold

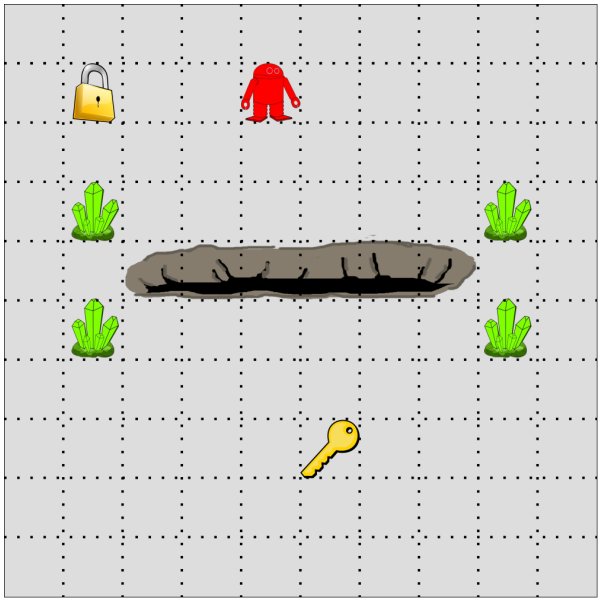


Autonomous Sequencing

Step 1

- Assume learning budget β
- Attempt to solve target task directly in β steps. Save samples
- Solvable?
  - Target task easy to learn
  - Started with policy that made it easy to learn. Done
- Goal: incrementally learn subtasks to build a policy that can learn the target task

