MAMOF - Multi-Agent Maze Objective Finding
AASMA - 2024/2025 - Group 05

Xiting Wang ist1112191
Laura Cunha ist1112269
Rodrigo Correia ist1112270

1 Abstract

Multi-agent pathfinding in maze-like environments presents unique challenges, particularly regarding coordination and information sharing. This project proposes a cooperative strategy in which agents initially explore independently. Once one finds the exit, it actively searches for the remaining agents to guide them toward the goal. A simulated environment is developed to evaluate different pathfinding algorithms using heuristics such as completion time and steps required for all agents to escape.

Keywords: Multi-agent pathfinding, cooperation, autonomous agents, maze exploration, agent coordination, simulation

2 Introduction

2.1 Motivation

Pathfinding in maze-like environments is a foundational problem in AI, robotics, and operations research. While traditional algorithms such as DFS, BFS, Dijkstra's, and A* perform well for single-agent navigation, real-world applications require multi-agent systems (MAS). In such scenarios, agents must coordinate, share information, avoid collisions, and adapt dynamically. Solving multi-agent pathfinding (MAPF) effectively is vital in domains like autonomous robotics, video games, warehouse logistics, and search and rescue.

2.2 Related Work

Numerous studies have addressed MAPF. A survey by Semuil [1] emphasizes coordination challenges in dynamic environments. Nawaf [2] implemented A* for maze-solving in strategy games, while Vijaya [3] integrated Dijkstra and A* for adaptable navigation in robotic simulations. However, coordination under partial observability remains underexplored. This project addresses that gap by simulating agents in such settings and evaluating cooperative behavior through algorithmic comparisons.

2.3 Problem Definition and Relevance

This project focuses on solving a cooperative pathfinding problem in a maze-like, grid-based environment, where multiple agents must explore the maze and find a viable path to an unknown exit point. As they navigate the environment, the agents cooperate by sharing the information they discover, enabling them to find the goal more efficiently.

The problem is highly relevant due to its strong practical applications in areas such as swarm robotics, autonomous vehicles, disaster response, and planetary exploration, where multiple autonomous agents need to navigate unknown or partially known environments collaboratively. Understanding how agents can coordinate to explore and share knowledge is fundamental to improving autonomy and robustness in these domains.

This problem directly relates to two key topics addressed in the course:

- **Multi-Agent cooperation**: This chapter explores how agents work together to accomplish a common goal. In our system, agents share explored map tiles, their positions, and discovered information to coordinate their actions more efficiently.
- **Partial observability:** This topic explores games and environments where the agents do not have full observability of the full state of the simulation. In our system this is reflected by the agents not having knowledge of the full maze, only communicating limited information, and needing to do actions to gain more information of the system.
- **Roles**: This topic addresses the assignment of different roles to agents. In our context, we have a similar setup: initially, all agents take on the role of *Searcher*. When one of them reaches the goal node, and there are still agents who haven't found the path, the agent that reached the goal switches to the *Guider* role and goes back to assist the others.

2.4 Objective

The main objective of this project is to develop a cooperative multi-agent system capable of efficiently finding the exit in a grid-based labyrinth. The agents should:

- Explore the environment and progressively map unknown areas.
- Share discovered information (such as explored tiles or the exit location) with other agents.
- Assist others after finding the exit by guiding them, reducing the time taken for all agents to escape.

By implementing this system, we aim to evaluate different cooperation strategies and pathfinding heuristics, and determine which combination leads to the lowest total escape time for all agents.

3 Approach

3.1 Environment Specification

The environment is modeled as a 2D grid-based maze, where each cell represents either free space (a traversable path) or an obstacle (a wall). Agents are restricted to moving in the four cardinal directions: up, down, left, and right, and cannot pass through obstacles.

The maze is randomly generated, with parameters for configurable size and a random seed, ensuring reproducibility across different simulation runs.

The entire environment, including the maze generation, movement constraints, and simulation logic, was fully implemented by us. This also includes the integration of agent behavior within the environment, enabling interaction and information sharing.

This setup offers a flexible and controlled testbed for evaluating various multi-agent cooperation strategies and pathfinding heuristics in a dynamic and partially observable space.

3.2 System Architecture

To support cooperative behavior in a dynamic maze environment, the system was architected around discrete timesteps, allowing each agent to perform a single action per step. This approach simplifies synchronization and enables modular control over agent behavior.

