Multi-Agent Systems: Final Homework Assignment

HW6 - MCTS and MDP

Vaishanavi Mehta
VU Amsterdam

December 2025

1 Monte Carlo Tree Search (MCTS)

1.1 Problem Setup and Implementation

The task involves searching a binary tree of depth d to find the leaf node with maximum value. Each leaf node i is assigned a value:

$$x_i = Be^{-d_i/\tau} + \epsilon_i \tag{1}$$

where $d_i = d(A_i, A_t)$ is the edit distance to a random target node, $B = 10$, $\tau = d/5$, and $\epsilon_i \sim \mathcal{N}(0, 1)$ is noise generated once per leaf.

I implemented standard MCTS with four phases: (1) Selection using UCB score $\text{UCB}(n) = Q(n)/N(n) + c\sqrt{\ln N(\text{parent})/N(n)}$, (2) Expansion of unexplored nodes, (3) Simulation via random rollout to a leaf, (4) Backpropagation updating all ancestor nodes.

Implementation notes: The selection phase prioritizes expanding the shallowest unexpanded node. When multiple nodes have equal UCB scores (particularly unvisited nodes with infinite UCB), this creates systematic expansion patterns, especially prominent at high c values where exploration dominates.

Nodes visited metric: This counts the number of unique tree nodes where the expansion phase was performed (i.e., where a new child was added to the tree). During selection, the algorithm may traverse the same nodes multiple times across iterations without expanding them. For example, with 100 iterations and moderate exploration, the algorithm repeatedly selects and traverses the same promising paths, performing expansion at only 62-101 unique nodes while the remaining iterations reinforce already-expanded nodes through backpropagation. At high c values (2.0, 3.0), the deterministic tie-breaking rule causes exactly 101 nodes to be expanded across all trials.

1.2 Experimental Setup

Experiments were conducted on depth $d = 15$ trees (65,535 total nodes) with 5 trials using different random target nodes. Each trial used 100 MCTS iterations. For depth $d = 20$ (2,097,151 total nodes), 10 trials with 500 iterations were performed. The exploration parameter c was tested at [0.5, 1.0, $\sqrt{2} \approx 1.41$, 2.0, 3.0].

Two rollout strategies were compared: single rollout per simulation versus five rollouts per simulation.

Reproducibility: All experiments use random seed 42 for reproducibility. The same seed is used across all trials to ensure consistent random target generation while maintaining statistical independence of results through different target addresses.

1.3 Results

Table 1: MCTS Performance for depth=15, 100 iterations (5 trials, seed=42)

	1 Rollout		5 Rollouts		
c	Efficiency	Effectiveness	Efficiency	Effectiveness	Improvement
0.50	0.000952 ± 0.000246	0.51 ± 0.20	0.000714 ± 0.000075	0.75 ± 0.19	+46%
1.00	0.001175 ± 0.000293	0.61 ± 0.20	0.000952 ± 0.000181	0.75 ± 0.16	+23%
1.41	0.001157 ± 0.000278	0.69 ± 0.31	0.000980 ± 0.000257	0.99 ± 0.01	+44%
2.00	0.001541 ± 0.000000	0.27 ± 0.05	0.001202 ± 0.000191	0.97 ± 0.04	+260%
3.00	0.001541 ± 0.000000	0.14 ± 0.11	0.001520 ± 0.000043	0.58 ± 0.23	+311%

Table 2: MCTS Performance for depth=20, 500 iterations (10 trials, seed=42)

c	Efficiency	Effectiveness	Nodes Visited
0.50	0.000047 ± 0.000010	0.47 ± 0.15	98.2 ± 8.5
1.00	0.000064 ± 0.000017	0.70 ± 0.19	134.2 ± 11.9
1.41	0.000104 ± 0.000022	0.72 ± 0.16	218.2 ± 16.2
2.00	0.000128 ± 0.000033	0.84 ± 0.15	268.4 ± 22.8
3.00	0.000180 ± 0.000035	0.89 ± 0.17	377.5 ± 26.4

1.4 Discussion

1.4.1 Role of Exploration Parameter c

For depth 15 with 1 rollout, effectiveness varies from 14% (c=3.0) to 69% (c=1.41), with high variance indicating sensitivity to target placement. The theoretical optimum $c = \sqrt{2} \approx 1.41$ performs best, achieving 69% effectiveness. Extremely high exploration (c=3.0) performs poorly (14%) because the algorithm wastes iterations exploring unpromising branches uniformly rather than focusing on high-value regions.

