Optimization Models and Formulations III

Yinyu Ye
Department of Management Science and Engineering
Stanford University
Stanford, CA 94305, U.S.A.

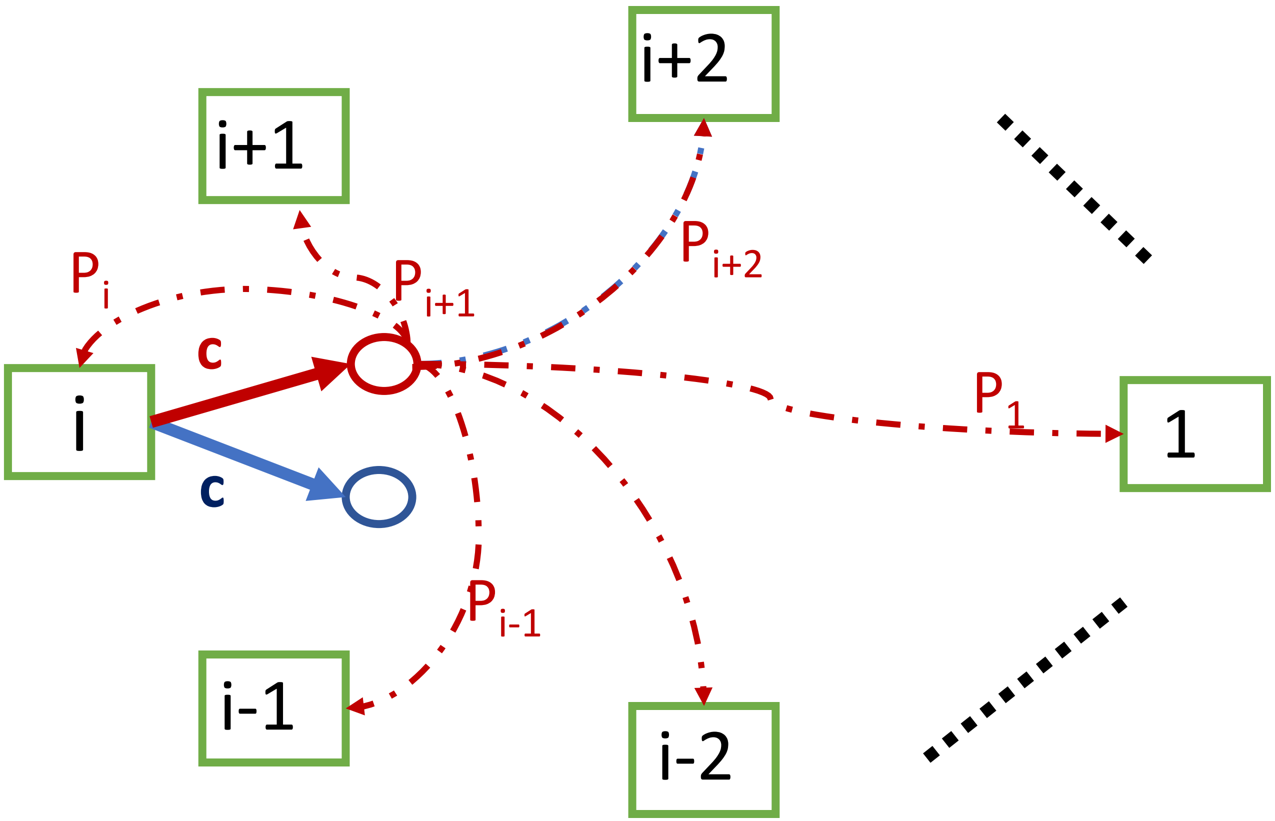
https://canvas.stanford.edu/courses/179677

Read Chapter 1.1, 1.2, 2.1, 2.2, Appendices A, B&D in Text-Book (hard copies would be available in the Book Store)