The core components of the architecture are as follows:

- **UI Module:** Manages the user interface and sets flags based on selected parameters.
- **Simulation Module:** Responsible for managing the environment and advancing discrete timesteps.
- **Maze Module:** Renders the maze as a 2D grid, with cells classified as either empty (traversable) or walls (obstacles).
- **Agent Module:** The module defines autonomous agents with memory, movement, and communication abilities. Their behavior

adapts to the current phase and cooperation setup. It abstracts various algorithms (custom DFS, A*, backtracking), letting agents choose based on context, and also manage and synchronizes shared information like positions and explored areas at each timestep.

- **Headless Module:** Allows running the simulation without a graphical interface, using command-line flags provided by the user to facilitate batch running.
- **Metrics & Evaluation Layer:** Runs simulations, captures data such as the number of timesteps, and generates output plots for performance evaluation and comparison across configurations.

3.3 Simulation Parameters

In order to set up the simulation and the environment, there are several parameters that can be configured:

- **Number of agents:** number of agents in the simulation;
- **Map size:** size of the maze map;
- **Simulation speed:** speed of the simulation;
- **Seed:** random seed to generate the maze.

In addition, the system includes several communication actions that can be enabled to support different strategies (further explained in section §3.5):

- **Share goal:** share the position of the goal;
- **Share positions:** share positions of the agents;
- **Share maze:** share explored tiles of the map;
- **Agent guiding:** backtrack for guide agents.

It's also important to refer that the agent guiding action is only effective when agent positions are also shared, as guiding requires knowledge of where other agents are located in order to provide accurate directions.

3.4 Pathfinding Algorithms

Initially, when no agent is aware of the goal's location, each agent performs an **exploration phase** using a custom **depth-first-inspired algorithm** enhanced with multiple heuristics to guide decision-making. These heuristics are evaluated in a specific order, and once one is decisive, the others are not considered for that step. The exploration heuristics include in this order:

- **Goal Check:** If any neighboring node is the goal, it is immediately selected.
- **Explored Check:** Prefer nodes that have not been explored yet by any agent.
- **Manhattan Distance to Self:** Prioritize nodes that are closer to the agent's current position.
- **Distance to Neighbors:** Prefer nodes that are farther from already-explored neighboring nodes, encouraging broader exploration early on.

Once any agent finds the goal, all agents switch to a **navigation phase**, using an **A*** algorithm to move efficiently toward the goal. The A* implementation also uses a chain of heuristics:

- **Goal Proximity:** Prioritize moves that lead closer to the goal.
- **Explored Check:** Avoid redundant exploration.
- **Manhattan Distance to Goal:** Prefer paths that reduce the distance to the goal.

3.5 Agent Communication

This behavioral design supports a cooperative exploration scenario, though several cooperation modes were implemented to evaluate different strategies, including:

- **Selfish Agents:** Do not share any information.
- **Goal Sharing:** Only share the goal location when found.
- **Map Sharing:** Share explored tiles with other agents.
- **Position Sharing:** Share current agent positions in real-time.
- **Agent Guiding:** Actively guide others once the goal is found.

To enable **agent guiding**, agents must share their current positions. When an agent reaches a node and detects it is the goal, it selects one of the agents that has not yet reached the goal and begins a backtracking procedure using A* to compute an optimal path from the current position to the target agent. If the selected agent does not yet have a known path to the goal, it will follow the guidance path provided by the guiding agent.

The backtracking process continues until one of the following conditions is met:

- The **guiding agent reaches** the agent still searching for the goal;
- The **target agent independently finds a valid path** to the goal, rendering further guidance unnecessary.

This approach allows agents to actively assist each other, improving overall system efficiency and reducing the total time required for all agents to escape the maze.

3.6 Limitations, Possible Solutions & Future Work

While the system architecture provides modularity and control through discrete timesteps, it introduces some limitations that impact realism and scalability. Below are key limitations along with potential directions for improvement:

- **Asynchronous Timesteps:** Agents act in discrete, asynchronous steps, which makes coordination easier but reduces realism. A finer-grained or event-driven timing model could improve this.
- **Collisions:** The current system doesn't handle collisions, allowing agents to overlap and waste steps. Collision detection and avoidance methods could improve this.
- **Scalability of Communication:** Currently the simulation runs all agents in a single machine but realistically agents would be physically independent autonomous systems that would need to use the network to share communication, in this scenario there would need to be changes to make this communication efficient and scalable, avoiding broadcasting from all agents to all other agents and also deal with all implications of network programming like packet loss and latency.

