Naive Multi-Agent Simulation: Mathematical Explanations and Algorithms

Benjamin Ocampo

August 15, 2024

1 Introduction

This document describes a naive multi-agent simulation model that generates and visualizes the behavior of a hierarchical tree structure. In this model, each agent is represented by a node within the tree, with parent nodes acting as higher-level agents that influence the movement and decision-making of lower-level agents, or child nodes. The system is designed to explore the interactions and pathfinding behavior of these agents within the tree, without the use of external forces such as reinforcement learning.

1.1 Overview of the Model

The core of this model is a tree generation algorithm that recursively constructs a tree of nodes, each with a defined position, color, and accuracy value. The accuracy value associated with each parent node dictates the likelihood that an agent, represented by a pulse, will follow the correct path towards a specified target within the tree. This probability-based decision-making process allows the simulation to model complex behaviors such as exploration, exploitation, and backtracking, which are common in multi-agent systems.

1.2 Multi-Agent System without Reinforcement Learning

Unlike many advanced AI systems, which employ reinforcement learning or other forms of external feedback to optimize agent behavior, this model operates without such influences. Instead, the agents' behavior is determined solely by the structural properties of the tree (such as node position and accuracy) and the probabilistic choices made at each decision point. This approach provides a baseline understanding of how agents might interact in a hierarchical structure when guided by simple, internal rules rather than by learned policies or external rewards.

1.3 Purpose of the Model

The primary goal of this simulation is to study the emergent behaviors that arise from the interaction of multiple agents within a constrained environment, specifically within a tree structure. By focusing on a simple, rule-based system, this model sheds light on the foundational dynamics that underpin more complex multi-agent systems, offering insights into how agents might navigate and interact in environments where their decisions are guided by inherent probabilities rather than by adaptive learning mechanisms.

2 Tree Generation Algorithm

2.1 Mathematical Description of Tree Generation

Let's define the process of generating a tree as follows:

- **Initial Setup:**
  - Start with a root node N_0 located at a position $\mathbf{p}_0$ in a 2D space.
  - Define the maximum depth of the tree as $d_{\max}$.
- **Recursive Tree Generation:**
  - For each node N_i at level l (starting with $l = 0$ for the root):
    - Assign a position $\mathbf{p}_i$ in the 2D space.
    - Assign a parent node P_i for all nodes N_i at levels $l > 0$, where P_i is the parent node of N_i.
    - Assign a color $\mathbf{c}_i$ to the node based on its position and the color of its parent P_i, ensuring that the color is distinct from the parent's color and any sibling nodes' colors.
    - If the node N_i has children, assign an accuracy value $\alpha_i \in [0.3, 0.6]$ that dictates the probability of a pulse choosing the correct path towards the target node.
- **Child Node Generation:**
  - Determine the number of children k_i for node N_i at level l:

$$k_i = \begin{cases} 1 + \text{rand}(1, 2) & \text{if } l = 0 \\ \text{rand}(0, 3) & \text{if } l > 0 \end{cases}$$

  - For each child node $N_{i,j}$ of N_i:
    - Assign a position $\mathbf{p}_{i,j}$ such that:

$$\mathbf{p}_{i,j} = \mathbf{p}_i + \Delta\mathbf{p}_j$$

      where $\Delta\mathbf{p}_j$ is a vector that spaces the child nodes horizontally relative to their parent.

    - Recursively apply the tree generation process to each child node until the maximum depth $d_{\max}$ is reached.

- **Color Assignment:**
  - For nodes without children (leaf nodes), assign a distinct color $\mathbf{c}_i$ by generating a random color vector $\mathbf{c}_i$ and ensuring it has a minimum color distance $d_{\min}$ from the parent node's color and sibling nodes' colors:
$$\|\mathbf{c}_i - \mathbf{c}_j\| \geq d_{\min} \quad \text{for all siblings } N_j \text{ of } N_i$$

- **Termination Condition:**
  - The recursion terminates when the tree reaches the maximum depth $d_{\max}$ or when a node N_i has no children, becoming a leaf node.

