CUNEF UNIVERSITY

ARTIFICIAL INTELLIGENCE

PLANETARY EXPLORATION

SIMULATION

Pablo Ceballos Arenas
Nicolás Mangas Álvarez
Raúl Salas Genero

April 2025

Contents

1 Introduction and Purpose

1.1 Project Context

This project simulates autonomous planetary exploration using a fleet of rovers equipped with artificial intelligence techniques. The system models planets as toroidal surfaces where rovers must collect resources like water and minerals while avoiding obstacles and accidents. The simulation features:

- Reproducible experiments through a three-level seed system
- Systematic testing via Cartesian product of probability combinations
- Modular architecture with clearly defined components

1.2 Main Objectives

- Implement a reproducible simulation system using hierarchical seeds
- Analyze resource collection efficiency across probability combinations
- Evaluate rover survival rates under varying accident probabilities

1.3 Code Repository

The complete source code for this project is available in our GitHub repository:

`https://github.com/NicolasMangas/IA_CUNEF.git`

The repository contains:

- All Python implementation files (sim.py, planet.py, rover.py)
- Jupyter Notebooks for analysis
- Example output datasets (results.csv)
- Complete documentation

2 Class Implementation Details

2.1 Rover Class (rover.py)

The Rover class implements the autonomous agents that explore planetary surfaces, with enhanced state management and resource tracking.

Class Purpose

Models individual exploration vehicles with:

- Precise movement with toroidal wrapping
- Three-state system (EXPLORING, GATHERING, LOST)
- Detailed resource tracking including collection locations

Attributes

`id`	Unique integer identifier
`planet`	Reference to Planet instance being explored
`location`	Current position as numpy array [x,y]
`state`	Current state from RoverState enum
`load`	Dictionary tracking water/mineral counts
`findings_locations`	Records coordinates of all collected resources

Key Methods

1. move(direction: str) → bool

- **Input**: Cardinal direction ('N','S','E','W')
- **Process**:
  1. Calculates new position with toroidal wrapping
  2. Checks for obstacles in new cell
  3. Updates position if move is valid
- **Output**: True if movement succeeded

2. gather()

- **Action**:
  1. Checks current cell for resources
  2. If resource present:
     - Increments load counter
     - Records collection location
     - Empties the cell on planet

3. _update_state()

- **Automatically** sets state based on current cell:
  - Water/Mineral → GATHERING
  - Accident → LOST
  - Empty/Obstacle → EXPLORING

2.2 Planet Class (planet.py)

The Planet class creates the exploration environment with validated probability distributions and seed-controlled generation.

Class Purpose

Represents the exploration environment with:

- Configurable size and resource distributions
- Toroidal coordinate system
- Seed-controlled random generation

CellType Enum

Value	Int	Description
EMPTY	0	Navigable empty space
WATER	1	Collectable water resource
MINERAL	2	Collectable mineral deposit
OBSTACLE	3	Impassable barrier
ACCIDENT	4	Hazard that destroys rovers

Key Methods

1. **__init__()**
   - **Parameters**:
     - Dimensions (width, height)
     - Probabilities for each cell type
     - Seed for reproducible generation
   - **Validation**:
     - Ensures probability sum ≤ 1.0
     - Calculates EMPTY probability as remainder
   - **Grid Generation**:

```
self.grid = np.random.choice(
    list(CellType),
    size=(width,height),
    p=[empty_prob, water_prob, mineral_prob,
       obstacle_prob, accident_prob]
)
```

2. **get_cell(pos) → CellType**
   - Implements toroidal wrapping via modulo arithmetic
   - Returns CellType at given coordinates
3. **empty_cell(pos) → bool**
   - Converts resource cells to EMPTY after collection
   - Returns success status

2.3 Simulation Controller (sim.py)

The Simulation Controller implements the complete experimental workflow with Cartesian product of probabilities and hierarchical seeding.

Core Components

Fleet Class

- Manages groups of rovers on a planet
- Tracks simulation metrics:
  - Resources collected (water/minerals)
  - Rovers lost to accidents
- Implements rover behavior as SimPy process

sim() Function

- Main simulation driver
- Implements experimental design:
  1. Generates probability combinations
  2. Selects configurations via global seed
  3. Runs multiple explorations per planet
  4. Aggregates results to DataFrame

Key Processes

1. Probability Combination Generation
We use the Cartesian product to systematically explore all possible combinations of our probability parameters. This mathematical operation creates a comprehensive test matrix that ensures we evaluate every possible scenario:

```
combinations = list(itertools.product(
    water_probs,
    mineral_probs,
    accident_probs
))
```

For our default configuration with 3 water probabilities, 3 mineral probabilities, and 3 accident probabilities, this generates $3 \times 3 \times 3 = 27$ unique planet configurations.