4 Results

4.1 Experimental Setup

To evaluate the performance of cooperative behaviors under various configurations, we conducted **20 simulations per configuration**, each using a different random seed. The experiments were run for **three different agent-environment** setups: 2 agents on an map size of 8, 5 agents on a map size of 8, and 32 agents on a map size of 64.

4.2 Configurations Compared

We compared **five** configurations, each representing a different level of cooperation and shared knowledge:

- **Config 1:** No information sharing.
- **Config 2:** Share agent positions.
- **Config 3:** Share agent positions and discovered goal.
- **Config 4:** Share positions, goal, and explored maze tiles.
- **Config 5:** Full sharing (positions, goal, map, and guiding assistance).
- **Config 6:** Share positions, map, and guiding assistance.

4.3 Metrics and Graphs

We analyzed the experimental results by plotting comparative graphs across different configurations and setups, allowing a clearer visualization of the performance and variability of each setup.

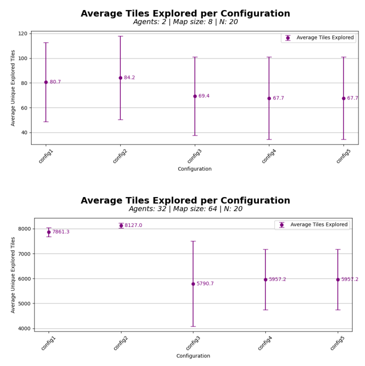


Figure 1: Number of tiles/nodes explored across the different setups.

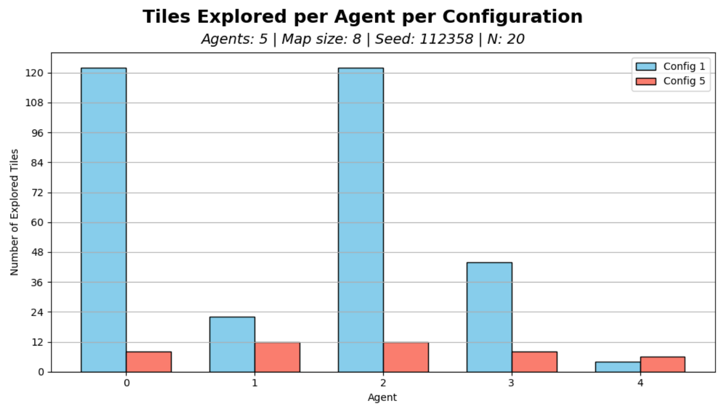


Figure 2: Number of tiles/nodes explored in a setup with 5 agents and 8x8 grid, with configs 1 and 5 for seed 112358.

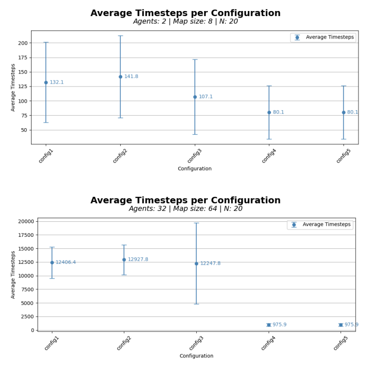
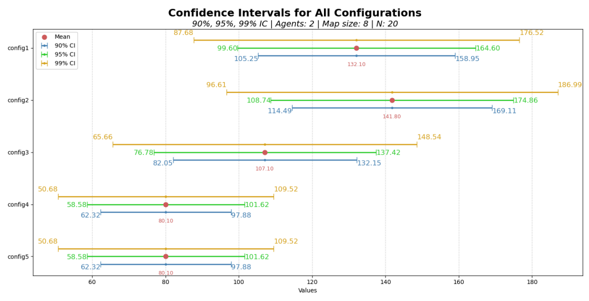


Figure 3: Timesteps distribution for each configuration across the different setups.



