Multi-Agent Systems Projects

Andrea Longo

AI Teacher using Reinforcement Learning

Setting

- **Agent**: Teacher
- **Environment**: Student
- **Goal**: Learn a policy that accelerate student's mastery

RL Formulation : Markov property

$$P(s_{t+1}=x|s_t) = P(s_{t+1}=x|s_t,s_{t-1},\dots,s_0)$$

"Student's current mastery levels fully summarize their future learning potential, independently of their past performance history"

RL Formulation : Core components

States

A **continuous state vector** $S \in [0, 1]^2$ representing the **student's mastery** in algebra and arithmetic

Actions

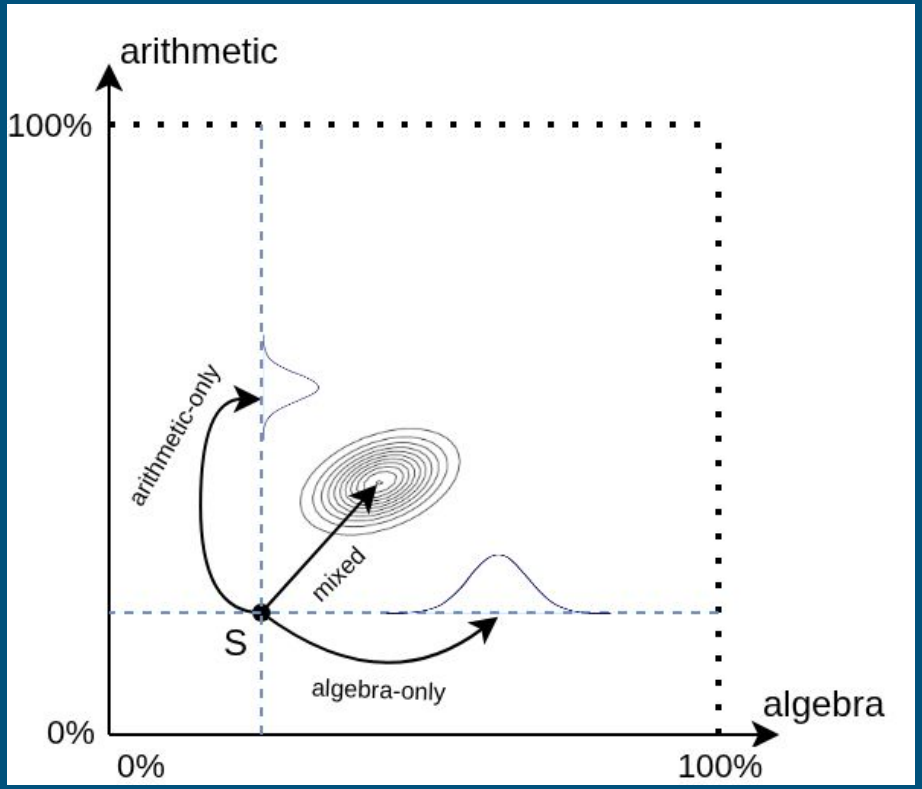
The **type of assignment** provided by the teacher, namely algebra-only, arithmetic-only, or mixed

Rewards

Either the sum of the **student's skill improvements** or the **reaching of a specific threshold** in both skills

Transitions

The student's skills evolve probabilistically based on their **current mastery levels** and the **assigned task**



Transition dynamics representation

RL Formulation : Knowledge & Observability

"Student learning dynamics are **unknown to the teacher** and cannot be explicitly modeled"

→ **Model-free Reinforcement Learning**

"Student mastery is **not directly observable**, so it must be inferred from assignment performance"

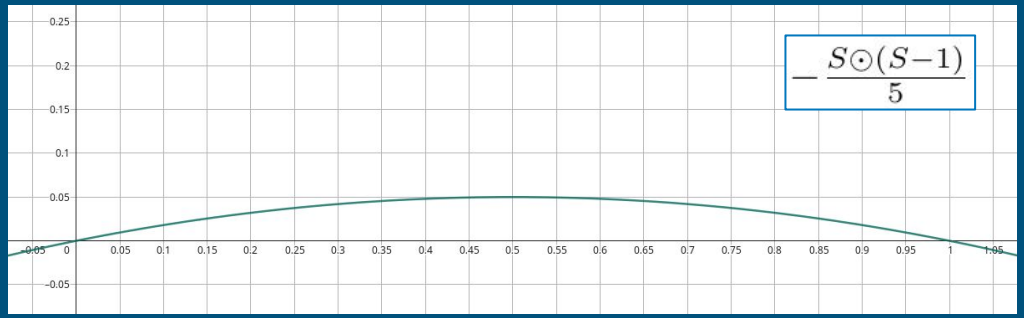
→ **Partial observability** (POMDP)

