WORKSHOP #2 – DYNAMICAL SYSTEMS ANALYSIS AND DESIGN

Johan Sebastián Gutiérrez Pérez - 20211020098

Francisco Jose de Caldas District University

System Sciences Foundations

Eng. Carlos Andrés Sierra

May 9th, 2025

Abstract

This project presents the second stage of the process of building and design an autonomous adaptative agent, focusing on how the dynamic systems theory is integrated in the architecture of a package delivery drone. Using, as well, the design developed in the Workshop #1, this stage introduces the analysis of the dynamics of the system to improve the understanding of the agent and with this, improving the behavior of the agent over the time, specially in uncertain situations. The proposed system considers the presence or absence of urban obstacles, as buildings, which are randomly configured in each iteration to simulate a changing environment. The adaptative capabilities of the agent are analyzed through the improving of the feedback loops and stability assessments, setting the stage for the reinforcement learning implementation. This document outlines the theorical framework, expected system behavior and the design considerations to reach a good performance.

Introduction

In the field of a cybernetic and AI system, the ability of an agent to adapt to changing environment is crucial for ensuring an optimal behavior. As systems become more complex and operate in increasingly dynamic contexts, understanding the internal mechanisms through the system sciences becomes essential. This stage of the project explores the principles of dynamical systems and their role in the behavior of the package delivery drone. The agent operates within a simulated urban environment that has some variations. These conditions demand a design capable of self-regulation, stability and continuous adaptation. It's important to analyze the system dynamics, including sensitivity to initial conditions and the feedback loops.

This workshop aims to stablish a stronger design for the agent´s efficiency. By connecting these concepts with the previous system design, we prepare the system for a more refined performance.

1. **System Dynamics Analysis**

1.1. Mathematical Model

To describe the behavior of the adaptative delivery drone over the time, it's interesting to consider a directed graph that is linked with the system dynamic. This graph represents the urban area where the drone navigates. The main components of the model are:

1.1.1 Environment Representation

The environment is modeled as a directed graph:

$$G = (N, E)$$

Where:

- **N** is the set of nodes representing navigable positions, in this context, we can understand these nodes as the intersections of the streets.
- **E** is the set of directed edges representing the paths between nodes.

1.1.2 System State

The state of the system over the time t, denoted S_t is represented as:

$$S_t = (P_t, O)$$

Where:

- $P_t \in N$ is the current position of the drone.

- $O \subseteq \mathrm{E}$ is the set of edges that are blocked by urban buildings. These obstacles are generated randomly at the beginning of each simulation and remain static during its execution.

1.1.3 **Next Movement of The Drone**

The evolution of the drone's position at each time step is determined by:

$$P_{t+1} = f(P_t, A_t O)$$

Where:

- A_t is the action taken by the agent, such as moving to de left, right, up or down.

- The function f uses the current position of the drone, the action and check which of the nodes are blocked (if they are or not in the set O) to check if the path to the selected node is passable before updating the position.

1.2. **Simulation Model**

The simulation model uses the mathematical formulation of the drone's navigation through a dynamic environment. This provides a virtual space where the agent can interact with its surroundings across discrete time steps. The principal aspects of the simulation model are:

1.2.1. **Environment Initialization**

At the start of each simulation, the environment is constructed as a grid-based directed graph $G = (N, E)$, with the addition of randomly placed obstacles (buildings) that determine the subset $O \subseteq \mathrm{E}$. These obstacles remain the same for the duration of the simulation but are regenerated in each iteration to simulate a non-static environment.

1.2.2. **Time Progression**

Time progression in discrete steps $t = 0, 1, 2, \ldots$ At each time step:

- The agent observes its current state $S_t = (P_t, O)$.
- Based on the decision that is constantly refined with the reinforcement learning module, the agent selects an action A_t.
- The system checks whether the action leads to a valid node.
- If valid, the drone transitions to a new state given by S_{t+1}. Otherwise, it remains in the same position and a penalty is applied

1.2.3. **Goal and Termination Conditions**

Each simulation has a predefined goal state, this goal is that de package arrives its destination node. The simulation ends when the drone reaches this goal or when maximum number of steps is exceeded. Performance metrics such as total steps, optimal path and success rate are tracked for evaluation.

