The Queueing Theory with a Cognitive Perspective

Kaiyan Fan (kaiyan@mit.edu)

Amanda Li (amandali@mit.edu)

Abstract

Queues are widely studied and have applications in computer networking, engineering, and designing public facilities. We first studied the Queueing Theory and analysed why the standard Queueing Theory does not model real world queues well. To improve the standard Queueing Theory model, we used Bayesian inference to model human's behavior in queues. We then designed and implemented a Bayesian queue simulator based on our model, and evaluated the model based on data collected from simulations.

1 Introduction

Queueing Theory (Bhat, 2008) mathematically formalizes a queue. In its simplest model, a queue has a service node and a waiting area. The queue is first-in, first-out. At the back, arriving customers join the queue in the waiting area. At the front of the queue, customers get serviced by the service node and leave the queue.

Queues are usually described by Kendall's notations (Kendall, 1953) in the form $A/S/c$. A is the arrival process of the queue. S describes the distribution of the service time at the service node, and c denotes and the number of servers at the service node. Consider an $M/M/1$ queue, as depicted in Figure 1. In this notation, M denotes Markov process. The first M indicates that customers arrive the queue according to a Poisson process (where the rate λ describes the expected time between each customer's arrival). The second M indicates that the amount of time it takes to serve each customer is sampled from an exponential distribution (where rate μ describes the reciprocal of the mean service time). Finally, 1 indicates that there is only 1 server at the service node (Willig, 1999).

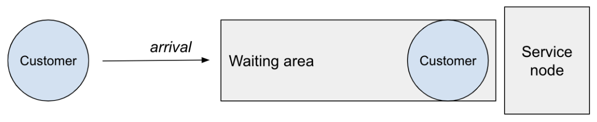

Figure 1: An M/M/1 queue.

One of the most common queues we encounter in our daily lives, cashier queues at supermarkets, can be modeled by $n \times M/G/1$ using Kendall's notations. $n\times$ comes from the fact that there are usually multiple cashier queues in the supermarket. M is the standard Poisson arrival process. G denotes a general distribution for the service time, rather than the specific exponential distribution denoted by M. At the cashier, the service time for each customer can be modeled as following:

$$\textit{service time} = N(\mu, \sigma^2) \times x$$

where the base service speed is sampled from a normal distribution to consider for some randomness in service time, with mean μ and variance σ^2 depending on the skills of the cashier. x is the load, denoting the amount of good purchased by the customer.

The Queueing Theory has wide applications (Willig, 1999; Bhat, 2008). In computer networking, the routers and the Network Interface Card hold internal queues when processing network packets they receive. Network architects can use the Queueing Theory to model the arrival and processing of network packets and analyse their algorithms. In industrial engineering, engineers can use the Queueing Theory to model the arrival and processing of manufactured components in the assembly line and analyse the manufacturing process.

Despite the success of the Queueing Theory, a key aspect of queues is omitted in this mathematical formalism, that is the behavior of humans. In the standard Queueing Theory, customers are treated as simple objects. They join the queue, wait, and get serviced. This is a good model for computer network packets or manufactured components in queues, because they are objects. However, this a bad model for humans in queues. In hospital queues or supermarket queues, for example, people always seek to minimise their waiting time in the queue. Therefore, they might attempt to find the shortest queue, switch to a different queue if they think that's quicker, or quit the queue if they find the queueing time too long. How do humans estimate the waiting time of the queue? How do humans decide to switch to a different queue or quit the queue? Can we find a way to model humans behaviors in the queue? These questions are left unanswered by the Queueing Theory.

In this project, we try to answer these questions by using a Bayesian model for human cognition. We propose that humans behaviors in queues are guided by their mental understanding of the queues. When they join the queue, they have some prior assumptions about the queue, such as how long are they expected to wait. Then, they observe the dynamics of the queue and update their understandings of the queue. They take actions, like switching to a different queue or leaving the queue, based on their posterior understandings of the queue.

