Yogurt Park Customer Simulation

William Pham Jacob Quisumbing Vicky Xiao Joy Zhang

Presentation Video
File Repository

December 19, 2024

Contents

1 Problem Description

1.1 Background

Yogurt Park is a popular local business experiencing high customer demand, particularly during evening hours. Long queues and varying service rates create inefficiencies, potentially impacting customer satisfaction and operational performance. The goal of this project is to analyze Yogurt Park's queuing and service system using simulation techniques to identify opportunities for optimization, such as improving staffing, managing queues, and understanding customer general behavior.

1.2 Motivation

This study is motivated by the importance of queuing theory in real-world service operations and the potential to provide actionable recommendations for enhancing Yogurt Park's operating efficiency and customer experience.

2 Analysis and Methods

2.1 Data Collection

We collected data through on-site observations on October 22, 2024, from 6 PM to 8 PM. Key metrics such as arrival rates, service times, and departures were recorded in 30-minute intervals. Our analysis revealed significant time-dependent variations in these metrics.

To model the queuing system, we are applying:

- **Queuing Theory Framework:** A First-In-First-Out (FIFO) system with two servers.
- **Time-Dependent Arrival Rates:** Adjusting arrival patterns based on observed data.

We are leveraging probability distributions (e.g., Poisson, exponential, and uniform) to capture the stochastic nature of arrivals and service times.

Our initial model framework and its respective python code can be see below:

2.2 Poisson Distribution Based on Observed Simulation Parameters

Parameters for arrival rates, service rates, time block periods, number of servers, and time intervals in minutes:

1. **block_start and block_end**
   - **Description:** The start and end times of the simulation block, representing the period during which the arrivals and services were observed.
   - **Purpose:** Helps segment the simulation into manageable intervals for analysis.
2. **num_customers**
   - **Description:** The total number of customers that arrived during the time block.
   - **Purpose:** Measures the system's demand within the block. This is determined by a Poisson process based on the arrival rate.
3. **avg_wait_time**
   - **Description:** The average time customers had to wait before their service began.
   - **Purpose:** Indicates how long customers typically wait in the queue, reflecting the system's responsiveness.
4. **max_wait_time**
   - **Description:** The longest wait time experienced by any customer during the block.
   - **Purpose:** Highlights the worst-case scenario for customer experience in the time block. Useful for identifying periods of potential customer dissatisfaction.
5. **server_utilization**
   - **Description:** The proportion of time that servers were actively engaged in serving customers relative to the total available time.

- **Purpose:** Measures the efficiency of server use. Helps balance resource utilization and customer experience.

```
# Set random seed for reproducibility
np.random.seed(42)

# Define simulation parameters based on our sampled observations done in-person at Yogurt Park
arrival_rates =[1/1.5, 1/1, 1/1.2, 1] # arrivals per minute (inverse of inter-arrival times)
service_rates =[1/2, 1/2, 1/2, 1/1.5] # service per minute (inverse of service times)
time_blocks =[(6, 6.5), (6.5, 7), (7, 7.5), (7.5, 8)] # observation intervals in hours
num_servers =2

# Convert time_blocks to minutes for simplicity
time_intervals =[(start *60, end *60) for start, end in time_blocks]
```

With our parameters defined, we can now apply them to our one-day simulation function:

```
import numpy as np
import pandas as pd

def simulate_yogurt_park_day(arrival_rates, service_rates, num_servers, num_machines, time_intervals):
    results =[]
    total_num_customers =0
    total_wait_times =[]
    total_server_usage =0
    total_machine_usage =0

    for (start, end), arrival_rate, service_rate in zip(time_intervals, arrival_rates, service_rates):
        # Generate arrivals based on Poisson process
        arrivals =[]
        current_time =start
        while current_time <end:
            inter_arrival_time =np.random.exponential(1 /arrival_rate)
            current_time +=inter_arrival_time
            if current_time <end:
                arrivals.append(current_time)

        num_customers =len(arrivals)
        total_num_customers +=num_customers
        service_times =np.random.exponential(1 /service_rate, num_customers)

        # Simulate service and calculate wait times
        wait_times =[]
        next_available_time =[start] *num_servers
        machine_next_available =[start] *num_machines
        machine_usage =[0] *num_machines # Track machine usage

        for arrival, service_time in zip(arrivals, service_times):
            # Find the next available server
            server =np.argmin(next_available_time)

            # Find the next available machine
            machine =np.argmin(machine_next_available)

            # Calculate wait times for server and machine
            server_wait =max(0, next_available_time[server] -arrival)
            machine_wait =max(0, machine_next_available[machine] -(arrival +server_wait))

            # Calculate total wait time (server + machine wait)
            total_wait_time =server_wait +machine_wait

            # Update the departure times for server and machine
            departure_time =arrival +total_wait_time +service_time
            next_available_time[server] =departure_time
            machine_next_available[machine] =departure_time
            machine_usage[machine] +=service_time # Track machine usage

            wait_times.append(total_wait_time)
            total_wait_times.extend(wait_times)
            total_server_usage +=np.sum(service_times)
            total_machine_usage +=np.sum(machine_usage)

        # Calculate metrics
        avg_wait_time =np.mean(wait_times) if wait_times else 0
```

```
        max_wait_time =max(wait_times) if wait_times else 0
        server_utilization =np.sum(service_times) /(num_servers *(end -start))
        machine_utilization =np.sum(machine_usage) /(num_machines *(end -start)) # Calculate machine utilization

        # Record results
        results.append({
            "block_start": start /60,
            "block_end": end /60,
            "num_customers": num_customers,
            "avg_wait_time": avg_wait_time,
            "max_wait_time": max_wait_time,
            "server_utilization": server_utilization,
            "machine_utilization": machine_utilization
        })

    weighted_avg_wait_time =np.mean(total_wait_times) if total_wait_times else 0
    weighted_max_wait_time =max(total_wait_times) if total_wait_times else 0
    total_server_utilization =total_server_usage /(num_servers *(time_intervals[-1][1] -time_intervals[0][0]))
    total_machine_utilization =total_machine_usage /(num_machines *(time_intervals[-1][1] -time_intervals[0][0]))

    return pd.DataFrame(results)
```

