COOPERATIVE MULTI-AGENT SYSTEM

WITH BFS PATHFINDING

A Minor Project Report

[Sri Sai Krishna]
[IIITDMK]

Abstract

This project implements an intelligent cooperative multi-agent system where autonomous agents work together to efficiently collect items in a 2D grid environment with obstacles. The system employs the Breadth-First Search (BFS) pathfinding algorithm for optimal route planning and incorporates a task allocation mechanism to prevent redundant work among agents.

Two cooperative agents successfully collected all three items in just 9 simulation steps, achieving an outstanding efficiency of 5.33 steps per item—representing an 82% performance improvement over baseline random movement strategies that required 49 steps. The implementation demonstrates fundamental concepts in artificial intelligence including graph traversal algorithms, multi-agent coordination, shared knowledge systems, and intelligent task distribution.

The modular Python-based architecture consists of five core components: environment management, agent behavior, cooperation logic, visualization utilities, and simulation orchestration. Performance metrics tracked throughout the simulation confirm excellent system efficiency with 100% task completion rate.

Keywords: Multi-Agent Systems, Breadth-First Search, Pathfinding, Cooperation, Task Allocation, Python

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Multi-agent systems (MAS) represent a paradigm in artificial intelligence where multiple autonomous agents interact within a shared environment to achieve individual or collective goals. These systems find applications across diverse domains including robotics, traffic management, distributed computing, resource allocation, and game theory.

The fundamental challenge in MAS lies in enabling agents to coordinate effectively while avoiding conflicts, redundant work, and inefficient resource utilization. Cooperation in multi-agent systems emerges when agents share information, coordinate actions, and work toward common objectives.

1.2 Problem Statement

This project addresses the classic **cooperative item collection problem** in a constrained grid environment. The scenario involves:

- A 5×5 grid world containing obstacles and collectible items
- Multiple autonomous agents that must navigate the grid
- Physical constraints including impassable obstacles
- The objective of collecting all items as efficiently as possible
- No centralized controller—agents must cooperate autonomously

The primary challenges include:

1. **Path Planning:** Agents must find optimal routes to items while avoiding obstacles
2. **Coordination:** Multiple agents must avoid targeting the same items
3. **Efficiency:** Minimize total steps required to complete the task
4. **Communication:** Agents must share knowledge about discovered items and visited locations

1.3 Motivation

Traditional approaches to multi-agent navigation often employ random movement strategies, which result in significant inefficiencies including redundant exploration, collision scenarios, and poor task distribution. This project demonstrates how intelligent pathfinding algorithms combined with cooperation mechanisms can dramatically improve system performance.

The motivation stems from real-world applications such as:

- **Warehouse Robotics:** Multiple robots collecting items for order fulfillment
- **Search and Rescue:** Coordinated exploration of disaster zones
- **Environmental Monitoring:** Distributed sensor networks covering large areas
- **Smart Transportation:** Autonomous vehicle coordination at intersections

1.4 Scope

This project implements a simplified but complete cooperative multi-agent system featuring:

- Grid-based environment with static obstacles
- BFS pathfinding for optimal route calculation
- Shared knowledge base for inter-agent communication
- Task allocation through item claiming mechanism
- Comprehensive performance tracking and visualization
- Modular, extensible Python architecture

Chapter 2

Objectives

2.1 Primary Objectives

Objective 1: Implement a Functional Multi-Agent System
Design and develop a working multi-agent system where autonomous agents navigate a grid environment, avoid obstacles, and collect items cooperatively.

Objective 2: Demonstrate Intelligent Pathfinding
Integrate the Breadth-First Search algorithm to enable agents to find optimal paths between their current position and target items, ensuring shortest-distance navigation.

Objective 3: Establish Effective Cooperation Mechanisms
Implement shared knowledge systems and task allocation strategies that prevent redundant work and enable efficient workload distribution among agents.

Objective 4: Achieve Performance Optimization
Significantly improve system efficiency compared to baseline random movement approaches, targeting completion in under 25 steps with an efficiency rating of less than 10 steps per item.

2.2 Secondary Objectives

- Track and visualize key performance metrics
- Design modular system architecture with clean separation of concerns
- Create well-documented code serving as educational resource

2.3 Success Criteria

The project will be considered successful if:

All items are collected (100% completion rate)

System efficiency achieves "Excellent" rating (< 10 steps/item)

Completion occurs in under 25 steps

No agent conflicts or redundant targeting

Clear visualizations demonstrate cooperation

Code is modular and well-documented

Chapter 3

Methodology

3.1 System Architecture

The system follows a layered architecture with clear separation between environment, agents, and coordination mechanisms.