We implemented a full queue simulator in Python which takes human behaviors into consideration. In part 2, we explain our mathematical model and the python implementation. In Part 3, we analyse the data collected from the simulation and a small human experiment we conducted. The source code is available at: https://github.com/kaiyanfan/bayesian_queue

2 Methodology

2.1 The Mathematical Model

We use the supermarket cashier queue as an example to describe our mathematical model. As explained previously, supermarket cashier queues can be modeled by $n \times M/G/1$ using Kendall's notations (Kendall, 1953). To model people's behaviors, we define 3 actions that customers may take on top of the standard Queueing Theory:

1. A customer may choose to not join the queue if the expected waiting time exceeds the maximum waiting time that the customer is willing to accept.
2. A customer may choose to quit the current queue and join a different queue, hoping that the latter queue is quicker. The customer abandons the current progress in the queue by doing so.
3. A customer may choose to leave the queue entirely. The difference of this action compared with action 1 is that for action 1, the customer doesn't join the queue at all. For this action, the customer join the queue, then realize the queue is taking longer than expected, and then leave the queue.

These actions are guided by human's understandings of the queues, so we need to quantify how humans estimate their waiting time in each queue. Coming back to the cashier model, the service time for each customer is modeled by:

$$service\ time = N(\mu, \sigma^2) \times x \tag{1}$$

where the service time for one customer is the product of the base service speed and each customer's load x (how many things this customer purchased). Parameters μ and σ^2 are different for each queue, since each queue's cashier may have different levels of skills. For a skilled cashier, the service speed is faster, therefore the value of μ is lower.

The waiting time for a customer in the queue is the sum of the service time of everyone in front. Since the base service speed is sampled from a normal distribution, the expected base service speed is μ, therefore:

$$expected\ service\ time = \mu x \tag{2}$$

$$expected\ waiting\ time = \sum_c \mu x_c \tag{3}$$

where c denotes all customers in the front and x_c denotes the load of customer c. This simple formula allows customers to figure out the waiting time for each queue, and make actions 1-3 based on the waiting time.

This model is not complete yet as customers do not know the value of μ. When they join the queue, they might have some prior understanding of the queue and assumes a prior mean μ. To deduce the real value of μ for each queue, the customer observe how much time does it take for each queue to service one customer. The observed service speed is derive from equation 1:

$$observed\ service\ speed = \frac{t}{x} \tag{4}$$

where t is the service time and x is the load. The observed speed is a sample from $N(\mu, \sigma^2)$.

With the observed data, we perform a Bayesian inference over Normal distribution (with unknown variance) for every observation that a customer makes. The conjugate prior of a Normal distribution with unknown variance (Murphy, 2007) is a Normal-Gamma distribution

$$NG(\mu, \lambda | \mu_0, \kappa_0, \alpha_0, \beta_0) = N(\mu | \mu_0, (\kappa_0 \lambda)^{-1}) Ga(\lambda | \alpha_0, rate = \beta_0) \tag{5}$$

and likelihood

$$p(D|\mu, \lambda) = \frac{1}{(2\pi)^{n/2}} \lambda^{n/2} exp(-\frac{\lambda}{2} \sum_{i=1}^{n} (x_i - \mu)^2) \tag{6}$$

.

The posterior can be derived using Bayes formula

$$p(\mu, \lambda | D) \propto NG(\mu, \lambda | \mu_0, \kappa_0, \alpha_0, \beta_0) p(D|\mu, \lambda) \tag{7}$$

We only concern about the posterior mean μ, so we only compute the posterior mean (omitting many intermediate calculations)

$$\mu_n = \frac{\kappa_0 \mu_0 + n\bar{x}}{\kappa_0 + n} \tag{8}$$

where μ_0 and κ_0 comes from the conjugate prior. Intuitively, κ_0 represents how strong does the customer believe in the prior mean.

As customers continuously update their posterior understanding of the service speed at each queue, they might slowly find out that some queues are faster than other queues, or maybe all queues are slower than the prior assumption. The customer then takes actions 1-3 based on the posterior.

In real life, humans are far from being complete rational and are affected by various psychological effects. In particular, people tend to stick with their default option and avoid making changes. This is known as the "default effect"

(Herrmann et al., 2011). In our queue model, this means that even if switching or quitting the queue is a better option, the customer might stick with their current queue because that's the default option. To model that effect, we add a threshold for taking the actions. Customers would only take actions if the benefit exceeds that threshold. If the benefit is too small, customers would stick to their default option.

2.2 Python Implementation

We implemented the above discrete-time event queueing system that simulates the behavior of human customers in Python. The code is here: https://github.com/kaiyanfan/bayesian_queue. Each run of the simulation contains a fixed number of first-in first-out queues. We cover the implementation of the queue simulator in the following sections in more detail. First, we describe the queues and their configuration setup. We then describe the customers, which we refer to in implementation as "agents". Lastly we describe the simulator and logging system, including an interactive version of the simulator using human input.

