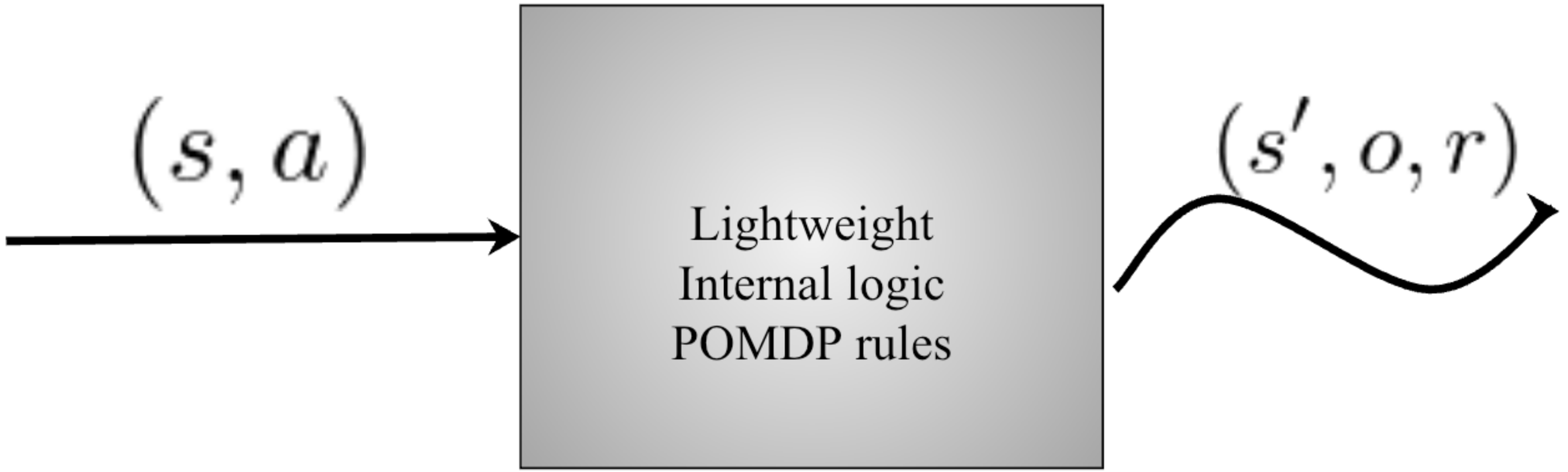
POMCP

Partially Observable Monte-Carlo Planning

Himanshu Dongre

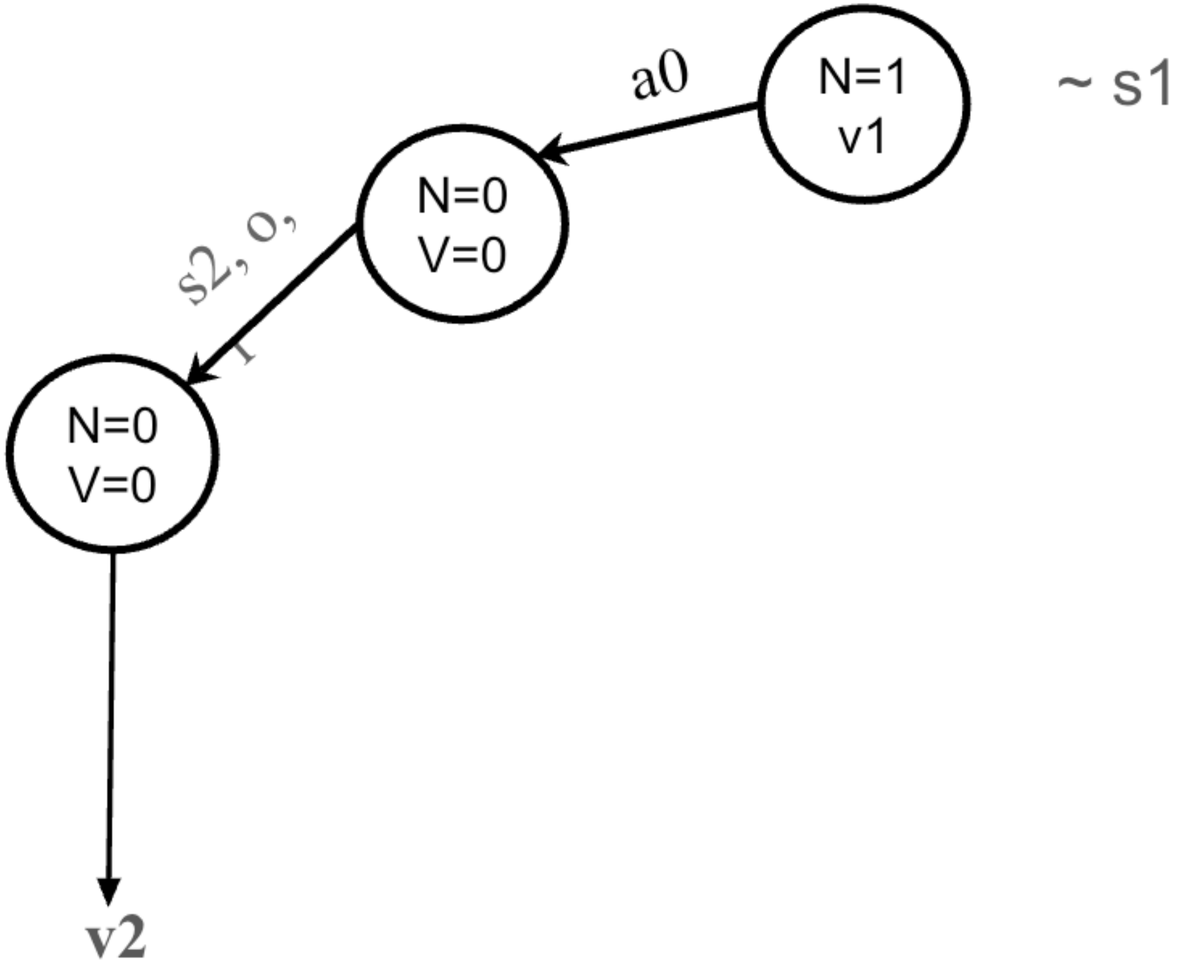
POMCP

Black-box generative simulator

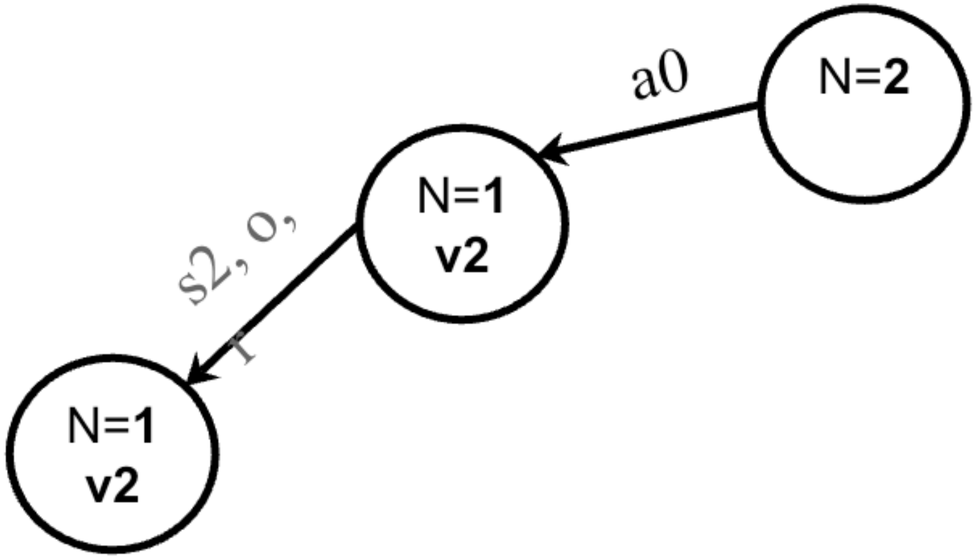