But assuming that the AI Teacher generates assignments that **accurately assess student mastery**, its possible to approximate the problem as a fully observable MDP

Simulation Environment

$$S' = S + \Delta(S, a, d),$$

$$d \sim \mathcal{U}([1, 1.5]^2)$$



$$\Delta(S, a, d) \sim \mathcal{N}\left(-\frac{S \odot (S-1)}{5} \odot d \odot A(a), \sigma^2 diag(A(a))\right), \text{ with } \sigma = 0.01, A(a) = \begin{cases} (1,0) & \text{if } a \text{ is algebra-only} \\ (0,1) & \text{if } a \text{ is arithmetic-only} \\ \left(\frac{1}{2}, \frac{1}{2}\right) & \text{if } a \text{ is mixed.} \end{cases}$$

Implementation

Configurations

Action-selection strategies

- **ε-Greedy**
- **Greedy**

Reward settings

- **Dense-reward**

  $R = \Delta(S, a, d)$

- **Sparse-reward**

  $R = 100$ if $S >= [0.9, 0.9]$ else 0

Algorithm = **Q-Learning**

Discretization = 100 bins

Goal = **[0.9, 0.9] mastery**

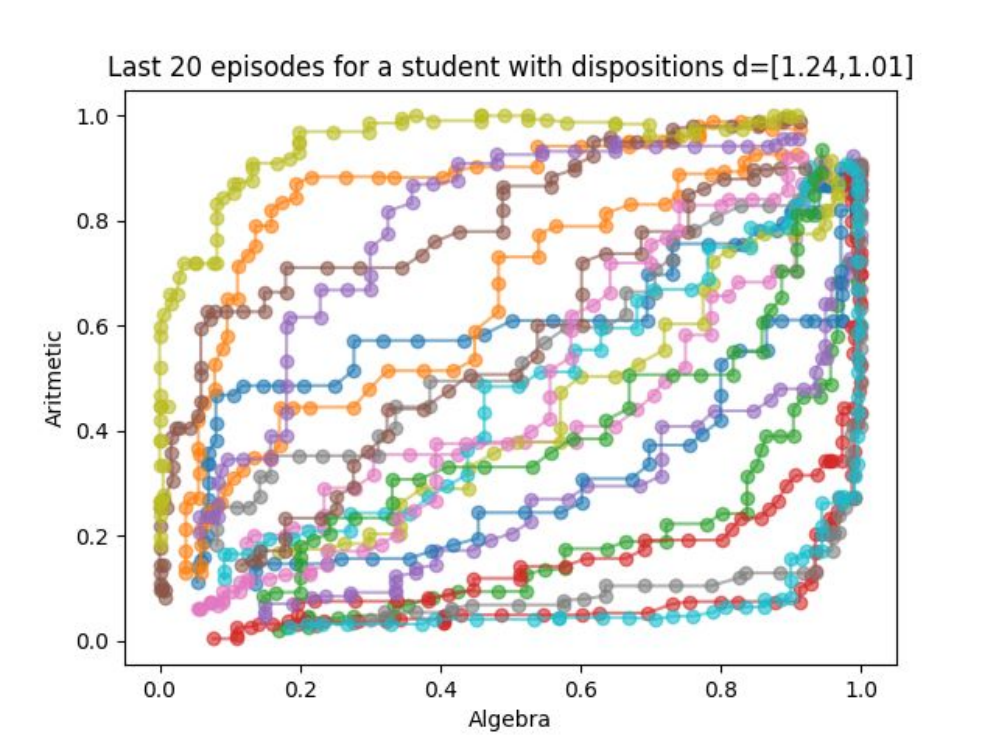
Samples per configuration = 200

Episodes = 1000

Learning Rate (α) = 0.1