Queues The queues used in this simulation are based off of $n \times M/G/1$ queues, as described in Section 2.1. To summarize, in $n \times M/G/1$ queues, each queue services agents first-in first-out at a single service point at the front of the queue. Each arrival for a given queue is dictated by a Poisson process, and the amount of time it takes to service each agent is sampled from a uniform distribution. The queue has no limits on the number of agents that it can hold at a given time.

The queues in our implementation take in constants from configuration files set by the user. The configuration file specifies how many queues are created. Then for each queue, the user supplies a value for μ, the expected service speed of the queue, and σ, the standard deviation of the queue. Additionally, each agent has some fixed load that is sampled from a uniform distribution in range [min_load, max_load]. These limits are also constants set by the user. Then given parameters μ and σ, and agent with load x, the queue calculates the service time for each agent via Equation (1).

As described in Section 2.1, our queues allow agents to exit prior to reaching the service point to allow for more human-like behavior. In our implementation, upon an agent's exit, the agent is removed from its position in the queue. This agent is not considered "serviced" by the queue. This exit action is public to other agents in both this queue and other queues. Each queue also privately tracks the number of successfully serviced jobs, total number of agents joining the queue, and other statistics for analysis purposes.

Agents We construct an abstract Agent class that allows for multiple agent types within the simulation; we focus on behavior of MallCustomer in this description. Each agent has a fixed global ID. At a high level, the agent class goes through the following states: arrival, waiting, and exit. We will describe how the configuration of a given agent determines the behavior at each state.

Agents arrive in the simulation at timestamps generated from a Poisson process. Upon arrival, the agent first selects an initial queue based on their selection criteria. We created a number of different initial selection criteria that agents can be configured to use. For example, agents may select a queue based on the observed length of the queue upon arrival. Agents may also factor in the load of other agents in the queue by selecting the queue with the smallest observed load. Notably, the speed and standard deviation of each queue is unknown to all agents. At the arrival state, each agent has no prior of the queue speeds in this discrete time simulation.

While an agent is waiting in a queue, the agent has visibility into other agents being serviced and the respective service times. After each observed service event, the agent updates its world view and performs Bayesian inference to estimate the speed of each queue as calculated in Equation 8. We optimize our implementation by accumulating the observed data over multiple time steps per agent, so that we can do a faster single inference rather than multiple consecutive inferences per update step.

There are multiple ways that an agent could exit a queue as described in Section 2.1. In our implementation, the configuration file specifies factors that impact queue switching behavior. For example, we highlight factors OBSERVE_TIME and SWITCH_THRESHOLD. OBSERVE_TIME dictates how long an agent must stay waiting in a given queue before deciding to exit the queue. SWITCH_THRESHOLD dictates the expected improvement in wait time required for an agent to switch, which is used to reflect the "default effect" as previously described. An agent can also exit a queue unsuccessfully without switching if the agent has reached a wait limit. This is intended to mimic human patience levels, and is configured for each agent variant.

Simulation and Logging The queue simulation propagates agents through the queues and logs relevant events and timings to disk. An overview of the simulation process is depicted in Figure 2. All agent events, including arrivals, switches, and departures, are tracked in a global list and ordered based on timestamp. This way, the simulation is able to "fast-forward" time until the next discrete event. On each event, the simulator updates the relevant queue state and broadcasts the event status to agents so they can update their world view.

At each discrete event timestamp, the simulator logs the length of each queue and their average wait times. Upon simulation complete, the statistics are reported for each queue: the number of agents arrived, the number of agents served, the number of agents that left the queue prior to being serviced, and the number of agents that joined the queue after having left a prior queue. The simulation is set with a known random seed so that experiments can be replicated.

We also implement an interactive version of the queue simulator that allows humans players to select queues, rather than using coded agents with Bayesian logic. In the inter-