Target Task ①



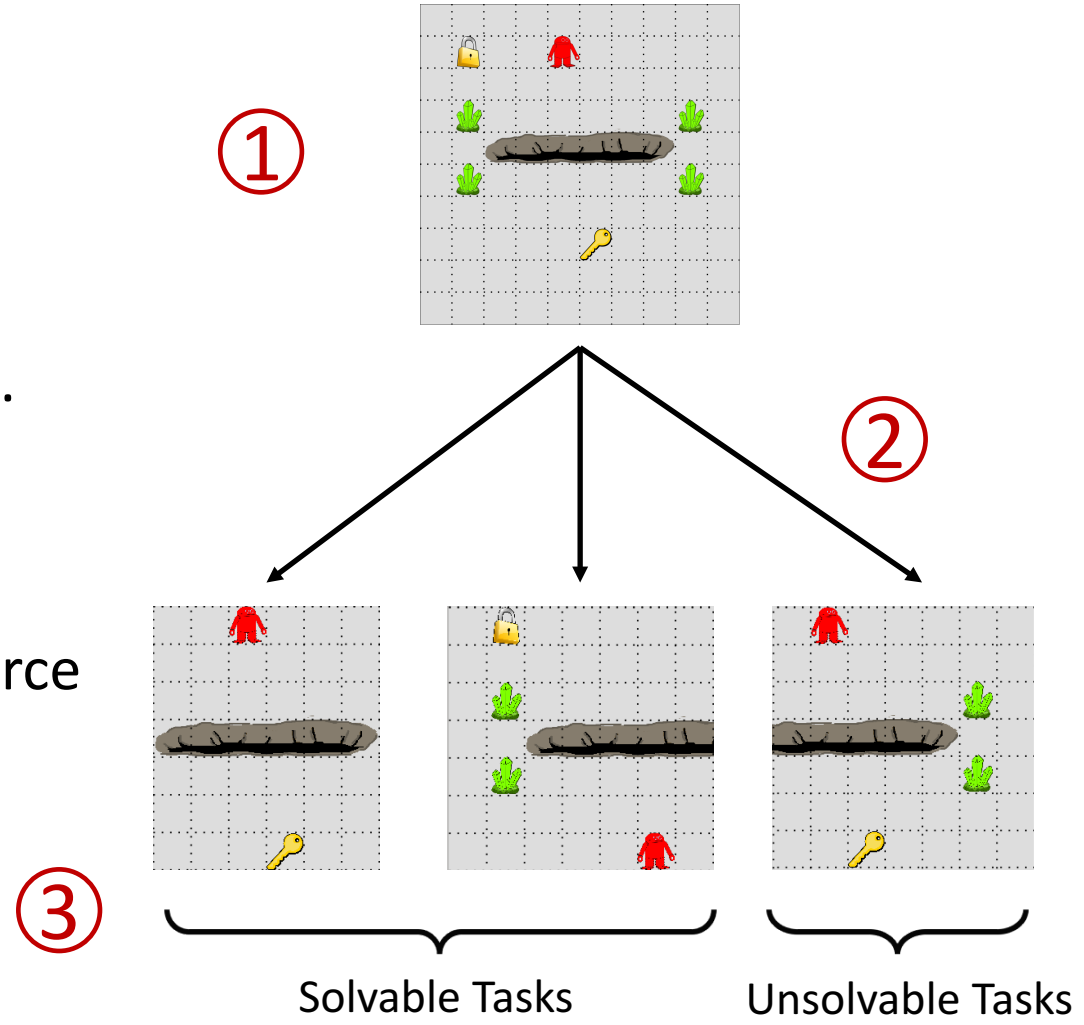
Autonomous Sequencing

Step 2

- Could not solve target
- Create source tasks using methods from AAMAS '16.

Step 3

- Attempt to solve each source in β steps
- Partition sources into solvable / unsolvable

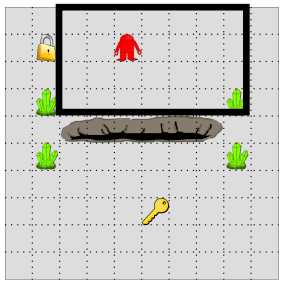


Autonomous Sequencing

Step 4

- If solvable tasks exist, select the one that updates the policy the most on samples drawn from the target task

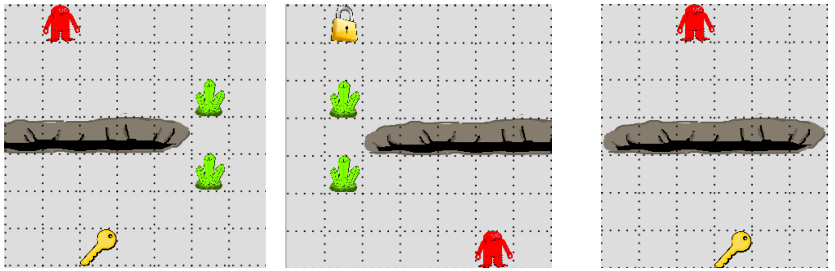
- Assumption
  - Source tasks that can be solved have policies that are relevant to the target task
  - Don't provide negative transfer



Initial Policy π_0

$[s_1, s_2, s_3, s_4 \ldots s_\beta]$

[➡, ⬇, ⬅, U … P]



Solvable Tasks

π_1 π_2

[➡, ➡, ➡, ⬇ … P] [➡, ⬇, ⬅, U … P]

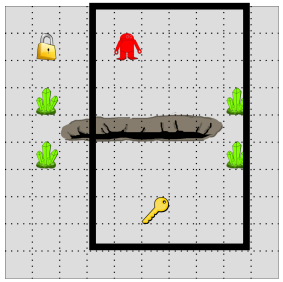
✔

4

Autonomous Sequencing

Step 4 (cont.)