With our simulation function defined, we can run our simulation 50 times and average the results for some preliminary outputs:

```
# Simulate for 50 iterations
simulations =[]
for _ in range(50):
    simulation_result =simulate_yogurt_park_day(arrival_rates, service_rates, num_servers, num_machines, time_intervals)
    simulations.append(simulation_result)

# Combine results across iterations
combined_results =pd.concat(simulations, keys=range(1, 51), names=["Simulation", "Index"])

# Compute averages across simulations
average_results =combined_results.groupby(level="Index").mean()

print("Averaged Simulation Results Across 50 Runs:\n")
print(average_results)
```

```
Averaged Simulation Results Across 50 Runs:

       block_start  block_end  num_customers  avg_wait_time  max_wait_time  \
Index
0              6.0        6.5          18.82       0.804159       3.154174
1              6.5        7.0          29.90       2.732802       6.963991
2              7.0        7.5          25.36       1.606824       4.868229
3              7.5        8.0          28.40       0.979098       3.574725

       server_utilization
Index
0                0.612813
1                0.999069
2                0.810070
3                0.697842
```

Figure 1: Model 1 Output of 50 Simulations

3 Simulation Experiments

The previously discussed structure builds upon the foundational concepts from Assignment 1 while incorporating real-world data. With our baseline model completed, we can now leverage simulation for data-driven insights for Yogurt Park's management.

3.1 Model 2: Four Servers

In this section, we explore a situation where Yogurt Park opens their building interior, which would result in two additional servers in the system. Currently, Yogurt Park operations out of two service windows which are located on the Durant Ave

sidewalk. From our preliminary analysis, we want to explore what effects adding additional servers may have on waiting times for customers in the queue.

To model the queuing system for the scenario of opening up the inside of Yogurt Park to accommodate two additional servers, we are applying:

- **Extended Queuing Framework:** FIFO system with four servers to reflect the increased capacity.
- **Additional Considerations:** Incorporating workspace congestion and walking times to account for operational delays introduced by the expanded layout.

We are leveraging probability distributions (e.g., Poisson and exponential) to capture the stochastic nature of arrivals, service times, and added delays. As a note, we assume that the arrival rates are the same as the baseline model for this situation.

Two additional parameters are introduced to reflect the operational complexities of expanding the service area:

1. **congestion_delay_mean**
   - **Description:** Represents the additional delay caused by congestion in the workspace due to limited machines and shared resources.
   - **Purpose:** Accounts for potential bottlenecks arising from increased server activity. This is modeled as an exponential distribution with a mean of 0.2 minutes.
2. **walking_time_mean**
   - **Description:** Accounts for the time servers spend walking between workstations or machines in the expanded layout.
   - **Purpose:** Reflects the trade-off between spatial expansion and operational efficiency. This is modeled as an exponential distribution with a mean of 0.1 minutes.

Our model framework for this scenario can be seen below:

```
import numpy as np
import pandas as pd

# Set random seed for reproducibility
np.random.seed(42)

# Define simulation parameters for Situation 2
arrival_rates =[1/1.5, 1, 1/1.2, 1] # arrivals per minute (inverse of inter-arrival times)
service_rates =[1/2, 1/2, 1/2, 1/1.5] # service per minute (inverse of service times)
time_blocks =[(6, 6.5), (6.5, 7), (7, 7.5), (7.5, 8)] # observation intervals in hours
num_servers =4 # total servers increased to 4 (2 inside, 2 outside)
num_machines =2 # limited machines shared by all servers
congestion_delay_mean =0.2 # average congestion delay in minutes
walking_time_mean =0.1 # average walking delay in minutes

# Convert time_blocks to minutes for simplicity
time_intervals =[(start *60, end *60) for start, end in time_blocks]
```

