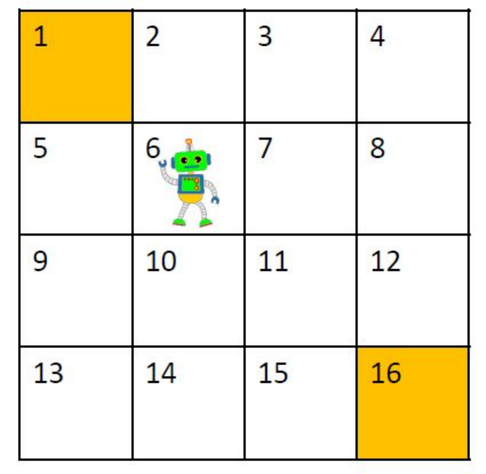
Asynchronous Dynamic Programming

By : Sumit Sharma



Recap - Dynamic Programming in RL

Dynamic Programming (DP) is a fundamental approach to solving Markov Decision Processes (MDPs) using the Bellman equations.

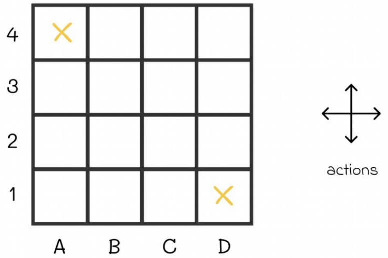
DP methods rely on two main processes:

- **Policy Evaluation** computes the value function for a given policy.
- **Policy Improvement** refines the policy based on updated values.

These methods leads to two algorithm

- **Value Iteration** : Combines evaluation and improvement into one step.
- **Policy Iteration** : Alternates between evaluation and improvement.

Source : (towardsdatascience)Vyacheslav Efimov



k = 0

	A	B	C	D
4	0	0	0	0
3	0	0	0	0
2	0	0	0	0
1	0	0	0	0

k = 1

	A	B	C	D
4	0	-1	-1	-1
3	-1	-1	-1	-1
2	-1	-1	-1	-1
1	-1	-1	-1	0

k = 2

	A	B	C	D
4	0	-1.7	-2	-2
3	-1.7	-2	-2	-2
2	-2	-2	-2	-1.7
1	-2	-2	-1.7	0

k = 3

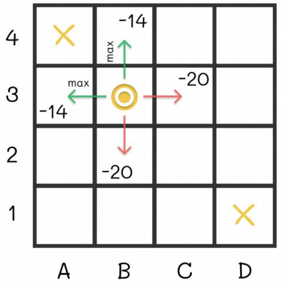
	A	B	C	D
4	0	-2.4	-2.9	-3
3	-2.4	-2.9	-3	-2.9
2	-2.9	-3	-2.9	-2.4
1	-3	-2.9	-2.4	0