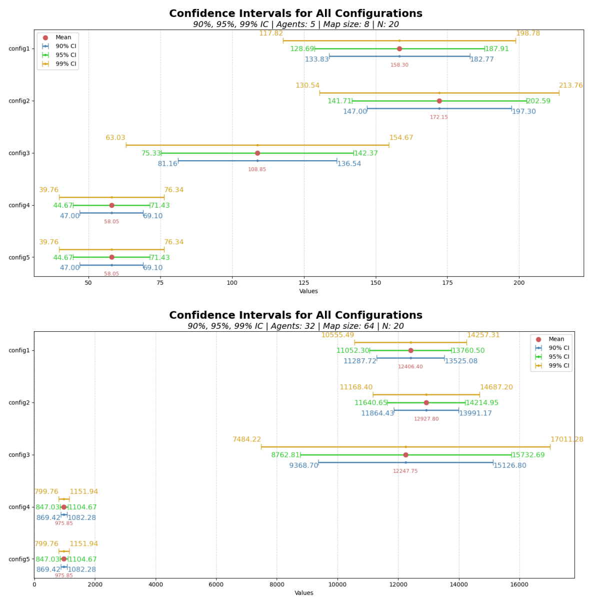


Figure 5: Confidence intervals of 90%, 95% and 99% with mean, for all configuration across the different setups.

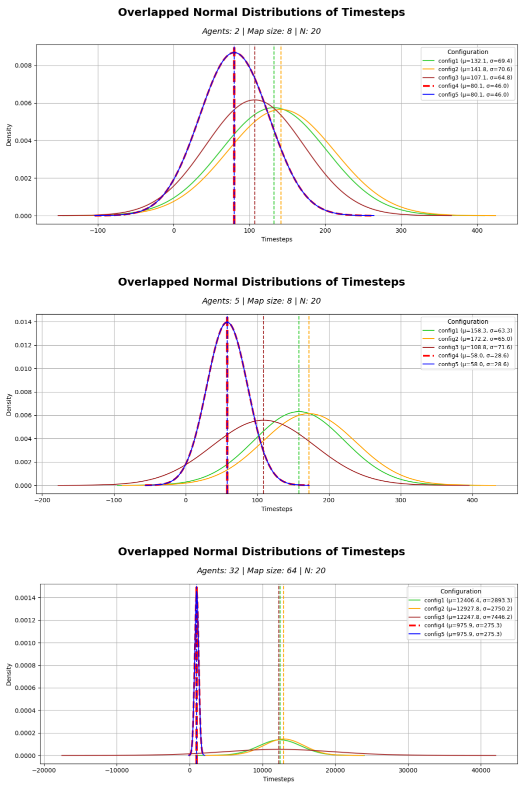


Figure 6: Timesteps distribution for different configurations and setups.

4.4 Critical Analysis

This section presents a performance analysis of various configurations across different setups.

In Figure 1, we observe the range of tiles explored by each configuration per setup, along with their averages. From this, we note that Configs 1 and 2 explore more tiles than the others. This is expected, as these configurations involve either no communication or only position sharing, and without shared information, agents tend to explore independently, leading to greater coverage. Furthermore, Config 2 consistently explores slightly more tiles than Config 1, this is likely because sharing positions can lead agents to pursue tiles farther from others, due to heuristic decisions in the algorithms, potentially deviating more from the goal and increasing exploration time.

Configs 4 and 5 exhibit a similar number of tiles explored, suggesting that the guiding mechanism in Config 5 did not provide a significant improvement, having may have helped in specific samples. Also, for instance, with only two agents, the guided agent might explore more tiles than the one assisting, while without guiding, the roles might reverse, leading to a balanced outcome overall.

During preliminary testing, we also evaluated a configuration that shared only the maze, and we conclude that it were the worst and was excluded from the main analysis. The issue stems from the agents' heuristics, since they prefer unexplored tiles, if one agent reaches the goal and doesn't share this information, others will continue exploring inefficiently, ignoring these tiles until the end.

In Appendix A (Figure 8), we provide detailed results of tiles explored per sample for each configuration and setup. For a more focused view, we selected the sample with seed 112358 using 5 agents in an 8×8 grid, and we plotted the number of tiles explored by each agent for Configs 1 and 5 in Figure 2. In Config 1, two agents explored approximately 120 tiles each, while the others also covered large areas. In contrast, Config 5 shows a more uniform exploration effort, where the two most active agents explored about 12 tiles each, while the remaining agents explored between 6 and 8 tiles. This demonstrates that increased communication also reduces the number of tiles each agent needs to explore.

We also analyzed the timesteps (Figure 3) across all setups, where Configs 4 and 5 consistently achieve the lowest step counts, as they feature the highest levels of communication, allowing agents to reach the goal faster. However, despite Config 5 including a guiding mechanism, its performance is nearly identical to Config 4, suggesting that the guiding adds computational overhead (due to extra path calculations) without meaningful benefits when all key information is already shared. Additionally, Config 2 shows a slightly higher average timestep count than Config 1, again due to agents potentially moving farther from the goal, as previously discussed in the tile exploration context. A more detailed data per sample for each configuration can be found in Appendix A (Figure 7).