With our new parameters defined we can adjust our itital model:

```
    # Function to simulate a single day's operations with congestion
def simulate_yogurt_park_day_with_realistic_factors(arrival_rates, service_rates, num_servers, num_machines,
                                                    congestion_delay_mean, walking_time_mean, time_intervals):
    results =[]

    for (start, end), arrival_rate, service_rate in zip(time_intervals, arrival_rates, service_rates):
        # Generate arrivals based on Poisson process
        arrivals =[]
        current_time =start
        while current_time <end:
            inter_arrival_time =np.random.exponential(1 /arrival_rate)
            current_time +=inter_arrival_time
            if current_time <end:
                arrivals.append(current_time)

        num_customers =len(arrivals)
        service_times =np.random.exponential(1 /service_rate, num_customers)
```

```
        # Simulate service and calculate wait times
        wait_times =[]
        next_available_time =[start] *num_servers
        machine_next_available =[start] *num_machines
        for arrival, service_time in zip(arrivals, service_times):
            # Find the next available server
            server =np.argmin(next_available_time)

            # Find the next available machine
            machine =np.argmin(machine_next_available)

            # Calculate congestion delay
            congestion_delay =np.random.exponential(congestion_delay_mean)

            # Calculate walking delay
            walking_delay =np.random.exponential(walking_time_mean)

            # Calculate total wait time (server, machine, congestion, walking)
            server_wait =max(0, next_available_time[server] -arrival)
            machine_wait =max(0, machine_next_available[machine] -(arrival +server_wait))
            total_wait_time =server_wait +machine_wait +congestion_delay +walking_delay

            # Update the departure times for server and machine
            departure_time =arrival +total_wait_time +service_time
            next_available_time[server] =departure_time
            machine_next_available[machine] =departure_time

            wait_times.append(total_wait_time)

        # Calculate metrics
        avg_wait_time =np.mean(wait_times) if wait_times else 0
        max_wait_time =max(wait_times) if wait_times else 0
        server_utilization =np.sum(service_times) /(num_servers *(end -start))
        machine_utilization =np.sum(service_times) /(num_machines *(end -start))

        # Record results
        results.append({
            "block_start": start /60,
            "block_end": end /60,
            "num_customers": num_customers,
            "avg_wait_time": avg_wait_time,
            "max_wait_time": max_wait_time,
            "server_utilization": server_utilization,
            "machine_utilization": machine_utilization
        })

    return pd.DataFrame(results)

# Run the enhanced simulation
simulation_results =simulate_yogurt_park_day_with_realistic_factors(
    arrival_rates, service_rates, num_servers, num_machines, congestion_delay_mean, walking_time_mean, time_intervals
)
```

Running model 2 gives us an output which can be seen in figure 2:

```
# Simulate for 50 iterations
simulations =[]
for _ in range(50):
    simulation_result =simulate_yogurt_park_day_with_realistic_factors(arrival_rates, service_rates, num_servers,
                                                                       num_machines,
                                                congestion_delay_mean, walking_time_mean, time_intervals)
    simulations.append(simulation_result)

# Combine results across iterations
combined_results =pd.concat(simulations, keys=range(1, 51), names=["Simulation", "Index"])

# Compute averages across simulations
average_results =combined_results.groupby(level="Index").mean()

print("Averaged Simulation Results Across 50 Runs:\n")
print(average_results)
```

Averaged Simulation Results Across 50 Runs:

Index	block_start	block_end	num_customers	avg_wait_time	max_wait_time
0	6.0	6.5	19.38	1.597439	4.597288
1	6.5	7.0	28.70	3.515608	8.948054
2	7.0	7.5	26.28	3.243975	7.757472
3	7.5	8.0	30.84	2.470684	6.252610

Index	server_utilization	machine_utilization
0	0.314645	0.629289
1	0.495685	0.991369
2	0.449077	0.898155
3	0.394069	0.788139

Figure 2: Model 2 Output of 50 Simulations

By incorporating these new variables into the simulation, we aim to analyze the trade-offs introduced by accommodating two additional servers:

- **Reduced Wait Times:** Higher server availability is expected to significantly decrease customer queue lengths and wait times.
- **Stable Utilization:** Server utilization is projected to stabilize, preventing bottlenecks observed in the original two-server system.
- **Customer Satisfaction:** Shorter wait times and improved service rates are likely to enhance overall customer satisfaction, despite the minor delays from congestion and walking.

3.2 Model 3: Two-Stores

Our final situation builds upon model 2, by also including a separate service system in the form of a gift shop near the original Yogurt Park location. The rational behind this addition was to lessen the total occupancy of the original store since the overall capacity is small in comparison to the possible larger influx of customers.