$$(s', o, r) \sim \mathcal{G}(s, a)$$

POMCP

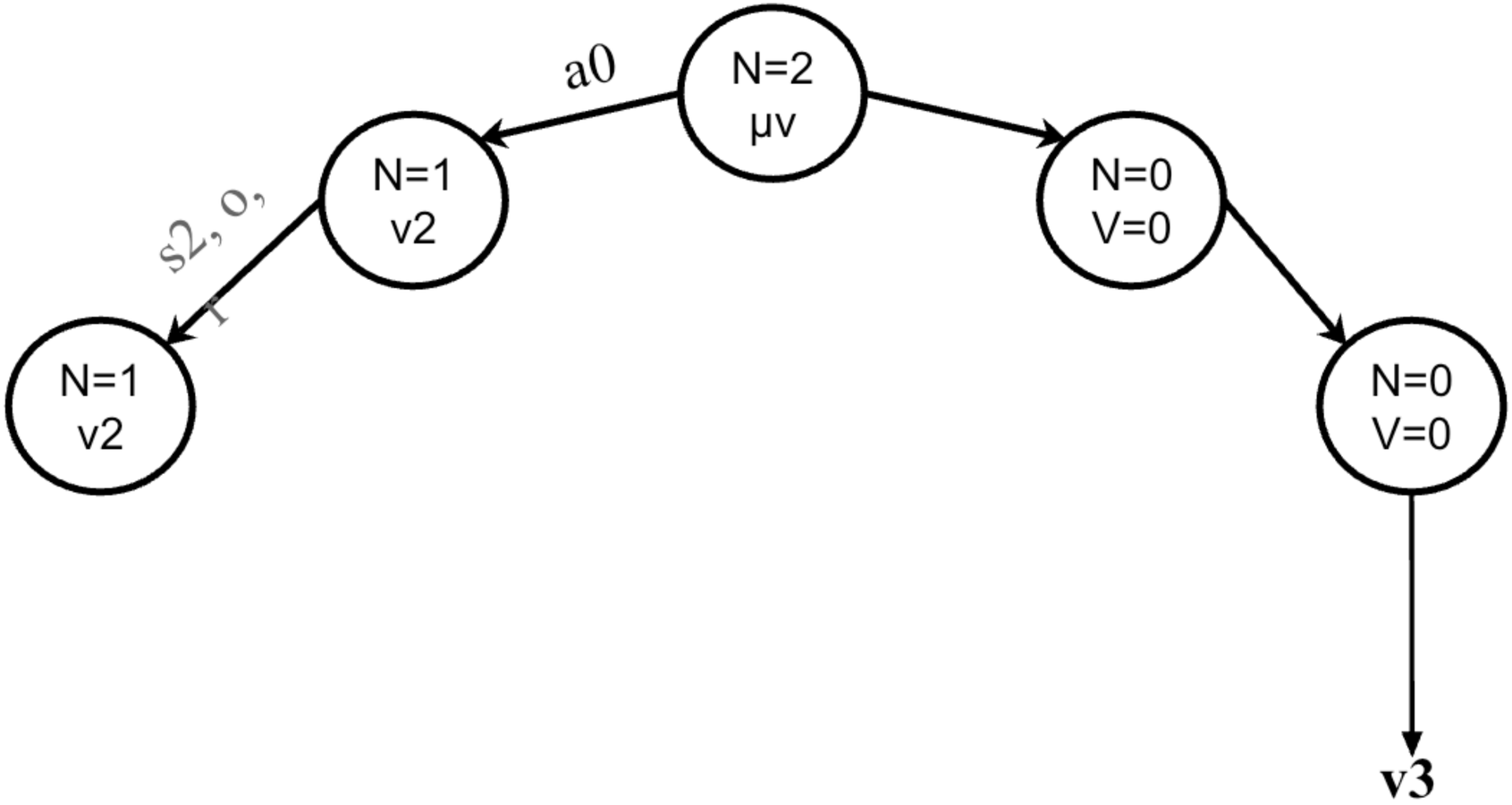


POMCP

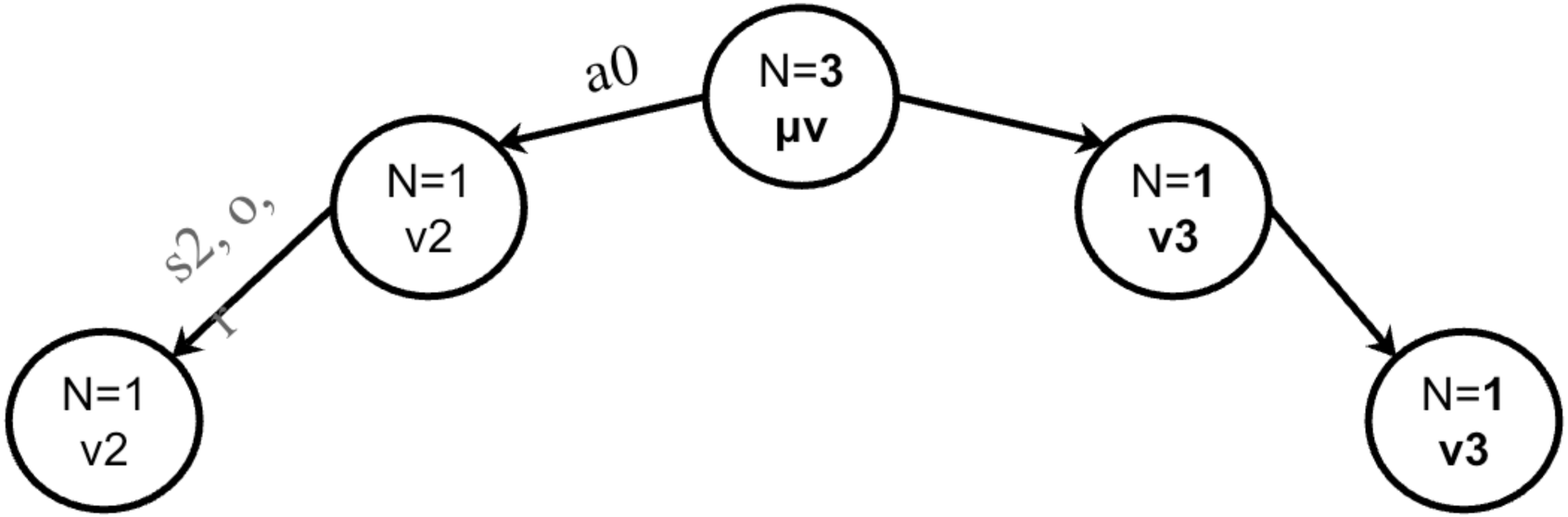


V = **(v1+v2)/2 = μv**

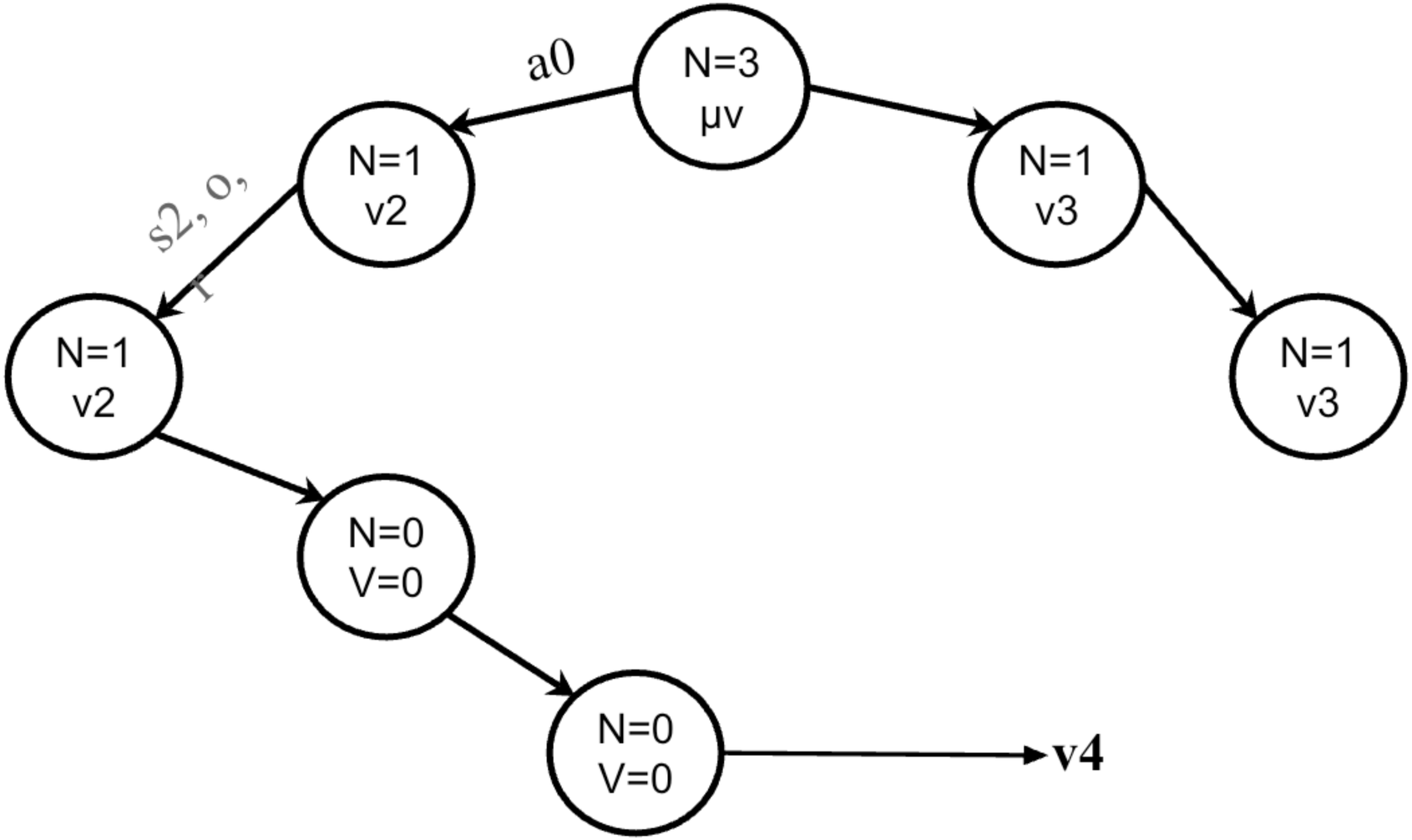