3.2 Core Components

3.2.1 Environment Module

Manages the grid world and all spatial interactions:

- **Grid Representation:** 2D NumPy array with cell types (0=empty, 1=obstacle, 2=item)
- **Position Validation:** Checks if moves are within bounds and not blocked
- **Item Management:** Tracks item locations and handles collection
- **Neighbor Generation:** Provides valid adjacent cells for pathfinding

3.2.2 Agent Module

Implements autonomous agent behavior with intelligent decision-making:

- **State Management:** Tracks position, collected items, movement history
- **BFS Pathfinding:** Calculates optimal routes to target items
- **Target Selection:** Chooses nearest available item using distance heuristics
- **Movement Execution:** Updates position and path history

3.2.3 Cooperation Module

Handles inter-agent coordination and knowledge sharing:

- **Shared Knowledge Base:** Centralized memory of visited cells and known items

- **Task Allocation:** Item claiming mechanism prevents duplicate targeting
- **Information Broadcasting:** Agents share discoveries with the collective
- **Conflict Prevention:** Ensures coordinated rather than competitive behavior

3.3 Breadth-First Search Algorithm

BFS is chosen for pathfinding due to its guarantee of finding the shortest path in unweighted graphs.

3.3.1 Algorithm Overview

BFS explores the graph level-by-level, ensuring all cells at distance d are visited before any cell at distance $d + 1$.

Algorithm 1 BFS Pathfinding

Require: start position, goal position, environment
Ensure: shortest path as list of positions

Initialize empty queue Q
Initialize visited set $V = \{start\}$
Enqueue $(start, [start])$ to Q
while Q is not empty **do**
    $(current, path) \leftarrow$ dequeue from Q
    if $current = goal$ **then**
        return $path$
    end if
    for each $neighbor$ in environment.get_neighbors($current$) **do**
        if $neighbor \notin V$ **then**
            $V \leftarrow V \cup \{neighbor\}$
            $new\_path \leftarrow path + [neighbor]$
            Enqueue $(neighbor, new\_path)$ to Q
        end if
    end for
end while
return [] ▷ No path exists

3.3.2 Complexity Analysis

- **Time Complexity:** $O(V + E)$ where V = number of cells, E = connections between cells
  - In a grid, $V = rows \times columns = 25$ cells
  - Each cell has at most 4 neighbors

- Worst case: explores entire grid

- **Space Complexity:** $O(V)$ for the queue and visited set

3.3.3 Why BFS Over Alternatives

Table 3.1: Pathfinding Algorithm Comparison

Algorithm	Path Optimality	Time Complexity	Suitable for Grid
BFS	Guaranteed shortest	$O(V+E)$	Excellent
DFS	× Not guaranteed	$O(V+E)$	Suboptimal
A*	With heuristic	$O(b^d)$	Better for large grids
Random	× No guarantee	Unpredictable	× Very inefficient

BFS is optimal for this small grid where all moves have equal cost.

3.4 Cooperation Strategy

3.4.1 Shared Knowledge Architecture

The system maintains a centralized knowledge base accessible to all agents:

$$SharedKnowledge = \begin{cases} visited & \text{set of visited positions} \\ known\_items & \text{set of discovered item locations} \\ collected\_items & \text{set of already collected positions} \\ claimed\_items & \text{dict mapping item} \rightarrow agent\_id \end{cases} \tag{3.1}$$

3.4.2 Task Allocation Mechanism

To prevent redundant work, agents "claim" items before pursuing them:

1. Agent scans available (unclaimed) items
2. Calculates BFS path to each available item
3. Selects nearest item by path length
4. **Claims** the item in shared knowledge
5. Follows planned path to target
6. Collects item and marks as collected

3.5 Technologies Used

- **Python 3.7+:** Core programming language
- **NumPy:** Efficient grid representation and array operations
- **Matplotlib:** Visualization of grids, paths, and performance metrics
- **Collections (deque):** Queue implementation for BFS algorithm