Example 9: Reinforcement Learning and Markov Decision Process

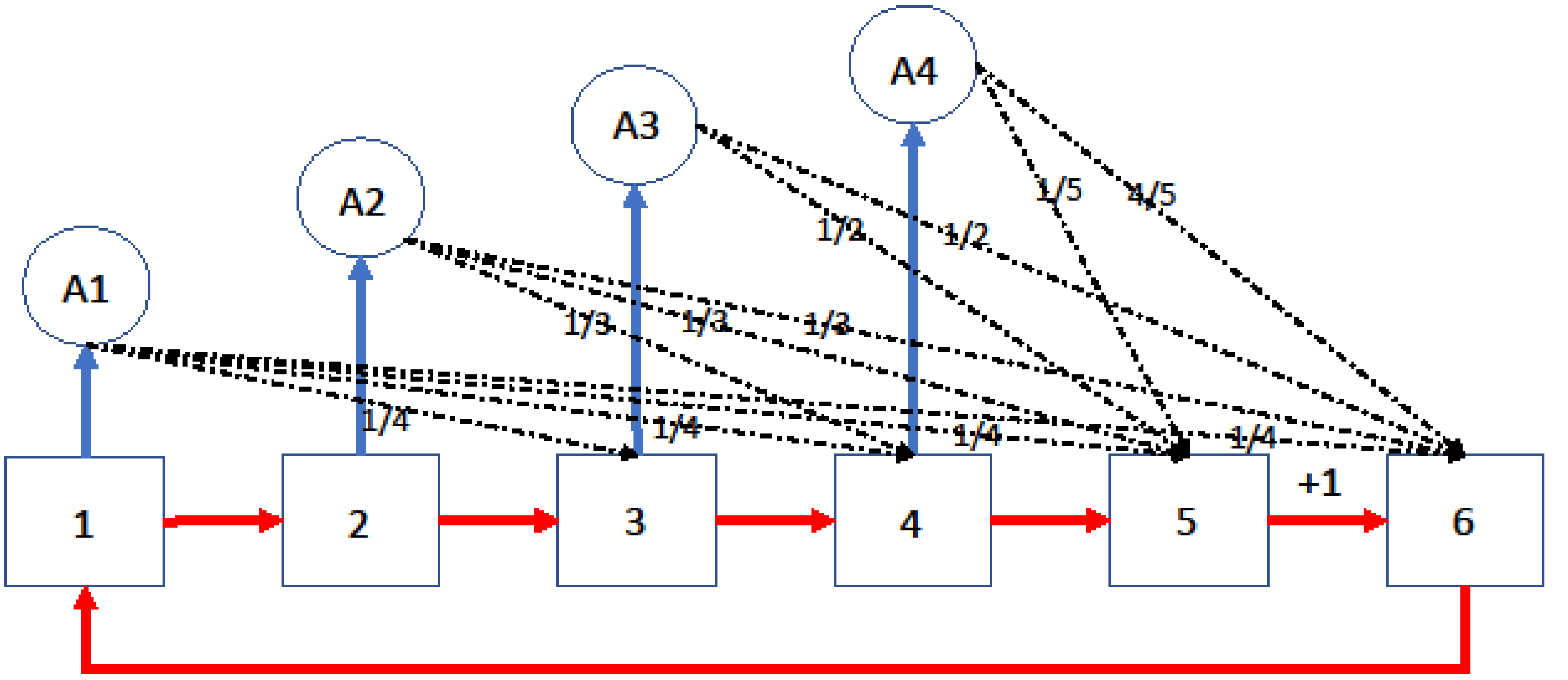
- **Markov Decision Process (MDP)** provides a mathematical framework for modeling **sequential** decision-making in situations where outcomes are partly **random** and partly under the control of a decision maker, and it is called Reinforcement Learning lately.
- MDPs are useful for studying a wide range of optimization problems solved via **stochastic dynamic programming**, where it was known at least as early as the 1950s (cf. Shapley 1953, Bellman 1957).
- Modern applications include dynamic planning, social networking, and almost all other dynamic/sequential-decision-making problems in real life.
- MDP is characterized by **States and Actions**; and at each time step, the process is in a state and the decision maker takes an action to optimize the long-term goal.

State/Action Environment



At each state, when the decision make takes an action (e.g., red), he or she pays an immediate cost (c) and with a probability distribution (p) ends at a state at the next time period.