k = 10

	A	B	C	D
4	0	-6.1	-8.4	-9
3	-6.1	-7.7	-3	-8.4
2	-8.4	-3	-7.7	-6.1
1	-9	-8.4	-6.1	0

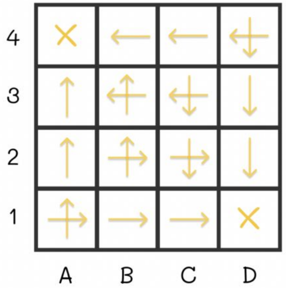
k = ∞

	A	B	C	D
4	0	-14	-20	-22
3	-14	-18	-20	-20
2	-20	-20	-18	-14
1	-22	-20	-14	0



$v_\pi(s)$

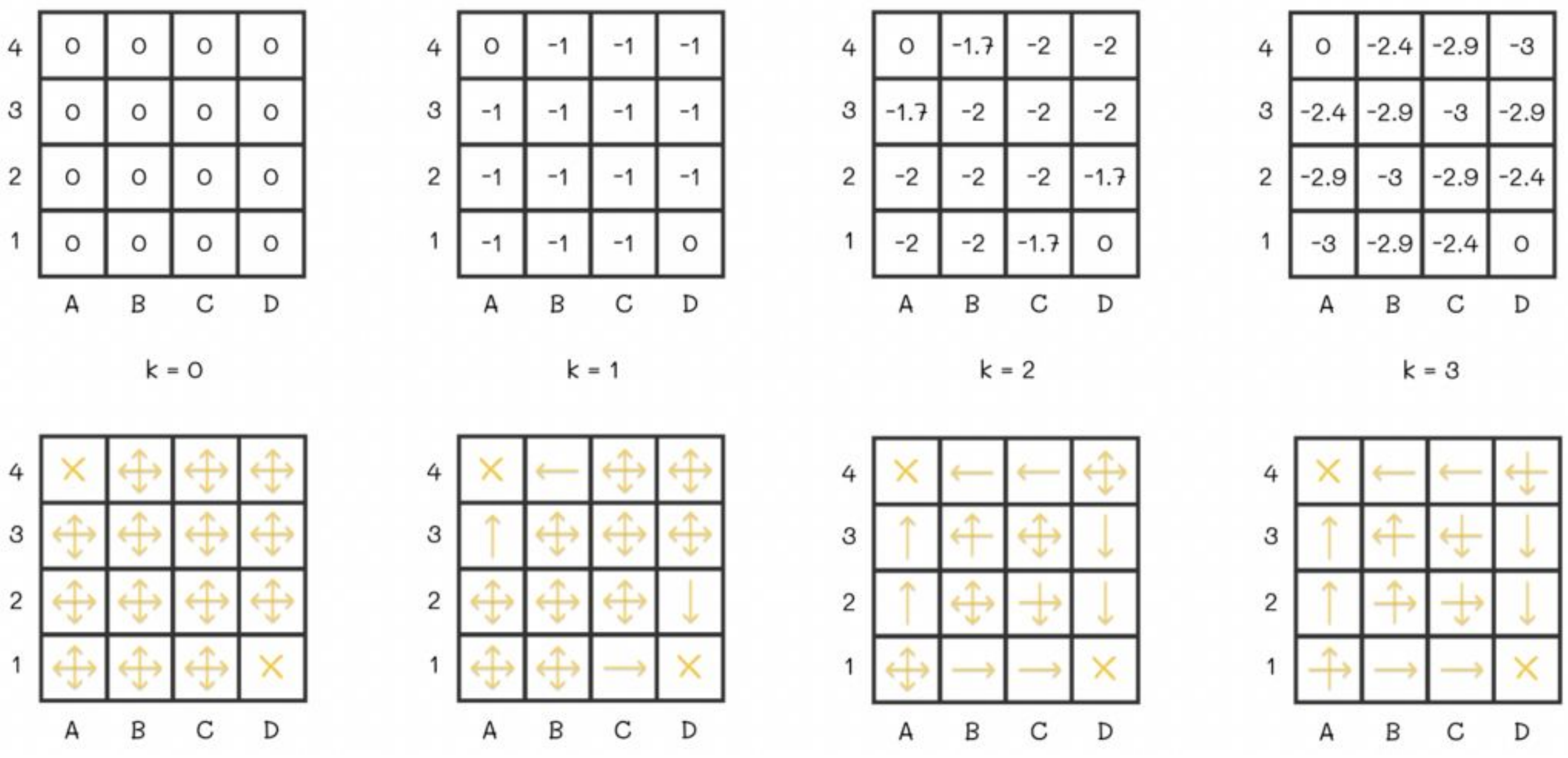
	A	B	C	D
4	0	-14	-20	-22
3	-14	-18	-20	-20
2	-20	-20	-18	-14
1	-22	-20	-14	0



$\pi(s)$

Policy Iteration

Source :
(towardsdatascience)Vyacheslav Efimov



Value Iteration

Challenges with Standard DP

- **Computationally expensive :**

  Each iteration requires updating all states, which becomes infeasible for large state spaces.

- **Inefficiency in large or continuous MDPs:**

  Since DP updates all states, it does not prioritize important regions of the state space.

- **Does not scale well for real-world applications:**

  Many applications involve millions of states, making full sweeps impractical.

Asynchronous Dynamic Programming (ADP)

The key idea is that instead of sweeping through the entire state space, only a subset of states is updated at each step.

This selective updating makes ADP more computationally efficient and applicable to large-scale problems.

Key Differences from Synchronous DP:

- Updates only a subset of states in each iteration.
- Can prioritize important states instead of treating all states equally.
- Reduces computation time and memory usage.

How Does ADP Work?

ADP modifies the standard DP process by introducing asynchronous updates. Instead of iterating over all states, ADP selects specific states and updates their values based on the Bellman equation.

Steps in ADP:

1. Select a subset of states (randomly, based on priority, or from recent experiences).
2. Apply the Bellman backup equation to update these states.
3. Repeat the process iteratively until convergence.

For example, in a gridworld environment,

ADP **selectively updates** a **small subset** of cells **per iteration**, often <u>focusing on states with the highest potential impact.</u>

Types of Asynchronous Updates

Several strategies can be used to determine which states are updated in ADP:

- **Random State Updates**: States are chosen randomly at each iteration.
- **Prioritized Sweeping**: Updates focus on states with the highest Bellman error (largest change in value estimates).
- **Real-time Dynamic Programming (RTDP)**: Only updates states encountered in actual experience.
- **Gauss-Seidel Updates**: Uses the most recent updated values instead of waiting for a full sweep.

Each approach has different trade-offs in terms of computation and convergence speed.