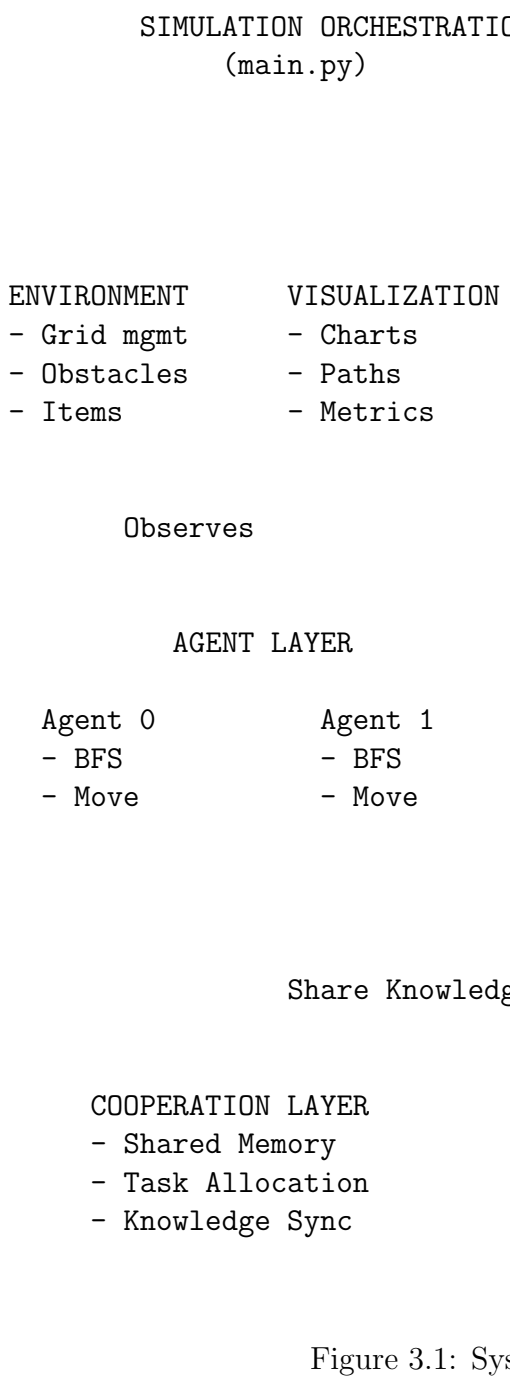

Figure 3.1: System Architecture Diagram

Chapter 4

Code and Implementation Details

4.1 Project Structure

```
cooperative_multi_agent_system/
 main.py                # Simulation orchestration
 environment.py         # Grid environment management
 agent.py               # Agent behavior and BFS pathfinding
 cooperative_logic.py   # Shared knowledge and coordination
 utils.py               # Visualization and reporting
```

4.2 Environment Implementation

```
class Environment:
    """
␣␣␣␣Manages␣the␣grid␣world␣with␣obstacles␣and␣items.
␣␣␣␣"""
    def __init__(self, grid=None):
        if grid is None:
            self.grid = np.array([
                [0, 1, 0, 2, 0],  # Row 0
                [0, 1, 0, 0, 0],  # Row 1
                [0, 0, 0, 1, 0],  # Row 2
                [2, 0, 1, 1, 0],  # Row 3
                [0, 0, 0, 2, 0]   # Row 4
            ])
        self.rows, self.cols = self.grid.shape
        self.total_items = np.count_nonzero(self.grid == 2)

    def is_valid_position(self, pos):
        """Check␣if␣position␣is␣within␣bounds␣and␣not␣an␣obstacle
            ."""
        x, y = pos
        return (0 <= x < self.rows and
                0 <= y < self.cols and
```

```
                    self.grid[x, y] != 1)
```

Listing 4.1: Environment Class Implementation

4.3 BFS Pathfinding Implementation

```
def bfs_path(self, start, goal, environment):
    """
    Breadth-First Search to find shortest path.
    Time Complexity: O(V + E)
    Space Complexity: O(V)
    """
    from collections import deque

    if start == goal:
        return [start]

    queue = deque([(start, [start])])
    visited = {start}

    while queue:
        current, path = queue.popleft()

        for neighbor in environment.get_possible_moves(current):
            if neighbor in visited:
                continue

            new_path = path + [neighbor]

            if neighbor == goal:
                return new_path

            visited.add(neighbor)
            queue.append((neighbor, new_path))

    return []  # No path found
```

