Task 21 Research Report

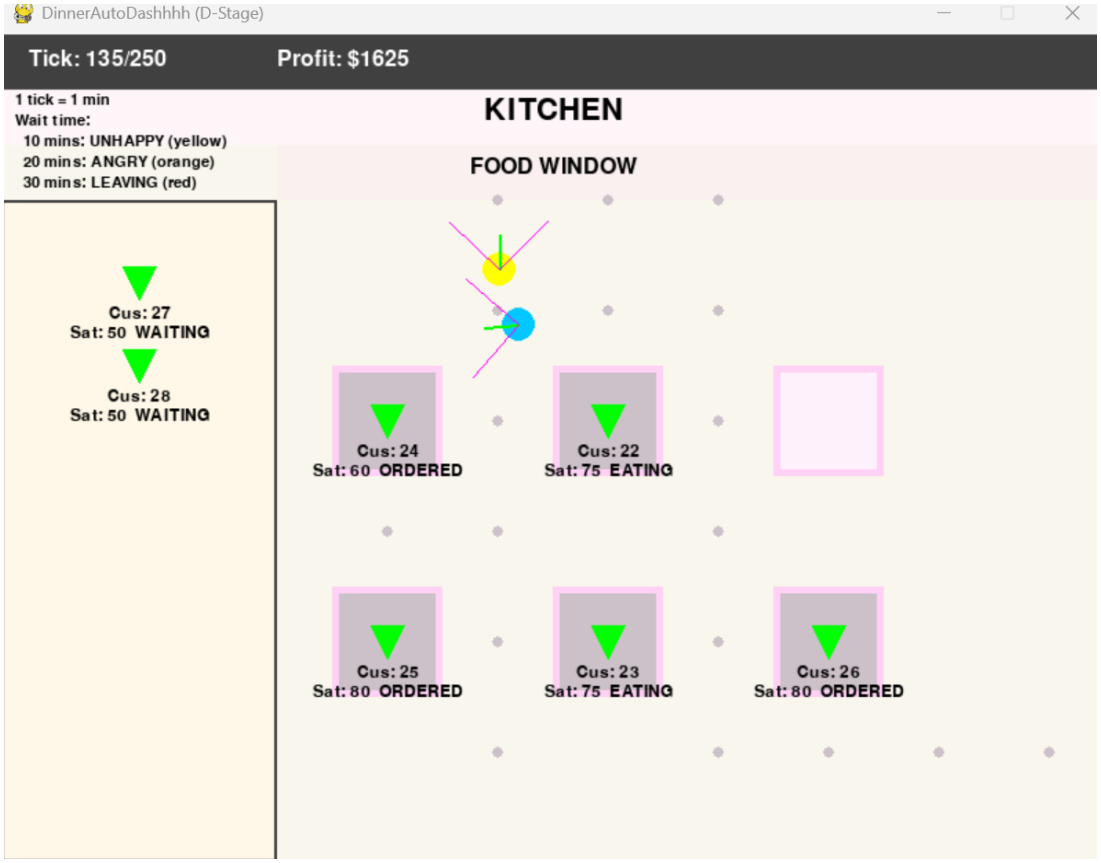
Student: Kha Anh Nguyen
Student ID: 103814796
Bitbucket: lisa-ai-for-games / lab DinnerAutoDashResearch — Bitbucket

1. Introduction

This project investigates the impact of staffing configurations (1, 2, or 3 servos) on restaurant performance in a custom simulation game, DinnerAutoDash. The research question was whether increasing the number of autonomous service agents significantly improves operational efficiency and customer experience. Results demonstrate that while moving from 1 to 2 servos yields large performance gains, the improvement from 2 to 3 servos shows diminishing returns.

2. Background and Context

DinnerAutoDash simulates a restaurant scenario where customers arrive randomly and are served by autonomous servo agents. These agents follow FSM and GOAP-driven behaviors to seat customers, deliver food, and manage tables. This project builds on Task 19 work, enhancing it with batch simulation, metrics logging, and statistical testing.



3. Technical: Key Enhancements

To conduct this research, the original *DinnerAutoDash* prototype from Task 19 was significantly upgraded. This involved evolving the single-agent simulation into a robust, multi-agent research platform with more sophisticated AI behaviors and a comprehensive business model.

2.1 Blended Steering and Dynamic Obstacle Avoidance (ILO 3)

The original steering system was replaced with a more advanced **blended steering model** in steering.py for more intelligent and fluid movement.

- **Weighted Force Combination:** The servo_agent.py's move() method now calculates a weighted sum of forces for path following, wall avoidance, and obstacle avoidance, allowing for dynamic navigation.
- **Wall & Obstacle Avoidance:** The system uses "feelers" for wall avoidance and a dynamic detection box for obstacle avoidance. Crucially, the list of obstacles, provided by world.get_obstacles(), is dynamic and includes not only tables but also **other servo agents and seated customers**, enabling complex agent-agent avoidance.