2.2 Step-by-Step Algorithmic Description

1. **Initialize the root node** at position $\mathbf{p}_0$.
2. **Set the maximum depth** $d_{\max}$ for the tree.
3. **For each node** N_i **at level** l**:**
   (a) Assign a position $\mathbf{p}_i$ in 2D space.
   (b) Assign a parent node P_i if $l > 0$.
   (c) Determine the number of children k_i.
   (d) If $k_i > 0$:
   - Assign an accuracy value α_i.
   - Generate and position the child nodes $N_{i,j}$.
   - Recursively apply this process to the child nodes.
   (e) If $k_i = 0$ (leaf node):
   - Assign a distinct color $\mathbf{c}_i$ that is different from its parent and siblings.
4. **Repeat** the process until the maximum depth $d_{\max}$ is reached.
5. **Terminate** when all nodes are generated up to depth $d_{\max}$.

3 Pulse Movement Algorithm

3.1 Mathematical Description of Pulse Movement

The pulse movement algorithm can be described as follows:

- **Initial Setup:**

- Define a set of pulses $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$, where each pulse p_k starts at the root node N_0 and has a target node N_t.
- The position of pulse p_k at time t is denoted by $\mathbf{q}_k(t)$.
- Each pulse p_k is associated with a current node N_c, a next node N_n, and a previous node N_p.

- **Pulse Movement:**
  - The pulse moves towards its next node N_n with a constant speed v.
  - The direction vector $\mathbf{d}_k$ for pulse p_k at time t is given by:

$$\mathbf{d}_k = \frac{\mathbf{p}_n - \mathbf{q}_k(t)}{\|\mathbf{p}_n - \mathbf{q}_k(t)\|}$$

    where $\mathbf{p}_n$ is the position of node N_n.
  - The position of pulse p_k is updated as:

$$\mathbf{q}_k(t + \Delta t) = \mathbf{q}_k(t) + v \cdot \mathbf{d}_k \cdot \Delta t$$

- **Path Selection:**
  - At each node N_c, the pulse must choose the next node N_n to move towards.
  - Each parent node N_c has an accuracy value $\alpha_c \in [0.3, 0.6]$, which determines the probability of the pulse choosing the correct path towards its target node N_t.
  - The pulse selects the next node N_n based on the following criteria:
    - With probability α_c, the pulse chooses the correct path by finding the node N_n that leads towards N_t.
    - With probability $1 - \alpha_c$, the pulse randomly selects a node that is not on the correct path.
  - If no valid next node is found, the pulse backtracks to its previous node N_p.

- **Termination Condition:**
  - The pulse terminates when it reaches its target node N_t.
  - The pulse is removed from the simulation upon termination.

3.2 Step-by-Step Algorithmic Description

1. **Initialize each pulse** p_k at the root node N_0 with a target node N_t.
2. **For each pulse** p_k:
   (a) Set the current node N_c to the starting node.

(b) Set the previous node N_p to null.

(c) Determine the next node N_n based on the accuracy value α_c of the current node:

- If the pulse should follow the correct path, select N_n as the child node leading towards the target node N_t.
- If the pulse should deviate, randomly select a child node that does not lead towards the target node.
- If no valid child node is available, backtrack to the previous node N_p.

(d) Move the pulse p_k towards N_n at a constant speed v:

$$\mathbf{q}_k(t + \Delta t) = \mathbf{q}_k(t) + v \cdot \mathbf{d}_k \cdot \Delta t$$

(e) If the pulse reaches N_n:

- Update N_p to N_c, and set N_c to N_n.
- Re-evaluate the next node N_n for the pulse.

(f) If the pulse reaches its target node N_t, terminate the pulse and remove it from the simulation.

3. **Repeat** the process for all pulses until all pulses have either terminated or reached their target nodes.