POMCP



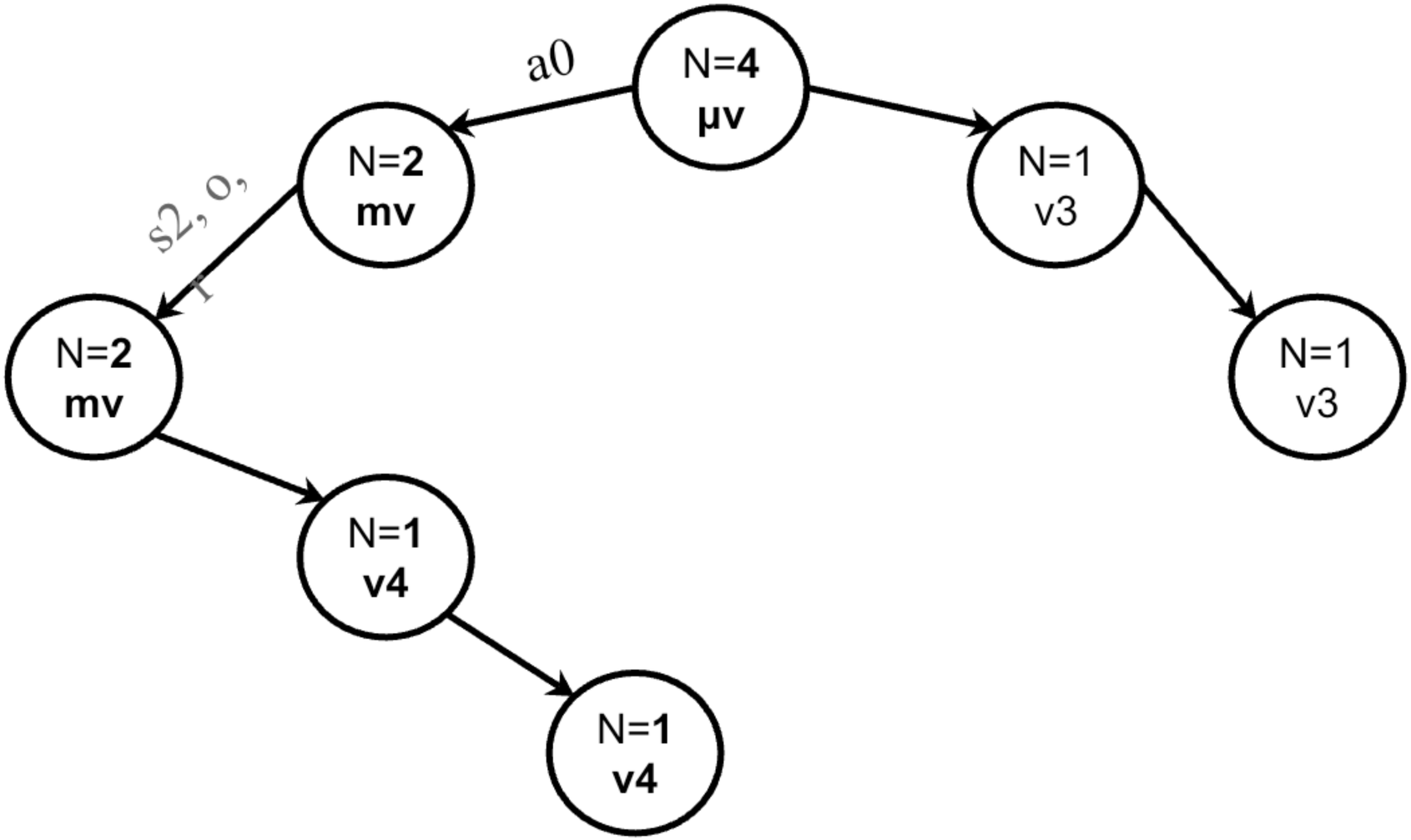
POMCP



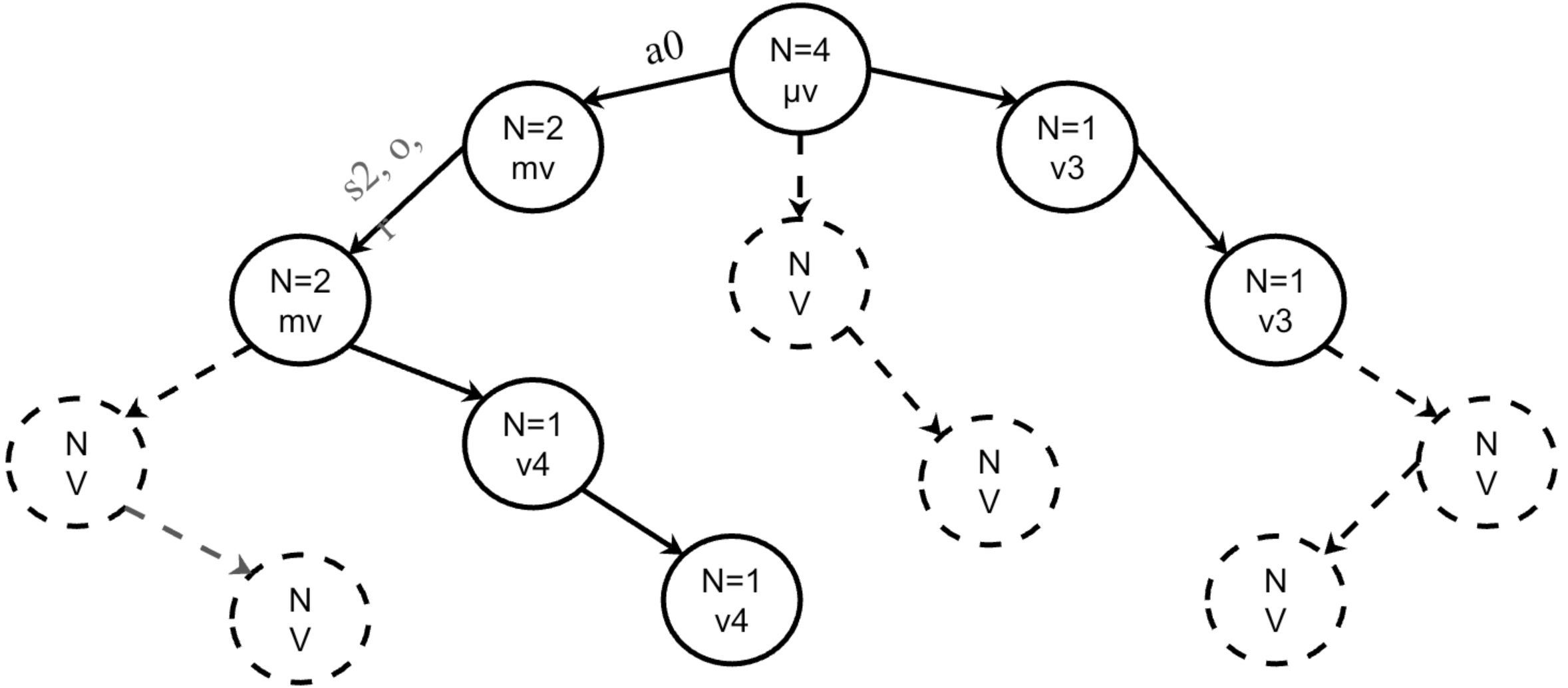
POMCP



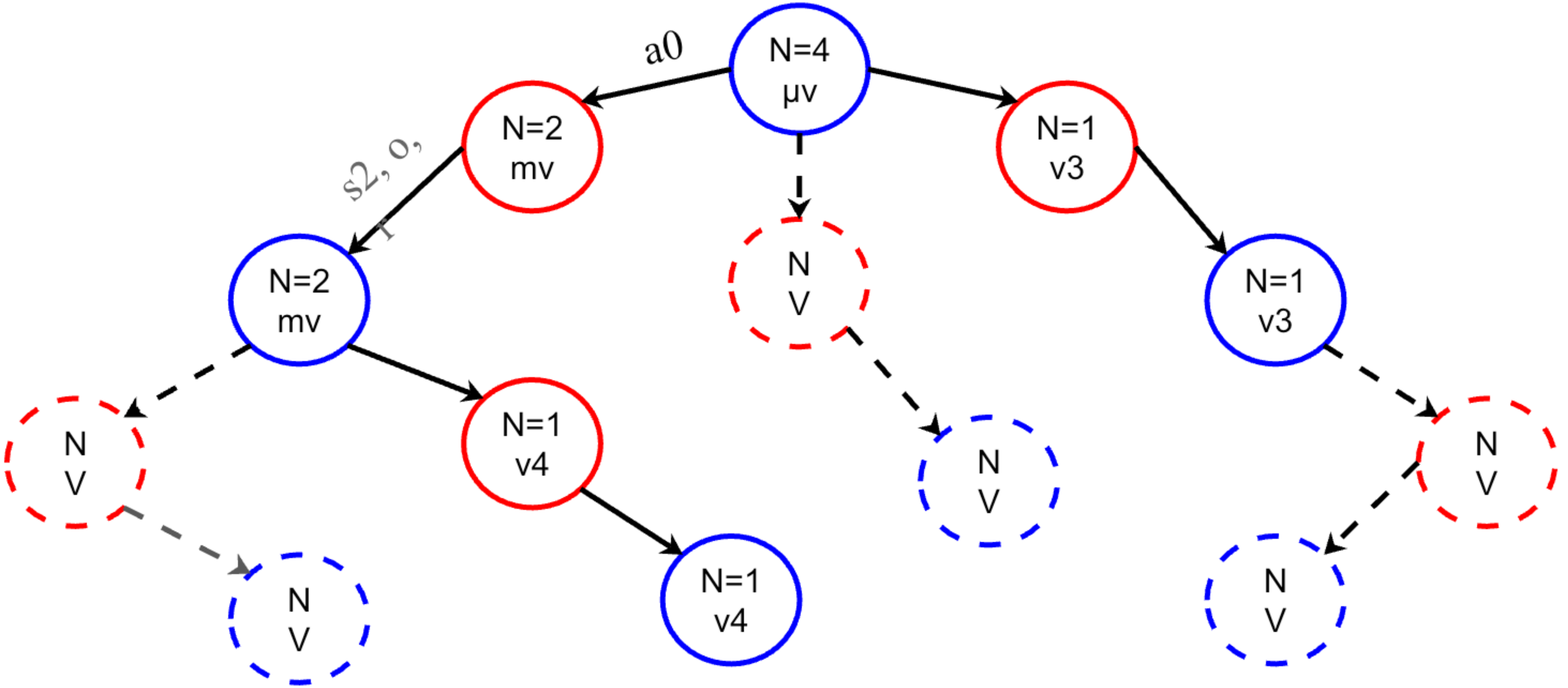
POMCP



POMCP

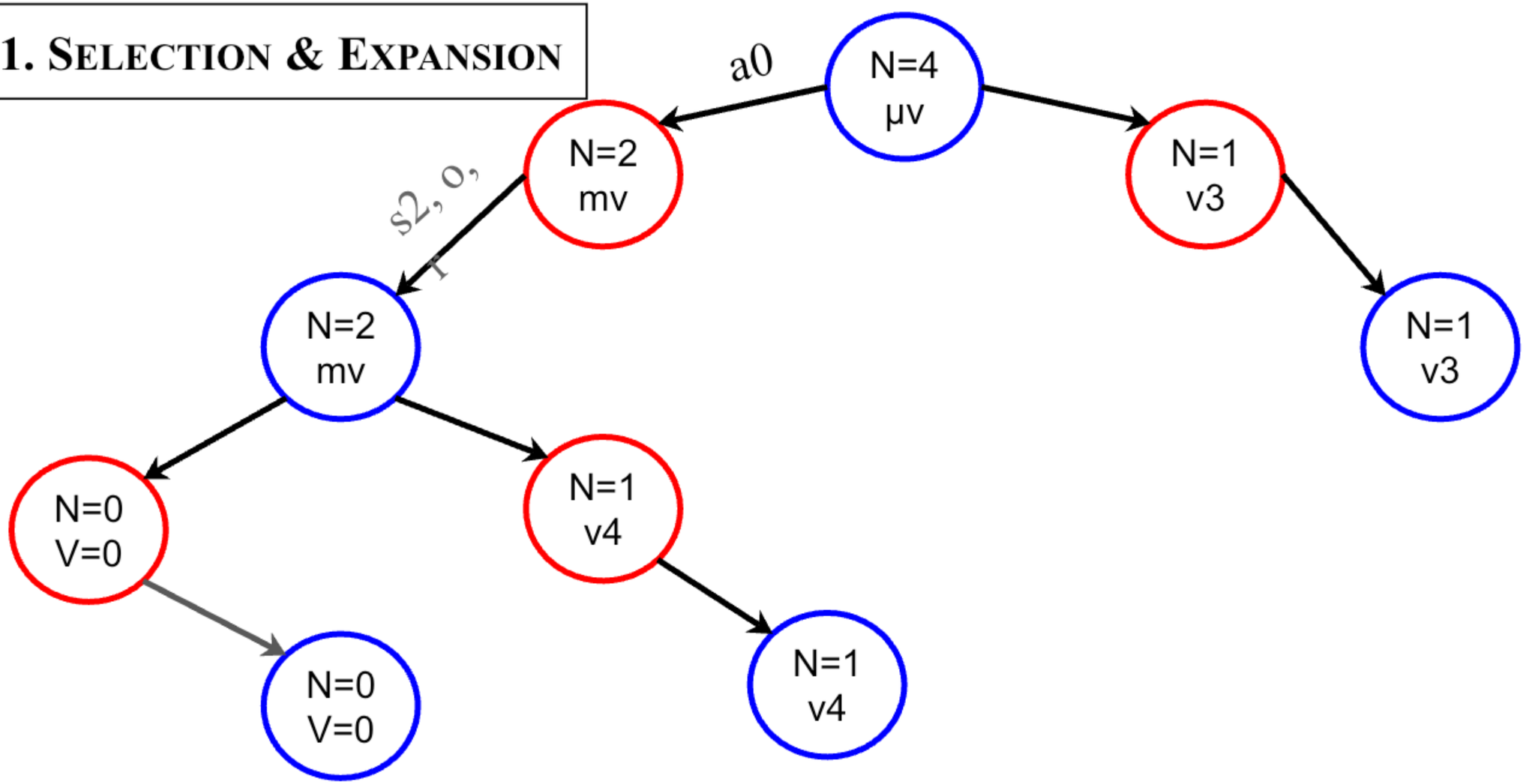