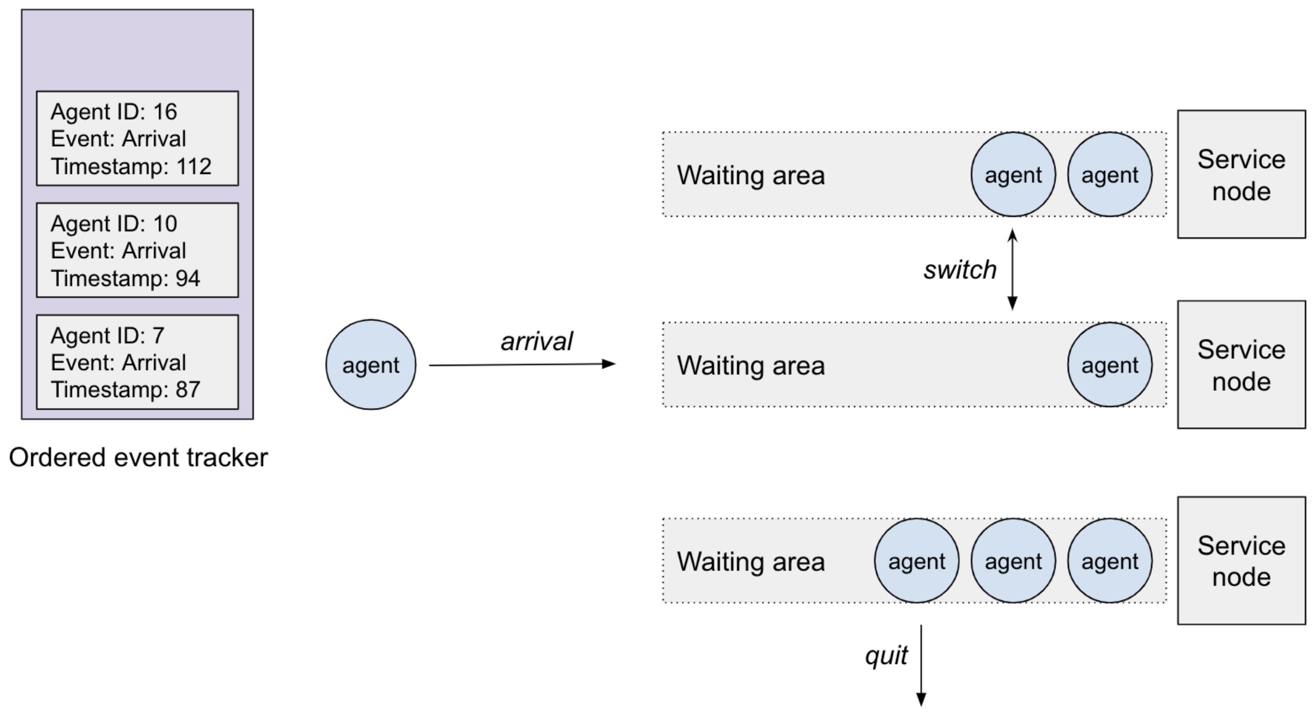

Figure 2: Queue simulation overview.

Queue	N. Served	N. Arrival	N. Left	N. Join
0	677	1784	1099	4
1	1020	1590	609	52
2	1994	599	112	1519

Table 1: Simulation statistics with standard setup

active version, a human player is prompted upon an agent's arrival and asked to select a queue for the agent to join. The simulation displays a visualization of each queue, including other agent's load amount, as well as relevant time stamps. Unlike the standard simulation, the interactive version does not prompt the human player to initiate early exits from a queue, as keeping track of the many agents in the simulation is unreasonable for a single player. Instead, as the simulation proceeds, the human player is expected to observe the queue behavior and modify their selections as they learn about the queue speeds.

3 Simulation Results and Evaluation

3.1 Standard Setup

In our standard setup, the simulation runs 3 queues. Queue 1 is a slow queue, Queue 2 is medium, and Queue 3 is a fast queue. We set the rate of arrival slightly greater than the rate of processing, so the total queue length is expected to grow gradually. The maximum amount of time that a customer is willing to wait for is 300 seconds. The average rate of processing is approximately 20 seconds per customer. We run the simulation for 20000 seconds using these setups.

First, we gather some statistics from each queue in the simulation, including number of customers served (N. Served), number of customers arrived at that queue (N. Arrival), number of customers leaving this queue (N. Left), and number of customers joining this queue from another queue (N. Join). The results are shown in table 3

N. Served: The number of served customers is a sanity check that queues 0, 1, 2 are indeed slow, medium, and fast queue respectively.

N. Arrival: We discover that the number of customers joining each queue is almost inversely proportional to the speed of each queue. A likely reason is that faster queues are usually longer than slower queues because customers move from slower queues to faster queues. New customers joining the queue always join the shortest queue, since their prior doesn't distinguish the speed of each queue. Therefore, new customers tend to join a slower queue and move to a faster queue later.