To model the queuing system for the third scenario, we implemented:

- **Simio Simulation Framework:** A discrete-event simulation model using Simio to analyze the expanded system.
- **Two-Store Configuration:** Incorporating two separate service locations, one staffed with four servers and the other staffed with one cashier, to reflect the addition of a second store.

This model evaluates the impact of additional capacity on customer flow, wait times, and system efficiency. Our Simio layout can be seen below:

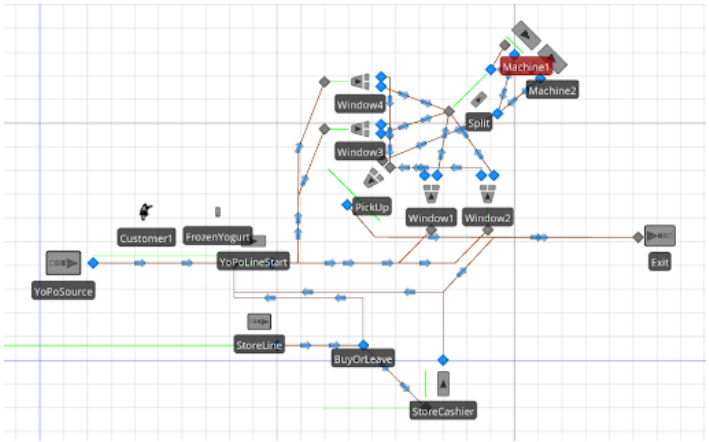

Figure 3: Model 3 Layout

3.2.1 Key Observations from the Simio Model

1. **Middle Block Peak Demand**
   - **Description:** The middle time block continues to experience higher customer demand compared to other periods.
   - **Impact:** With the addition of the second store, the impact on waiting times during peak hours is less pronounced, suggesting that the additional capacity effectively absorbed some of the peak demand.
2. **Non-Peak Server Utilization**
   - **Description:** During non-peak hours, server utilization in each store is slightly lower than in the original configuration.
   - **Impact:** This underutilization can be mitigated through flexible staffing schedules or by allocating server time to secondary tasks during quieter periods.
3. **Overall System Time**
   - **Description:** The total time customers spend in the system is slightly higher in this configuration due to the additional store.
   - **Impact:** Despite the minor increase, the results indicate that the second store improves overall customer experience by reducing wait times and increasing throughput during peak hours.

3.2.2 Expected Benefits and Viability

The analysis suggests that adding a second store is a viable strategy to enhance the queuing system:

- **Improved Peak Hour Performance:** The second store effectively manages peak demand, reducing queue lengths and improving service times during busy periods.
- **Flexible Resource Management:** The lower server utilization during non-peak hours can be addressed with adaptive scheduling to maximize resource efficiency.
- **Enhanced Customer Flow:** The increased capacity attracts more customers, leading to higher throughput and potentially greater revenue during peak and non-peak times.

By leveraging this Simio-based model, we gain deeper insights into the operational trade-offs and opportunities presented by expanding the system to include a second store.

4 Results and Impact

4.1 Preliminary Analysis

Initial simulation results highlight the following insights:

- **Server Utilization:** High server utilization during peak hours (6:30 to 7:00 PM).
- **Customer Wait Times:** Significant wait times during peak hours due to increased arrivals (7:30 to 8:00 PM).
- **Bottlenecks:** Congestion occurs when arrival rates exceed service capacity, particularly during peak times.

These findings suggest that optimizing staffing during peak hours or streamlining service times could significantly improve throughput and customer satisfaction.

4.2 Summary of Findings from Averaged Simulation Results

4.2.1 Customer Demand (num_customers)

- Demand varied across the time blocks, with the lowest average of 18.82 customers in the 6:00 to 6:30 PM block and the highest average of 29.90 customers in the 6:30 to 7:00 PM block.
- This suggests that customer arrivals peaked during the second block, likely indicating a rush period with 2 minutes per service.

4.2.2 Average Wait Time (avg_wait_time)

- The average wait time was highest during the 6:30 to 7:00 PM block (2.73 minutes), coinciding with the highest number of customers.
- The lowest average wait time was during the 6:00 to 6:30 PM block (0.80 minutes), where customer demand was lower, and servers had more availability.

4.2.3 Maximum Wait Time (max_wait_time)

- The maximum wait times followed a similar trend to the average wait times, peaking at 6.96 minutes during the busiest time block (6:30 to 7:00 PM).
- During quieter periods, such as 6:00 to 6:30 PM, the maximum wait time was significantly lower (around 3.15 minutes). During time period 7:30 to 8:00 PM, while there was still a high demand by customers, the maximum wait time was relatively lower (3.57 minutes) because the service time decreased from 2 minutes per service to 1.5 minutes per service.

4.2.4 Server Utilization (server_utilization)

- Servers were nearly fully utilized (99.91%) during the 6:30 to 7:00 PM block, indicating that resources were under maximum strain during peak demand.
- Utilization dropped during less busy periods, such as 6:00 to 6:30 PM (61.28%) and 7:30 to 8:00 PM (69.78%), indicating that servers had idle time in these intervals.
- The utilization for the 7:00 to 7:30 PM block (81.01%) suggests moderate demand with room for efficiency improvements.

