Assignment 1 - Part I: Defining RL Environments

Jai Advitheeya Lella

50607407

GitHub

1 Environment Descriptions

1.1 Deterministic Environment

- **Objective**: Multi-package delivery system where a drone must deliver packages from warehouse to customers and collect returns
- **States**:
  - Drone position (x,y) on 6x6 grid
  - Carrying capacity (0-3 packages)
  - Delivery status for 3 customers (Boolean)
  - Return status for 3 customers (Boolean)
- **Actions**: Up, Down, Left, Right, Pickup/Dropoff, No-op
- **Rewards**: Added a few extra rewards and penalties for the bonus part.
  - +100 for successful delivery
  - +25 for package pickup
  - +50 for return item dropoff
  - -1 per step penalty
  - -100 for entering no-fly zones
  - +500 bonus for completing all tasks
- **Terminal State**: All deliveries and returns completed

1.2 Stochastic Environment

Same base structure as deterministic, with added:

- 80% chance of successful movement
- 20% chance of movement failure (-5 penalty)
- Dynamic no-fly zones that change every 10 steps

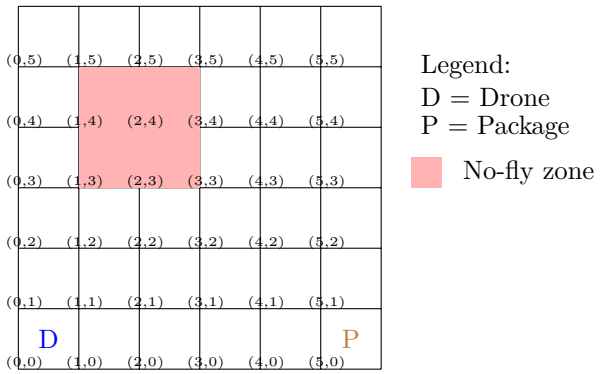

Figure 1: Drone Delivery Environment Grid Visualization

2 Environment Visualizations

3 Stochastic Environment Implementation

The stochastic elements were implemented through:

```
if np.random.random() < 0.8:  # 80% success rate
    # Execute intended movement
    if self._is_valid(new_pos):
        self.drone_pos = new_pos
    else:
        reward = -100  # No-fly zone penalty
else:
    reward = -5  # Movement failure penalty
```

Dynamic no-fly zones update every 10 steps:

```
if self.steps % 10 == 0:
    self._update_dynamic_no_fly_zones()
```

4 Deterministic vs Stochastic Environments

Key differences:

Feature	**Deterministic**	**Stochastic**
Movement	Always succeeds	80% success rate
No-fly zones	Static	Static + Dynamic
Policy requirements	Fixed optimal path	Adaptive routing
Reward variance	Consistent	High variability

5 Safety in AI

The environment implements several safety measures:

1. **Action Space Constraints**: Using gymnasium.spaces.Discrete(6) ensures only valid actions (0-5) can be selected.
2. **State Validation**: The _is_valid() method prevents collision with no-fly zones and keeps the drone within grid boundaries.
3. **Carrying Capacity**: Maximum package limit of 3 prevents overloading.
4. **Reward Structure**: Significant penalties (-100) for entering no-fly zones discourage unsafe behavior.
5. **Type Safety**: Gymnasium spaces enforce proper observation and action types, preventing type-related errors.