N.Left / N. Join: Many customers switch from slower queues to join faster queues. This is the result of the Bayesian inference we implemented. This matches people's behavior in queues in the real life. As people observe the queues and update their posterior, they should understand which queues are faster and which queues are slower, therefore they may choose to switch the queue. Notice the total number of customers leaving the queue is greater than the total number of customers joining another queue. This corresponds to customers who quit the queue and never join another queue.

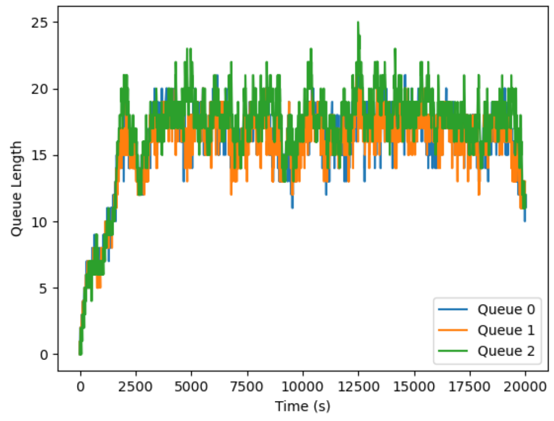

Figure 3: Standard Setup: Queue Length

Next, we investigate how the length of the queue changes over time. We show the results of simulation in figure 3. We find that after the initial rapid growth of the queue length, the total length of the queue stays below a maximum. This corresponds to the fact that customers stop joining the queue and start leaving the queue when the queue is too long. Furthermore, we find that the length of 3 queues tend to be very close to each other, despite that some queues are faster than others. This is because new customers join the shortest queues, balancing the queue length over time. These observations on the queue length roughly matches our experiences with queues in the real life.

We also collect the average wait time of customers joining each queue, shown in figure 4. Note that if a customer joins queue 1 initially and switches to queue 2, the customer's wait time contribute toward queue 1's average wait time.

The average wait time for queue 2 is the lowest. This is expected, since queue 2 is the fastest queue. An interesting observation is that the average wait time of queue 0 and queue 1 are similar, despite queue 1 being faster than queue 0. This can be explained by the fact that customers switch between queues. From the tabulated statistics, queue 0 has the most customers leaving the queue. Therefore, for a customer who stay on queue 0, many customers in the front choose to switch to a different queue, effectively reducing the queue length and wait time. Hence, the average wait time for queue 0 and queue 1 are similar.

Queue	N. Served	N. Arrival	N. Left	N. Join
0	666	2529	3216	1372
1	984	1387	4864	4479
2	2017	65	4076	6051

Table 2: Simulation statistics with lower prior mean

Queue	N. Served	N. Arrival	N. Left	N. Join
0	674	1340	653	5
1	992	1366	368	12
2	1984	1305	68	764

Table 3: Simulation statistics with higher pseudo-observation

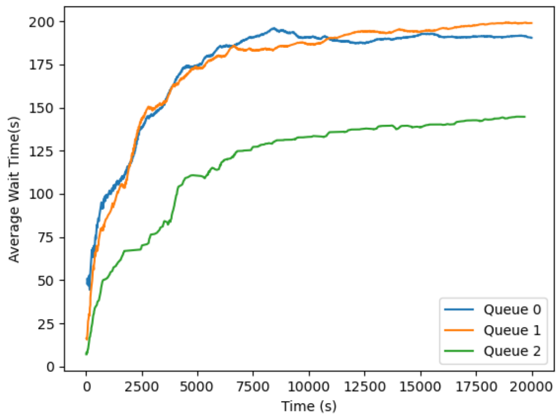

Figure 4: Standard Setup: Average Wait Time

3.2 Variant Setup

To demonstrate how different conditions can affect the queue simulation, we attempt different setups.

Change prior: We change μ_0. Recall from formula 8 that μ_0 is the prior mean and represents people's prior understanding of the speed of the queue. In the standard setup, $\mu_0 = 1$, and the actual μ is 1.5, 1, 0.5 for queues 0, 1, 2 respectively. We run the simulation using a lower prior mean $\mu_0 = 0.5$ and collect the results shown in table 3.1.

We observe that customers switch between queues much more frequently given a lower prior mean. With a lower prior mean, customers would find that the queues progress slower than their expectation, therefore are more likely to seek for a different queue. Hence, there are more switches between queues.

