LAB VIII : Policy Iteration for the Gbike Bicycle Rental Problem

Soham Naukudkar, Siddhikesh Gavit, Shreyash Borkar
Department of Computer Science and Engineering
Indian Institute of Information Technology, Vadodara
Email: {202351135, 202351040, 202351132}@iiitvadodara.ac.in
Under the Guidance of Prof. Pratik Shah

Code repository for this work: `github`

***Abstract*—This report presents a complete mathematical and algorithmic solution for the Gbike Bicycle Rental problem formulated as a Markov Decision Process (MDP). Two variants are solved using Policy Iteration: (1) the standard two-location model (Problem 2), and (2) an extended model with free transfer, asymmetric move costs, and parking penalties (Problem 3). The report explains the problem statements, derives the expected-return equations using Poisson distributions, describes the implementation logic, shows full textual outputs including the optimal policy matrices, and interprets the results. Both problems converged in 5 iterations with significant policy differences due to the additional constraints.**

I. Learning Objectives

- Formulate the Gbike system as a finite MDP with discrete states and actions.
- Derive the expected-return $Q(s,a)$ using Poisson rental/return models.
- Implement Policy Iteration: policy evaluation and greedy policy improvement.
- Interpret optimal policy matrices and value function ranges under different constraints.
- Analyze the impact of asymmetric costs and operational constraints on optimal policies.

II. Introduction

Gbike is a two-location bicycle rental service. Each day customers request bicycles and return them; requests and returns can be modeled by Poisson processes. Each successful rental yields revenue; moving bikes overnight between locations incurs costs. The goal is to determine an overnight transfer policy that maximizes expected discounted long-run reward. We use Policy Iteration to compute the exact optimal policy on the discrete state space.

III. MDP Formulation and Expected Return

A. State and Action Space

- State $s = (b_1, b_2)$, where $b_i \in \{0, 1, \dots, 20\}$ denotes bikes at location i at the start of the night.
- Action $a \in \{-5, -4, \dots, 5\}$. $a > 0$ means move a bikes from location 1 to 2; $a < 0$ means move $|a|$ bikes from 2 to 1.

B. Transition Dynamics Summary

Overnight:

1) Apply the action a (with bounds: final bikes clipped to $[0, 20]$).
2) Requests $R_1 \sim \text{Poisson}(\lambda_1 = 3)$, $R_2 \sim \text{Poisson}(\lambda_2 = 4)$ are sampled.
3) Serve up to available bikes; each served rental yields reward (INR 10).
4) Returns $T_1 \sim \text{Poisson}(\mu_1 = 3)$, $T_2 \sim \text{Poisson}(\mu_2 = 2)$ are sampled.
5) Next-state s' is computed and clipped to $[0, 20]$.

C. Expected-Return Equation (Single Action)

For a state s and action a we compute:

$$\begin{aligned} Q(s,a) = &-C(a) \\ &+ \sum_{r_1,r_2} P(r_1)\,P(r_2)\Big[R(r_1,r_2;s,a) \\ &\qquad + \gamma \sum_{t_1,t_2} P(t_1)\,P(t_2)\,V(s_{\text{next}})\Big]. \end{aligned}$$

where:

- $C(a)$ is the immediate movement cost (problem-dependent).
- $R(r_1, r_2; s, a)$ is the rental income obtained that night (INR 10 per served rental).
- $P(\cdot)$ are Poisson probabilities precomputed and truncated at MAX_POISSON=11 for speed.
- $V(s')$ is the value of the next state after returns.
- $\gamma = 0.9$ is the discount factor.

D. Policy Iteration Algorithm

1) Initialize policy π (e.g., zeros).
2) **Policy Evaluation:** solve $V(s) = Q(s, \pi(s))$ iteratively until $\max_s |V_{new}(s) - V(s)| < 10^{-4}$.
3) **Policy Improvement:** for each state s, choose $\pi(s) = \arg\max_a Q(s,a)$ checking only valid as for that state.
4) Repeat until policy is stable.

IV. Part 1 — Problem 2 (Standard Gbike)

A. Problem Statement

- Bikes per location capped at 20.
- Max move per night: 5 bikes.
- Rental reward: INR 10 per served rental.

- Movement cost: INR 2 per bike moved (symmetric).
- Poisson means: rental requests $\lambda = (3, 4)$, returns $\mu = (3, 2)$.
- Discount factor: $\gamma = 0.9$.