- Add source task to curriculum
- Return to Step 1



New Policy π_1

$[s_1, s_2, s_3, s_4 \ldots s_\beta]$

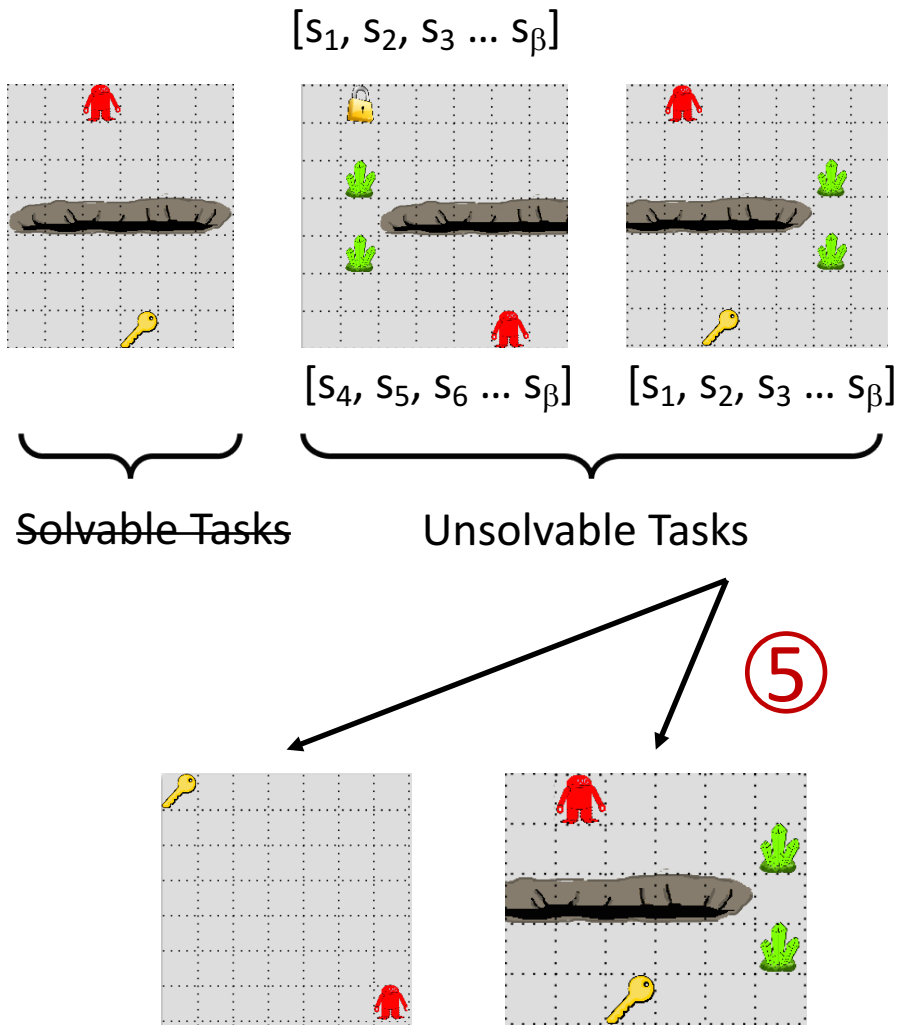
[➡, ➡, ⬇, P … P]

- (Re-evaluate on target task)
- Policy has changed, so we will get a new set of samples
- Samples biased towards agent's current set of experiences
- This in turn guides selection of source tasks

