WASHINGTON UNIVERSITY IN ST. LOUIS

Cashier Simulation

by

Philip I. Thomas

Semester Project

ESE 407: Analysis and Simulation of Discrete Event Systems

May 2013

Contents

1 Problem **1**

1.1 Motivating Example 1

1.2 Background 1

1.3 Problem Statement 2

2 Theory and Methods **3**

2.1 Assumptions 3

2.2 Model 4

3 Implementation and Conclusion **5**

3.1 Solving Method 5

3.2 Interface 5

3.3 Results 6

3.4 Motivating Example Solution 6

4 Source Code **8**

Chapter 1

Problem

1.1 Motivating Example

A client owns a sandwich shop in a major city. They have 8 cash registers in the shop. Customers form a single queue, and the number of active registers varies at the manager's discretion.

The manager has historical data on the customer load, and is able to quantify how many customers are expected every hour and how long it takes a cashier to check out a customer, on average. For instance, during lunch hour they may expect 150 customers between noon and 1PM, and checkout time may take an average of 2 minutes. The manager wishes for time from entering the line to paying for each customer to an average of less than 3 minutes.

The manager seeks a system that allows him to predict how many servers need to be scheduled during a given hour, based on anticipated customer load.

1.2 Background

This problem was motivated by the Integrated Labor-Management System for Taco Bell.[1]. Taco Bell uses software to model their workforce as a system and optimize to minimize labor costs. The system, known as "TACO" (total automation

[1] http://interface.highwire.org/content/28/1/75.full.pdf

of company operations) successfully reduces labor costs by about $5 million per calendar year.

The system integrates three parts to manage the workforce. First, statistical analysis is conducted in a forecasting model using historical restaurant data to predict customer load at the restaurant. Then, a queuing theory model is used to assign the number of cashiers and cooks required for every shift based on acceptable wait time parameters. Finally, the system uses an integer programming model to schedule every employee restaurants.

This total automation of labor is an intriguing problem, and my senior design project, Scheduling Algorithm with Optimization of Employee Satisfaction[2], was motivated by creating a similar employee scheduling system with an objective function that was applicable to small businesses.

With regards to the queuing model used at Taco Bell perceived wait time was a main concern. "After actual waiting time exceeds ve minutes, customer perception of waiting time increases exponentially." In addition, "[. . .] consistently delivering food to customers within an average of three minutes would be a signicant improvement."

Thus, for this project we reasonably conclude that optimal customer wait times are under 3 minutes, and that customers who exceed a wait time of over 5 minutes should be minimized.

1.3 Problem Statement

Design a system for a manager to input anticipate customer load and an average service time, then return the average wait time and other commentary on anticipated lines based on having between one and eight servers, inclusive.

[2]http://students.cec.wustl.edu/ pit1/

Chapter 2

Theory and Methods

2.1 Assumptions

When creating an interface for management, precise details such as arrival process are difficult to glean from project parameters. For this project, multiple assumptions are made to create a usable program for non-technical users.

It is assumed that customer arrival process is a Poisson process. This assumption is reasonable because customers arrivals are likely independent of other factors, thus making a memoryless arrival process desired.

The service time is expected to be exponentially distributed. The exponential was chosen instead of a General distribution becuase it is assumed that checkout time has greater variance, with a higher likelihood of short service than a long service.

It is assumed that the number of servers during a given time period is constant. The software is intended for use during scheduling, and scheduling decisions are often constant over a discrete amount of time. Hence, the simulation operates over a discrete amount of time with constant labor.

Finally, it is assumed that no balking occurs. This assumption is justified because the goal service time chosen by the manager is chosen to minimize balking, so the parameter under which the simulation runs assume that turnaway is not an issue.

2.2 Model

Based on the above assumptions, an M/M/c queue was selected. This multi-server model adequately represents the single-queue system outlined by the client, it allows for a varying number of servers, and analytic solutions are well documented.

Chapter 3

Implementation and Conclusion

3.1 Solving Method

Because the M/M/c queue has been thoroughly solved analytically, an analytical implementation was selected. This yields the fastest computation time for the client. The model formulation was obtained from the book Discrete-Event System Simulation by Jerry Banks.

Simulation methods were rejected because of their run time and applicability to the client. Running an simulation of the queue would require significant computation power and memory, and time complexity would have a high marginal increase with each additional server. If the arrival process or service rate distribution were different, a simulation approach would have been considered. In addition, factors such as queue size limit or balking would require a simulation approach.

3.2 Interface

In order suit the needs of the client, it was decided that three metrics of the model were necessary to display: The average queue length, the average total system time ("checkout time"), and the cashier utilization. The average queue length is important to consider because space must be considered by the manager. In addition, it presents a heuristic for visualization of the model. The average total system time, referred to as the "checkout time", is the most important metric

because literature shows that this value is most closely correlated to perceive service time and balking. Finally, utilization is included because management may wish to consider how often during a shift a cashier may be working. Having low utilization leads to inflated workforce costs.