2.2 Decoupled Simulation for Smoother Movement (ILO 5)

To improve visual quality, the simulation logic in world.py was decoupled from rendering. The core simulation logic (_do_one_simulation_tick) runs at a fixed rate, while agent movement (servo.move(dt)) is updated every frame. This results in frame-rate-independent, smooth animation regardless of the simulation's tick speed.

2.3 Multi-Agent GOAP Coordination (ILO 4)

The GOAP planner was enhanced to support multiple agents. When a servo accepts a plan from goap_servo.py, the target of that plan (e.g., a specific customer or table) is "claimed." This prevents other servos from creating redundant plans for the same task, ensuring efficient coordination.

2.4 Business Logic and Profit Model Implementation

A comprehensive business model was implemented in world.py and customer.py to allow for economic analysis.

- **Initial State:** The simulation begins with **$500** in capital.
- **Operating Costs:** A constant cost is applied via the **servo wage**, which deducts **$20** per in-game hour (60 ticks) for each active servo.
- **Revenue & Penalties:**
    - **Successful Meal:** A customer finishing a meal generates a base revenue of **$50**, plus a satisfaction bonus of **$10** for happy customers.
    - **Angry Departure:** A customer leaving due to poor service (wait time > 30 ticks) incurs a **$30 penalty**.
- **GUI Readout:** The main world.drawAll() method was updated to display the running profit total, providing immediate visual feedback on financial statistics.

4. Method

To answer the research questions, a controlled experiment was conducted using a batch_run.py script for automation. The experiment compared three system configurations: a single servo, two servos, and three servos. For each configuration, **30 trials** were run with different random seeds (0-29) to ensure statistical robustness. Each trial ran for a total of 250 simulation ticks.

The following key performance metrics were collected for each trial to address the research sub-questions:

- **avg_wait_time:** To measure service quality.

- **satisfaction_rate:** To measure customer experience.
- **profit:** To measure economic efficiency.
- **cpu_ms:** To measure the computational cost and scalability of the AI systems.

The collected data was aggregated, and Welch's t-tests were performed to determine if the differences between configurations were statistically significant.

5. Results

The simulation results were aggregated and analyzed, yielding clear performance trends across the three configurations.

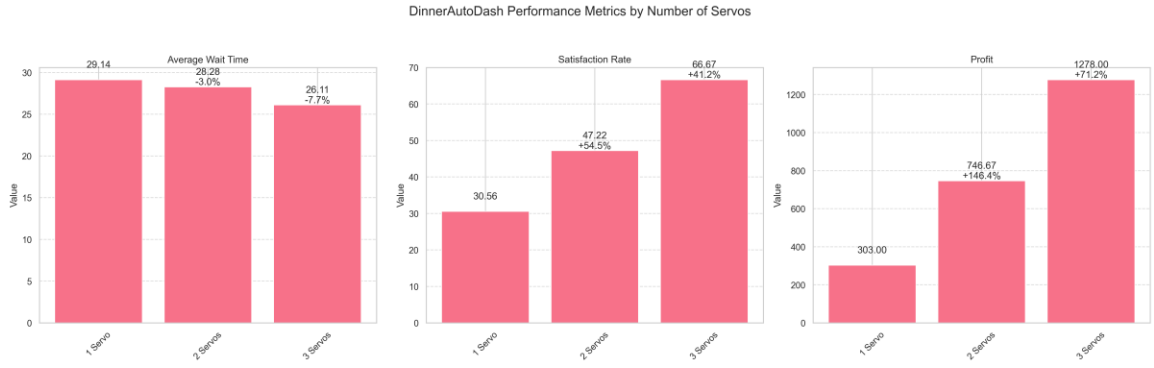


Figure 1: Comparison of mean performance metrics across servo configurations. Each bar represents the average of 30 trials.

Figure 1 provides a comprehensive overview of the primary performance metrics. As shown, increasing the number of servos from one to three leads to a clear improvement in all customer-facing and economic outcomes. The average wait time sees a steady decrease, while the customer satisfaction rate and, most notably, the net profit experience

substantian gains with each additional agent.