To evaluate reliability, we computed confidence intervals (CIs) at 90%, 95%, and 99% levels for each configuration and setup (Figure 5), with detailed sample breakdowns in Appendix A (Figure 9). As CIs indicate the range within which the true population mean likely falls. For example, we can say with 95% confidence that, for the 32-agent, 64×64 grid setup, the mean of 975.85, for the Config 4, would fall within this range in 95% of repeated experiments.

Configs 4 and 5 consistently have the narrowest intervals, especially as grid size and agent count increase. Narrow intervals imply high precision and consistency, while wide intervals suggest variability or limited sampling. Non-overlapping intervals imply statistically significant differences—for example, between Configs 2 and 4 in the 2-agent, 8×8 setup. Overall, Configs 4 and 5 exhibit the most reliable performance, particularly in larger environments. Config 3 also performs well when computational overhead are lower.

Regarding confidence interval levels, 90% being less rigorous with a higher risk of false positives, 95% providing a standard balance between precision and reliability, and 99% offering greater rigor at the cost of wider intervals, configs 4 and 5 demonstrate the most reliable and consistent performance with fewer errors.

Finally, we examined timestep distributions across configurations and setups (Figure 6), with further details in Appendix A (Figure 10), categorized by standard deviation. Assuming normal distributions, we observed:

- **2 agents, 8×8 grid:** Wide distributions with high standard deviations, the mean values vary significantly. Config 3 has the lowest mean and deviation, indicating better performance, and some distributions are skewed or overlapping, suggesting variability.
- **5 agents, 8×8 grid:** Narrower distributions with higher peaks. Config 5 achieves the lowest mean (58.0), indicating superior efficiency. Also, a better separation is observed, and Configs 4 and 5 clearly outperform others.
- **32 agents, 64×64 grid:** Very high mean timesteps (12,000) and large standard deviations (>2700). Configs 4 and 5 achieve the lowest means and similar deviations, suggesting both efficiency and stability. In contrast, Configs 1 and 2 have the highest means and deviations, indicating poor performance. Here, standard deviation is especially important given the high variability, making Configs 4 and 5 the safest choices.

Interestingly, in several scenarios, Config 3 performs comparably to the best configurations, this could be attributed to reduced redundancy and avoidance of information overload, allowing agents to act more independently and in parallel.

So we can conclude that there is a clear trade-off between communication and performance: too little communication limits coordination, while too much—especially with guiding—adds unnecessary computational costs, and also, Configs 3 and 4 appear to strike the best balance, offering strong efficiency and consistency without overwhelming agents.

It's also important to mention that not all configuration combinations were plotted. Preliminary tests showed that some yielded unremarkable results. The focus of this analysis was to progressively test increasingly sophisticated heuristics.

Configuration	Pros	Cons
1 (None)	Zero overhead; high independence	No cooperation; redundant exploration
2 (Position)	Basic agent awareness	No goal or map sharing; still redundant
3 (Position + Goal)	Faster convergence; less redundancy	No map sharing
4 (Position + Goal + Map)	Strong coordination; fewer steps	Increased communication cost
5 (Position + Goal + Map + Guiding)	Best performance in large mazes; balanced workload	High computational overhead; minimal gain over Config 4

Table 1: Qualitative summary of the analyzed configurations

5 Conclusion

This project successfully demonstrated an agent-based simulation with basic orientation and movement behaviors. While the current model works well for simple scenarios, there is room for improvement in collision handling, communication realism, and computational efficiency. Future enhancements will make the simulation more accurate and scalable, enabling more complex and realistic agent interactions.