POMCP



POMCP

1. Selection & Expansion



Introduction

- Definition:-

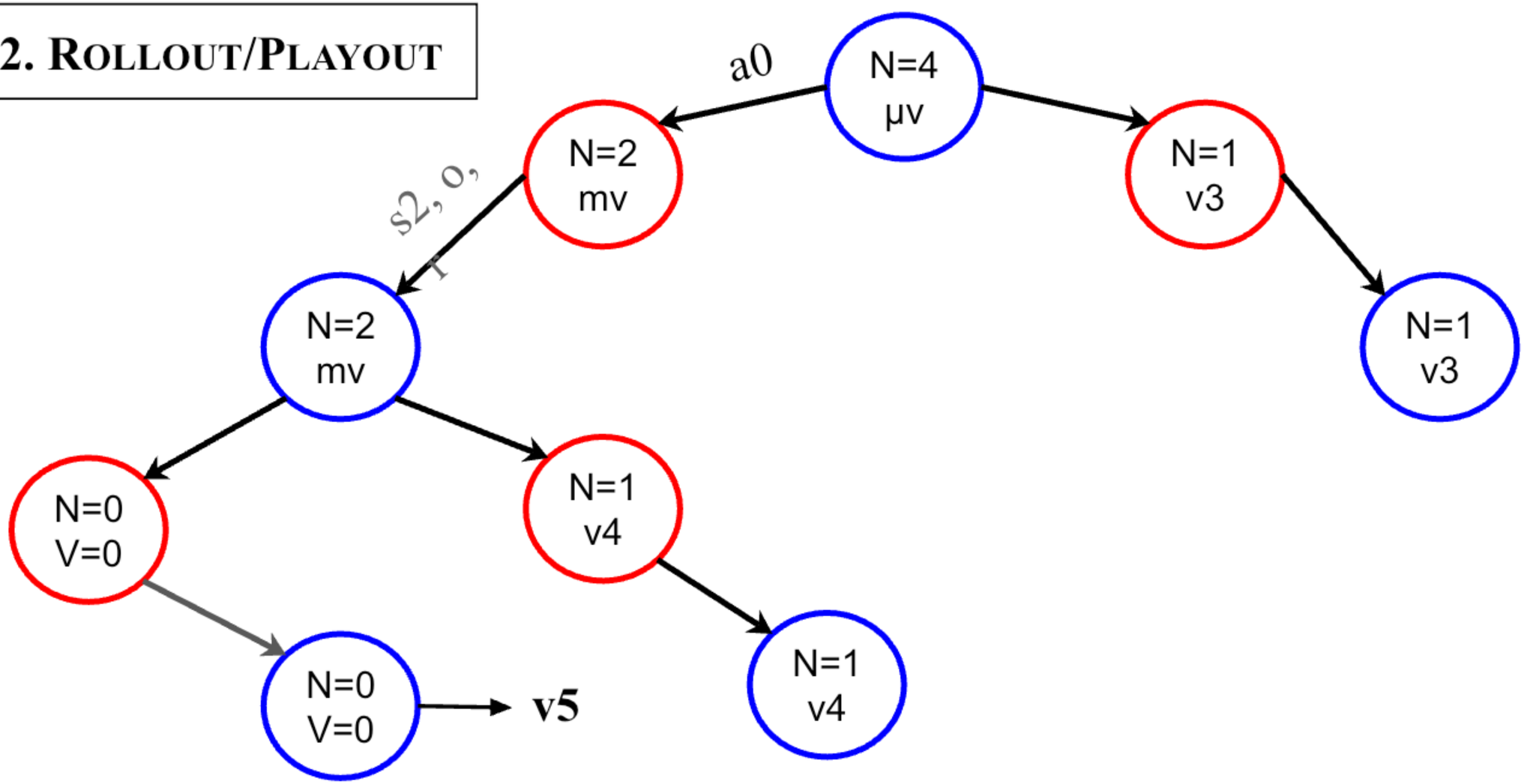
$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \Omega, \mathcal{O}, \mathcal{R} \rangle$$

- $$\mathcal{B} = \{b \mid b \text{ is a probability distribution over } \mathcal{S}\}$$

- Planning problem in POMDPs: Maximizing **expected long-term rewards** by figuring out the best **way to act**.