B. Implementation Details

Implementation notes (matching ‘2.py‘ logic):

- Precompute Poisson probabilities up to `MAX_POISSON` for both requests and returns; caching reduces repeated work.
- For each state and candidate action:
  1) Apply movement with bounds to obtain bikes after movement.
  2) For all request pairs (r_1, r_2) compute served rentals and rental income.
  3) For all return pairs (t_1, t_2) compute next-state s' and accumulate discounted $V(s')$ weighted by Poisson probabilities.
  4) Combine immediate rental income and discounted expected future value, subtract move cost to get $Q(s, a)$.
- Policy evaluation uses iterative updates until $\Delta < 10^{-4}$.
- Policy improvement checks all valid actions (clipped by available bikes and max move).

C. Results and Analysis

Problem 2: Terminal Output

```
Starting Policy Iteration for Problem 2...
Policy Iteration 1... Changes: 318, Time: 772.86s, Stable: False
Policy Iteration 2... Changes: 260, Time: 854.12s, Stable: False
Policy Iteration 3... Changes: 82, Time: 576.88s, Stable: False
Policy Iteration 4... Changes: 10, Time: 393.85s, Stable: False
Policy Iteration 5... Changes: 0, Time: 200.45s, Stable: True
Converged after 5 iterations! Total time: 2798.17s (46.64 minutes)
PROBLEM 2 POLICY ANALYSIS
Policy Statistics: Min action: -4, Max action: 5, Mean action: 0.60
Move Directions: Location 1 → 2: 125 states, Location 2 → 1: 45 states, No movement: 271 states
Value Function range: [405.43, 617.01]
```

D. Problem 2 Optimal Policy Matrix

Interpretation:

- Large central region with zeros indicates many states where **no overnight movement** is optimal
- Positive values (1–5) in lower rows indicate shifts from L1 to L2 when L1 has surplus
- Negative values in upper-right region indicate moving bikes from L2 back to L1 when L2 is overstocked

TABLE I: Problem 2 Optimal Policy Matrix (Partial View)

Bikes at	**Bikes at Location 2**						
Loc 1	0-2	3-5	6-8	9-11	12-14	15-17	18-20
0-2	0	0	0	-1	-1 to -2	-2 to -3	-3 to -4
3-5	0	0	0	0 to -1	-1 to -2	-2 to -3	-3
6-8	0	0	0	0	0 to -1	-1 to -2	-2
9-11	1	0	0	0	0	0	-1 to -2
12-14	2	1	0	0	0	0	0
15-17	3	2	1	1	0	0	0
18-20	4-5	3-4	2-3	1-2	1	0	0

TABLE II: Detailed Policy Pattern Analysis for Problem 2

Pattern Type	**Description**
Conservative Zone	States with few bikes at both locations show minimal or negative movements (bikes moved to location 1)
Balancing Zone	Moderate bike counts show strategic movements from location 1 to 2 (positive values)
Aggressive Zone	High bike counts at location 1 show maximum allowed movements (up to +5 bikes)
No-Movement Belt	Large central region where maintaining current distribution is optimal

- The asymmetry reflects boundary conditions and demand rates ($\lambda_2 = 4 > \lambda_1 = 3$)

V. Part 2 — Problem 3 (Extended Model)

A. Problem Statement

Problem 3 modifies Problem 2 by:

- **Free first move:** First bike moved from Location 1 to 2 costs zero; subsequent bikes cost INR 2 each
- **Asymmetric movement costs:** Movement 2→1 costs INR 2 per bike (no free move)
- **Parking penalty:** INR 4 per location if bikes ¿ 10 overnight
- Other parameters same as Problem 2

B. Modified Cost Function

$$\text{MovementCost}(a) = \begin{cases} 2 \times \max(0, a-1) & \text{if } a > 0 \\ 2 \times |a| & \text{if } a < 0 \end{cases}$$

$$\text{ParkingPenalty}(s, a) = 4 \times \mathbb{I}[\text{bikes1} > 10] + 4 \times \mathbb{I}[\text{bikes2} > 10]$$

C. Implementation Details

Implementation details aligned with ‘3.py‘:

- Movement cost logic: if $a > 0$ then cost = $2 \cdot \max(0, a - 1)$ (first bike free), if $a < 0$ cost = $2|a|$
- After applying movement we check for parking penalties and subtract them before adding rental income
- Policy Iteration uses the same stopping tolerance for evaluation