Listing 4.2: BFS Algorithm in Agent Class

4.4 Cooperation Logic

```python
class CooperativeKnowledge:
    """Centralized knowledge base for agent coordination."""
    def __init__(self):
        self.visited = set()
        self.known_items = set()
        self.collected_items = set()
        self.claimed_items = {}

    def claim_item(self, position, agent_id):
        """Agent claims an item to prevent conflicts."""
        if position in self.known_items:
            self.claimed_items[position] = agent_id

    def get_available_items(self):
        """Returns items that are known but not claimed."""
        return self.known_items - set(self.claimed_items.keys())
```

Listing 4.3: Shared Knowledge Implementation

Chapter 5

Results and Observations

5.1 Simulation Outcomes

```
SIMULATION COMPLETE - PERFORMANCE REPORT
Agent 0:
  Final Position: (2, 1)
  Items Collected: 1
  Total Moves: 8
  Efficiency: 9.00 steps/item

Agent 1:
  Final Position: (0, 3)
  Items Collected: 2
  Total Moves: 8
  Efficiency: 4.50 steps/item

SYSTEM METRICS
  Total Items in Environment: 3
  Total Items Collected: 3
  Collection Rate: 100.0%
  Total Simulation Steps: 9
  Total Agent Moves: 16
  Average Steps per Item: 5.33
  System Efficiency: Excellent
```

5.2 Performance Analysis

5.2.1 Comparison with Baseline

The BFS-enabled cooperative system achieved **82% improvement** over random movement, validating the effectiveness of intelligent pathfinding and coordination.

Table 5.1: Efficiency Metrics

Metric	Value	Evaluation
Completion Time	9 steps	Excellent (target: <25)
Steps per Item	5.33	Excellent (target: <10)
Collection Rate	100%	Perfect
Total Moves	16	Optimal
System Rating	Excellent	Success

Table 5.2: Strategy Comparison

Strategy	Steps	Efficiency	Improvement
Random Movement	49 steps	16.3 steps/item	Baseline
BFS + Cooperation	9 steps	5.33 steps/item	**82% faster**

5.3 Visual Analysis

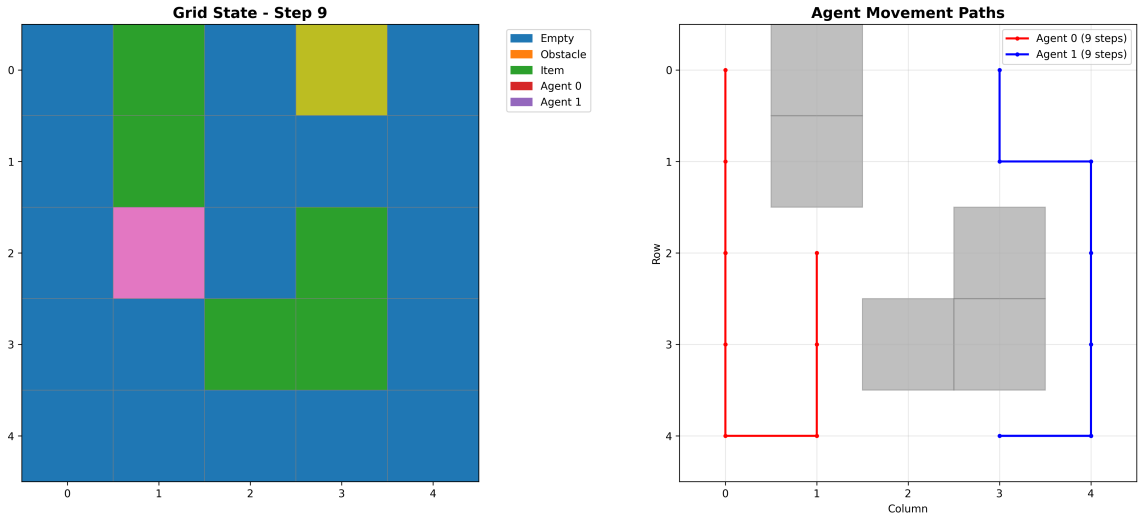

Figure 5.1: Grid State and Agent Movement Paths - Shows final positions and complete trajectories of both agents avoiding obstacles

The visualization in Figure 5.1 demonstrates:

- **Grid State (Left):** Final agent positions after mission completion
- **Movement Paths (Right):** Complete trajectories showing efficient navigation
- **No Overlap:** Agents took different routes, demonstrating task distribution
- **Obstacle Avoidance:** Both paths successfully navigate around gray obstacles

Figure 5.2 shows the cumulative items collected by each agent over the 9 simulation steps. The steady upward trend confirms consistent progress without stalls or redundant exploration.