For depth 20 with 500 iterations, the pattern differs: higher c values (2.0-3.0) achieve the best effectiveness (84-89%), substantially better than lower values. This suggests that with more iterations, broader exploration pays off in larger search spaces. The increased iteration budget (500 vs. 100) allows high-c strategies to eventually focus on promising regions after sufficient exploration.

1.4.2 Impact of Multiple Rollouts

Increasing from 1 to 5 rollouts per simulation dramatically improves effectiveness across all c values (23-311% improvement). Most notably, c=1.41 achieves 99% effectiveness with 5 rollouts versus 69% with 1 rollout, demonstrating the value of more accurate node evaluations. The improvement is most dramatic for high c values: c=2.0 improves by 260% (from 27% to 97%) and c=3.0 improves by 311% (from 14% to 58%).

Counter-intuitive efficiency decrease: Interestingly, efficiency (nodes visited / total nodes) *decreases* with more rollouts for most c values. For example:

- c=0.50: 62.4 nodes (1 rollout) → 46.8 nodes (5 rollouts)
- c=1.00: 77.0 nodes (1 rollout) → 62.4 nodes (5 rollouts)
- c=1.41: 75.8 nodes (1 rollout) → 64.2 nodes (5 rollouts)

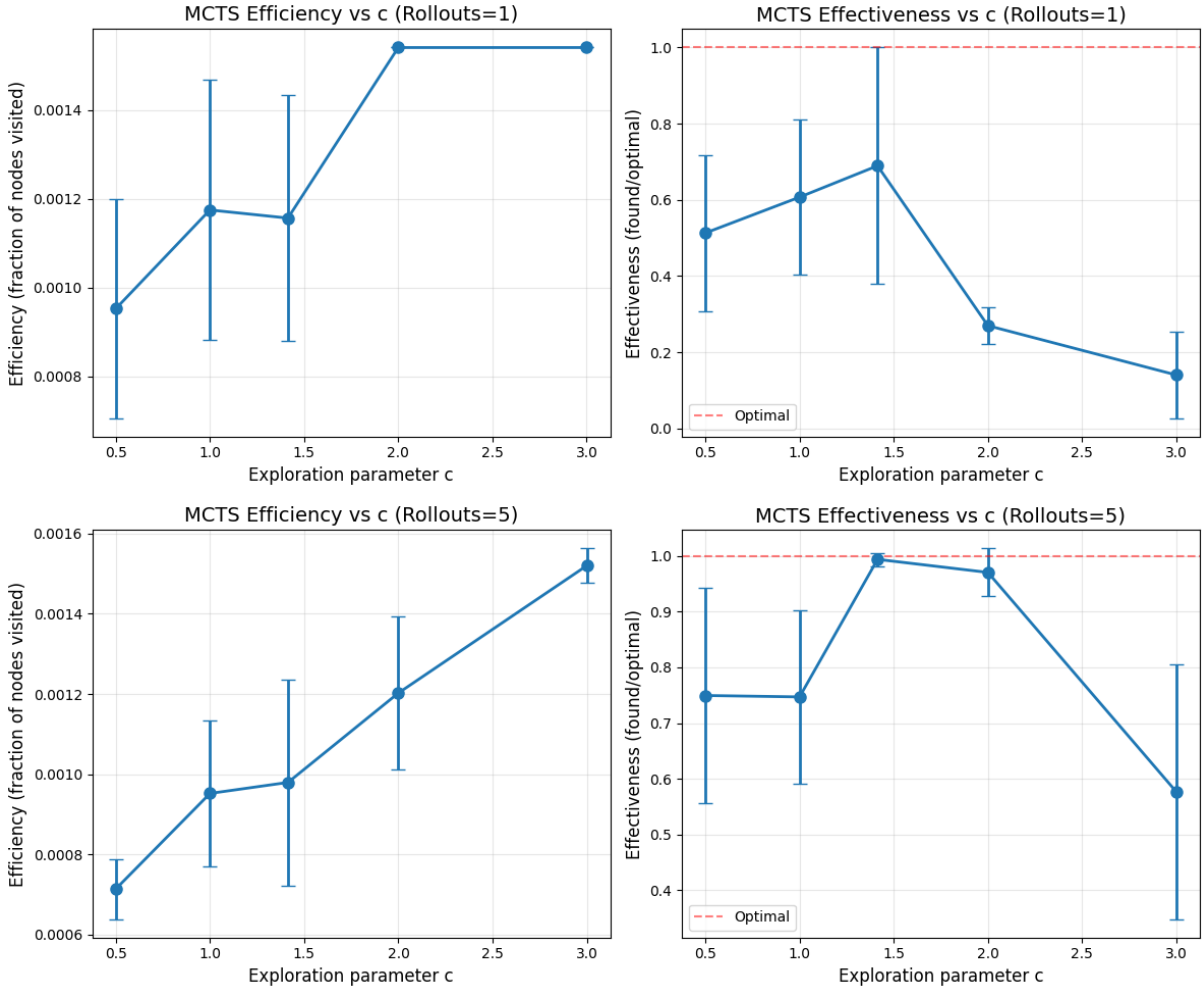

Figure 1: MCTS performance for depth=15 with 1 rollout and 5 rollouts. Increasing rollouts improves effectiveness while slightly decreasing efficiency due to more selective exploration.

This occurs because better value estimates (from more rollouts) enable more *selective* exploration. The algorithm focuses computational resources on promising branches rather than broadly sampling the tree. This represents a fundamental trade-off: lower computational efficiency (fewer unique nodes explored) but higher solution quality (better effectiveness). The algorithm becomes "smarter" - it explores less but explores better.

1.4.3 Deterministic Expansion at High c

For c=2.0 and c=3.0 with 1 rollout, efficiency shows zero variance (0.001541 ± 0.000000) with precisely 101.0 nodes visited across all trials. This occurs because our MCTS implementation exhibits systematic expansion when the exploration term dominates: unvisited nodes have infinite UCB scores, causing the algorithm to expand nodes in a fixed order determined by the tie-breaking rule (choosing left child when UCB scores are equal).