D. Results and Analysis

Problem 3: Terminal Output

```
Starting Policy Iteration for Problem 3... Special Con-
ditions: First bike 1→2 FREE, Parking cost: INR 4
if bikes ¿ 10
Policy Iteration 1... Changes: 372, Time: 389.75s,
Stable: False
Policy Iteration 2... Changes: 264, Time: 319.81s,
Stable: False
Policy Iteration 3... Changes: 104, Time: 290.37s,
Stable: False
Policy Iteration 4... Changes: 1, Time: 2344.06s, Sta-
ble: False
Policy Iteration 5... Changes: 0, Time: 65.16s, Stable:
True
Converged after 5 iterations! Total time: 3409.15s
(56.82 minutes)
PROBLEM 3 POLICY ANALYSIS
Policy Statistics: Min action: -5, Max action: 5, Mean
action: 0.68
Move Directions: Location 1 → 2: 250 states (56.7%),
Location 2 → 1: 76 states (17.2%), No movement:
115 states (26.1%)
Parking Constraint: 247/441 states (56.0%) with po-
tential parking cost
Free Move Utilization: 154 states use exactly 1 bike
transfer (free)
Value Function range: [413.68, 583.37]
```

E. Problem 3 Optimal Policy Matrix

TABLE III: Problem 3 Optimal Policy Matrix (Partial View)

Bikes at	**Bikes at Location 2**						
Loc 1	0-2	3-5	6-8	9-11	12-14	15-17	18-20
0-2	0 to 1	0	0	-1 to -2	-2 to -4	-4 to -5	-5
3-5	1	0	0	-1	-2 to -4	-4 to -5	-3 to -4
6-8	1	0	0	0 to -1	-1 to -4	-4 to -5	-3
9-11	1	1	0	0	-1 to -4	0 to -1	-1
12-14	2	1	1	0 to -1	0	0	0
15-17	3	2	1	1	1	1	1
18-20	4-5	3-4	2-3	1-2	1	1	1

TABLE IV: Policy Changes from Problem 2 to Problem 3

Change Type	**Observation**
Increased 1→2 Movements	56.7% of states vs 28.3% in Problem 2 due to free movement incentive
Free Move Clustering	154 states specifically use +1 movement to leverage free transfers
Parking Avoidance	Policy shows patterns avoiding states where both locations exceed 10 bikes
Reduced Conservatism	Significant reduction in no-movement states (26.1% vs 61.5%)
Strategic Violations	Some states accept parking costs when movement benefits outweigh penalties

Interpretation:

- Large increase in states choosing positive actions (1→2) due to free movement incentive
- Many small positive integers (1 or 2) exploit free first transfer or avoid parking penalty
- Negative actions appear where L2 is overstocked relative to L1
- Parking violation statistics show policy shifts to mitigate penalties

VI. Comparative Analysis and Conclusions

TABLE V: Problem 2 vs Problem 3 Comparative Analysis

Metric	**Problem 2**	**Problem 3**
Convergence Iterations	5	5
Computation Time	46.64 min	56.82 min
Mean Action	0.60	0.68
L1→L2 Movements	28.3%	56.7%
No-Movement States	61.5%	26.1%
Value Range	[405.43, 617.01]	[413.68, 583.37]
Policy Complexity	Conservative	Dynamic

A. Key Insights

- **Convergence**: Policy Iteration converged in 5 iterations for both problems, demonstrating numerical stability
- **Value Ranges**: Problem 3 compresses maximal achievable value due to penalties, despite free move benefits
- **Policy Adaptation**: Optimal policies adapt intelligently to asymmetric cost structures
- **Incentive Effectiveness**: Free movement incentive significantly increases bike redistribution
- **Constraint Management**: Parking penalties successfully limit overnight storage at locations

VII. Implementation Verification

- **Code Structure**: `expected_return()` computes $Q(s, a)$ using nested Poisson sums
- **Algorithm**: Main policy iteration loop with evaluation and improvement phases
- **Numerical Stability**: Convergence achieved with $\Delta < 10^{-4}$, proper boundary handling
- **Efficiency**: Poisson probability caching and optimized NumPy operations

VIII. Conclusion

In this laboratory exercise, we successfully implemented Policy Iteration algorithms to solve two variants of the Gbike bicycle rental management problem:

- Developed MDP formulations for both basic and enhanced rental scenarios with realistic operational constraints
- Implemented efficient Policy Iteration that converged within 5 iterations for 441-state problems
- Derived optimal policies showing intuitive patterns of bike redistribution from lower-demand to higher-demand locations
- Demonstrated significant behavioral changes due to asymmetric costs and parking constraints
- Provided comprehensive policy analysis with statistical characterization of movement patterns

- Delivered actionable insights for real-world bicycle sharing system management

The results demonstrate that Policy Iteration is an effective algorithm for solving complex resource allocation problems, and the optimal policies provide valuable guidance for operational decision-making in shared mobility systems.

References

[1] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[2] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 2005.

[3] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2012.