2. Seed Management
Our three-level seed system ensures complete reproducibility while maintaining experimental integrity:

- **Global seed**: Controls combination selection
- **Planet seeds**: Ensure consistent grid generation (seed = planet number)
- **Exploration seeds**: Unique per simulation run ($seed = planet\_num100 + exploration\_num$)

3. Data Collection

- Tracks all key metrics per exploration
- Uses pandas DataFrame for structured storage
- Outputs to CSV:

```
df.to_csv("results.csv", index=False)
```

3 Data Collection and Analysis

3.1 CSV Output Structure

The generated results.csv contains comprehensive simulation data:

Column	Type	Description
Planet	int	Planet identifier (1-n)
Exploration	int	Run number per planet
Water_prob	float	Water probability used
Mineral_prob	float	Mineral probability used
Accident_prob	float	Accident probability used
Water_gathered	int	Total water collected
Mineral_gathered	int	Total minerals collected
Total_gathered	int	Sum of all resources
Lost_rovers	int	Rovers destroyed by accidents

3.2 Jupyter Notebook Analysis

The analysis notebook performs the following results study:

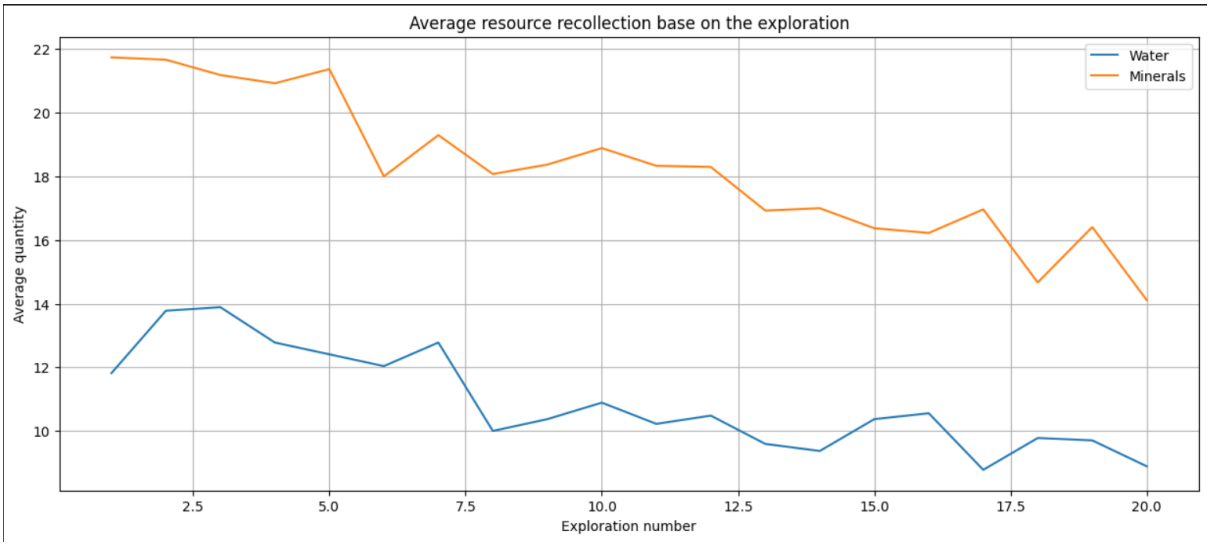

Figure 1: Average resource recollection based on the exploration

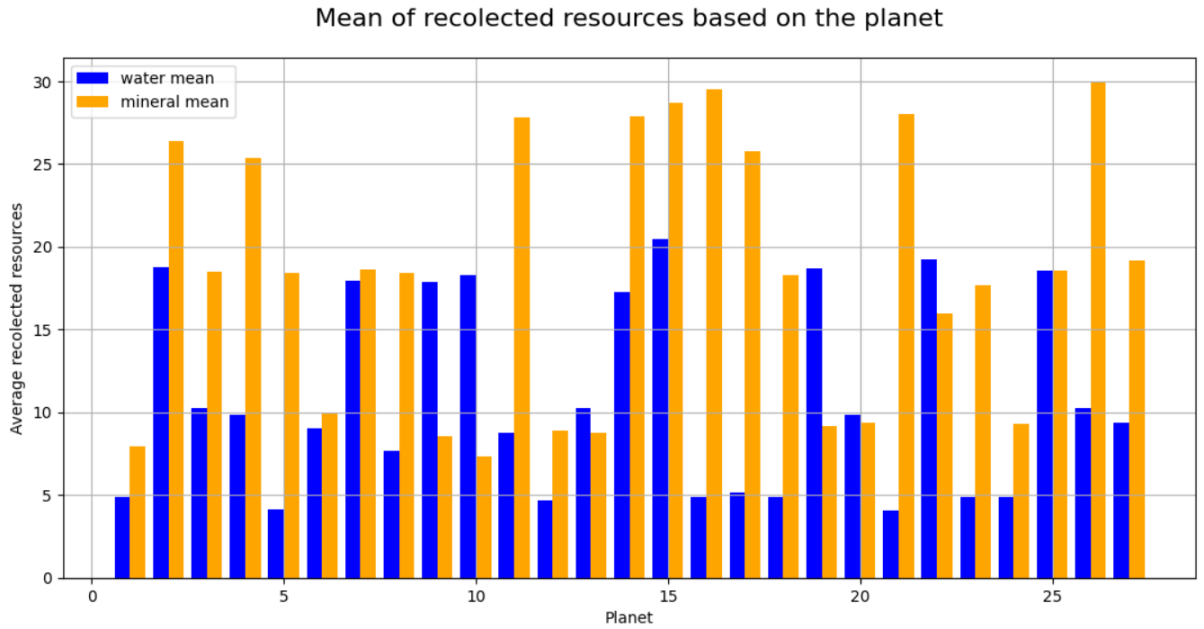

Figure 2: Mean of recolected resources based on the planet

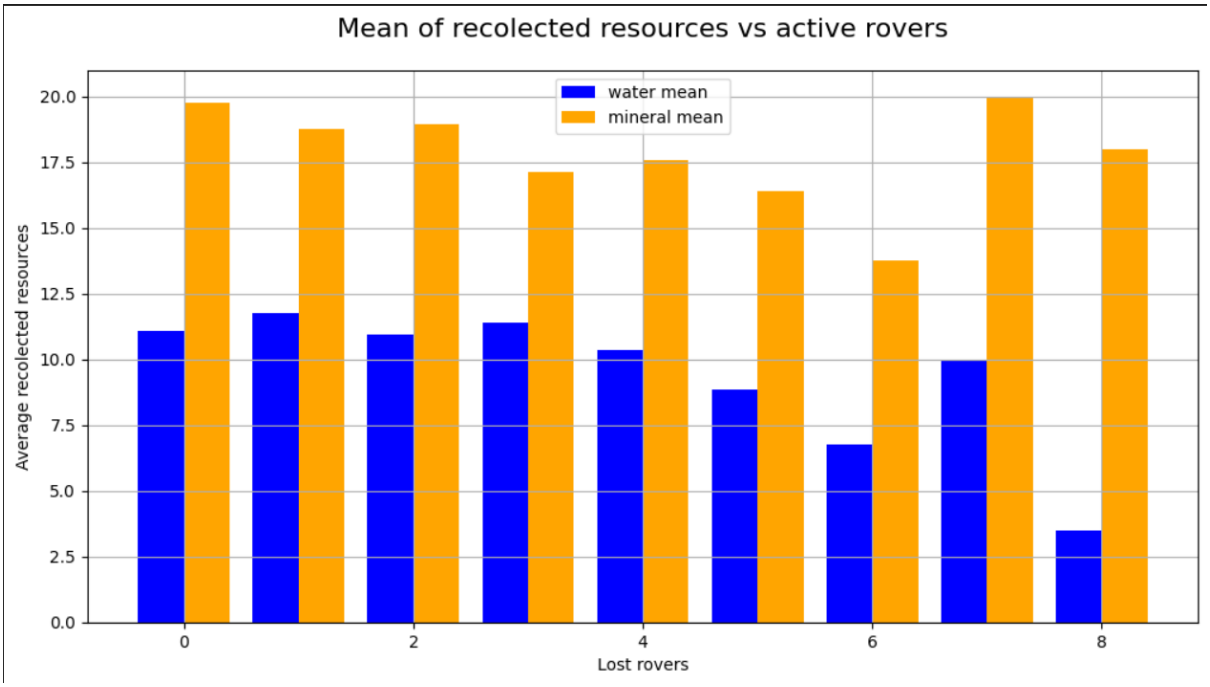

Figure 3: Average resources gathered by planet type

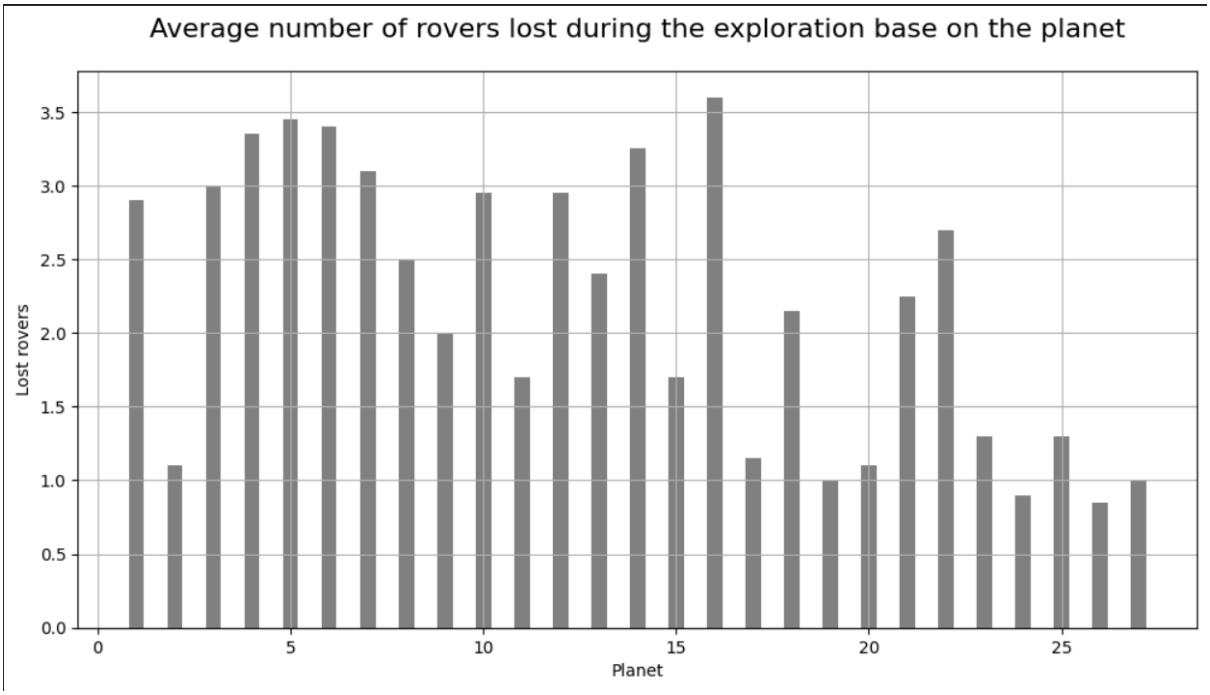

Figure 4: Rover loss average analysis

