Restaurant Discrete Event Simulation

Simulating Restaurant Performance and Rating

Vincent Cheong

General

The goal is to generate a fairly accurate assessment of how well a restaurant will perform under given conditions for the time period of a working day. General facts are as listed:

- *The restaurant will contain several major areas: the waiting area, entrance, dining area, bar, and kitchen.*
- *The restaurant will be run by several types of employees: hosts, servers, bartenders, cooks, and dishwashers.*
- *The customers will be grouped and referred to as "parties".*
- *A restaurants success is defined by its performance, rating, and net profit*.
- *The restaurant is simulated to open at 11AM and close at 9PM*.

More specific details include:

- *The trend of the arrival rate is predefined but can be adjusted by the popularity rating.*
- *The physical setting of the restaurant is predefined. Layout, distances between areas, and area capacities are set.*
- *There are three presets available – small, medium, large. The medium is generally 2x the size of the small, and the large is 2x the size of the medium.*
- *The way a party rates an experience is calculated based on 3 factors: the amount of time it takes to be seated after waiting in line, the amount of time it takes for a server to attend to the party and take its order, and the amount of time it takes for an order to arrive.*
- *The features of a party (number of customers, order size, order choice) are all randomly generated based off a list of probabilities.*
- *Walking speed is assumed to average to 85 meters per minute.*
- *Employee hourly salaries are predefined.*
- *Overtime salary is 1.5x the employee's base salary.*
- *At closing time, all customers waiting in line are dismissed. The consequence is that each dismissed customer will rate the restaurant with the lowest rating possible.*
- *The restaurant goes into overtime after closing hours if there is unfinished work. This most often applies to dishwashing but can be the result of customers not being served yet.*

Program Options

- **Restaurant Size –** Pick between small (s), medium (m), large (l).
- **Number of Cooks** – Must be greater than 1.
- **Number of Bartenders** – Must be greater than 1.
- **Number of Hosts** – Must be greater than 1.
- **Number of Servers** – Must be greater than 1.
- **Number of Dishwashers** – Must be greater than 1.
- **Popularity –** Default is at 1.0. Can range anywhere between 0.0 to Double.MAX_VALUE.

Simulation Options

1) **Simulate one operational day** – Outputs arrival rate distribution, event log, and relevant results.
2) **Simulate N times and output** – Outputs the mean and standard deviation of the total earnings, average rating, overtime hours, total upkeep cost, and net profit.
3) **Testing N** parties – This feature is mainly to customize the amount of arriving parties to simulate in order to validate a simplified event log. Relevant results are also displayed on finish.

Actors/Entities

- **Cook**
  - There will always be one Chef who leads the kitchen. The rest of "Cooks" are lower waged workers. **Example:** Cook = 3, which would imply there is 1 Chef and 2 Cooks.
  - Salary: $15 per hour or $30 per hour for the Chef.
  - Does not move from the kitchen.
  - Checks the list of food orders and cooks them at a certain rate.
  - Puts completed orders back into a list so a server can bring them to a table.
- **Bartender**
  - Salary: $15 per hour.
  - Does not move from bar.
  - Checks the list of drink orders and prepares them at a certain rate.
  - Puts completed orders back into a list so a server can bring them to a table.
- **Host**
  - Salary: $15 per hour.
  - Moves from the entrance to the dining area or the bar and back.
  - Checks the line for the dining area and the bar area. If there is available seating, leads the party to their table and returns to their post.
  - Handles the register by moving to the party table when payment is ready. Goes back to register to sort the check out and returns to the party table with the receipt.
- **Server**
  - Salary: $12 per hour.
  - Moves between the dining/bar area and the kitchen.
  - Takes orders and delivers them to the kitchen/bar.
  - Takes completed orders from the kitchen/bar to the party table.
  - Cleans tables that are marked as unclean.
  - Takes dishes from an unclean table to the kitchen where they are added to the dishwashing queue.
- **Dishwasher**
  - Salary: $8 per hour.
  - Does not move from the kitchen.
  - Checks dishwashing queue and cleans dishes when applicable.

- **Party**
    - Upon arrival, waits in a chosen line (dining or bar line).
    - When lead to a table, sits until a server takes its order.
    - Once order is given to a server, waits until food arrives.
    - When food arrives, eats food.
    - When food is finished, waits for host to give check.
    - Pays check, rates the experience, and leaves.
    - Rating is based on: time taken to be seated, time taken for server to take order, and time taken for order to arrive.