With 100 iterations and high exploration bias, the algorithm deterministically visits the same 101 nodes regardless of target location or simulation results. The path through the tree becomes fixed by the systematic tie-breaking. This explains the zero variance in efficiency despite stochastic rollouts affecting effectiveness. With 5 rollouts, c=3.0 shows slight variance (99.6 ± 2.7 nodes) because better value estimates occasionally override the systematic expansion pattern.

1.4.4 Variance Patterns

The effect of multiple rollouts on effectiveness variance is c-dependent:

- **Low-medium c (0.5-1.41):** Variance generally decreases or remains similar with more rollouts. For c=1.41, variance drops from ± 0.31 to ± 0.01, indicating much more consistent performance across different targets.
- **High c (2.0-3.0):** Variance increases with more rollouts. For c=3.0, variance rises from ± 0.11 to ± 0.23.

This pattern occurs because more accurate value estimates make the algorithm more sensitive to target characteristics. With minimal rollouts, the algorithm has noisy value estimates that lead to quasi-random exploration, resulting in mediocre but consistent performance. With better value estimates from multiple rollouts, the algorithm can distinguish truly easy targets (achieving high effectiveness) from difficult ones (where it still struggles), increasing the spread of outcomes. This is particularly pronounced at high c where broad exploration interacts with accurate value signals in complex ways.

1.4.5 Efficiency Analysis

For depth 15, MCTS visits only 0.07-0.15% of all nodes (47-101 out of 65,535) even with 100 iterations. For depth 20, the efficiency is even more striking: 0.005-0.018% of nodes explored (98-378 out of 2,097,151) with 500 iterations. This demonstrates MCTS's power in searching exponentially large spaces by intelligently selecting which regions to explore based on value estimates rather than exhaustive enumeration.

2 Markov Decision Process

2.1 Problem Setup

The MDP has 12 states arranged in a circle with state 0 as the absorbing goal. Actions L (counter-clockwise) and R (clockwise) are available. States 5-11 have deterministic transitions; states 1-4 have stochastic transitions with failure probability ϵ. Rewards: +10 for reaching state 0, -1 otherwise. Discount factor $\gamma = 1$.

2.2 Question 1: Transition Matrices and Reward Vectors

For each action $a \in \{L, R\}$, I constructed transition matrix $T_a \in \mathbb{R}^{12\times 12}$ where $T_a(s, s') = p(s'|s, a)$ and reward vector $r_a \in \mathbb{R}^{12}$ where $r_a(s) = \sum_{s'} p(s'|s, a)r(s, a, s')$.