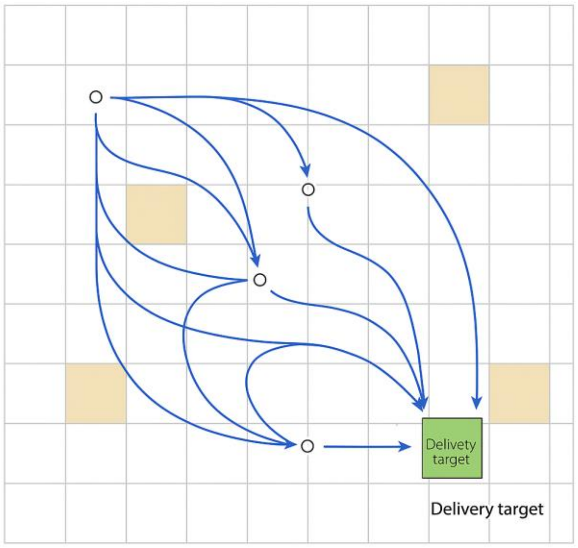
1.3. **Phase Diagrams**

To understand how the agent's behavior evolves under varying conditions, we analyze its state space using a 2D grid representation of the environment. Each node in the grid represents a possible position of the drone, while the blocked paths simulate the presence of dynamic urban obstacles.

As the agent interacts with the environment, its trajectory across the grid reveals patterns in its decision-making process. Initially, the agent may exhibit chaotic movements—exploring various directions and sometimes choosing suboptimal or penalized paths. However, over time, as it receives feedback from its environment and updates its policy, the system begins to converge toward attractors, i.e., efficient paths that consistently lead to successful deliveries.

These attractors represent stable behavioral patterns, where the agent learns to avoid obstacles and minimize the number of steps taken. In contrast, during the early stages of learning or in highly obstructed configurations, the system may fall into chaotic regimes, where the drone repeatedly fails or explores inefficient paths due to lack of information or incorrect strategy.

The figure below illustrates a sample 2D environment used to visualize the agent's evolving state trajectories:



This diagram highlights how the agent's movement evolves across different runs, depending on obstacle placement and the refinement of its internal policy. Over multiple episodes, a trend toward convergence and improved decision-making becomes apparent.

2. **Feedback loop Refinement**

2.1. **Enhanced control mechanisms**

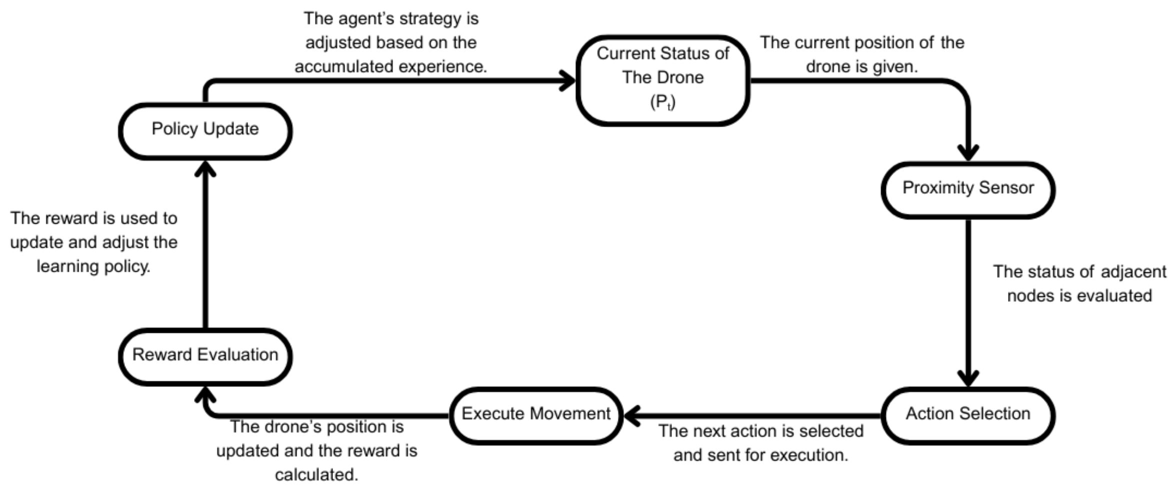
The adaptive behavior of the drone is driven by a continuous feedback loop that refines its decision-making strategy over time. This loop is essential for learning from interaction with the environment and is structured around a reinforcement learning process.