Modeling Party Arrival Rate

The time between arrivals of parties is generated based on an ***exponential*** distribution. However, rather than having the exponential distribution rate remain constant, I included a variation in the rate based on the expected peak hours of 1pm – 2pm and 6pm – 7pm. This can be generally confirmed by examining the amount of arrivals per hour that is displayed in simulation options 1 and 3.

$$Rate = \begin{cases} \frac{1}{4} & 11am - 12pm \\ \frac{1}{3} & 12pm - 1pm \\ \frac{1}{2} & 1pm - 2pm \\ \frac{1}{3} & 2pm - 3pm \\ \frac{1}{4} & 3pm - 4pm \\ \frac{1}{4} & 4pm - 5pm \\ \frac{1}{3} & 5pm - 6pm \\ \frac{1}{2} & 6pm - 7pm \\ \frac{1}{3} & 7pm - 8pm \\ \frac{1}{2} & 8pm - 9pm \end{cases}$$

Additionally, the rate can be scaled by a popularity modifier: $ActualRate = Popularity * Rate$.

Popularity

The default popularity is set to 1.0. Too much popularity can be a curse - lines become too long and the staff cannot keep up with the pace. If there is a significant line left during closing hours, the restaurant rating will also take a large hit. This can be remedied by hiring more staff or renting a larger venue. Conversely, if popularity is too low, staff will remain idle and profit margins will be low or negative.

Operational Hours

Operational hours of the restaurant run from 11am to 9pm – after which anyone left in the waiting lines is removed. The system then continues to run under overtime pay until the event queue is empty and all entities have finished executing.

Generating a Party

Two factors must be generated when creating a new Party – the party size and the chosen seating area.

The size of a party is determined based off the probabilities below:

Party Size	Probability
1	0.05
2	0.2
3	0.25
4	0.3
5	0.05
6	0.05
7	0.05
8	0.05

The chosen seating area has a **70% chance to be the dining area** and a **30% chance to be the bar area**.

Generating an Order

The first step to generating an order is to generate the amount of items that guest wishes to order. This is modeled based off of the probabilities shown below:

	Number	Probability
Appetizer	0	0.4
	1	0.5
	2	0.1
Entrees	0	0.05
	1	0.9
	2	0.05
Drinks	1	0.6
	2	0.3
	3	0.1
Dessert	0	0.4
	1	0.5
	2	0.1

The second step is to generate the choice of item within that category. Each item is also tagged with a price, time required to cook, and time required to consume. This allows us to further estimate how long the order would take to be prepared by the cooks and bartenders. There is additional variation added to these estimates so that they aren't just flat values (more on this later).

	Item Name	Price	Cook Time	Consumption Time	Probability
Appetizers	**Chicken Wings**	3.99	1	5	0.1
	Eggroll	1.99	0.3	3	0.1
	Garlic Bread	1.99	0.5	3	0.1
	Nachos	1.99	0.2	5	0.1
	Onion Rings	2.99	1	5	0.2
	Salad	1.99	0.5	5	0.2
	Fries	1.99	0.3	3	0.2
Entrees	**Sandwich**	7.99	2	10	0.2
	Pizza	9.99	8	15	0.25
	Burger	5.99	2	5	0.2
	Steak	12.99	8	10	0.15
	Ribs	14.99	10	10	0.1
	Tacos	6.99	3	5	0.1
Drinks	**Water**	0	0	1	0.4
	Tea	1.99	0.2	2	0.05
	Soda	1.99	0.2	2	0.1
	Juice	2.99	0.3	1	0.05
	Milk	1.99	0.2	1	0.05
	Coffee	1.99	0.2	1	0.1
	Beer	4.99	0.2	4	0.2
	Wine	9.99	0.5	4	0.05
Dessert	**Chocolate Cake**	1.99	0.2	1	0.2
	Cheesecake	3.99	0.2	1	0.3
	Brownie	0.99	0.2	0.5	0.1
	Ice Cream	1.99	0.4	1	0.3
	Cookie	0.99	0.1	0.5	0.1

Variation in Execution Time

There are multiple occasions where a set action is to be executed in a certain time frame. This list includes:

- **Travel Time –** Affected by distance and rate of walking (85 meters per minute).
- **Taking Order Time** – Affected by the number of items in the order.
- **Cooking Time** – Affected by the number of cooks, number of dishes and type of each dish.
- **Preparing Drink Time –** Affected by the number of bartenders, number of drinks and type of each drink.
- **Cleaning Table Time** – Affected by the number of items in the order.
- **Handling Payment Time –** A set constant (1 minute) to handle the register.
- **Dishwashing Time –** Affected by the number of dishwashers and the number of items to wash.