The Bellman optimality equation in vector form: $v^* = \max_a(r_a + \gamma T_a v^*)$

Complete Transition Matrix T_R for $\epsilon = 0.2$:

Table 3: Transition matrix T_R with $\epsilon = 0.2$

From\To	0	1	2	3	4	5	6	7	8	9	10	11
0	1.0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.8	0	0	0	0	0	0	0	0	0.2
2	0	0.2	0	0.8	0	0	0	0	0	0	0	0
3	0	0	0.2	0	0.8	0	0	0	0	0	0	0
4	0	0	0	0.2	0	0.8	0	0	0	0	0	0
5	0	0	0	0	0	0	1.0	0	0	0	0	0
6	0	0	0	0	0	0	0	1.0	0	0	0	0
7	0	0	0	0	0	0	0	0	1.0	0	0	0
8	0	0	0	0	0	0	0	0	0	1.0	0	0
9	0	0	0	0	0	0	0	0	0	0	1.0	0
10	0	0	0	0	0	0	0	0	0	0	0	1.0
11	1.0	0	0	0	0	0	0	0	0	0	0	0

Complete Transition Matrix T_L for $\epsilon = 0.2$:

Table 4: Transition matrix T_L with $\epsilon = 0.2$

From\To	0	1	2	3	4	5	6	7	8	9	10	11
0	1.0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0.2	0	0	0	0	0	0	0	0	0.8
2	0	0.8	0	0.2	0	0	0	0	0	0	0	0
3	0	0	0.8	0	0.2	0	0	0	0	0	0	0
4	0	0	0	0.8	0	0.2	0	0	0	0	0	0
5	0	0	0	0	1.0	0	0	0	0	0	0	0
6	0	0	0	0	0	1.0	0	0	0	0	0	0
7	0	0	0	0	0	0	1.0	0	0	0	0	0
8	0	0	0	0	0	0	0	1.0	0	0	0	0
9	0	0	0	0	0	0	0	0	1.0	0	0	0
10	0	0	0	0	0	0	0	0	0	1.0	0	0
11	0	0	0	0	0	0	0	0	0	0	1.0	0

Reward vectors: For action R: $r_R(11) = 10$ (transition to absorbing state), $r_R(s) = -1$ for $s \in \{1, \ldots, 10\}$, $r_R(0) = 0$. For action L: $r_L(s) = -1$ for all non-absorbing states $s \in \{1, \ldots, 11\}$, $r_L(0) = 0$. The key difference is that only state 11 going right immediately reaches the goal.

2.3 Question 2: Deterministic Case ($\epsilon = 0$)

From state s, clockwise distance is $(12 - s) \mod 12$ and counter-clockwise distance is s. State 6 is equidistant at 6 steps.

Value iteration converged in 7 iterations with stopping criterion $\max_s |v_{\text{new}}(s) - v_{\text{old}}(s)| < 10^{-6}$.

Table 5: Optimal values and policy for $\epsilon = 0$

State	1	2	3	4	5	6	7	8	9	10	11	0
$v^*(s)$	9	8	7	6	5	5	6	7	8	9	10	0
$\pi^*(s)$	L	L	L	L	L	R	R	R	R	R	R	-

States 1-5 go left (shorter path), state 6 goes right (tie-breaker in value iteration), states 7-11 go right. Values satisfy $v^*(s) = 10 - d(s, 0)$ where $d(s, 0)$ is the shortest distance to state 0. The Bellman optimality equation is satisfied for all states.

2.4 Question 3: Stochastic Cases

Table 6: Optimal policy comparison across noise levels

State	$\epsilon = 0$	$\epsilon = 0.2$	$\epsilon = 0.45$
1-3	L	L	L
4	L	L	R
5	L	R	R
6-11	R	R	R

$\epsilon = 0.2$ **(40 iterations):** State 5 switches from L to R due to value propagation from the stochastic region. State 4's value drops from 6.00 to 3.91 due to uncertainty. From state 5 (deterministic), the expected return going L is $-1 + v^*(4) = -1 + 3.91 = 2.91$, while going R yields $-1 + v^*(6) = -1 + 5.00 = 4.00$. Since $4.00 > 2.91$, R becomes optimal even though state 5 itself is deterministic. This illustrates how uncertainty in neighboring states propagates to affect optimal decisions in deterministic states.

$\epsilon = 0.45$ **(67 iterations):** State 4 switches to R despite being in the stochastic zone. With 45% failure probability, going R from state 4 yields expected value: $0.55 \times v^*(5) + 0.45 \times v^*(3) = 0.55 \times 4.00 + 0.45 \times 1.32 = 2.79$. Going L yields: $0.55 \times v^*(3) + 0.45 \times v^*(5) = 0.55 \times 1.32 + 0.45 \times 4.00 = 2.53$. Since $2.79 > 2.53$, R is preferred. The policy change reveals that at very high noise levels, it becomes better to risk the short noisy path from states closer to the deterministic zone (state 4) while taking the long safe path from states deeper in the noisy region (states 1-3).