A Simple RL/MDP Problem: Maze Run

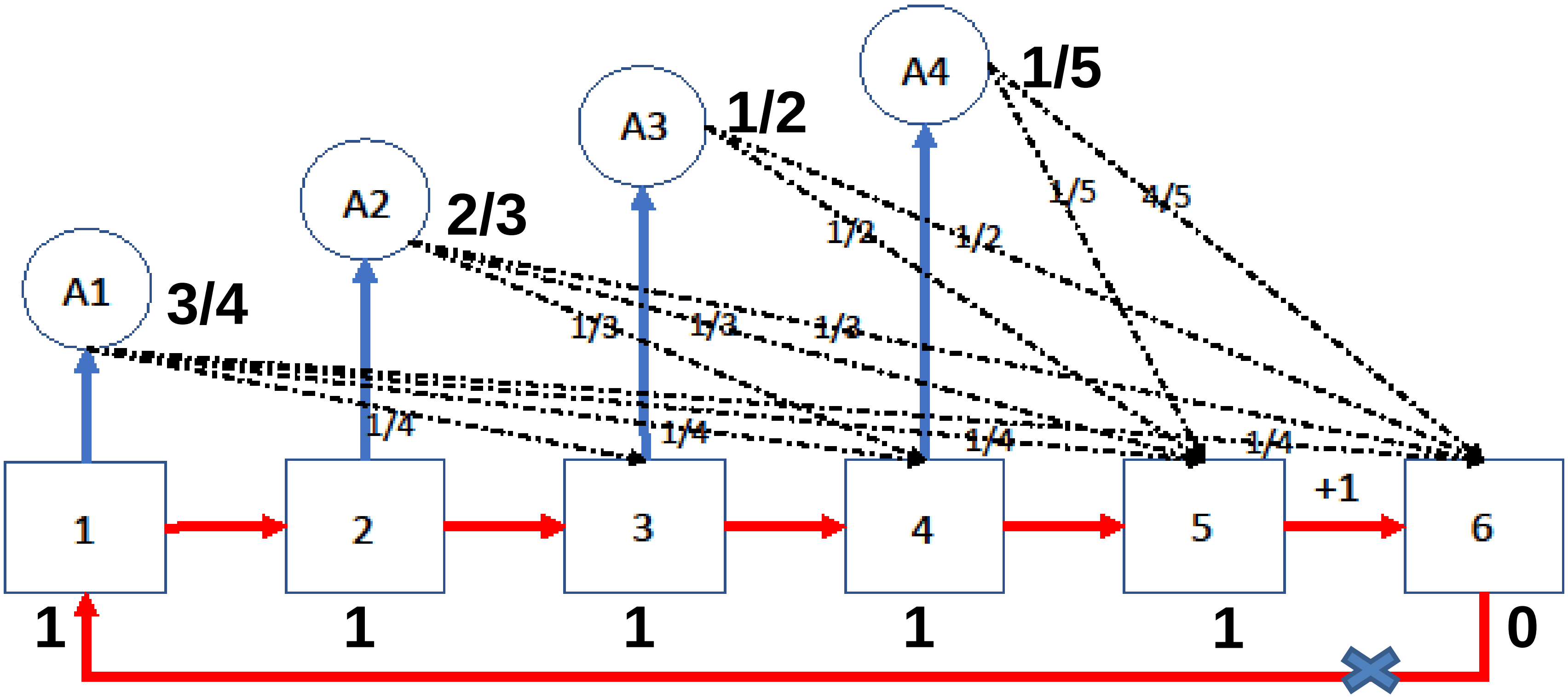


Each state *i* (in Square) is equipped with a set of actions A_i , and they are colored in red (status quo move), blue (shortcut move); and each of them incurs an immediate cost c_j. In this example, all actions have zero cost except the one from the state 4 (trap) to the final termination state 5 (Exit state which goes back to itself). Each action is associated with transition probability node (circle) with distribution vector P_j to all states.

Markov Decision Process with Finite or Discounted Infinite Horizon

- **The Process** can end at a finite horizon or time steps
- **It can also extend to infinite horizon** with a **discount factor** γ
- A (stationary) **policy** is a set of actions taken, one per State, at anytime step
- A (stationary) policy defines an expected and **discounted present Cost-to-Go value** for every state over all future time steps, that is, the overall expected present cost if starting from this very state
- The MDP is to find the optimal stationary policy such that its overall expected present cost is minimized from an initial state
- It turns out that the optimal policies are identical for all possible initial states

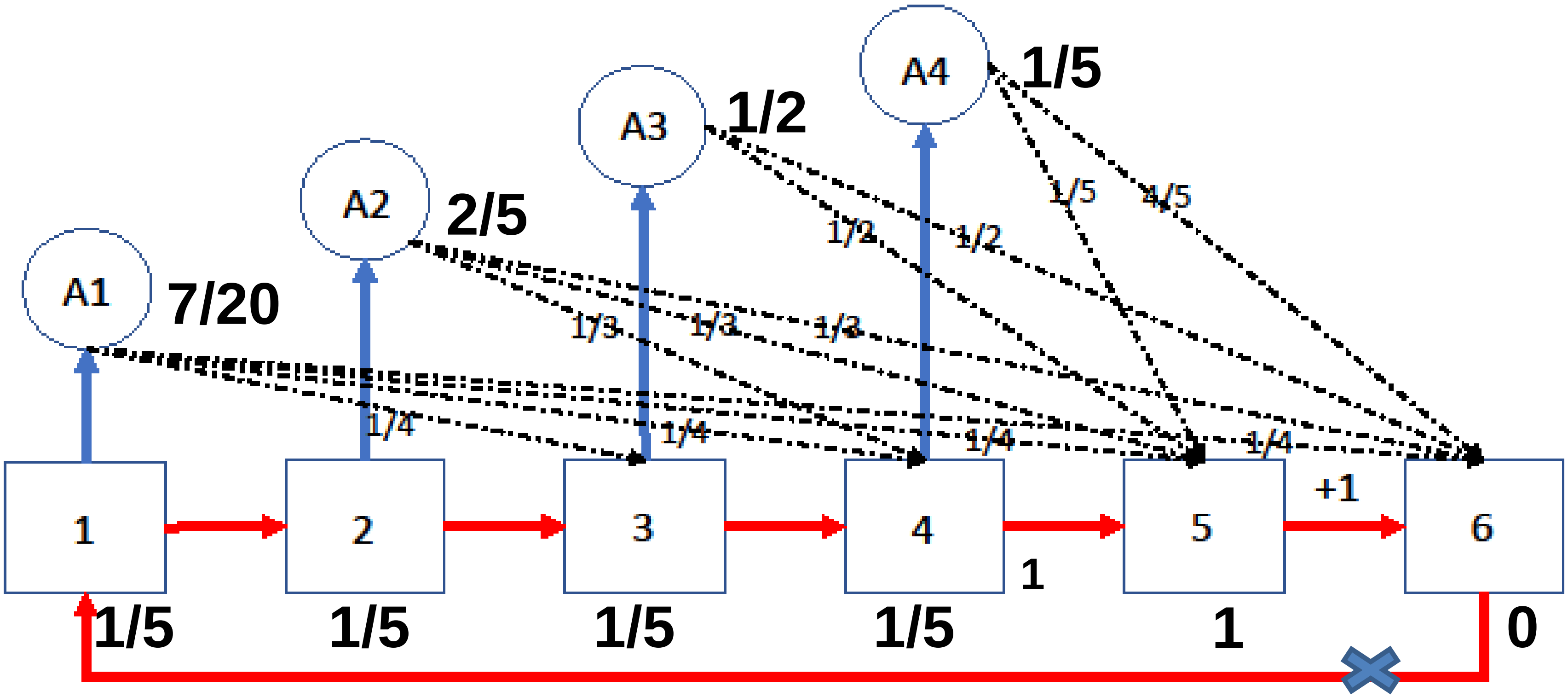
Expected Cost-to-Go values of the Red Policy