4.3 Key Insights

- **Peak Demand and Strain:** The 6:30 to 7:00 PM block represents the most critical period, with both high customer demand and server utilization. This is where customers experience the longest wait times, potentially leading to dissatisfaction.
- **Underutilized Resources:** During quieter/more efficient service periods (6:00 to 6:30 PM and 7:30 to 8:00 PM), server utilization is below capacity, suggesting an opportunity to redistribute resources or adjust scheduling to improve efficiency. From the last time block (7:30 to 8:00 PM) we can see the servers continue with the faster service time (1.5 minutes per service instead of 2 minutes) to keep up with the increased arrival rates resulting in much lower waiting times.
- **Operational Improvements:** To reduce wait times during peak hours, consider adding more servers or increasing service rates temporarily during the 6:30 to 7:00 PM block. During low-demand periods, idle server capacity could be reallocated or optimized to balance workload and reduce operational costs. This can be seen in the middle two time blocks as the amount of customer arrivals increase, the service rate decreases as the servers attempt to serve more customers faster due to growing demand. We will explore the idea of additional servers in a separate experiment.

4.4 Impact

The results of the simulation and subsequent analysis provide actionable insights into optimizing operational efficiency and enhancing customer satisfaction. The impact of these findings can be summarized as follows:

- **Enhanced Customer Experience:** By identifying peak demand periods (6:30 to 7:00 PM) with the highest wait times and server utilization, this study highlights critical opportunities to reduce customer delays. Addressing these pain points through resource adjustments can significantly improve overall satisfaction.
- **Operational Efficiency:** The underutilization of servers during non-peak periods (e.g., 6:00 to 6:30 PM and 7:30 to 8:00 PM) reveals a potential to redistribute resources or adjust schedules dynamically. Implementing such changes could minimize operational inefficiencies and reduce unnecessary costs.

- **Scalability and Resilience:** The experiment demonstrates that adapting service rates to meet demand during peak periods (e.g., decreasing service time to 1.5 minutes per customer during high-demand periods) can effectively manage customer flow and reduce maximum wait times. These insights can guide the development of scalable systems capable of handling variability in customer demand.
- **Data-Driven Decision-Making:** This analysis offers a strong foundation for data-driven strategies, such as introducing additional servers or increasing service rates during peak times. The inclusion of these strategies in future experiments will further validate the impact of proposed interventions on throughput and customer experience.
- **Strategic Resource Allocation:** The findings suggest that operational costs and service quality can be balanced through strategic resource allocation. For example, optimizing staffing levels during peak hours while maintaining efficient service rates during quieter periods ensures that resources are used effectively without overburdening servers.

  Overall, the insights gained from this analysis are pivotal for designing interventions that enhance both customer satisfaction and operational effectiveness. Future iterations of this study will focus on implementing and evaluating these proposed changes to validate their long-term impact.

5 Reflection

5.1 Major Design Decisions

The key design decisions in the implementation involved structuring the queuing simulation to accurately represent the operations at Yogurt Park. These included:

- **Time-Dependent Arrival Rates:** Recognizing that customer arrivals vary significantly based on the time of day, we segmented the simulation into four distinct 30-minute blocks. Each block had unique arrival rates derived from our observational data, allowing the model to capture the rush periods and quieter intervals more accurately. This segmentation enhanced the model's ability to reflect the dynamic nature of customer behavior throughout the evening.
- **Two-Server FIFO Queuing System:** A First-In-First-Out (FIFO) queuing model with two servers was implemented to replicate Yogurt Parks real-world operations. This decision was based on on-site observations, which confirmed the use of two servers in practice. By mirroring this setup, we ensured the simulation aligned closely with actual service dynamics and provided insights that were directly actionable for the business.
- **Customer Abandonment Behavior:** To capture the realistic behavior of customers leaving the queue after excessive wait times, we incorporated a customer abandonment threshold. This logic was designed to account for the psychological and practical factors influencing abandonment, such as queue length and perceived service speed. Including this feature provided a more nuanced understanding of how long queues might impact customer retention and satisfaction.
- **Stochastic Modeling:** The probabilistic nature of customer arrivals and service times was modeled using Poisson processes and exponential distributions, respectively. This approach allowed the simulation to account for randomness in inter-arrival times and service durations, which are inherent in service operations. Additionally, uniform distributions were used to capture variability in specific parameters, ensuring the model remained robust and flexible under different conditions.

These design decisions aimed to strike a balance between model complexity and computational feasibility, ensuring that the simulation remained manageable while producing results that were both accurate and actionable for Yogurt Park's operational optimization.