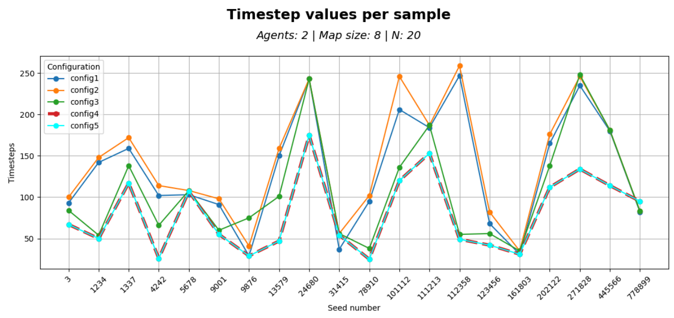
Bibliography

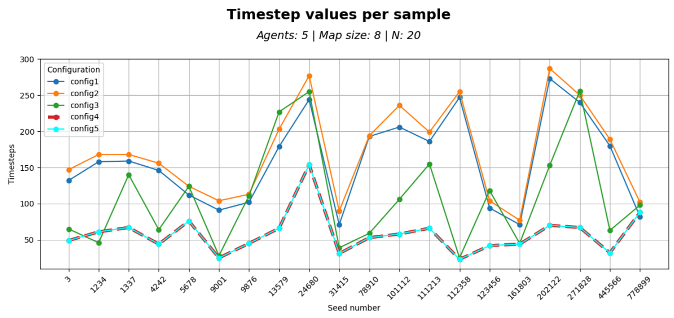
[1] S. Tjiharjadi, S. Razali, and H. A. Sulaiman, "A Systematic Literature Review of Multi-Agent Pathfinding for Maze Research," *Journal of Advances in Information Technology*, vol. 13, no. 4, pp. 358–367, 2022, [Online]. Available: https://www.jait.us/uploadfile/2022/0629/20220629044024452.pdf

[2] N. H. Barnouti, S. S. M. Al-Dabbagh, and M. A. S. Naser, "Pathfinding in Strategy Games and Maze Solving Using A* Search Algorithm," *Journal of Computer and Communications*, vol. 4, no. 11, pp. 15–25, 2016, doi: 10.4236/jcc.2016.411002.

[3] J. Vijaya, A. K. Baghel, and A. S. Mishra, "Dynamic Maze Solver: Integrating Advanced Algorithms for Optimal Path-finding in Varied Environments," 2024, [Online]. Available: https://ieeexplore.ieee.org/document/10503462

Appendix A - Metrics and Graphs





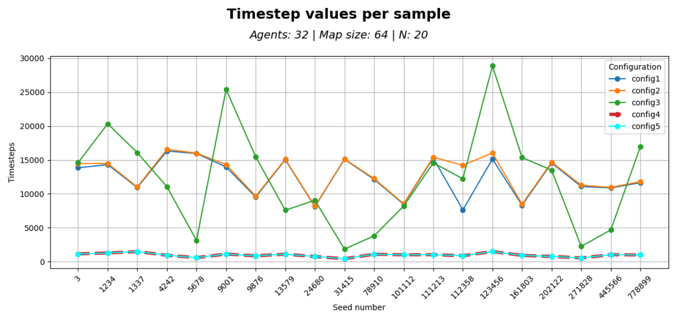
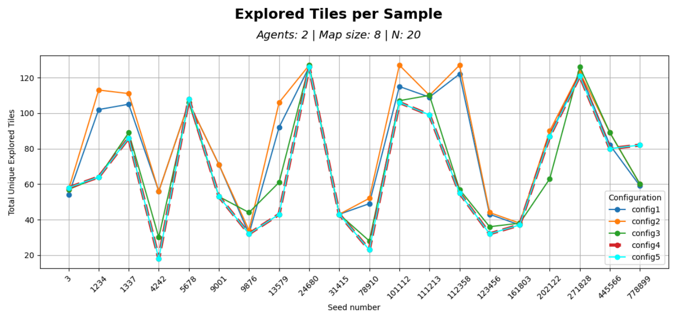
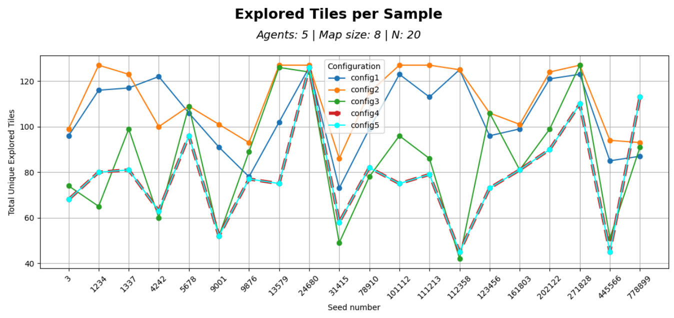


Figure 7: Timesteps distribution for each configuration across the different setups.