Consider a finite horizon maze run where there is no discount. Assume the current policy takes all-red actions, then the corresponding expected cost-to-go state-values would be given above, together with the expected values when taking alternative action in a state.

Clearly, this policy is not optimal… The optimal policy is?

The Optimal Policy

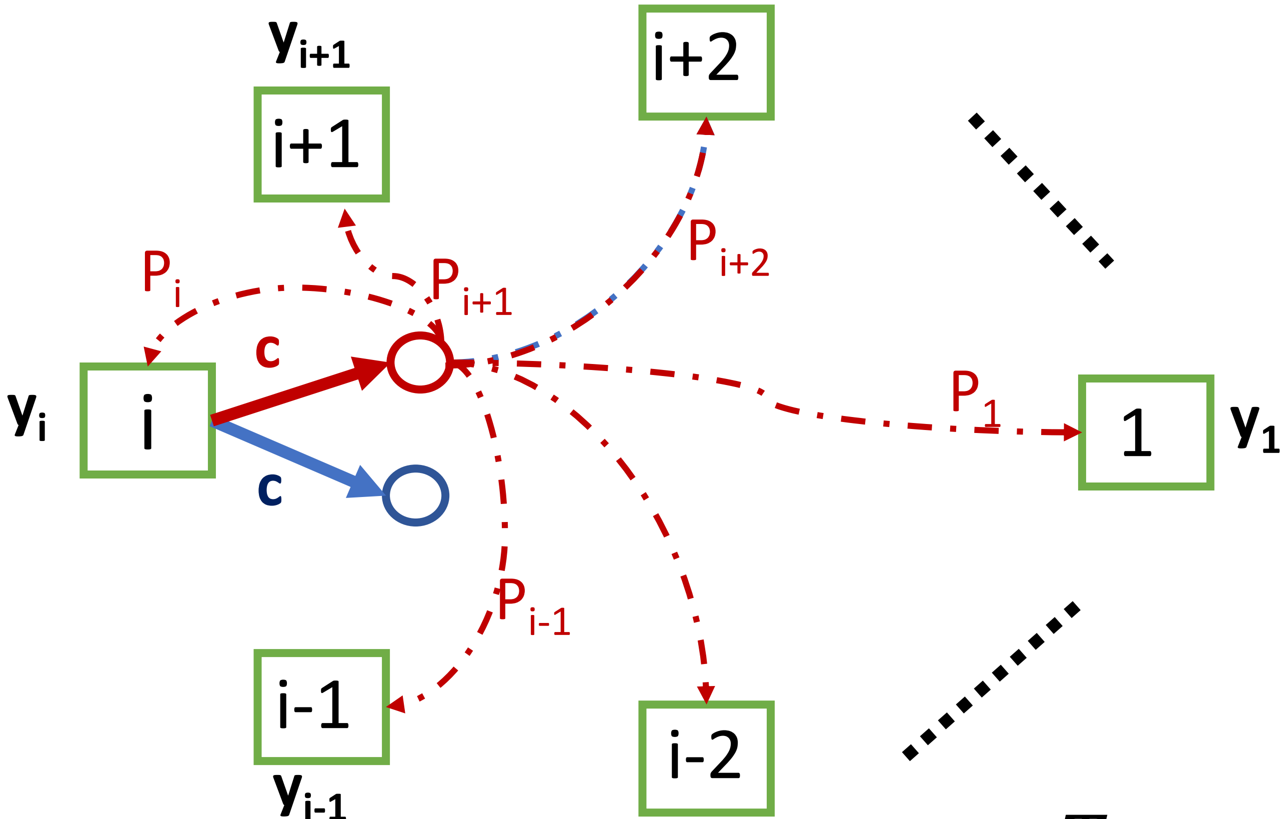


The optimal policy takes (red, red, read, blue, red) action for state (1,2,3,4,5).
The corresponding expected overall cost-to-go values would be given above, together with the expected values when taking alternative action in a state.

Why is this policy optimal?

Because for each state there is no action-switch that results in a lower cost.