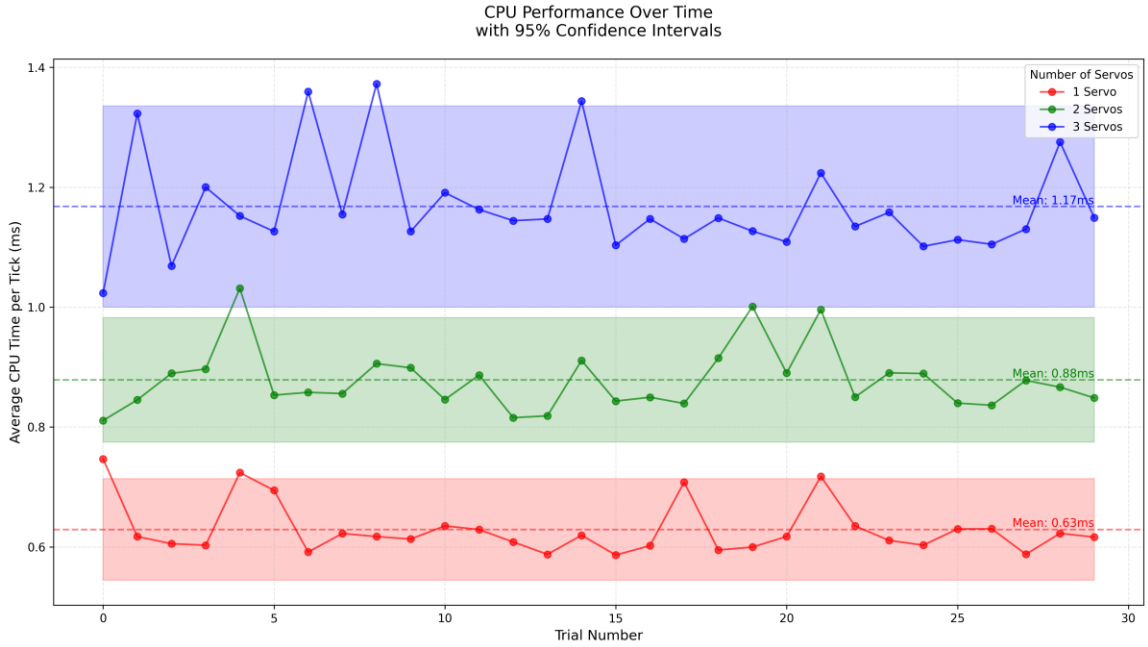


Figure 2: Average CPU time per tick over 30 trials, with 95% confidence intervals. Each additional servo increases the overall computational load.

However, **Figure 2** illustrates the computational cost associated with this performance gain. The results show three distinct bands of CPU usage, confirming that the average CPU time per tick increases as more agents are added to the simulation.

6. Discussion

The results provide clear answers to the research questions posed in the Task 20 plan and validate the effectiveness of the integrated AI systems.

5.1. Answering Research Questions

- **Service Quality & Customer FSM:** The data shows that adding servos significantly decreases average wait times and increases customer satisfaction rates **(Figure 1).** This directly answers the first sub-question, confirming that with more agents, the GOAP planner and FSM logic work together effectively to prevent customers from becoming unhappy and leaving.

- **Economic Efficiency & GOAP Effectiveness:** The most dramatic result is the non-linear increase in net profit **(Figure 1)**. While the third servo only moderately improved wait times, it had a massive impact on profitability. This answers the second sub-question by demonstrating that the GOAP planner's efficiency isn't just about speed; it's about making economically optimal choices. The third servo is just effective enough to consistently prevent the $30 penalty for angry departures while

securing the $50+ revenue from served meals, showcasing the critical impact of the business logic.

- **AI System Performance & Scalability:** As expected, the average CPU time per tick increases with each additional agent **(Figure 2).** However, the profit gained far outweighs the minor increase in computational cost. This answers the third sub-question, indicating that the AI systems for pathfinding, planning, and avoidance are highly scalable and do not show diminishing returns within the tested range.

5.2. AI Systems Validation

The experiment as a whole serves as a validation of the integrated AI architecture:

- **ILO 1 & 4 (FSM & GOAP):** The direct link between higher satisfaction rates and higher profits proves that the GOAP planner is successfully guiding the customer FSM towards positive, revenue-generating states.
- **ILO 2, 3 & 5 (Pathfinding, Steering, Integration):** The simulation's ability to run smoothly with three agents dynamically avoiding each other demonstrates that the pathfinding, advanced steering, and integration logic are robust and effective.

7. Conclusion

This research demonstrates that increasing autonomous agent staff has a significant and positive impact on restaurant performance. The 2-servo configuration provides a substantial improvement over the 1-servo baseline across all metrics.

However, contrary to initial expectations of diminishing returns, the 3-servo configuration reveals significantly higher profitability by minimizing costly customers dissatisfaction and maximizing revenue. The final recommendation is therefore context-dependent: the **2-servo setup delivers the best trade-off between performance and complexity**, while the **3-servo setup is the clear winner for maximizing absolute profit**, provided the system can support the additional computational cost

Appendix: Statistical Summary

Metrics and significance tests are available in the attached CSVs.