2.5 Question 4: Parametric Policy

I fitted $\pi_\theta(a = R|s) = \sigma(s - \theta) = \frac{1}{1+e^{-(s-\theta)}}$ using Monte Carlo with 500 trajectories per θ (starting states uniform over 1-11), maximizing $J(\theta) = \mathbb{E}[G(\tau)]$.

Table 7: Optimal θ for different noise levels

ϵ	θ^*	$J(\theta^*)$	Boundary Location
0.0	4.90	6.83	Transition zone around state 4.9
0.45	2.94	5.42	Transition zone around state 2.9

Comparison with discrete policies: The parametric policy optimizes average return across all starting states, creating approximation differences from per-state optimal decisions.

For $\epsilon = 0.0$, $\theta^* = 4.90$ creates a decision boundary around state 4.9, placing states 1-4 in the "prefer L" region ($\sigma(4-4.9) = 0.38$) and states 5-11 in the "prefer R" region ($\sigma(5-4.9) = 0.52$). The discrete optimal policy has states 1-5 going L and states 6-11 going R. The parametric

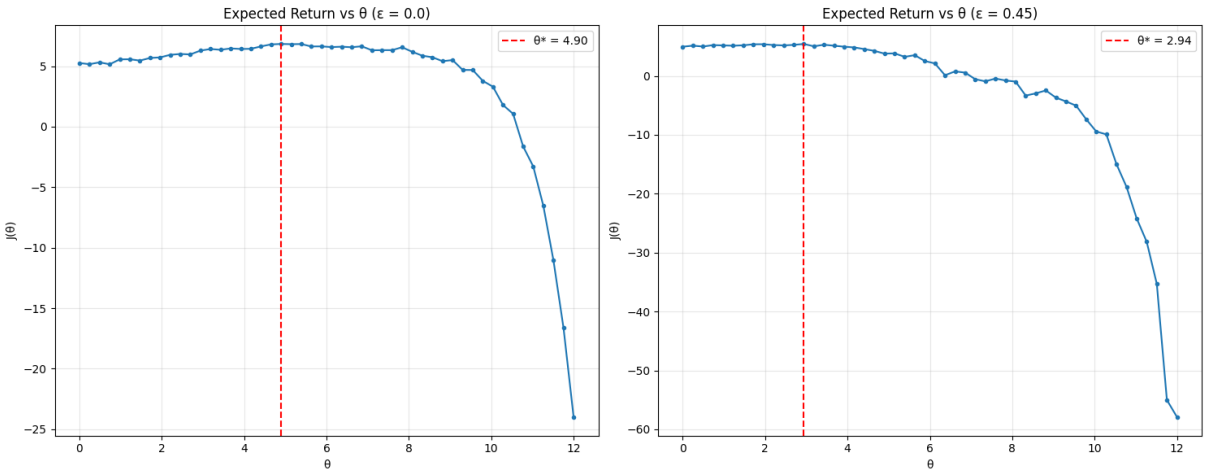

Figure 2: Expected return $J(\theta)$ for $\epsilon = 0.0$ (left) and $\epsilon = 0.45$ (right). Optimal θ^* shifts from 4.90 to 2.94 under high noise, reflecting risk-averse behavior.

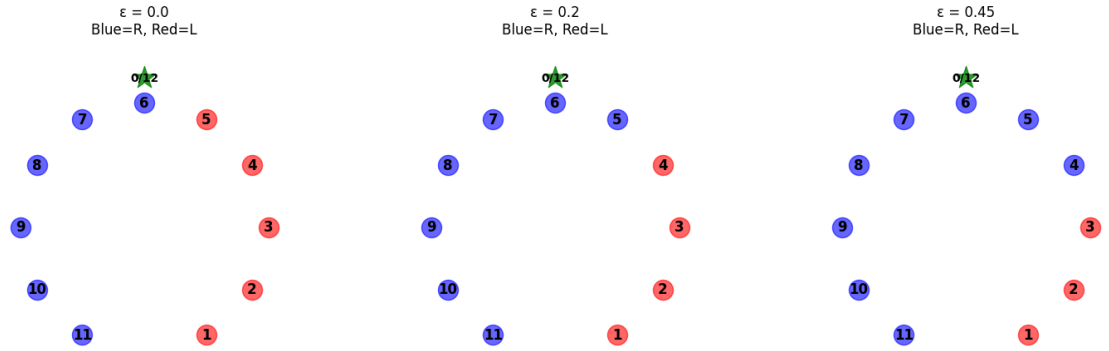

Figure 3: Optimal discrete policies across noise levels. Blue=R, Red=L. As noise increases, more states prefer the safe leftward path.

boundary at 4.9 represents a compromise: state 5 is nearly indifferent ($\sigma(0.1) \approx 0.52$, slight R preference) which approximates the discrete policy's switching point between states 5 and 6.