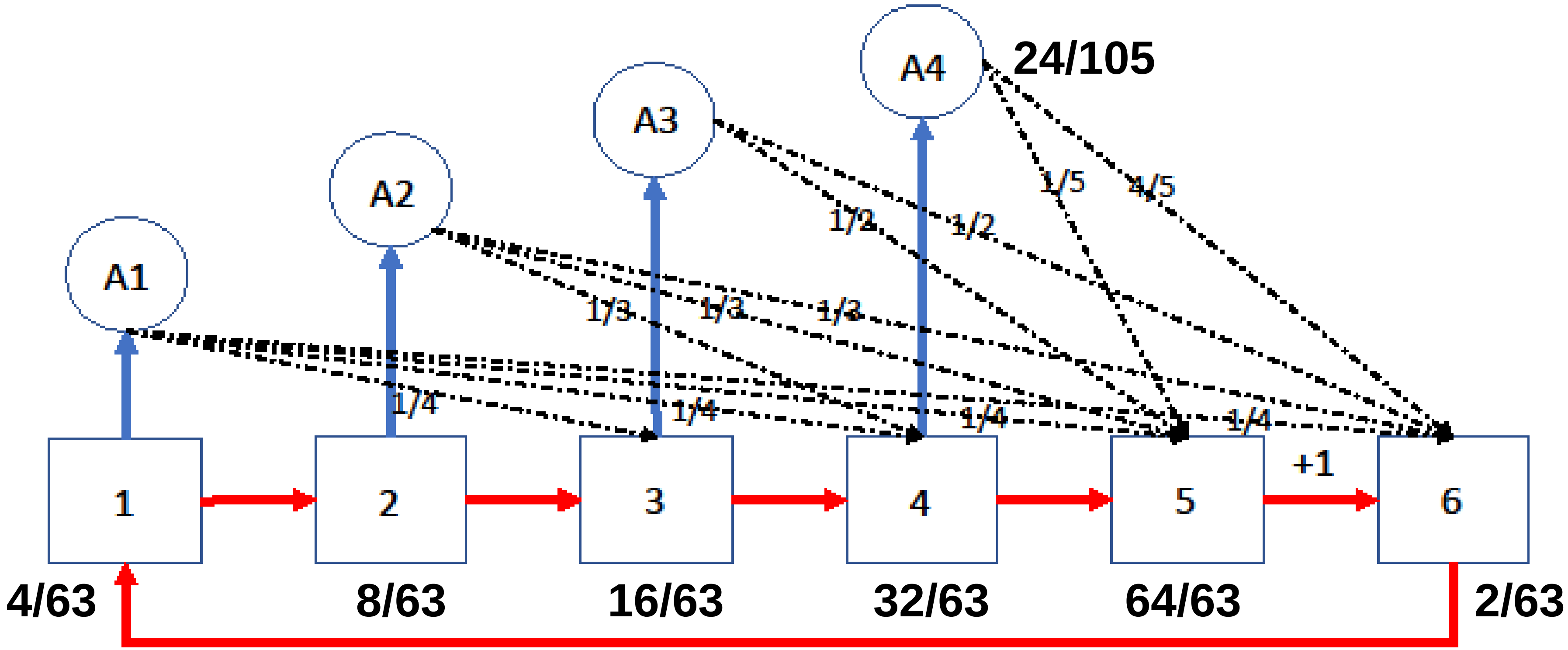
Infinite Discount Horizon: Compute the Cost-to-Go Value at State *i* in General



$$y_i \quad = \quad c \qquad + \quad \gamma \quad p^T y,$$

immediate cost   expect future cost

Expected Cost-to-Go values of the Red Policy

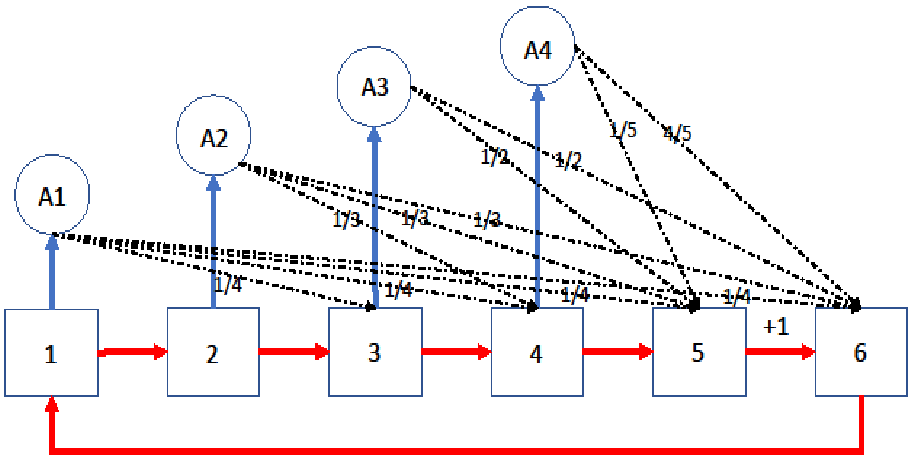


Consider an infinite horizon maze run where the discount factor is 0.5. Assume the current policy takes all-red actions, then the corresponding expected cost-to-go state-values would be given above, together with the expected values when taking alternative action in a state

Is this policy optimal? No, action-switch at State 4 results a lower cost

Cost-to-Go values of the Maze Run

- y_i: the expected overall present cost if stating from State i.
- State 5 is a trap
- State 6 is the exit state
- Each other state has two options:
  Go directly to the next state or
  a short-cut go to other states
  with uncertainties