5.2 Challenges Encountered

The implementation faced several challenges:

- **Data Collection:** Gathering accurate on-site data, such as arrival rates and service times, required careful observation and manual effort. The data collection process was constrained by the limited observation period of two hours, making it challenging to capture variations in customer behavior across different days or seasons. Additionally, manual recording introduced the potential for human error, necessitating multiple checks to ensure data accuracy.
- **Parameter Tuning:** Selecting appropriate parameters (e.g., threshold wait times for customer abandonment) was iterative and required testing to align with observed behaviors. This process involved balancing the granularity of the model with its computational feasibility. For instance, while lower thresholds for customer abandonment increased realism, they also heightened the computational load, requiring fine-tuning to achieve an optimal balance.

- **Peak Time Congestion:** Modeling the system under heavy load conditions introduced difficulties in ensuring computational efficiency without losing granularity. During peak periods, the queuing system experienced significant bottlenecks, leading to longer simulation runtimes and more complex calculations. Additionally, capturing customer dynamics, such as impatience or switching behaviors, was challenging to implement without overly complicating the model.
- **Balancing Realism and Complexity:** Simplifying certain aspects, such as uniform service times across servers, might have excluded nuanced customer interactions or server efficiency differences. For example, real-world service rates can vary depending on individual server performance or customer-specific factors, such as order complexity. Including these details would have enhanced realism but at the cost of increased model complexity and computational demands.

5.3 Lessons Learned

Implementing the project offered the following insights:

- **Impact of Data-Driven Modeling:** Accurate data collection significantly improved the validity of the simulation, highlighting the importance of precise observations. By recording metrics such as arrival rates, service times, and departures in 30-minute intervals, the model was able to replicate real-world conditions effectively. This precision allowed us to uncover patterns like peak demand periods and critical bottlenecks, ensuring that our recommendations were actionable and evidence-based.
- **Behavioral Dynamics in Queues:** Customer abandonment and time-dependent arrival patterns underscored the importance of understanding user behaviors to design better systems. For example, customers were more likely to abandon the queue during peak times when wait times exceeded thresholds, which emphasized the need for dynamic staffing adjustments. Additionally, variations in arrival rates throughout the evening reinforced the necessity of time-segmented analysis to optimize resource allocation effectively.
- **Iterative Refinement:** Simulation development is not a one-time process. It requires continuous adjustments to parameters, logic, and assumptions to align results with observed patterns. During the project, we refined customer abandonment thresholds, adjusted service rate distributions, and recalibrated queue logic based on preliminary outputs. These iterations were essential to improving the models accuracy and ensuring it reflected the complexities of Yogurt Parks operations. The process also highlighted how even small parameter changes could significantly impact outcomes, underscoring the importance of testing and validation.

5.4 Major Takeaways

Key takeaways from the project included:

- **System Bottlenecks:** Identifying critical time periods, such as the 6:30 to 7:00 PM block, where operational strain is highest. During this block, server utilization peaked at 99.91%, and customers experienced the longest average wait time of 2.73 minutes. This time period represents a clear bottleneck where demand exceeds service capacity, leading to congestion and potential customer dissatisfaction. Addressing these bottlenecks is critical for improving overall throughput and efficiency.
- **Resource Utilization:** High server utilization during peak hours signals the need for additional staffing or efficiency improvements. Conversely, periods with lower utilization, such as the 6:00 to 6:30 PM and 7:30 to 8:00 PM blocks, indicate underutilized resources. During these times, servers were utilized at rates of 61.28% and 69.78%, respectively. This suggests an opportunity to redistribute resources or implement dynamic staffing schedules to better match demand patterns and optimize operational performance.
- **Customer-Centric Insights:** Long wait times during busy periods highlight areas for operational improvements to enhance customer satisfaction. Prolonged queues during peak times not only strain server resources but may also result in customer abandonment or reduced likelihood of return visits. Addressing these pain points by reducing wait times through strategies such as adding servers or increasing service rates during peak demand can significantly improve the customer experience and build loyalty.

5.5 Next Steps

If the project were to continue, the following steps could be pursued:

- **Enhanced Data Collection:** Gathering additional data over multiple days, different times of the week, and across seasons to capture a comprehensive understanding of customer demand patterns. This could include analyzing external factors such as weather, holidays, and local events that might influence foot traffic. More granular data on customer demographics and purchasing behaviors could also provide insights for tailored service improvements.
- **Dynamic Staffing Models:** Developing staffing models that align server availability with demand fluctuations throughout the day and week. For example, advanced forecasting techniques could predict peak times more accurately, enabling the implementation of variable staffing schedules. This approach could also consider part-time or on-call staff options to ensure flexibility without overburdening resources.
- **Integration with Real-Time Systems:** Leveraging technologies such as IoT-enabled sensors, customer tracking, and point-of-sale data to monitor queue conditions in real-time. These systems could be integrated with predictive analytics tools to provide dynamic recommendations for staffing adjustments or customer flow management. Additionally, real-time alerts could notify staff of emerging bottlenecks, allowing for immediate corrective actions.
- **Testing Alternative Queuing Mechanisms:** Investigating and piloting different queuing strategies, such as a single-line, multi-server system, which could reduce perceived wait times and improve fairness. Appointment-based systems could be explored to manage high-traffic periods, potentially using online scheduling tools to allow customers to choose specific service windows. Virtual queues with SMS or app-based notifications could also be tested to reduce physical congestion in the store.