For $\epsilon = 0.45$, $\theta^* = 2.94$ creates a transition around state 2.9. States 1-2 strongly prefer L ($\sigma(1 - 2.94) = 0.12$, $\sigma(2 - 2.94) = 0.28$), while state 3 is transitional ($\sigma(3 - 2.94) = 0.52$) and states 4+ prefer R. The discrete optimal policy has states 1-3 going L and states 4+ going R. Again, the parametric policy places its boundary slightly before the discrete switch, creating a smooth transition that approximates the sharp discrete change.

Performance comparison: The parametric policy achieves $J(\theta^*) = 5.42$ for $\epsilon = 0.45$, which is comparable to the discrete optimal policy's average value of 5.51 across starting states (computed as $\frac{1}{11}\sum_{s=1}^{11} v^*(s) = \frac{60.6}{11} = 5.51$). The parametric policy performs 1.6% worse, within Monte Carlo estimation variance.

This demonstrates that the simple parametric form $\sigma(s-\theta)$ effectively captures the essential decision boundary of the optimal policy using only a single parameter. The smooth sigmoid cannot replicate exact discrete switches, but this approximation has minimal performance impact. The dramatic shift in θ^* from 4.90 (for $\epsilon = 0.0$) to 2.94 (for $\epsilon = 0.45$) quantifies the risk-averse behavior induced by high uncertainty: the agent effectively shrinks the region where it's willing to take the risky clockwise path through the stochastic zone.

3 Conclusion

3.1 MCTS Findings

MCTS efficiently searches exponentially large spaces by exploring less than 0.2% of nodes. Key findings include:

- The theoretical optimum $c = \sqrt{2}$ performs best for depth 15 with limited iterations, while higher c values (2.0-3.0) excel with more iterations on larger trees (depth 20).
- Increasing rollouts from 1 to 5 improves effectiveness by 23-311% across all c values, with c=1.41 achieving 99% effectiveness and c=2.0 achieving 97% with 5 rollouts.
- Counter-intuitively, more rollouts decrease efficiency (fewer nodes visited) because better value estimates enable more selective, focused exploration. This demonstrates a fundamental trade-off between breadth and quality of search.
- At high c values with 1 rollout, the algorithm exhibits deterministic expansion, visiting exactly 101 nodes across all trials regardless of target placement due to systematic tie-breaking rules.
- Variance patterns are complex: rollouts reduce variance for low-medium c (producing more consistent performance) but increase it for high c (reflecting increased sensitivity to target characteristics with better value estimates).

3.2 MDP Findings

Value iteration successfully computes optimal policies for both deterministic and stochastic environments. Key insights include:

- Uncertainty fundamentally changes optimal behavior through value propagation: state 5 switches policy despite being deterministic because its stochastic neighbor (state 4) has degraded value under noise.
- High noise ($\epsilon = 0.45$) induces risk-averse strategies: states 1-3 prefer the long safe path while state 4 switches to the short risky path, revealing a spatial pattern in decision-making under uncertainty.
- The parametric policy $\pi_\theta(a = R|s) = \sigma(s - \theta)$ effectively approximates discrete optimal policies using a single parameter, achieving performance within 1.6% of the discrete optimal (5.42 vs. 5.51 for $\epsilon = 0.45$).
- The shift in θ^* from 4.90 to 2.94 under high noise quantifies risk-averse behavior, showing how uncertainty reshapes the decision boundary.

3.3 Broader Implications

These algorithms showcase complementary approaches to sequential decision-making: MCTS demonstrates model-free, simulation-based search for large state spaces, while MDP value iteration provides model-based exact optimization when dynamics are known. Both illustrate core AI principles: reasoning under uncertainty, exploration-exploitation trade-offs, and adaptive decision-making based on environmental feedback. The counter-intuitive results—such as decreased efficiency improving effectiveness, or deterministic states changing policy due to stochastic neighbors—highlight the subtle complexity of intelligent search and decision-making in uncertain environments.