Change pseudo-observation of prior: We change κ_0. Recall from formula 8 that κ_0 is the pseudo-observation and represents how strong do people believe in their prior. In the standard setup, $\kappa_0 = 3$. We run the simulation using $\kappa_0 = 10$

Queue	Served (Bayesian)	Served (Human)	Arrived (Bayesian)	Arrived (Human)
0	2	3	6	5.5
1	4	4	7	7
2	8	8.5	10	11

Table 4: Human vs. Bayesian agent statistics.

and collect the statistics shown in table 3.1.

Compared with the standard setup, there are much fewer customers who choose to switch between queues. This is the expected result of choosing a higher κ_0 value. Since people have a stronger belief in their prior, their posteriors are less sensitive to the observations. Therefore, customers are less convinced that the speed of different queues are different, so fewer customers choose to switch between queues.

3.3 Interactive Setup

In the interactive setup, the simulation runs 3 queues with the same settings as in the standard setup, wherein Queue 0 is slow, Queue 1 is medium, and Queue 2 is fast. The average rate load per agent is approximately 20 seconds. In this experiment, we first the standard simulation for 120 seconds on a set seed. We then have 12 human participants independently play the interactive simulation for 120 seconds on the same seed. We then compare the queue selections made by the Bayesian agents to the selections made by human players to see if our coded agents effectively capture natural human behavior.

In Table 3.3, we compare the number of agents served per queue in standard setup versus the median number of agents served per queue across the 12 interactive setup runs. Note that we take the median rather than the mean due to the small sample size; in the future, we'd ideally like to increase the player count to at least 100 to collect more robust data. We see that the way humans select queues is consistent with the Bayesian agents. A notable observation is that the humans arrive at Queue 2 (fast queue) more often and Queue 0 (slow queue) less often than the Bayesian agents. This may be a result of humans being able to intuitively infer queue speeds faster than the Bayesian agent logic in this relatively short simulation.

3.4 Case Study: Highway Congestion

Although our queue simulator focuses on modelling supermarket cashier queues, it is general enough to be used for modelling many different types of queues. To demonstrate the generality of our model and our implementation, we write a highway congestion simulation.

The highway congestion problem can be modelled by queues. A queue represents a lane, and customers represent the automobiles. The processing speed at each service node represents the flow of each lane, and can be customized to fit specific scenarios. For example, if we want to model some lanes being blocked on the highway, we can set the processing speed of these lanes to 0.

We add a number of rules on top of the standard queue simulator in order to simulate a highway congestion:

1. A car is not allowed to quit the queue.
2. A car is only allowed to switch to an adjacent lane.
3. When switching lanes, a car does not go to the back of the queue.

Due to the page limit, we are not presenting the full results from highway congestion simulation. In short, similar to how customers switch queues to seek the least wait time, automobiles frequently switch lanes to seek the least queueing time. This phenomenon matches our real life experiences in a highway congestion. Our simulation provides a tool for traffic engineers to study the behavior of drivers in a congestion, and can help them to design better traffic systems.

4 Conclusion and Future Works

The Queueing Theory is widely used to study the dynamics of queues. However, the standard Queueing Theory does not consider human behaviors in queues. Our queue simulator uses a Bayesian approach to model people's behavior in queues, not only providing a more accurate simulation, but also contributing insights into how humans behave in queues. We believe it is a better tool to help engineers designing better systems and public facilities. We have only conducted a small scale experiment with human agents to verify our simulation. As future works, we should conduct large scale studies on queues in the real life, and compare the results of real life human behaviors in queues with our simulation.

5 Author Contribution

Kaiyan Fan implemented the Bayesian queue logic and Amanda Li implemented the agent logic, and both authors collaborated on integration into the simulation. Kaiyan Fan implemented the variant experiment and the highway congestion case study. Amanda Li implemented the interactive simulation and conducted the human agent experiments. Both authors wrote half the paper.

References

Bhat, U. N. (2008). *An introduction to queueing theory : modeling and analysis in applications*. Birkhauser.

Herrmann, A., Goldstein, D. G., Stadler, R., Landwehr, J. R., Heitmann, M., Hofstetter, R., & Huber, F. (2011). The effect of default options on choice—evidence from online product configurators. *Journal of Retailing and Consumer Services*, *18*(6), 483-491.

Kendall, D. G. (1953). Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, *24*(3), 338–354.

Murphy, K. (2007, 11). Conjugate bayesian analysis of the gaussian distribution.

Willig, A. (1999). A short introduction to queueing theory.