Autonomous Sequencing

Step 5

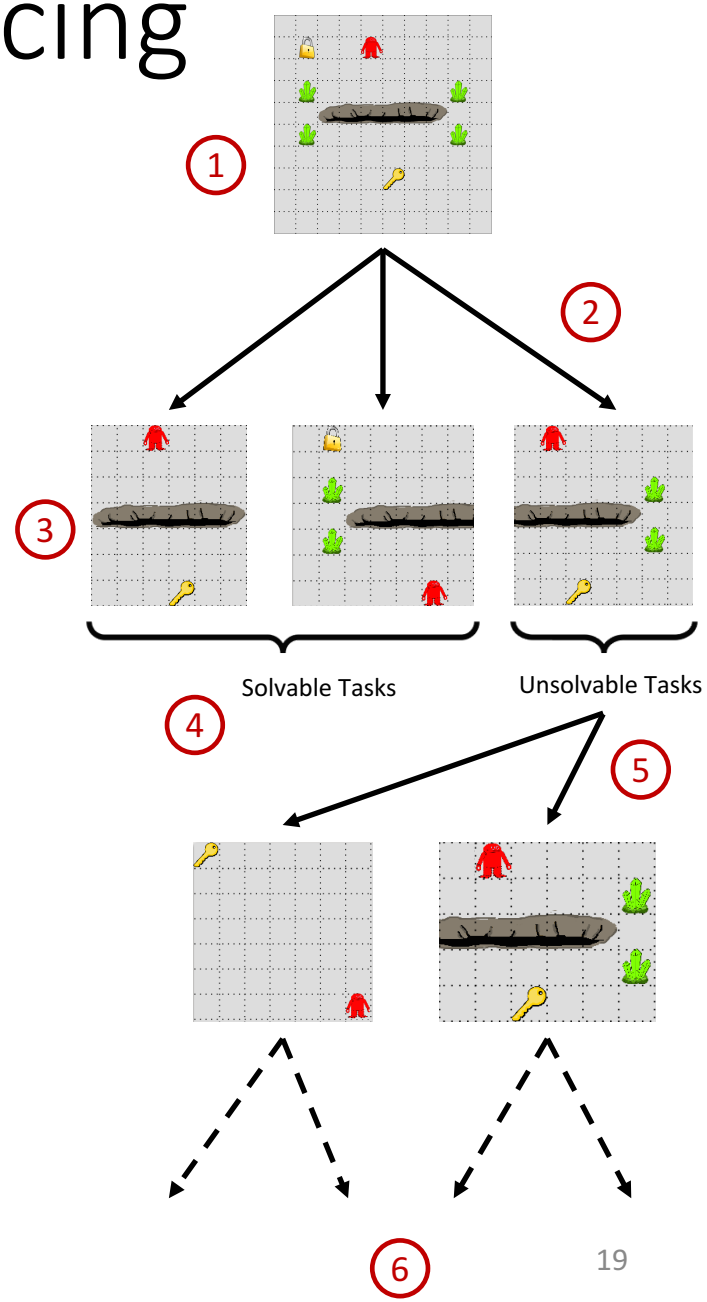
- No sources solvable
- Sort tasks by sample relevance
  - Compare states experienced in target task with those in experienced in sources
- Recursively create sub-source tasks
- Return to Step 2 with the current source task as the target task