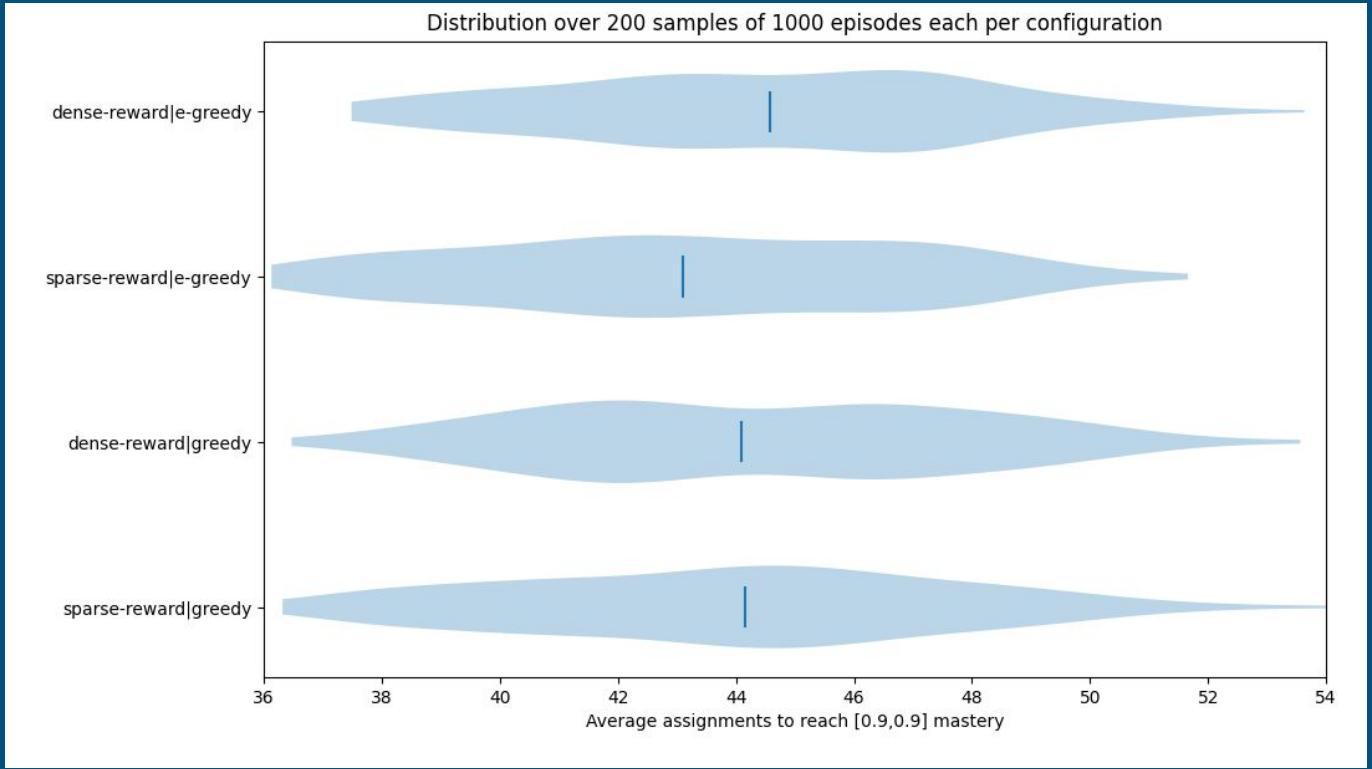
Exploration Rate (ε) = 0.1

Discount Factor (γ) = 0.99

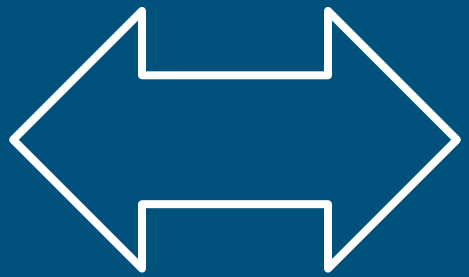


Paths with sparse-reward and ϵ-greedy

Results

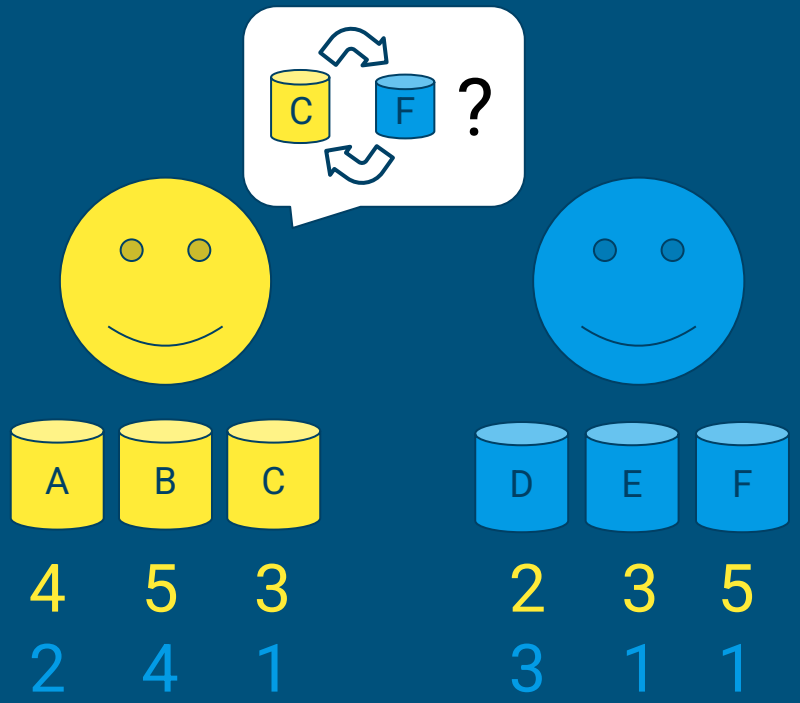

Distribution over 200 samples of 1000 episodes each per configuration
dense-reward|e-greedy
sparse-reward|e-greedy
dense-reward|greedy
sparse-reward|greedy
36
38
40
42
44
46
48
50
52
54
Average assignments to reach [0.9,0.9] mastery

Automated Mechanism Design for Bartering



Setting

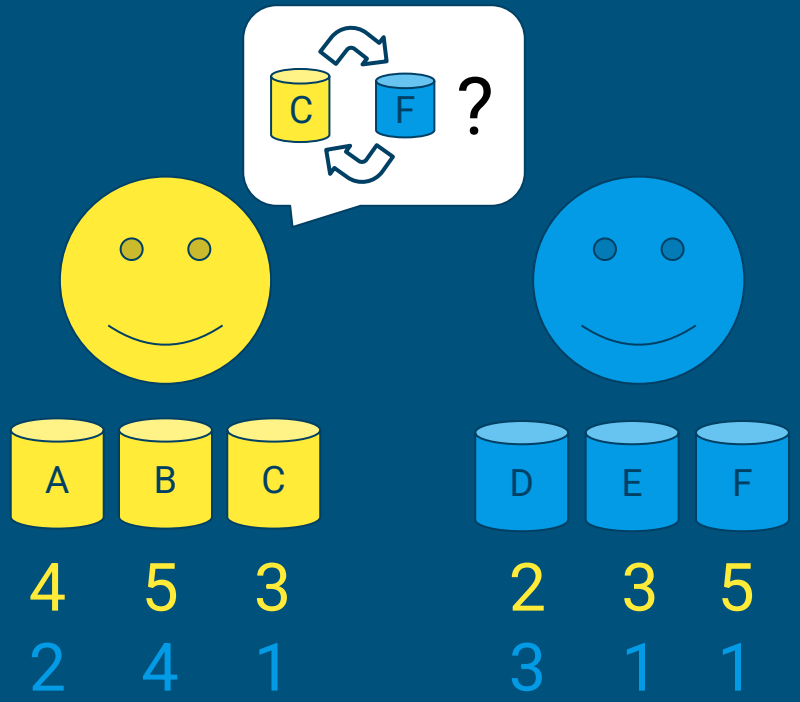
- Mechanism Designer (Agent 1)
- Player (Agent 2)
- **Utility**: Subjective difference in value between exchanged goods
- **Goal**: Find a **direct**, **truthful** and **individually rational** mechanism that maximizes Agent 1 utility



Model

- **Outcomes** are the disposition of the belongings at the end of the exchange ($O = \{0, 1\}^{2N}$):
  [1,1,1,0,0,0] ⟶ Initial disposition
  [1,1,0,0,0,1] ⟶ Trade of good C with F

- **Types** are the possible individual valuation of Agent 2 of each good ($\Theta = \{1,2,3,4,5\}^{2N}$):
  [2,4,1,3,1,1]



Constraints and Objective

The mechanism (A, M: A⟶O) should be:

- **Direct**: "Agent 2 action is to declare a valuation for each good" (A=Θ)
- **Truthful**: "Agent 2 should benefit in reporting its own valuation instead of lying"

$$\forall \theta, \underline{\theta} \in \Theta \qquad utility(\theta, M(\theta)) \geq utility(\theta, M(\underline{\theta}))$$

- **Individual Rationality**: "Agent 2 should never incurs a loss by participating"

$$\forall \theta \in \Theta \qquad utility(\theta, M(\theta)) \geq 0$$

Under those constraints Agent 1 maximizes its expected utility

Algorithm

From "An Algorithm for Automatically Designing Deterministic Mechanisms without Payments"

- Outcome-Set-Based Mechanism

  When the outcome is chosen from a fixed set of options $X \subseteq O$ by maximizing Agent 2 utility (with ties broken in favor of Agent 1), **truthfulness** holds for every subset X