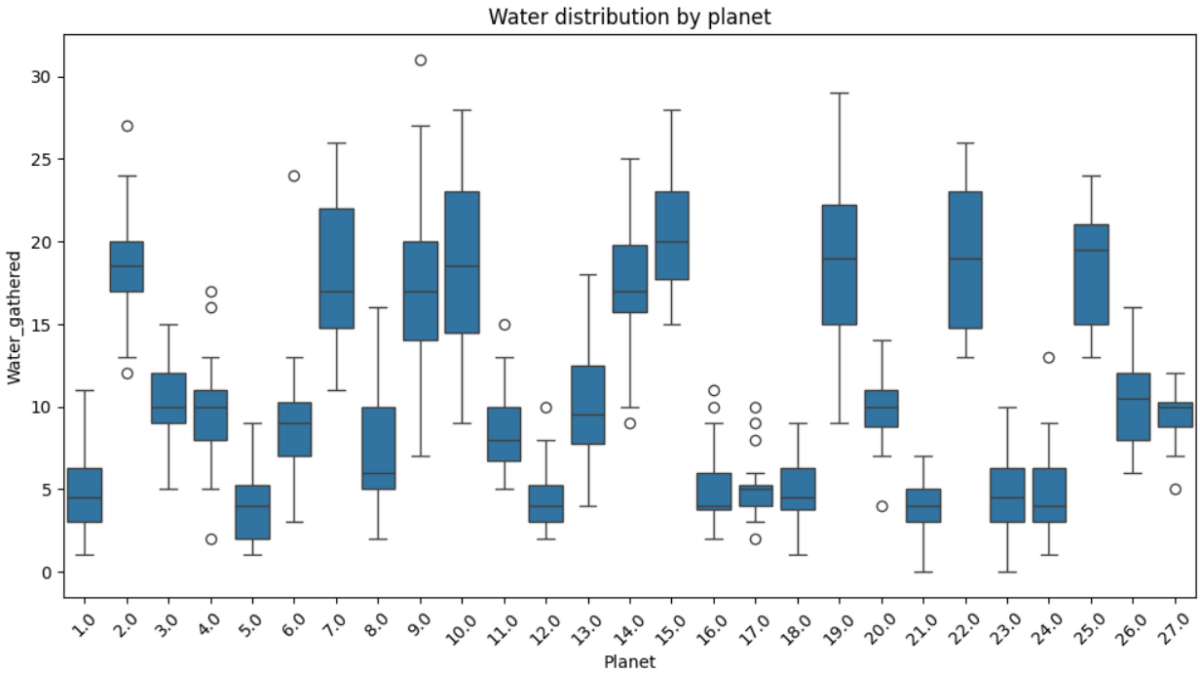

Figure 5: Water distribution by planet

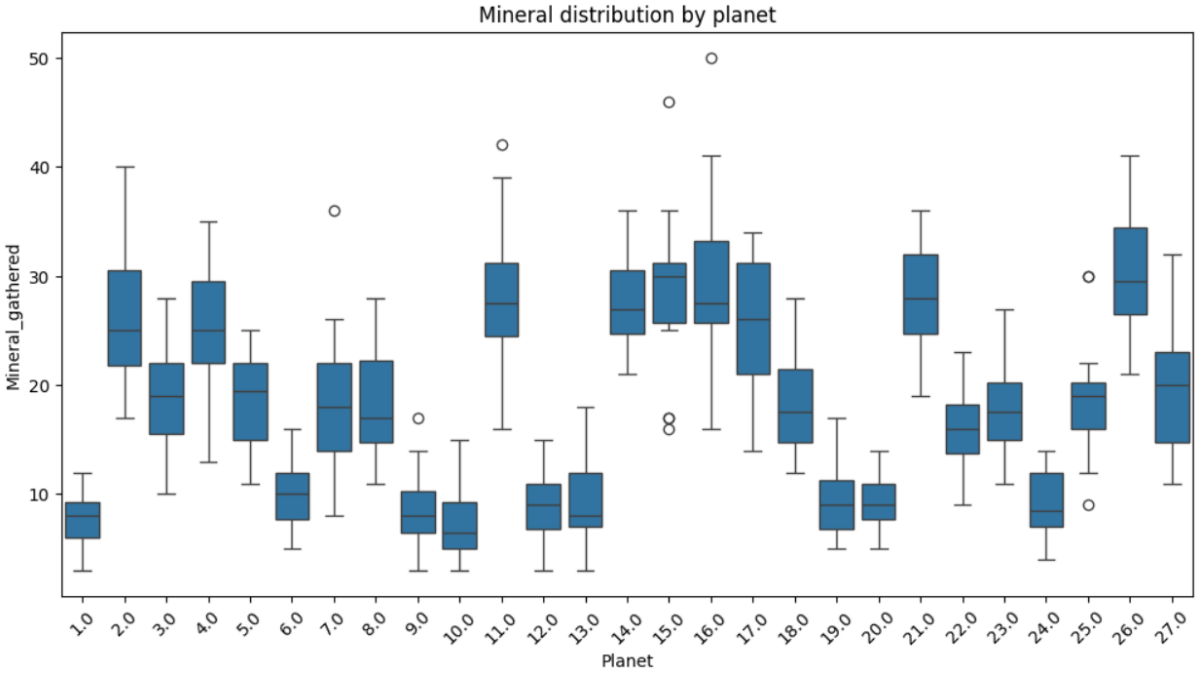

Figure 6: Mineral distribution by plane

4 Conclusions

4.1 Technical Achievements

- Implemented reproducible experimental design
- Systematic exploration of parameter space
- Implemented probabilities through a cartesian product

4.2 Future Extensions

- Advanced visualization of rover paths
- New movements for each rover, learning from one fleet to another