Example - ADP Algorithm (Pseudocode)

Initialize V(s) arbitrarily for all states s

Repeat until convergence:

  Select a subset of states S' (randomly or using priority)

  For each s in S':

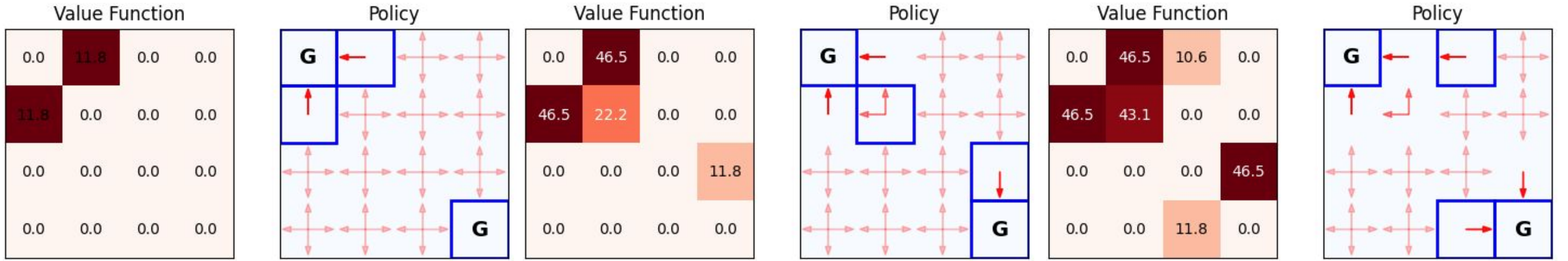
    Update V(s) using Bellman Backup:

    $V(s) = \max_a [R(s, a) + \gamma * \sum P(s' | s, a) * V(s')]$

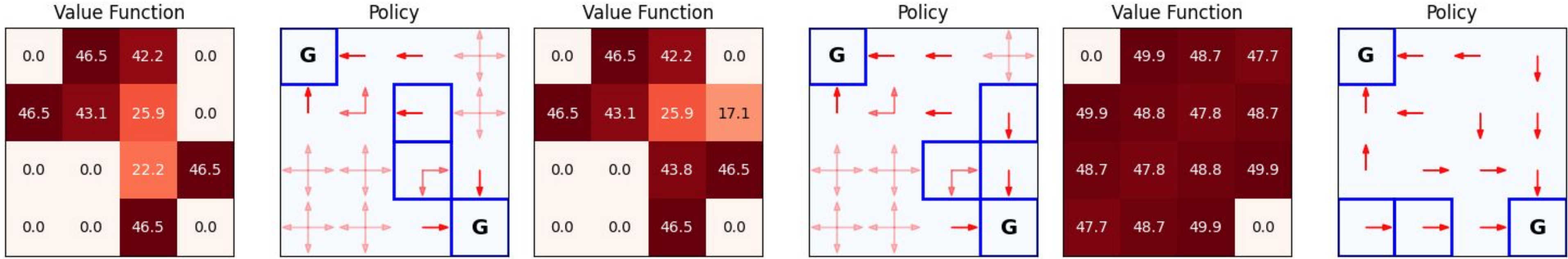
Asynchronous Value Iteration & Policy Iteration

ADP can be applied to both Value Iteration and Policy Iteration.

- Asynchronous Value Iteration:
  - Updates a subset of states at each iteration instead of sweeping through all states.
  - Uses the Bellman backup equation selectively.
  - Converges to the optimal value function given enough iterations.
- Asynchronous Policy Iteration:
  - Performs policy evaluation and improvement in an asynchronous manner.
  - Instead of evaluating the value function for all states, updates occur selectively.
  - Retains the convergence properties of standard policy iteration but with better efficiency.



Iteration : 1 2 3



Iteration : 4 5 Final

Asynchronous Policy Iteration

Advantages of ADP

ADP has several benefits over standard DP:

- **Computational Efficiency:** Avoids unnecessary updates, reducing overall computation.
- **Flexibility:** Different update strategies can be used depending on the problem.
- **Scalability:** Works well for large MDPs where full sweeps are infeasible.
- **Bridges DP and Model-Free RL:** ADP shares similarities with Q-learning, making it a stepping stone to model-free reinforcement learning.

Challenges in Asynchronous Dynamic Programming

- **Stale Information :**
  - States may use outdated values from earlier iterations, leading to slower propagation of information.
- **Redundant Updates :**
  - Asynchronous updates may repeatedly refine the same state or group of states before propagating changes to other parts of the state space.
- **Convergence Behavior :**
  - Asynchronous methods may require more iterations to converge compared to synchronous methods due to the lack of consistency across updates.
- **Scalability Trade-offs :**
  - Although asynchronous DP is designed to handle large state spaces, improper implementation can lead to inefficiencies, such as over-prioritizing irrelevant states or failing to scale effectively.

References

- Sutton & Barto - "Reinforcement Learning: An Introduction"
- Dynamic Programming - AnalyticsVidhya
  https://www.analyticsvidhya.com/blog/2018/09/reinforcement-learning-model-based-planning-dynamic-programming/