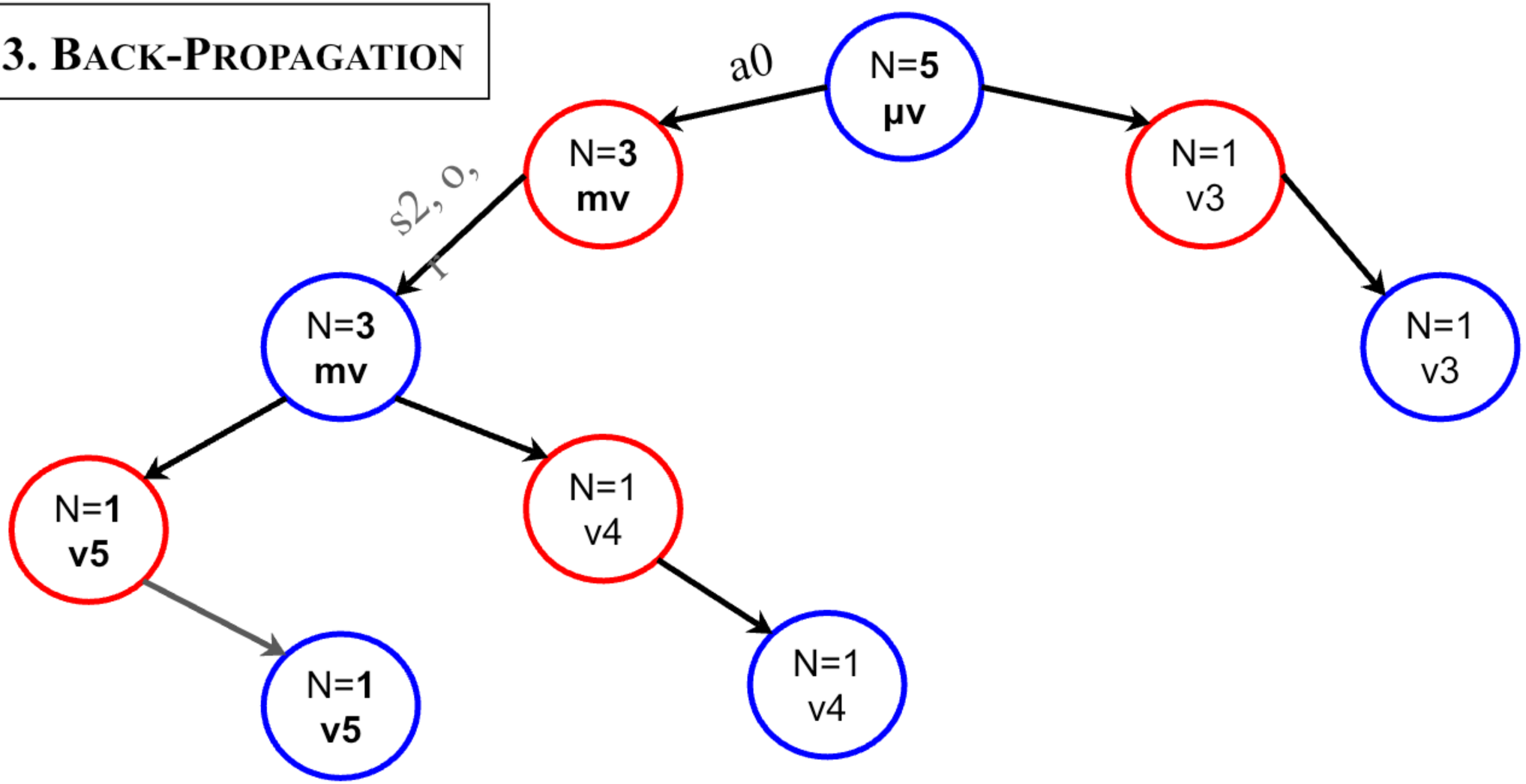
POMCP

2. Rollout/Playout



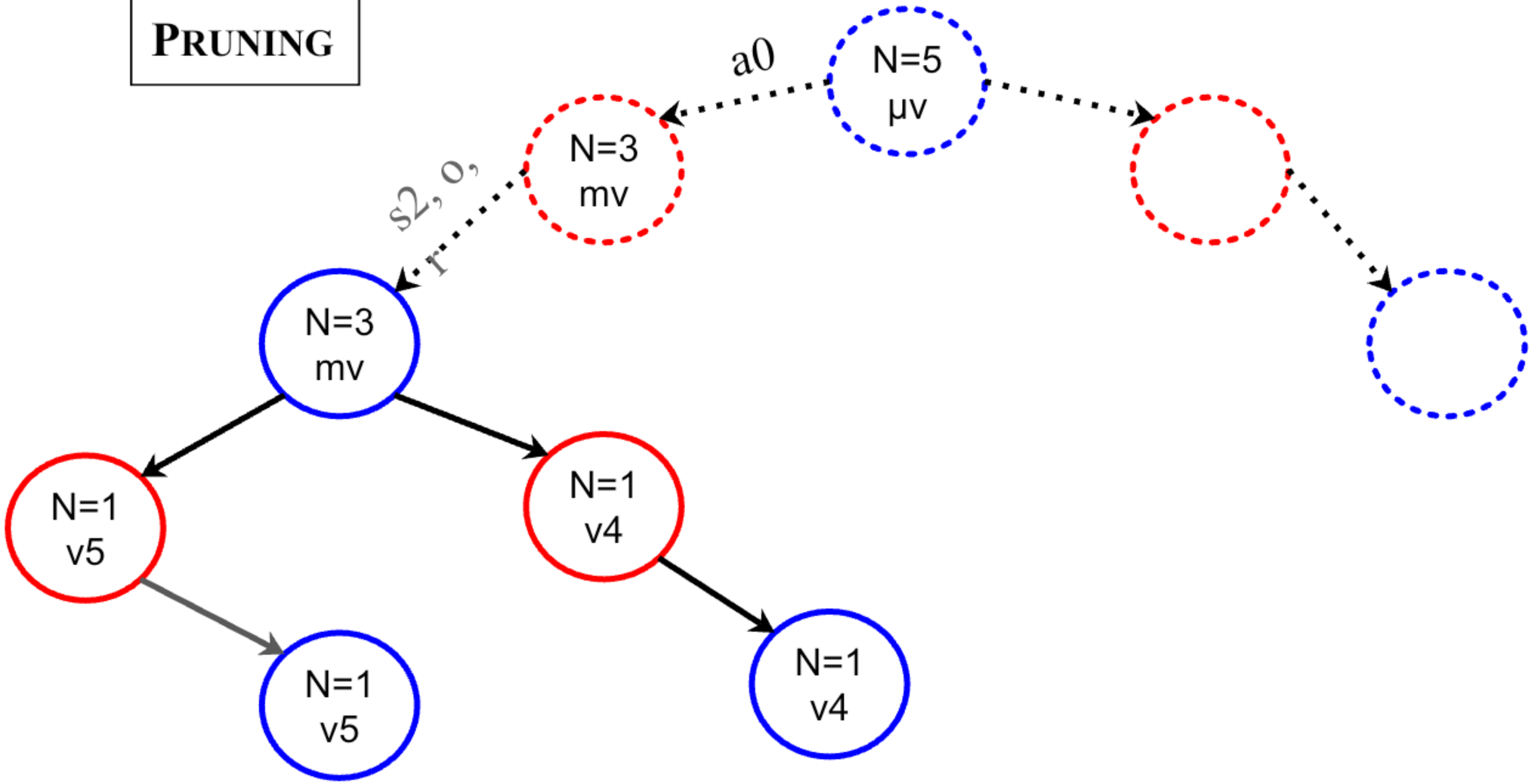
POMCP

3. Back-Propagation

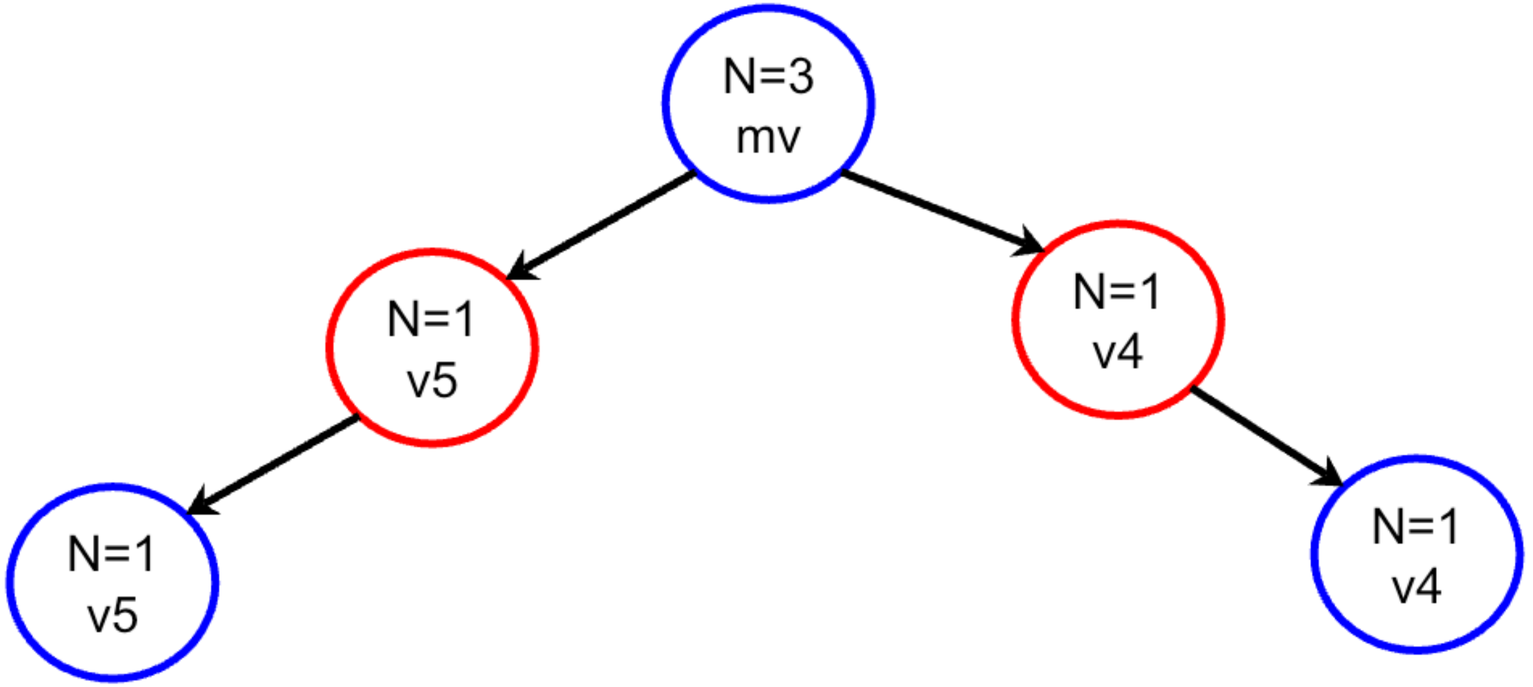


POMCP

PRUNING



POMCP



POMCP

- The Law of Large Numbers states that as the number of trials or samples increases, the **sample mean** of the results will get closer and closer to the **expected value** (theoretical mean) of the population.

- We need to make optimal choices to reduce the error in value as fast as possible

- Formalism:-
  - "Do a rollout" → **Rollout policy** $\pi\text{rollout}(b, a)$
  - "Choose an action" → **Selection policy** $\pi\text{tree}(b, a)$

POMCP

Rollout policy

$$\pi_{\text{rollout}}(b, a) = \mathbb{P}[A^{(t)}_{\text{rollout}} = a \mid B^{(t)}_{\text{rollout}} = b]$$

- Lightweight

$$a \sim \pi_{\text{rollout}}(b, \cdot)$$

- **Move ordering** is important for rollouts of a better quality, which estimate the true value better.