- The cost-to-go values of the optimal policy with discount factor ϒ for this simple example should meet the following conditions

$y_6 = 0 + \gamma y_1$, $y_5 = 1 + \gamma y_6$

$y_4 = \min\{ 0 + \gamma y_5 , 0 + \gamma(0.2y_5 + 0.8y_6) \}$,

$y_3 = \min\{ 0 + \gamma y_4 , 0 + \gamma(0.5y_5 + 0.5y_6) \}$

$y_2 = \min\{ 0 + \gamma y_3 , 0 + \gamma(y_4/3 + y_5/3 + y_6/3) \}$

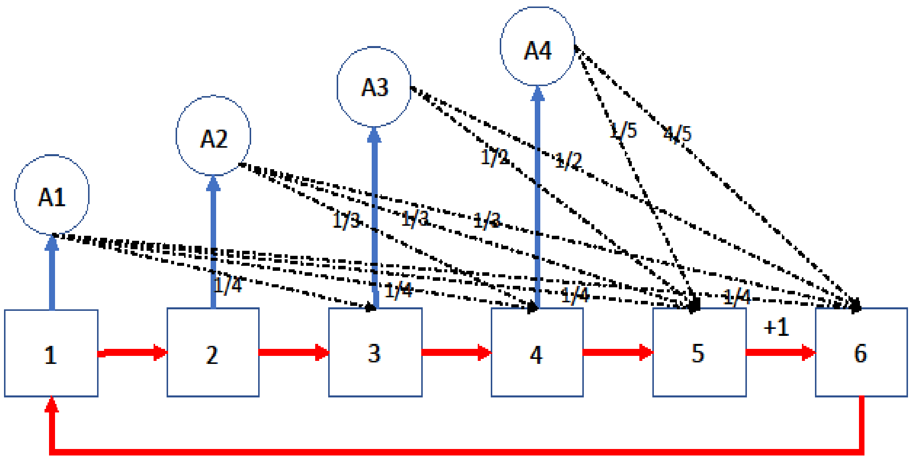
$y_1 = \min\{ 0 + \gamma y_2 , 0 + \gamma(0.25y_3 + 0.25y_4 + 0.25y_5 + 0.25y_6) \}$

See "How to Linearize the Min-Function"
Lecture Note #2, Slide 10

LP Formulation of the Maze Run

$$\max\ y_1 + y_2 + y_3 + y_4 + y_5 + y_6$$

s.t.
$$y_6 \le 0 + \gamma y_1$$
$$y_5 \le 1 + \gamma y_6$$
$$y_4 \le 0 + \gamma y_5$$
$$y_4 \le 0 + \gamma(0.2y_5 + y_6)$$
$$y_3 \le 0 + \gamma y_4$$
$$y_3 \le 0 + \gamma(0.5y_5 + 0.5y_6)$$
$$y_2 \le 0 + \gamma y_3$$
$$y_2 \le 0 + \gamma(0.33y_4 + 0.33y_5 + 0.33y_6)$$
$$y_1 \le 0 + \gamma y_2$$
$$y_1 \le 0 + \gamma(0.25y_3 + 0.25y_4 + 0.25y_5 + 0.25y_6)$$



See "How to Linearize the Min-Function" Lecture Note #2, Slide 10

The LP Formulation in General

- In general, let $\boldsymbol{y} \in R^m$ represent the expected present cost-to-go values of the m states, respectively, for a given policy. Then, the cost-to-go vector of the optimal policy, with the discount factor γ, by Bellman's Principle is a Fixed Point:

$$y_i = \min\{ c_j + \gamma p_j^T y, j \in A_i \}, \forall i,$$

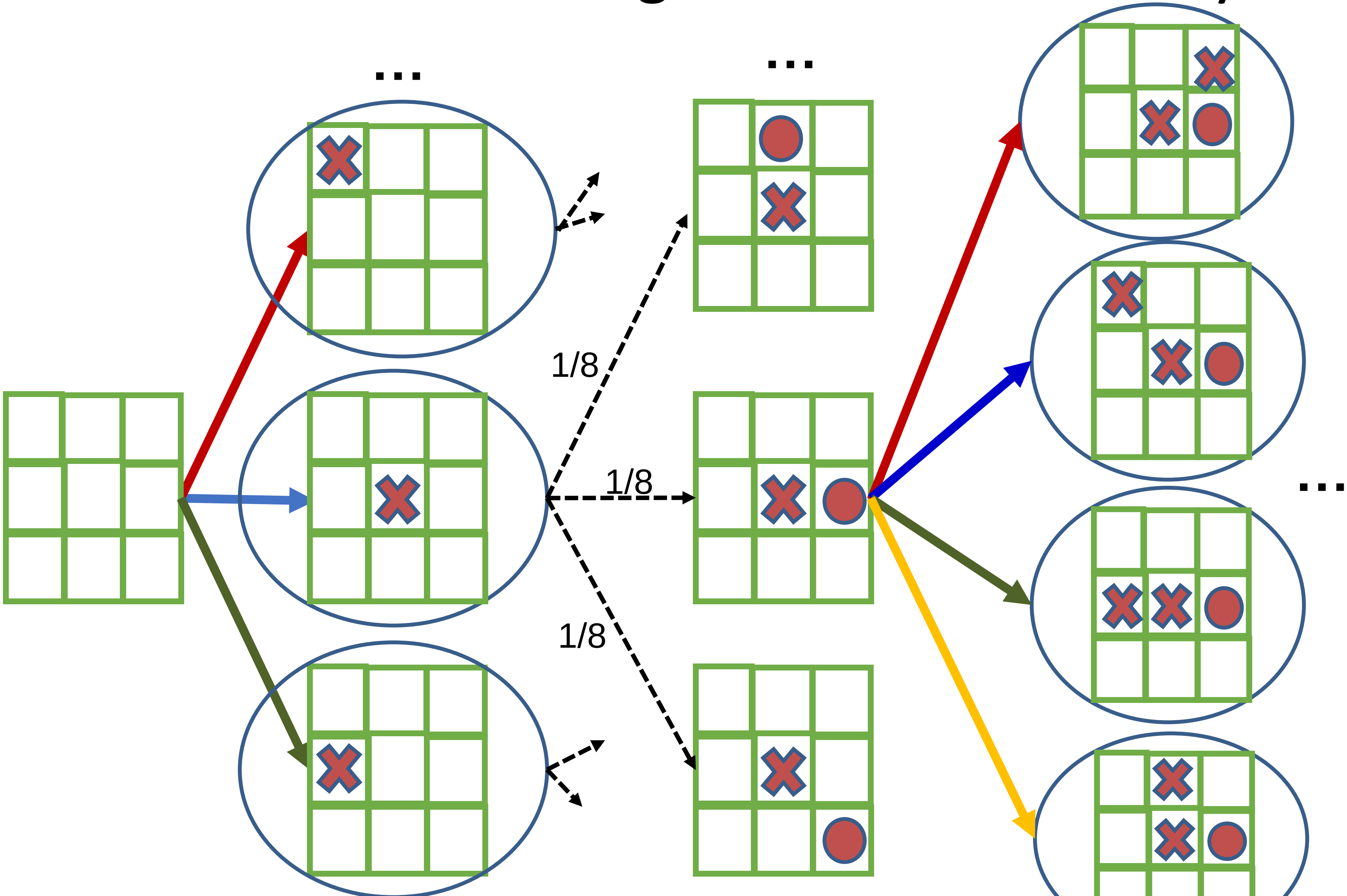
$$j_i = \arg\min\{ c_j + \gamma p_j^T y, j \in A_i \}, \forall i.$$

- Such a fixed-point computation can be formulated as an LP

$$\begin{aligned} \max \quad & \sum_i y_i \\ \text{s.t.} \quad & y_i \le c_j + \gamma p_j^T y, \forall j \in A_i; \forall i. \end{aligned}$$

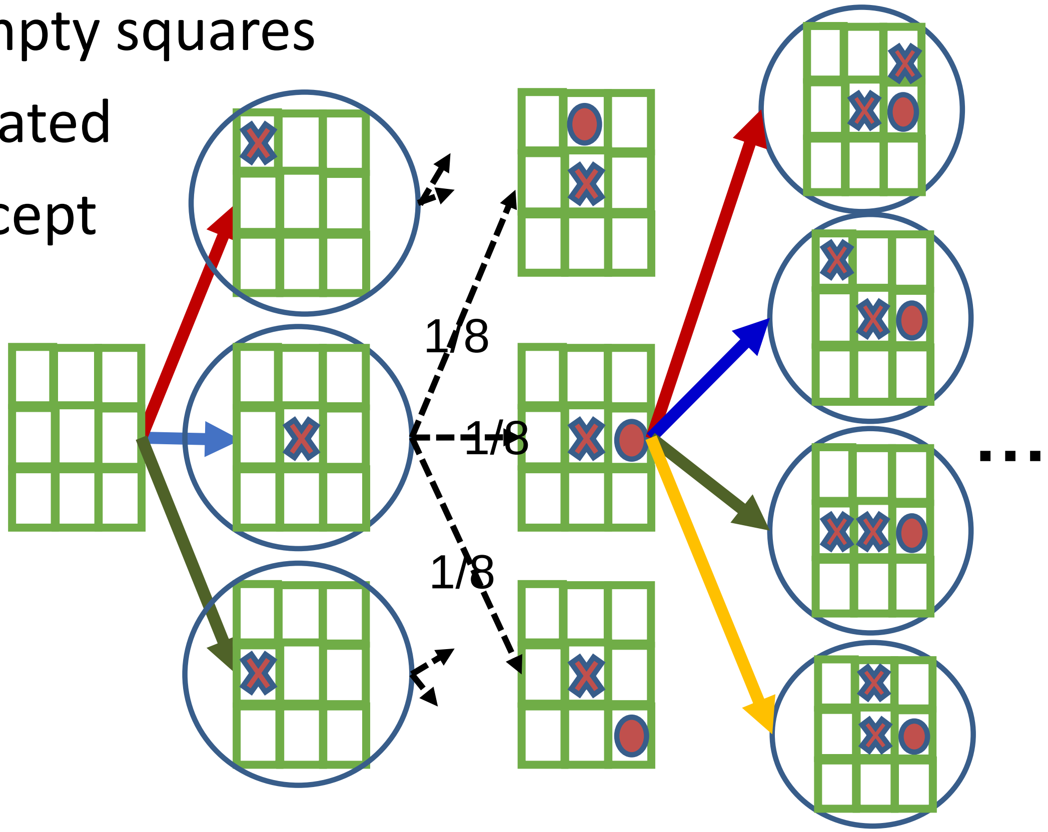
- The maximization is trying to pushing up each yi to the highest possible so that it equal to min-argument. When the optimal y is found, one can then find the index of the original optimal action/policy using argmin.