Each iteration of the loop begins with the current state of the agent, defined by its position in the environment and the obstacle configuration. A proximity sensor evaluates the status of adjacent nodes whether they are blocked or free. Based on this perception, the action selection module chooses a movement direction (up, down, left, right).

The execution of the movement leads to a new position and triggers the reward evaluation step. If the drone moves closer to the delivery point, it receives a positive reward. If it hits an obstacle or moves inefficiently, the reward is reduced. Finally, the policy update module adjusts the learning strategy based on the received feedback, closing the feedback loop.

This process allows the drone to improve within a single episode as well as across multiple runs. Over time, the agent refines its behavior, reinforcing successful paths and discouraging unproductive ones.

The diagram below illustrates the feedback loop that rules the adaptive learning cycle of the drone:


The agent's strategy is adjusted based on the accumulated experience.
Current Status of The Drone (P_t)
The current position of the drone is given.
Policy Update
Proximity Sensor
The reward is used to update and adjust the learning policy.
The status of adjacent nodes is evaluated
Reward Evaluation
Execute Movement
Action Selection
The drone's position is updated and the reward is calculated.
The next action is selected and sent for execution.

Through this iterative cycle, the system transitions from exploratory behavior to a more stable and efficient strategy, shaping an attractor around optimal delivery patterns and avoiding chaotic movement regimes.

2.2. **Stability and Convergence**

To ensure that the drone does not behave erratically or chaotically in a dynamic environment, its learning strategy must converge toward stable policies. This is achieved by:

- **Reward shaping**, which progressively reinforces optimal behaviors by providing rewards proportional to the reduction in distance to the delivery point.
- **Penalization of undesirable actions**, such as moving toward obstacles or making inefficient detours.
- **Cumulative learning**, where the agent uses prior experience to avoid repeating past mistakes.

The result is the emergence of behavioral attractors preferred paths and strategies that the drone repeatedly follows under similar conditions. This convergence reduces the occurrence of chaotic patterns, improving performance over time and leading to stable navigation behavior.

3. **Iterative Design Outline**

3.1. Project Plan Update

To support the advanced dynamic behavior of the adaptive delivery drone, several additions and refinements have been incorporated into the project plan:

- Data Structures:
  - A directed graph ($G = (N, E)$) is used to represent the urban grid with N as nodes and E as directed edges.
  - A state representation ($S_t = P_t, O$) where P_t is de drone's position and O is the set of blocked edges (urban obstacles).

- Algorithms
  - **Reinforcement Learning (RL)** will be used to allow the agent to improve its decision-making based on rewards.
  - A **distance-based reward function**, which increases as the drone approaches the package location and decreases when it hits obstacles or moves inefficiently.
  - Optional integration of **Dijkstra's algorithm** to calculate shortest paths and assist in reward shaping.
- **Frameworks & Libraries**:
  - **Gymnasium**: For creating and managing the simulation environment.
  - **NumPy**: For numerical operations and handling matrices/vectors.
  - **Matplotlib**: For visualizing trajectories, performance trends, and state evolution.
  - **Stable-Baselines3**: For a higher-level implementation of RL algorithm.

3.2. Testing Strategy

To evaluate the dynamic behavior and learning progress of the agent, the following testing strategies will be employed:

Simulation Parameters:

- Grid size (e.g., 10×10, 20×20).
- Number of buildings per simulation.
- Maximum steps per episode.

- Learning rate, exploration rate (epsilon), and discount factor for RL.

Random Seeds:

- Different random seeds will be used to generate varying initial conditions (drone position, target position, obstacle distribution).
- Seeds will ensure reproducibility of each experiment and facilitate controlled comparisons.

Scenario Variations:

- Different positions for the delivery target.
- Varying obstacles in the map.

By systematically varying these factors, we can observe how well the drone adapts to new challenges, how quickly it converges to stable behavior, and whether its learned policies remain effective across scenarios.

References

OpenAI. (2024). *Gymnasium documentation*. https://www.farama.org/projects/gymnasium/

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., & Ernestus, M. (2021). *Stable-Baselines3: Reliable reinforcement learning implementations*. https://github.com/DLR-RM/stable-baselines3

NumPy Developers. (2024). *NumPy*. https://numpy.org/

Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. *Computing in Science & Engineering, 9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Waskom, M. L. (2021). *Seaborn: Statistical data visualization*. *Journal of Open Source Software, 6*(60), 3021. https://doi.org/10.21105/joss.03021

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.