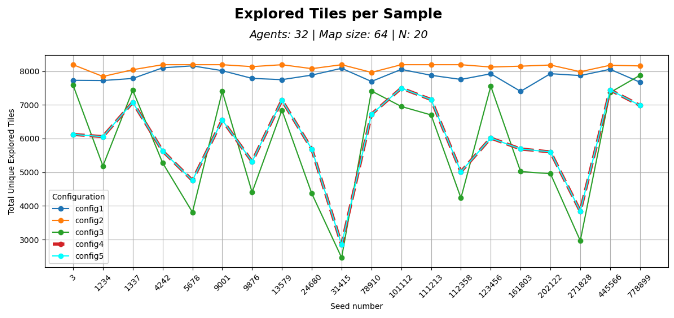


Figure 8: Tiles explored distribution for each configuration across the different setups.

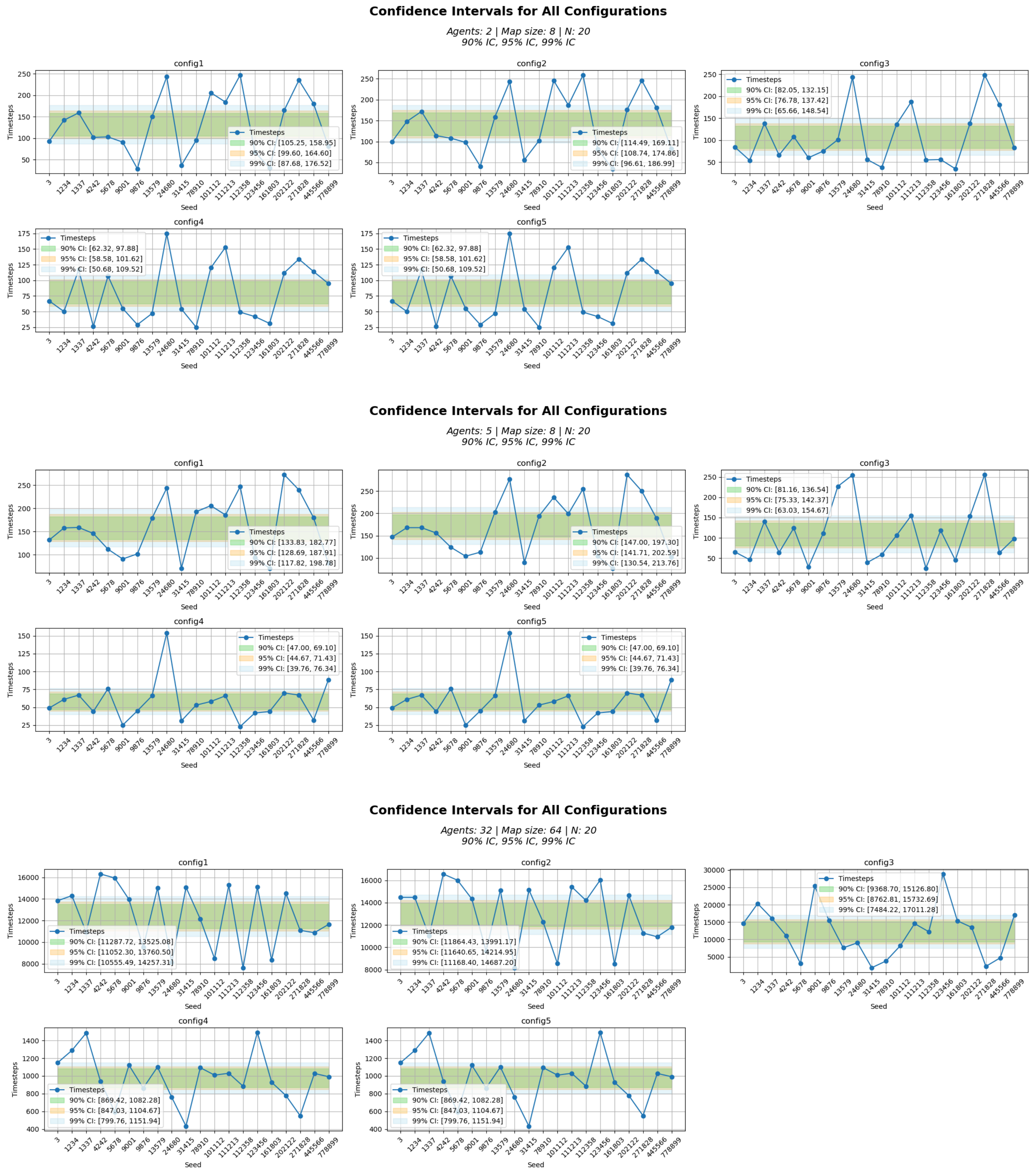


Figure 9: Confidence intervals of 90%, 95% and 99% with mean, for all configuration across the different setups, with discriminate samples.