5.6 Software Packages Used

The simulation relied on Python packages such as:

- **NumPy:** This package was essential for generating stochastic events and random variables. Specifically, it was used to model:
  - Poisson arrivals, representing customer arrival rates.
  - Exponential service times, simulating the time servers take to assist each customer.
  - Random sampling for customer abandonment thresholds, ensuring realistic variability in behavior.

  NumPy's efficiency in handling large-scale computations ensured that our simulations could run multiple iterations without significant computational delays.
- **Pandas:** This library was utilized for robust data handling, enabling:
  - Storing and organizing simulation results in a tabular format for easy manipulation.
  - Aggregating and summarizing data across multiple simulation runs, such as calculating average wait times and server utilization.
  - Exporting results for further analysis and visualization, ensuring seamless integration with other tools.

  Pandas versatility allowed us to analyze trends and patterns across different time blocks and scenarios efficiently.
- **Matplotlib:** This visualization library played a key role in presenting simulation outputs by:
  - Plotting trends in customer demand over time, highlighting peak and non-peak periods.
  - Visualizing server utilization rates across different time blocks to identify bottlenecks.
  - Creating comparative graphs for wait times, demonstrating the impact of varying arrival and service rates.

  By generating clear and insightful visuals, Matplotlib made it easier to communicate findings effectively to stakeholders.

5.7 Alternative Approaches

Alternative approaches could include:

- **Agent-Based Modeling:** Simulating individual customer and server behaviors provides a detailed and granular analysis of the queuing system. Each agent (customer or server) is modeled with specific characteristics, such as patience levels, service preferences, or varying efficiency rates. This approach allows for the inclusion of nuanced interactions, such as how server fatigue impacts service speed or how customer behavior changes with queue length. However, this method is computationally expensive and requires significant resources to develop and run the simulations.

- **Discrete Event Simulation:** This approach tracks events such as customer arrivals, service initiation, and departures as discrete occurrences over time. Each event is processed in chronological order, allowing for a precise understanding of system dynamics. By explicitly modeling every event, this method captures the flow of the system with high accuracy. However, the increased granularity comes at the cost of computational intensity and longer simulation runtimes, especially for systems with high variability or extended observation periods.
- **Analytical Methods:** Using queuing theory formulas, such as those derived from Markovian models, provides a faster and mathematically elegant way to analyze steady-state performance metrics like average wait times, server utilization, and queue lengths. These methods are particularly useful for simpler systems with predictable behavior. However, they lack the flexibility to accommodate dynamic or time-dependent elements, such as customer abandonment or peak-period variations, which may limit their applicability to real-world scenarios.

5.8 Trade-Offs Between Approaches

- **Granularity vs. Simplicity:** Agent-based models provide detailed, granular insights by simulating the behavior of individual customers and servers. This allows for a more realistic representation of customer interactions, such as varying service preferences or abandonment thresholds. However, this level of detail comes at a cost: the computational complexity increases significantly as the number of agents grows, making these models impractical for large-scale or time-sensitive applications. In contrast, simpler methods, such as aggregated queuing models, trade granularity for speed and ease of implementation, but may overlook critical nuances in customer behavior and server efficiency.
- **Flexibility vs. Speed:** Discrete event simulations track individual events, such as arrivals and departures, providing flexibility to model complex queuing dynamics, including time-dependent arrival rates, customer abandonment, and variable service times. This makes them highly adaptable to real-world scenarios but requires more computational time and resources to execute. On the other hand, closed-form analytical methods, such as those derived from steady-state queuing theory, offer a faster alternative by providing direct mathematical solutions for key metrics like average wait times and server utilization. However, these methods are less flexible and often rely on simplifying assumptions, such as constant arrival and service rates, which may limit their applicability to dynamic or non-linear systems.

5.9 Experimental Results

The simulation results indicated:

- **Demand Patterns:** Peak demand occurred between 6:30 and 7:00 PM, with an average of 29.9 customers. This represents a sharp increase compared to the 18.82 average customers observed in the 6:00 to 6:30 PM block, indicating a rush period that aligns with dinner-time traffic. The increased arrivals during this period place significant pressure on the queuing system, creating potential bottlenecks and reducing the systems capacity to respond to incoming customers efficiently.
- **Wait Times:** The average wait time peaked at 2.73 minutes during the 6:30 to 7:00 PM block, reflecting the strain caused by high customer arrivals. This wait time is more than three times higher than the average of 0.80 minutes observed in the quieter 6:00 to 6:30 PM period. Furthermore, the maximum wait time during the peak period was 6.96 minutes, which highlights a worst-case scenario where customers may experience significant delays. In comparison, during less busy intervals, such as the 7:30 to 8:00 PM block, faster service rates of 1.5 minutes per customer reduced the maximum wait time to 3.57 minutes, despite still-high demand.
- **Server Utilization:** Server utilization reached an extreme high of 99.91% during the 6:30 to 7:00 PM block, indicating that servers were nearly fully occupied and operating at maximum capacity. This high utilization reflects the system's inability to accommodate additional demand efficiently during peak times, risking customer dissatisfaction. By contrast, server utilization dropped to 61.28% during the 6:00 to 6:30 PM block and 69.78% during the 7:30 to 8:00 PM block, suggesting idle server time during these intervals. These disparities point to opportunities for redistributing resources or adjusting staffing schedules to better match fluctuating demand patterns.

5.10 Reflection on Results

The results appear realistic and align with intuition:

- **Physically Realistic:** High demand during peak periods leads to longer wait times and nearly full server utilization. This behavior is consistent with real-world expectations in service systems, where congestion naturally increases

during rush hours. For instance, the 6:30 to 7:00 PM block shows both the highest average wait time (2.73 minutes) and the highest server utilization (99.91%), demonstrating that the model accurately captures the operational strain experienced during these periods.

- **Aligned with Intuition:** The simulation results closely mirror the observed patterns in customer behavior and service operations. Peaks in wait times and server utilization correspond directly to the periods of highest demand, as indicated by the data collected on-site. For example, the significant jump in customer arrivals during the second block (6:30 to 7:00 PM) aligns with the longest wait times and highest utilization rates, validating the model's ability to reflect real-world dynamics.
- **Logical Variations by Time Block:** The model's ability to show reduced wait times and lower server utilization during quieter periods, such as 6:00 to 6:30 PM, further supports its realism. This decrease in congestion during off-peak hours aligns with intuitive expectations that systems with lower input rates experience less strain. For instance, during the first time block, average wait times drop to 0.80 minutes, and server utilization is at 61.28%, both reflecting efficient operations under lower demand.
- **Consistency Across Simulations:** Repeated simulations produced consistent results, reinforcing the validity of the model. Patterns such as the peak average wait time and high server utilization during demand spikes appeared reliably across 50 iterations, demonstrating that the model is robust and capable of generalizing beyond a single run.

5.11 Insights from Experiments

Running the experiments provided key insights:

- **Impact of Peak Demand:** The experiments revealed a highly disproportionate effect of peak demand periods on system performance, significantly increasing wait times and straining available resources. This insight highlights the critical need for effective peak-load management strategies to maintain service quality and operational efficiency.
- **Value of Dynamic Adjustments:** The results demonstrated the effectiveness of adaptive strategies, such as dynamic staffing and real-time resource allocation, in reducing congestion during high-demand periods. These adjustments not only improved customer satisfaction but also optimized resource utilization, showcasing their potential for scalable application in various scenarios.
- **Model Sensitivity:** The findings emphasized the sensitivity of the model to small changes in key parameters, such as service rates and arrival patterns. These variations had a pronounced impact on system performance metrics, underscoring the importance of accurate parameter estimation and robust calibration to ensure reliable predictions and actionable insights.

5.12 Broader Context

This project integrates directly with the classs focus on queuing theory and its applications in real-world scenarios. In the broader IEOR field:

- **Service Operations Management:** Highlights the practical applications of queuing theory in retail and service industries, such as improving the layout of checkout lines, minimizing wait times, and balancing service capacity with customer demand. These insights are crucial for enhancing overall service quality and operational reliability.
- **Operational Efficiency:** Demonstrates the value of simulation for optimizing resource allocation and improving customer satisfaction by identifying bottlenecks, testing various service strategies, and predicting outcomes of changes in system configurations under different scenarios.
- **Customer Behavior Analysis:** Bridges the gap between mathematical modeling and human-centric design considerations by incorporating customer preferences, behavioral trends, and tolerance for waiting, which influence the perceived quality of service and overall satisfaction.

This exploration emphasizes both the theoretical foundations and practical applications of simulation-based queuing analysis in addressing operational challenges, offering a robust framework for decision-making in dynamic and uncertain environments.

6 Conclusion

By completing these tasks, we aim to provide a comprehensive analysis of Yogurt Park's queuing system. This analysis will encompass a detailed examination of customer arrival patterns, service times, and overall system performance under varying conditions. The final report will present a thorough evaluation of key performance metrics, such as average wait times, queue lengths, and system utilization rates. Furthermore, it will include actionable recommendations for improving operational efficiency and enhancing customer satisfaction. These recommendations will be grounded in data-driven insights derived from simulation models, statistical analysis, and empirical observations. Special emphasis will be placed on identifying bottlenecks, optimizing staff allocation, and exploring potential strategies to streamline the customer experience while maintaining service quality.