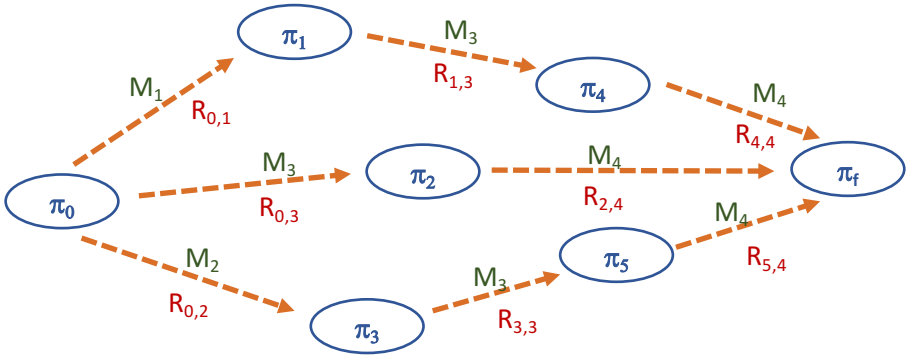
Autonomous Sequencing

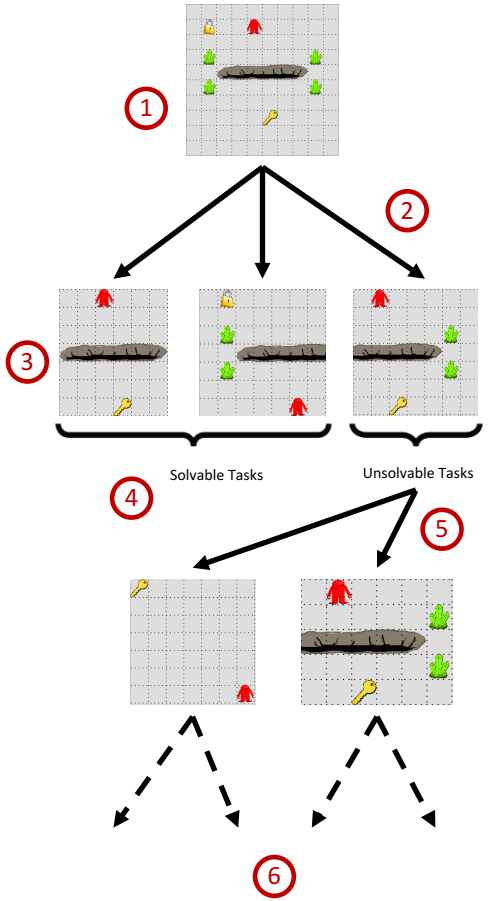
Step 6

- No sources usable after exhausting the tree
- Increase budget, return to Step 1
- Learning can be cached, so agent can pick up where it left off



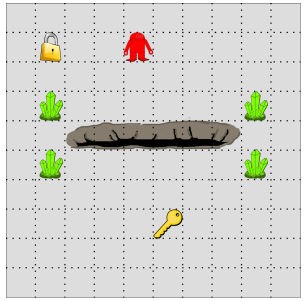
Connection to CMDPs





- An optimal path in CMDP is one that reaches π_f with least cost
- Selection in Step 4 picks tasks that update most towards π_f
- Learning budget minimizes cost
- Algorithm behaves greedily to balance updates and cost

Experimental Setup

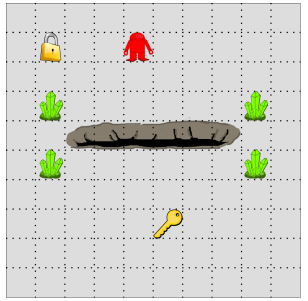


- Grid world domain presented previously

Create multiple agents

- Multiple agents shows the algorithm is not dependent on implementation of RL agent
- Evaluate whether different agents benefit from individualized curricula