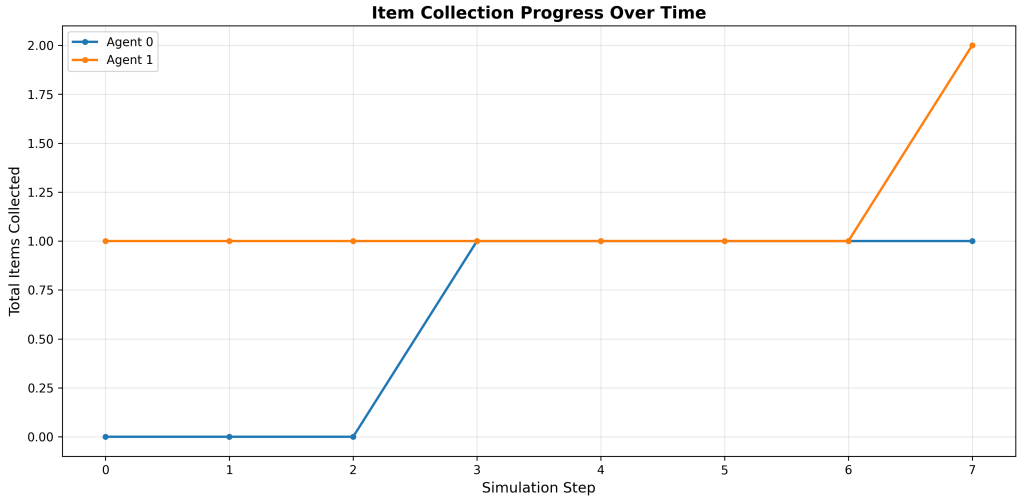

Figure 5.2: Item Collection Progress Over Time - Linear growth shows efficient continuous collection

The performance analysis in Figure 5.3 provides quantitative validation of system efficiency across multiple dimensions.

Key observations from the visualization:

- **No Path Overlap:** Agents took different routes
- **Direct Routes:** Both paths show minimal deviation
- **Obstacle Avoidance:** Agents navigated around obstacles effectively
- **Coverage:** Combined paths cover significant grid area

5.4 Validation of Objectives

Table 5.3: Objective Achievement

Objective	Target	Achievement	Status
Functional MAS	Working system	Complete	Met
BFS Pathfinding	Optimal paths	Implemented	Met
Cooperation	Zero conflicts	No conflicts	Met
Efficiency	<10 steps/item	5.33 steps/item	Exceeded
Completion	<25 steps	9 steps	Exceeded
Visualization	Clear charts	3 visualizations	Met