POMCP

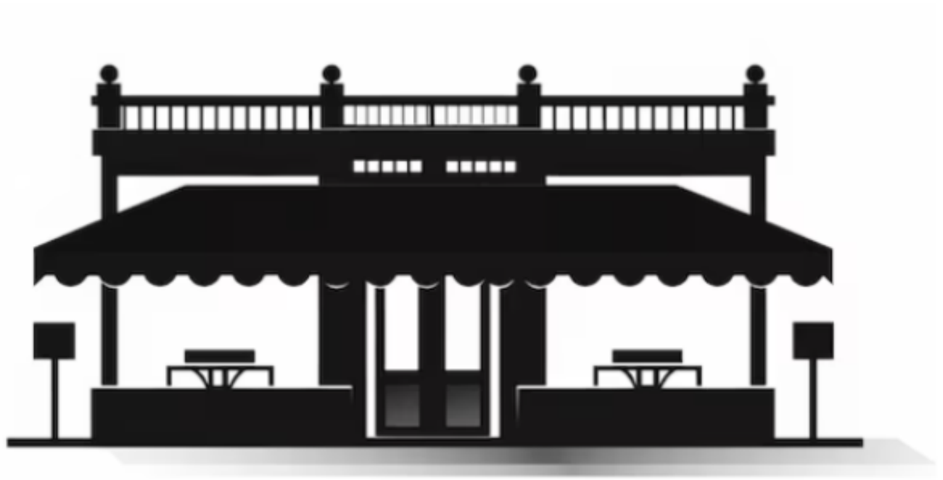
Selection policy

$$\pi_{\text{tree}}(b, a) = \mathbb{P}[A_t = a \mid B_t = b]$$

- A better selection policy focuses the search better.
- Simple random sample.
- Exploration vs. exploitation trade-off.

POMCP

Exploration vs. Exploitation



POMCP

Exploration vs. Exploitation: PO-UCT

b' is the belief state obtained on executing action a in belief state b

$$V^{\oplus}(b,a) = V(b') + C\sqrt{\frac{\ln N(b)}{N(b')}}$$

- C is the exploration constant.

$$\pi_{\text{tree}}(b,\cdot) = \arg\max_a \; V^{\oplus}(b,a)$$

$$V^{\oplus}(b,a) = w_a^{(1)} V(b') + w_a^{(2)} C\sqrt{\frac{\ln N(b)}{N(b')}}$$

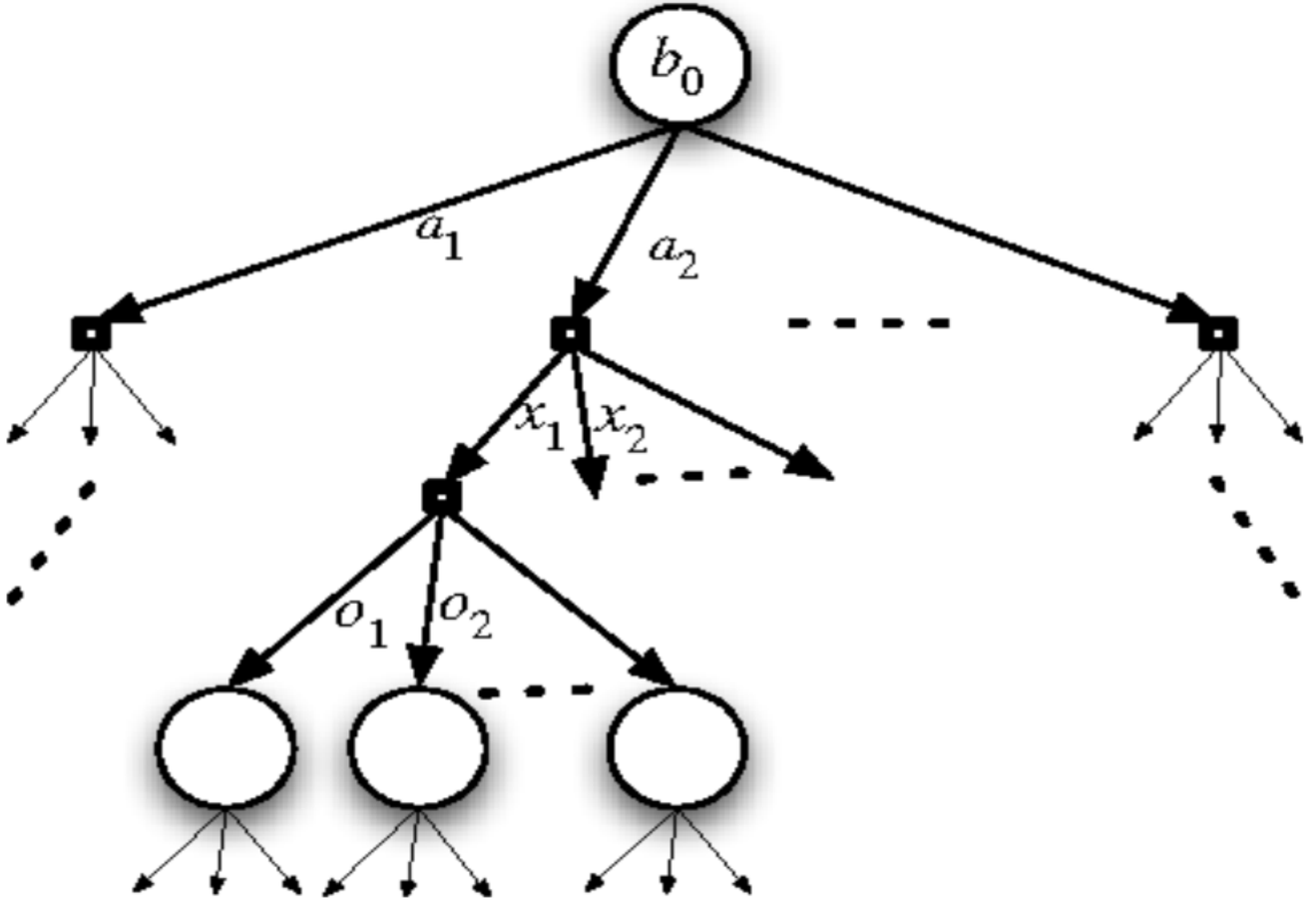
POMCP

Parameters

- N **:** = number of simulations
- γ **:** = discount factor
- **w**a, C **:** = PO-UCT params
- ε **:** = discount horizon

Introduction

- Exhaustive tree search: Online planning



POMCP

Algorithm 1 Partially Observable Monte-Carlo Planning

procedure SEARCH(h)
  repeat
    if $h = empty$ **then**
      $s \sim \mathcal{I}$
    else
      $s \sim B(h)$
    end if
    SIMULATE($s, h, 0$)
  until TIMEOUT()
  return $\underset{b}{\text{argmax}}\ V(hb)$
end procedure

procedure ROLLOUT($s, h, depth$)
  if $\gamma^{depth} < \epsilon$ **then**
    return 0
  end if
  $a \sim \pi_{rollout}(h, \cdot)$
  $(s', o, r) \sim \mathcal{G}(s, a)$
  return r $+ \gamma$.ROLLOUT($s', hao, depth{+}1$)
end procedure

procedure SIMULATE($s, h, depth$)
  if $\gamma^{depth} < \epsilon$ **then**
    return 0
  end if
  if $h \notin T$ **then**
    for all $a \in \mathcal{A}$ **do**
      $T(ha) \leftarrow (N_{init}(ha), V_{init}(ha), \emptyset)$
    end for
    return ROLLOUT($s, h, depth$)
  end if
  $a \leftarrow \underset{b}{\text{argmax}}\ V(hb) + c\sqrt{\frac{\log N(h)}{N(hb)}}$
  $(s', o, r) \sim \mathcal{G}(s, a)$
  $R \leftarrow r + \gamma$.SIMULATE($s', hao, depth + 1$)
  $B(h) \leftarrow B(h) \cup \{s\}$
  $N(h) \leftarrow N(h) + 1$
  $N(ha) \leftarrow N(ha) + 1$
  $V(ha) \leftarrow V(ha) + \frac{R - V(ha)}{N(ha)}$
  return R
end procedure

Demo


Percept: <none,none,breeze,none,non
Current score: -1
Last action: GO_FORWARD
Time step: 1


Percept: <none,none,breeze,none,non
Current score: -3
Last action: GO_FORWARD
Time step: 3

Reference

D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," Advances in Neural Information Processing Systems, vol. 23, 2010.