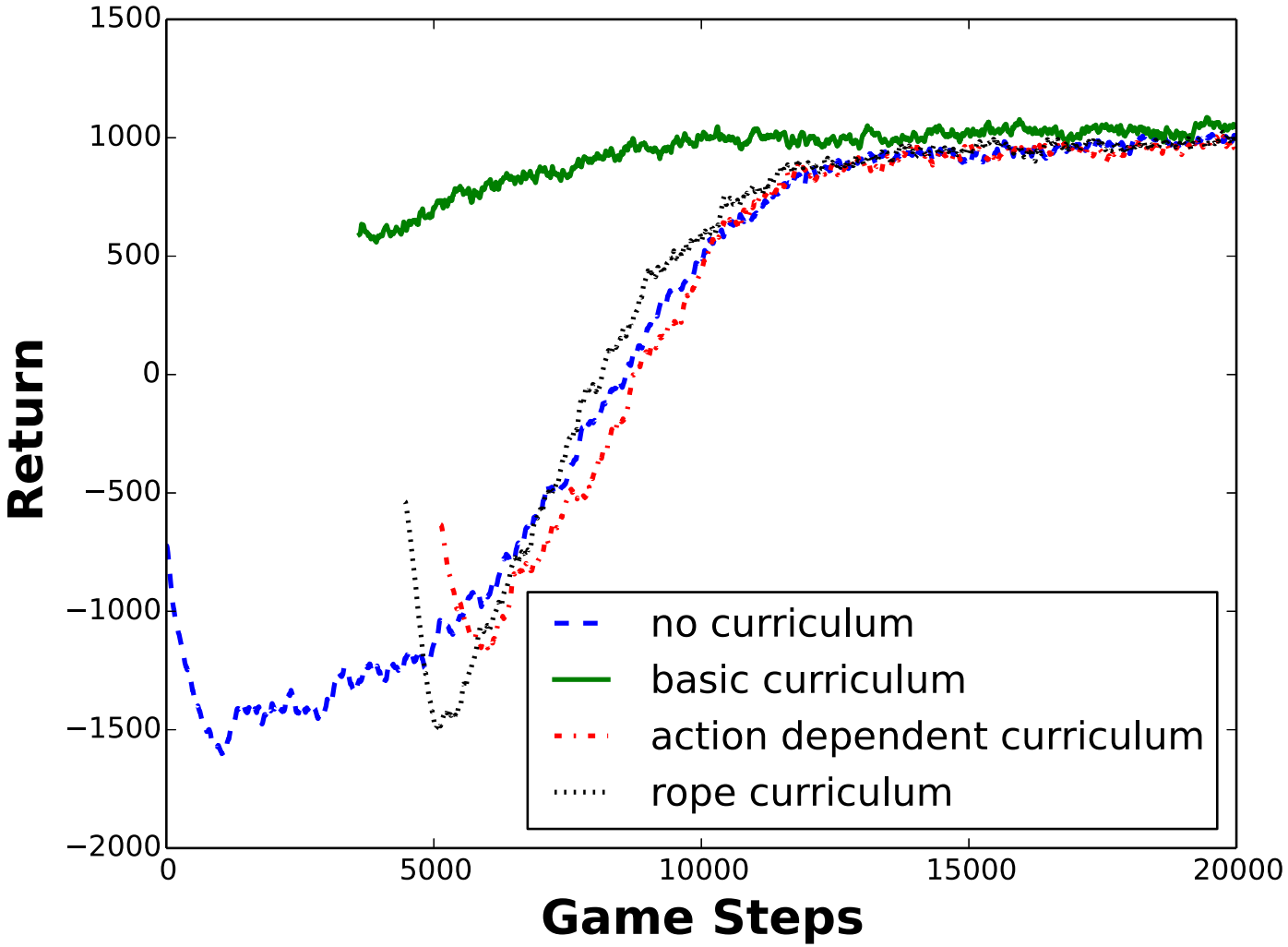
Experimental Setup



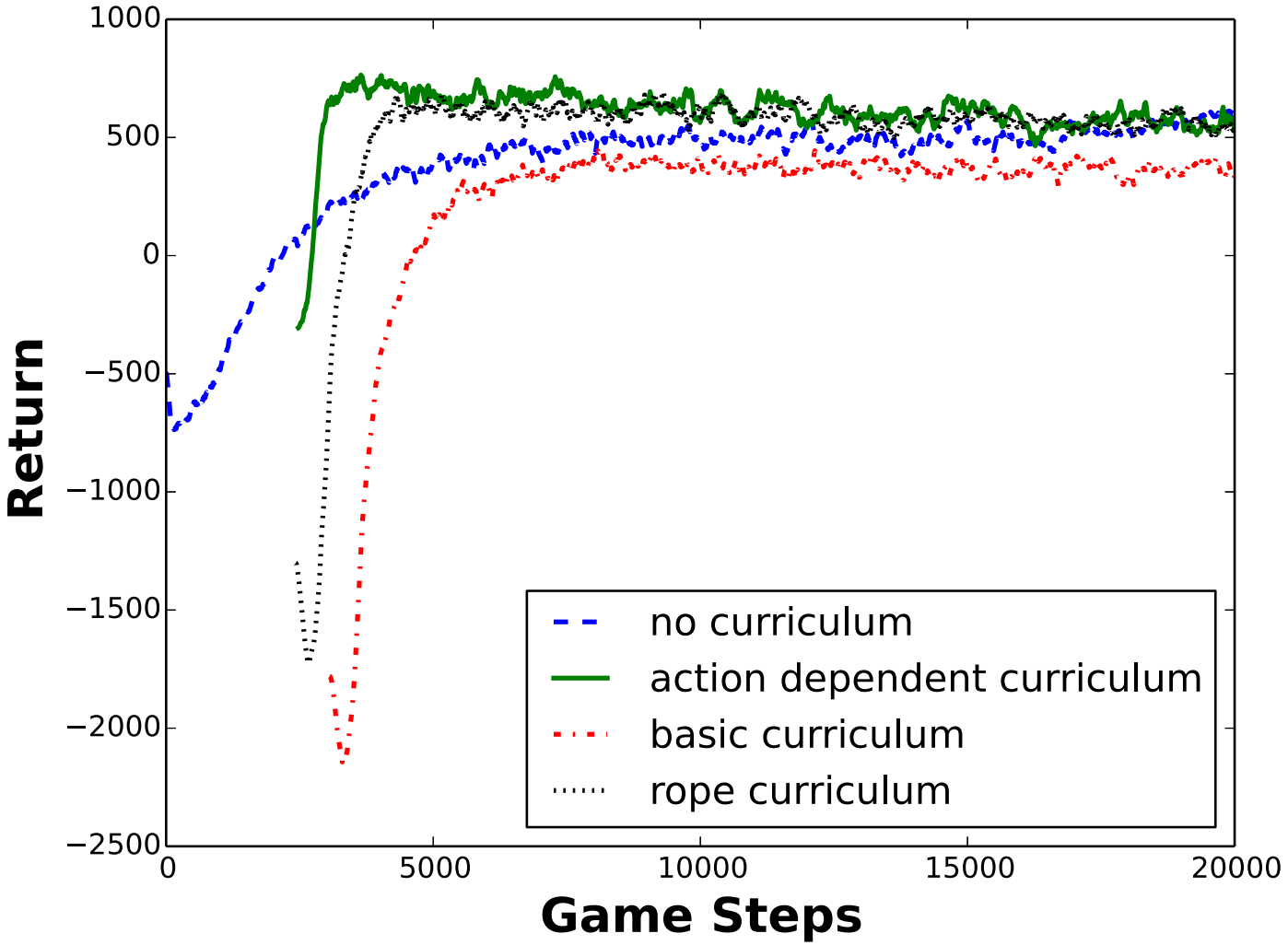
Agent Types

- 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