The interface must make a simple recommendation of the recommended number of cashiers based on target average checkout time. In addition, it must display the above metrics for each possible number of active cashiers provided in the parameters.

3.3 Results

The software was designed to take the three inputs: Average Customers per Hour, Average Cashier Service Time, and Goal Customer Order Time. The software then runs the analysis of the system for each possible number of cashiers. It then displays a recommendation of number of cashiers to schedule. A table is displayed, showing the relevant metrics described in the Interface section for each possible number of cashiers. For technical users, full system details are available.

The software was implemented as a web application to make accessibility easy. With a mobile-compatible interface that requires no additional software, it is easy to use for the client.

Calculations for the metrics are done dynamically client-side. Javascript calculations on the client's computer were selected because the analytic model require minimal computational power.

The recommended number of cashiers is derived as the minimum number of cashiers required so that the average total checkout time is less than or equal to the desired checkout time.

3.4 Motivating Example Solution

In the motivating example it was stated that the manager sought to know how many servers were recommended for an anticipated lunch load of 150 customers

with an average service time of 2 minutes, and a goal customer checkout time of 3 minutes.

The simulation model finds that between one and three, inclusive, servers are infeasible becuase the traffic intensity exceeds one, meaning that customers arrive more quickly than they may be serviced.

The model finds that 4 servers lead to an average queue length of 6.58 customers, with an average checkout time of 5:56. Because this checkout time exceeds the goal checkout time, this number of services.

The model thus selects 5 cashiers, for an average queue length of 1.31, and a n average checkout time of 2:47.

The model further analyzes having 6 cashiers on duty, and the average checkout time of 2:13 shows that there is little marginal benefit to scheduling a sixth employee.

Chapter 4

Source Code

The working model is available at:

$$http://queue.philipithomas.com$$

In addition, the full source code is available at:

$$https://github.com/philipithomas/Queue\_Simulator$$

The full implementation includes multiple files, but the core of the calculations is in the file */custom/queue.js*. The source code of this file is printed below:

```
$("#reset").click(function() {
    $("#output").addClass('hide');
$("#service_input").val('');
$("#arrival_input").val('');
$("#goal_service").val('');

});

$("#simulate").click(function() {
if ( ($("#service_input").val() != "" ) && ($("#arrival_input").val() != "") &&

// There are inputs
```

```
service_input = $("#service_input").val();
arrival_input = $("#arrival_input").val();
goal_service = $("#goal_service").val();
lambda = ( arrival_input / 60 );
mu = 1/service_input;
$("#arrival_display").text(arrival_input);
$("#service_display").text(service_input);
suggestion = 0;
for (c=1;c<=8;c++)
{
// Reset rows
$("#row" + c).removeClass('success error warning');

// Utilization
rho = lambda / c / mu;

// Customers in queue
Pnaught = 0;
// summation term
for (n=0;n<c;n++) {
Pnaught = Pnaught + Math.pow((lambda/mu),n) / factorial(n);
}
// Add second term
Pnaught = Pnaught + Math.pow((lambda/mu),c) * (1/factorial(c)) * (c*mu / (c*mu-l

// final inverse
Pnaught = 1/Pnaught;

// Probabilility L of infinity is greater than c
Plinf = Math.pow(c*rho,c)*Pnaught/(factorial(c)*(1-rho));
// Now we calculate L
// init
L = c * rho;
L = L + Math.pow(c*rho, (c+1)) * Pnaught / (c * factorial(c) * Math.pow(1-rho,2)
L = L + rho * Plinf / (1-rho);
// L complete
```

```
w = L / lambda;

wq = w - 1/mu;
// number in line
Lq = lambda * wq;



// Now display all those numbers
if ( rho >= 1 ) {
$("#row" + c).addClass("error");
$("#rho" + c).text('');
$("#L" + c).text('');
$("#W" + c).text('');
}
else {
$("#rho" + c).html(Math.round( rho * 100) + "%");
$("#L" + c).text(Math.round( Lq * 100)/100);
min = Math.floor( w );
sec = Math.floor(w%1*60);
if (sec < 10 && sec>0) {
// make it pretty
sec = "0"+sec;
}
if (sec == 0) {
sec = "00";
}

$("#dumpbody"+c).html("<div>Servers: " + c + "</div><div>Arrival rate: " + lambd
$("#W" + c).text( min+ ":" +sec );
if ( w > goal_service ) {
$("#row" + c).addClass('warning');
}
else {
$("#row" + c).addClass('success');
```

```
if (suggestion == 0) {
suggestion = c;
}
}
}



}
// Now we print the suggestion
if (suggestion == 0 ) {
$("#rec_outer").addClass('hidden');

}else {
$("#rec_outer").removeClass('hidden');
$("#recommendation").text(suggestion);

}

$("#output").removeClass('hide');
}
});

function factorial(n)
{
if (n <= 1) return 1;
return n*factorial(n-1);
}
```