  Goal : Find a subset X that maximizes Agent 1 utility while preserving **individual rationality**

- Branch-and-Bound Depth-first search

  Explores subsets of outcomes in a binary tree by including ($O_i \in X$) or excluding ($O_i \in Y$) each outcome, using an admissible upper bound function v(X, Y) to prune suboptimal branches while ensuring **individual rationality** for every possible type

Implementation

N = 2 K = 15

N = 2 → 2^{16} possible subsets of O
N = 3 → 2^{64} possible subsets of O

~10^{14} times larger

- Utilities for both agents were cached to avoid recomputation
- The evaluation of v(X, Y) compares candidate outcomes only against the best utility per type
- Outcome exclusion (Y) was implemented using boolean masking
- A finite subset of possible types randomly sampled with |Θ| = K was precomputed

Verification

```
--- Results
Type 1 : [1 2 5 3]
Type 2 : [1 2 2 1], outcome: [0 0 1 1], agent1_util: 5, agent2_util: 0
Type 2 : [1 2 4 1], outcome: [0 1 0 1], agent1_util: 2, agent2_util: 0
Type 2 : [4 1 3 4], outcome: [0 1 1 0], agent1_util: 4, agent2_util: 1
Type 2 : [1 3 3 5], outcome: [1 0 1 0], agent1_util: 3, agent2_util: 0
Type 2 : [2 3 3 1], outcome: [0 0 1 1], agent1_util: 5, agent2_util: 1
Type 2 : [3 5 4 1], outcome: [0 0 1 1], agent1_util: 5, agent2_util: 3
Type 2 : [2 2 3 5], outcome: [1 1 0 0], agent1_util: 0, agent2_util: 0
Type 2 : [3 2 2 5], outcome: [0 1 1 0], agent1_util: 4, agent2_util: 1
Type 2 : [3 2 2 2], outcome: [0 0 1 1], agent1_util: 5, agent2_util: 1
Type 2 : [2 4 2 4], outcome: [1 0 1 0], agent1_util: 3, agent2_util: 2
Type 2 : [3 2 2 1], outcome: [0 0 1 1], agent1_util: 5, agent2_util: 2
Type 2 : [4 3 3 2], outcome: [0 0 1 1], agent1_util: 5, agent2_util: 2
Type 2 : [2 1 1 4], outcome: [0 1 1 0], agent1_util: 4, agent2_util: 1
Type 2 : [1 3 3 1], outcome: [0 0 1 1], agent1_util: 5, agent2_util: 0
Type 2 : [4 4 4 1], outcome: [0 0 1 1], agent1_util: 5, agent2_util: 3
--- Analysis on first type for agent 2
Type 1 : [1 2 5 3], Type 2 : [1 2 2 1], mechanism outcome : [0 0 1 1], agent1 util : 5, agent2 util : 0
[0 0 0 0]: ,X -> adding [0 0 0 0] reduces expected agent1 utility from 4.00 to -3.00
[0 0 0 1]: ,X -> adding [0 0 0 1] reduces expected agent1 utility from 4.00 to 1.20
[0 0 1 0]: ,X -> adding [0 0 1 0] reduces expected agent1 utility from 4.00 to 2.00
[0 0 1 1]:O,* -> best outcome chosen from the mechanism
[0 1 0 0]: ,X -> adding [0 1 0 0] reduces expected agent1 utility from 4.00 to -0.33
[0 1 0 1]:O,X -> same util for agent2 (0), not the maximum valuation for agent1 (2<5)
[0 1 1 0]:O,X -> individual rationality constraint (agent2_util=-1)
[0 1 1 1]:O,X -> individual rationality constraint (agent2_util=-2)
[1 0 0 0]: ,X -> adding [1 0 0 0] reduces expected agent1 utility from 4.00 to -0.73
[1 0 0 1]: ,X -> adding [1 0 0 1] reduces expected agent1 utility from 4.00 to 2.87
[1 0 1 0]:O,X -> same util for agent2 (0), not the maximum valuation for agent1 (3<5)
[1 0 1 1]:O,X -> individual rationality constraint (agent2_util=-1)
[1 1 0 0]:O,X -> same util for agent2 (0), not the maximum valuation for agent1 (0<5)
[1 1 0 1]:O,X -> individual rationality constraint (agent2_util=-1)
[1 1 1 0]:O,X -> individual rationality constraint (agent2_util=-2)
[1 1 1 1]:O,X -> individual rationality constraint (agent2_util=-3)
```