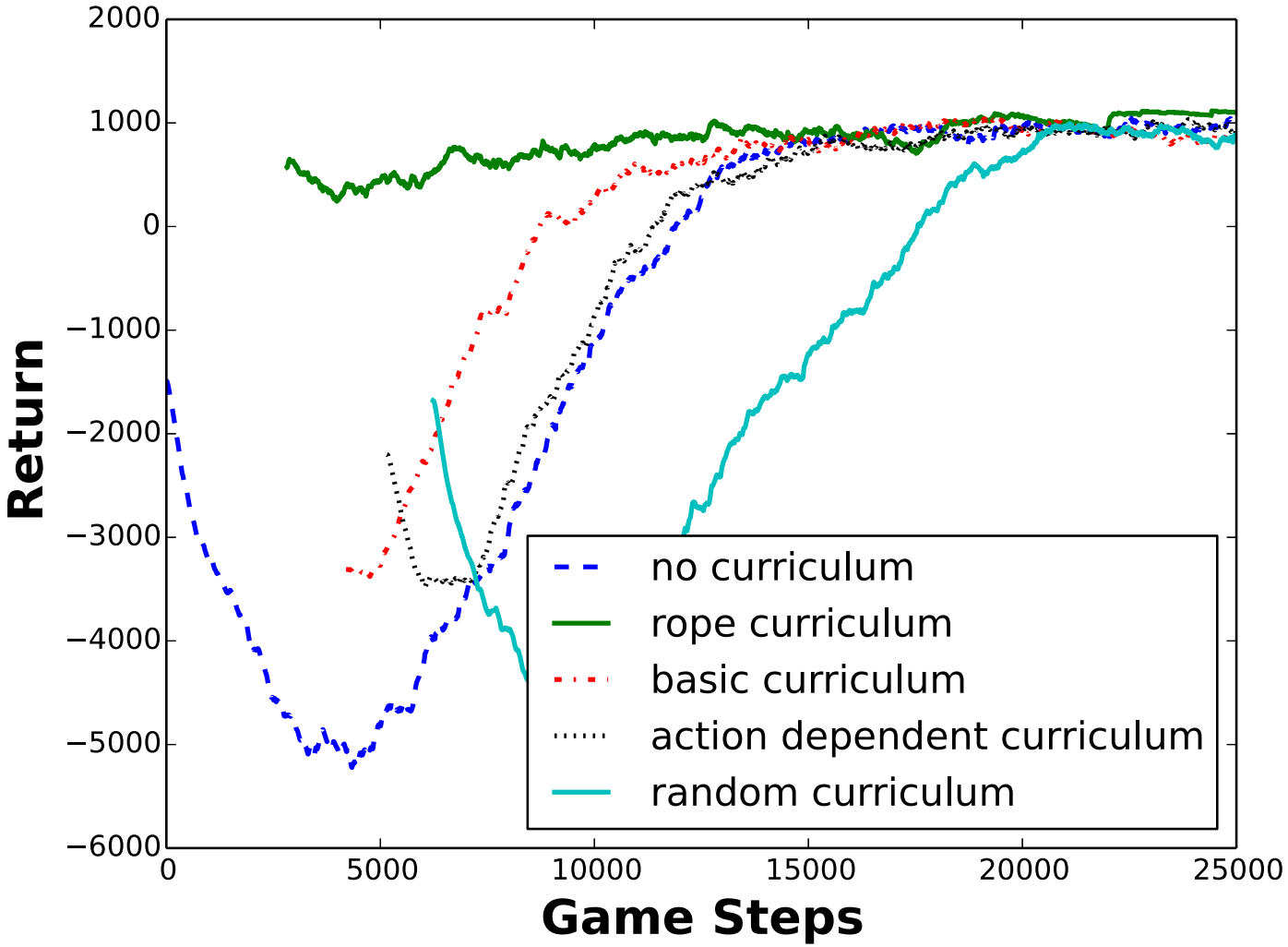
Basic Agent Results



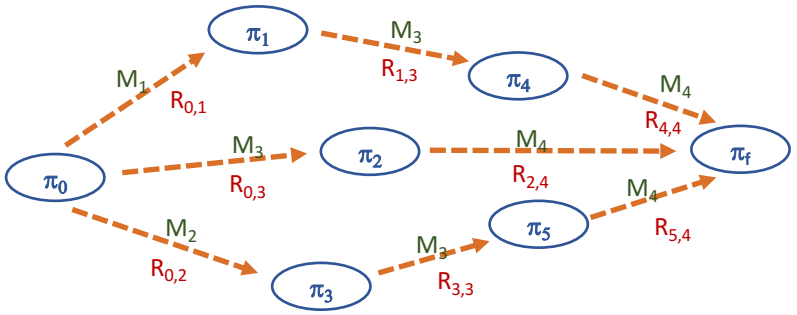
Action-Dependent Agent Results



Rope Agent Results



Summary



- Presented a novel formulation of curriculum generation as an MDP
- Proposed an algorithm to approximate a trace in this MDP
- Demonstrated method proposed can create curricula tailored to sensing and action capabilities of agents