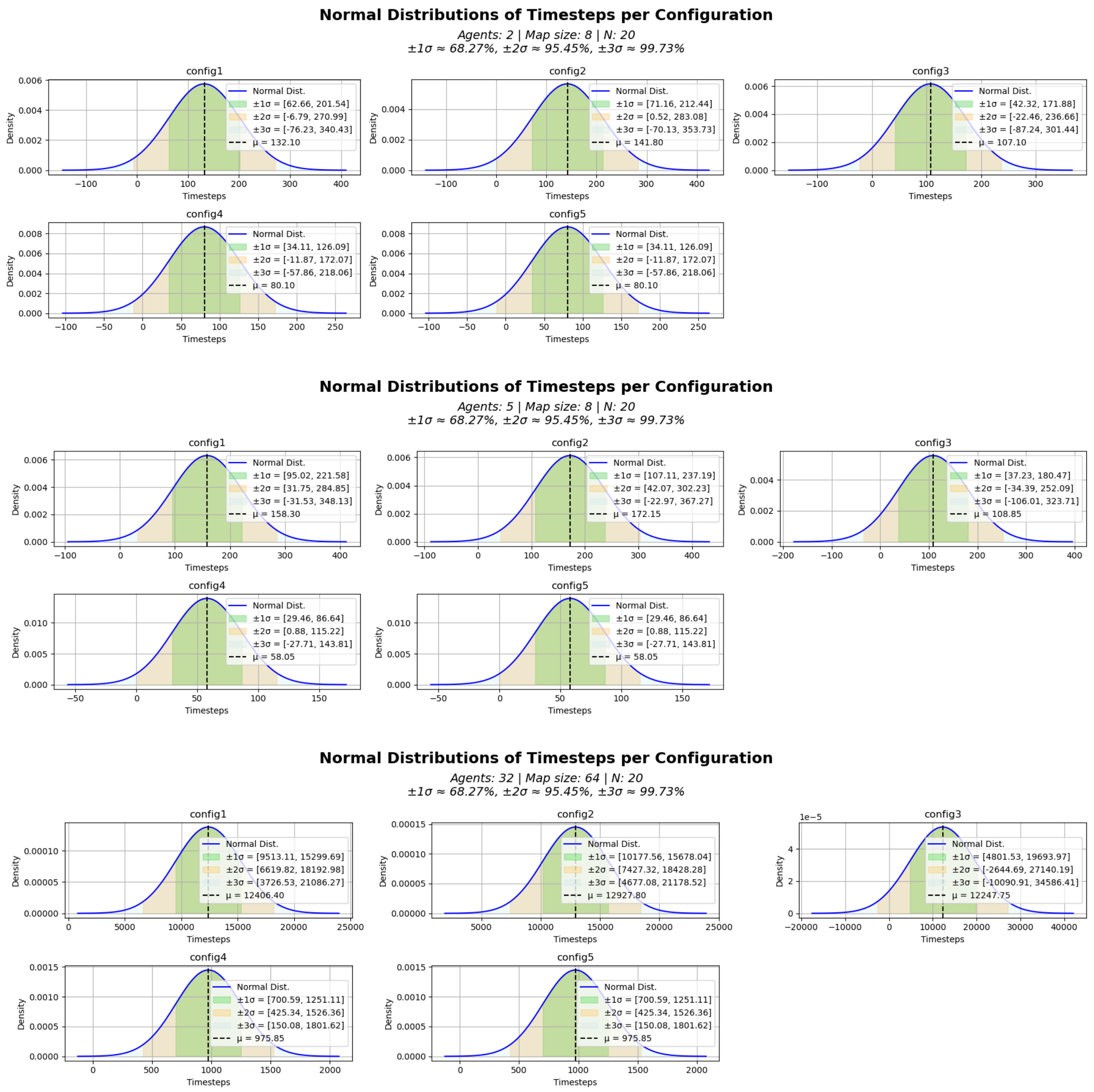


Figure 10: Timesteps distribution for different configurations and setups, with discriminate samples.