Each item is also affected by a uniformly distributed multiplier between 0.8 and 1.2 to add an element of randomness to these actions.

Estimating Customer Satisfaction

The general concept of modeling satisfaction was centered around equally weighing the time taken to be seated, time taken for server to take the order, and time taken for the order to arrive. Peak satisfaction is maintained until a certain time threshold where it begins to deteriorate until it reaches 0. The following equations model each element:

TimeTilSeated – How long it takes to transition from waiting in line to being seated. After 15 minutes of waiting in line, satisfaction decays. It hits 0 after 45 minutes.

$$TimeTilSeated = \begin{cases} 1 & for\ 0 \le t \le 15 \\ -\frac{1}{1800}x^2 + \frac{2025}{1800} & for\ 15 < x \le 45 \\ 0 & otherwise \end{cases}$$

TimeTilOrderPlace – How long it takes for a server to take an order after the party has been seated. After 10 minutes of waiting, satisfaction decays. It hits 0 after 30 minutes.

$$TimeTilOrderPlaced = \begin{cases} 1 & for\ 0 \le t \le 10 \\ -\frac{1}{400}x^2 + \frac{1}{20}x + \frac{3}{4} & for\ 10 < t \le 30 \\ 0 & otherwise \end{cases}$$

TimeTilOrderArrives – How long it takes for an order to arrive after the party has placed the order. After 90 minutes of waiting, satisfaction decays. It hits 0 after 150 minutes. Perhaps I was a bit lenient on this factor but it only represents a third of customer satisfaction.

$$TimeTilOrderArrives = \begin{cases} 1 & for\ 0 \le t \le 90 \\ -\frac{1}{3000}(x-90)^2 + 1 & for\ 90 \le t \le 150 \\ 0 & otherwise \end{cases}$$

Lastly, these values are averaged to output a satisfaction score between 0 and 1. This is then multiplied by 5 to get the final "rating" score (similar to a 0-5 star review).

DES Events, Entities, and States

Each actor/entity contains several lists responsible for storing elements of its state. The events transition the actors between states. Finally, after each event cycle, the list of states are checked to potentially trigger new events that should be added to the priority queue. For example, the addition of a new order to the Cook_list combined with a Cook becoming idle would place a CookingEvent into the priority queue.

Event List

1) **PartyArrivalEvent** – Party is added to appropriate wait list (line).
2) **SeatingEvent** – Party is assigned to a table.
3) **TakeOrderEvent** – Server arrives at a table and receives the order.
4) **OrderDeliveryEvent** – Server arrives at the kitchen and delivers the order.
5) **FoodDeliveryEvent** – Cooked food arrives at the assigned table.
6) **DrinkDeliveryEvent** – Prepared drinks arrive at the assigned table.
7) **FinishedEatingEvent** – Party finished eating. Host is signaled to handle payment.
8) **PaymentEvent** – Host arrives at table and handles payment.
9) **CleanTableEvent** – Server approaches table and cleans it. Dirty dishes are produced.
10) **DishesDeliveryEvent** – Server delivers dirty dishes to the kitchen.
11) **CookingEvent** – Cooks complete a food order.
12) **PrepareDrinkEvent** – Bartenders complete a drink order.
13) **DishwashingEvent** – Dishwasher finishes washing a load of dishes.

Entity States – Keeps track of idle actor/entities as well as important objects.

1) **Bartender_idle_list**
2) **Cooks_idle_list**
3) **Dishwasher_idle_list**
4) **Host_idle_list**
5) **Server_idle_list**
6) **Cook_list** – List of food orders that need to be cooked.
7) **Finished_cook_list** – List of cooked orders.
8) **Drink_list** - List of drink orders that need to be prepared.
9) **Finished_drink_list** – List of prepared drinks.
10) **Dirty_table_list** – List of tables that need to be cleaned.
11) **Dishes_list** – List of dishes that need to be washed.

Party States

1) **Dining_wait_list** – Line for the dining area.
2) **Bar_wait_list** – Line for the bar area.
3) **Seated_list** – List of seated parties who haven't had their order taken yet.
4) **Ordered_waiting_list** – List of parties who are waiting for their food/drink orders to arrive.
5) **Eating_list** – List of parties that are eating.
6) **Finished_eating_list** – List of parties that have finished eating and are waiting for a check.