Tic-Tac-Toe Game Against a Random Player

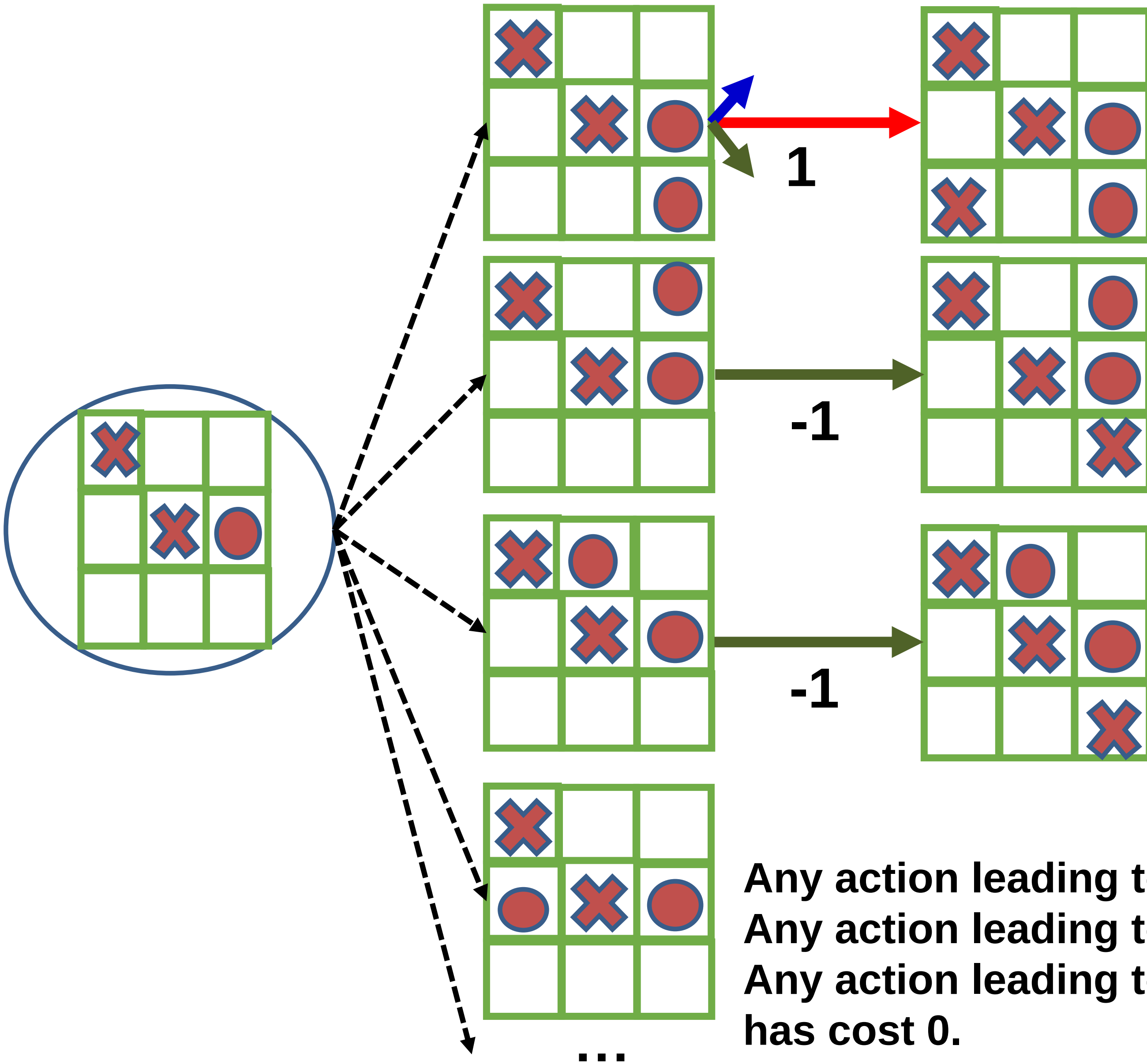


States/Actions of Tic-Tac-Toe Game

- Each **State** is a configuration of "**cross**" and "**circle**" locations
- **Each Action** at a state is to place a "**cross**" at an empty square
- After an action being taken at a state, the **probability distribution** of transferring to a new state at next time step is uniformly placing "**circle**" at every empty squares
- The **immediate cost** associated with each action is zero except at the very end …



State-Action Costs of the End Time



Any action leading to win has cost -1
Any action leading to lose has cost 1
Any action leading to tie or undecided has cost 0.