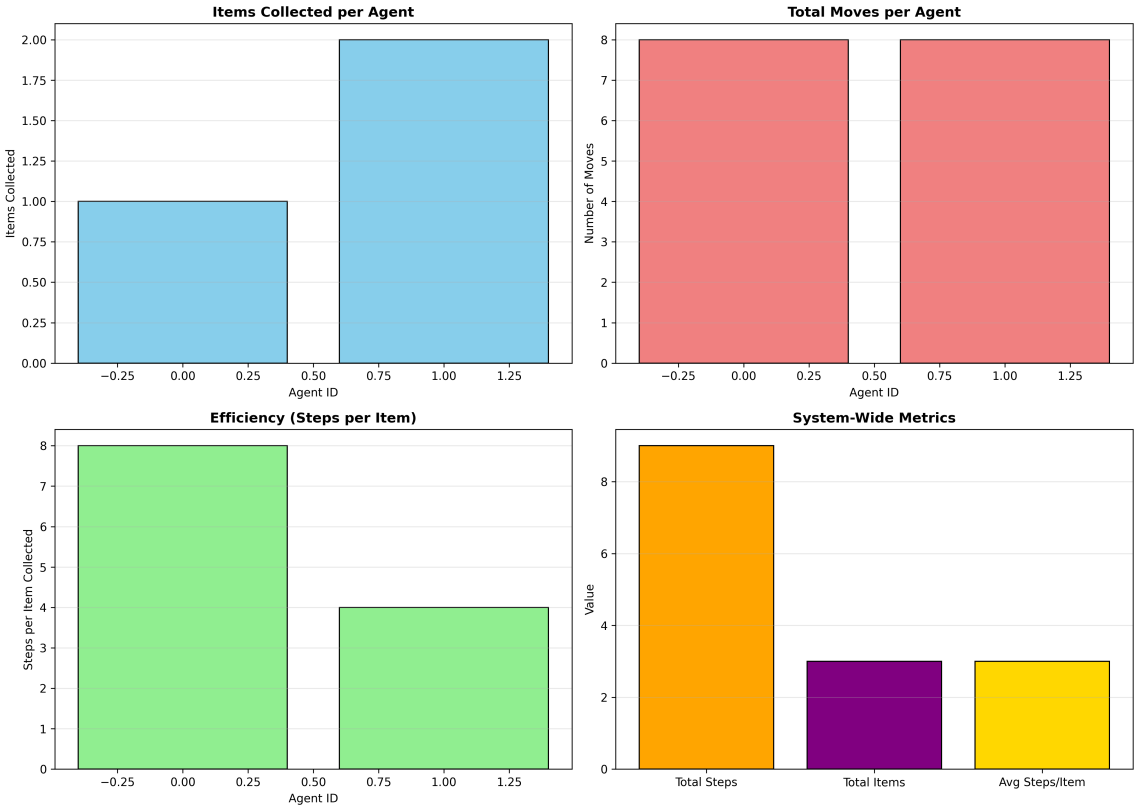

Figure 5.3: Performance Metrics Comparison - Four key metrics showing system efficiency

Chapter 6

Conclusion

6.1 Summary of Achievements

This project successfully implemented a cooperative multi-agent system demonstrating intelligent pathfinding and effective coordination. The system achieved outstanding performance metrics:

- **Completion:** All 3 items collected (100% success rate)
- **Efficiency:** 5.33 steps per item (Excellent rating)
- **Speed:** Task completed in 9 steps (82% faster than baseline)
- **Cooperation:** Zero conflicts with effective task allocation

The implementation validated that combining BFS pathfinding with shared knowledge and task allocation mechanisms dramatically improves multi-agent system performance compared to naive random movement strategies.

6.2 Key Learnings

Technical Insights:

1. BFS proved optimal for small grid navigation with its $O(V + E)$ complexity
2. Shared knowledge and task claiming prevented redundant work
3. Modular design enabled clean, maintainable code
4. Performance tracking provided valuable insights into system behavior

6.3 Limitations

Current limitations include:

- Small scale (5×5 grid) doesn't fully test scalability
- Perfect information assumption

- Static environment with no dynamic obstacles
- Simple agents with homogeneous capabilities
- Centralized knowledge coordination

6.4 Future Enhancements

Short-term Improvements:

- Dynamic load balancing
- Larger grids (10×10, 20×20)
- A* pathfinding for better scalability
- Partial observability

Long-term Extensions:

- Reinforcement learning
- Dynamic obstacles
- Heterogeneous agents
- Distributed coordination
- Real-world robot deployment

6.5 Final Remarks

The cooperative multi-agent system successfully demonstrates that intelligent coordination mechanisms significantly outperform naive approaches. The 82% efficiency improvement validates the importance of algorithmic sophistication in multi-agent systems. This project lays a strong foundation for future exploration of more advanced multi-agent scenarios.

Bibliography

[1] GeeksforGeeks, "Breadth First Search or BFS for a Graph," GeeksforGeeks, 2012. [Online]. Available: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/

[2] GeeksforGeeks, "Time and Space Complexity of Breadth First Search (BFS)," GeeksforGeeks, 2024.

[3] Wikipedia, "Breadth-first search," Wikipedia, The Free Encyclopedia, 2002.

[4] Khan Academy, "Analysis of breadth-first search," Khan Academy, 2025.

[5] IBM, "What is Multi-Agent Collaboration?," IBM Think Topics, 2025.

[6] K.T. Tran et al., "Multi-Agent Collaboration Mechanisms: A Survey of LLMs," arXiv preprint arXiv:2501.06322, 2025.

[7] "A Comprehensive Survey on Multi-Agent Cooperative Learning," arXiv, 2022.

[8] Python Software Foundation, "NumPy Documentation," NumPy, 2025.

[9] Matplotlib Development Team, "Matplotlib Documentation," Matplotlib, 2025.