Monte-Carlo Planning in Large POMDPs

David Silver
MIT, Cambridge, MA 02139
davidstarsilver@gmail.com

Joel Veness
UNSW, Sydney, Australia
jveness@gmail.com

Abstract

This paper introduces a Monte-Carlo algorithm for online planning in large POMDPs. The algorithm combines a Monte-Carlo update of the agent's belief state with a Monte-Carlo tree search from the current belief state. The new algorithm, *POMCP*, has two important properties. First, Monte-Carlo sampling is used to break the curse of dimensionality both during belief state updates and during planning. Second, only a black box simulator of the POMDP is required, rather than explicit probability distributions. These properties enable POMCP to plan effectively in significantly larger POMDPs than has previously been possible. We demonstrate its effectiveness in three large POMDPs. We scale up a well-known benchmark problem, *rocksample*, by several orders of magnitude. We also introduce two challenging new POMDPs: 10×10 *battleship* and *partially observable PacMan*, with approximately 10^{18} and 10^{56} states respectively. Our Monte-Carlo planning algorithm achieved a high level of performance with no prior

Introduction

It is a ***full-width planning*** algorithm.

It relies on an evaluation function (fallible heuristics) that estimates the value.

Monte-Carlo Method

- Curse of dimensionality? → Randomly sampled **simulations**
- Fallible evaluation? → Value estimates are based on simulated experience.

- A simulation from a belief state samples a state, and chooses actions, observations, and next states until a terminal state is reached. It is an *episode of experience* sampled from the POMDP.

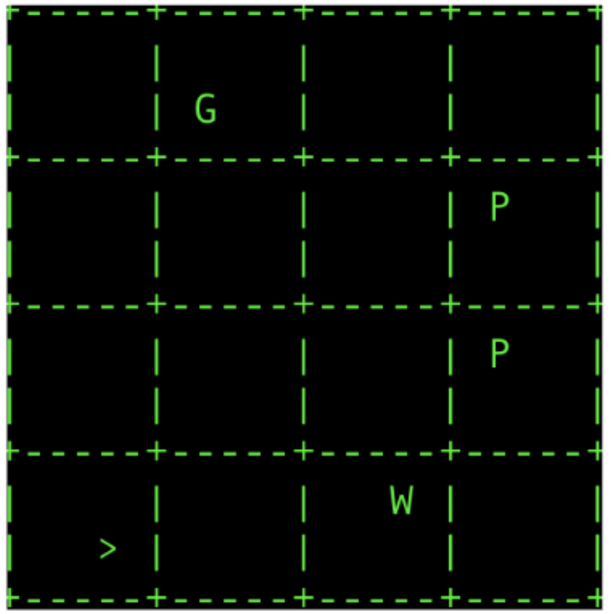
- The value of the belief state is estimated as the **average value** of all simulations carried out from that belief state.

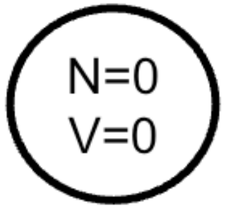
- In each simulation, the first new belief state encountered is added to the search tree.

Monte-Carlo Method

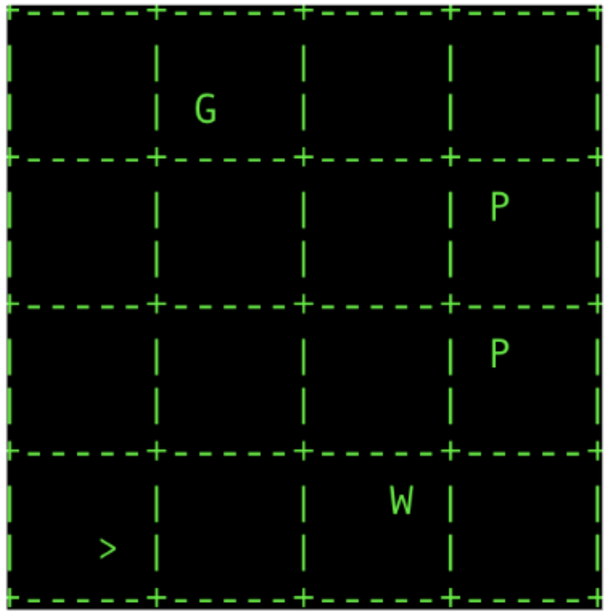
- In the wumpus world, we know that the agent starts out from (1, 1), facing east.
- The initial belief state *b0* in the wumpus world defines a uniform prior on 15C2 × 15 × 16 = 25,200 states. Note that all subsequent belief states are uniform distributions.
- The belief state is updated from *b0* to *b1* on observing the first percept.
- The Monte-Carlo search tree is initially empty. The first node added is *b1*.

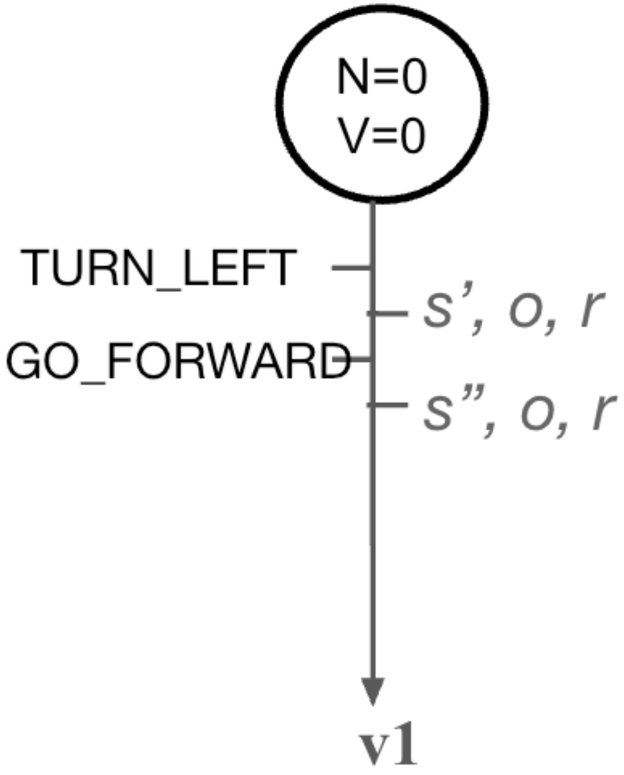
POMCP

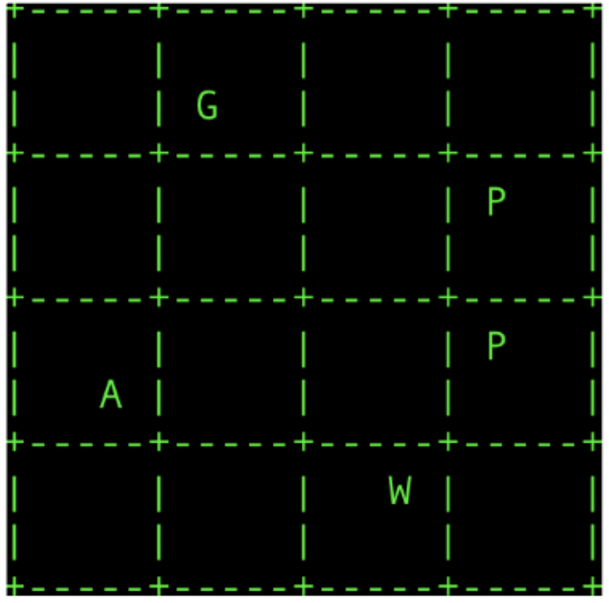




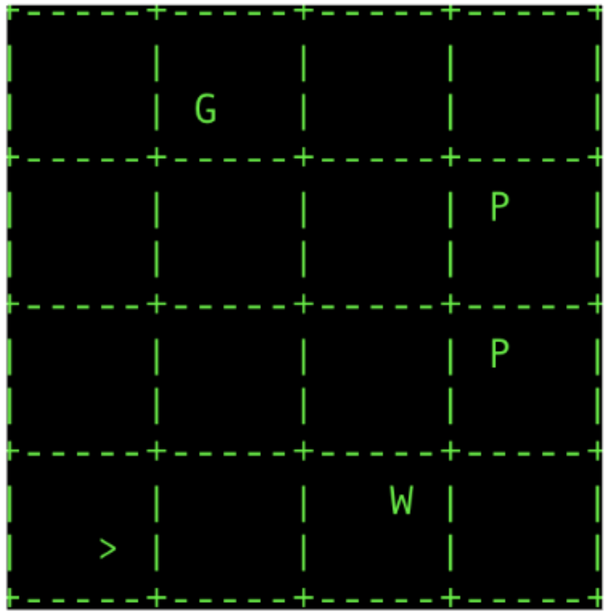
POMCP







POMCP



N=1
v1