Streamlining the Bike Share System of Sandy Hills

ORIE 4580

December 2013

Executive Summary

Sandy Hills' newly implemented bike sharing system plans to provide a growing tourist population with an effective and easy way to travel around the city. Sandy Hills undertook a pilot study in order to understand the best way to operate the system. Currently, the bike system provides a low level of service, as many customers arrive to find bike stations empty. We were hired to carry out a detailed study of the system, using data from the pilot study in order to provide recommendations about the best way to operate the system.

Using ProModel software, we modelled the bike share system in order to analyze the proposed situations. Using the data gathered during the pilot study, we fit different distributions to customer arrival times, probabilities of station use, and trip duration times. Using these distributions, we defined the input parameters to our model so it would effectively model the real system. At the request of Mr Reed, we focused our analysis on two different solutions. The solutions we consider either increase the bike supply, offer an incentive for customers to reposition the bikes between stations, or some combination of both. To model the incentive plan, we kept track of the station with the least amount of bikes and analyzed the effect of different proportions of customers who decide to return their bike to the short station. We used certain performance measures, specifically percentage of lost customers, total utilization, and peak hour utilization (between hours of 8:00 AM and 10:00 PM) when comparing alternative solutions.

After analyzing our performance measures for different scenarios, we made some discoveries that have led us to specific recommendations. First, only increasing the bike supply does not provide an effective solution. In order to attain a percentage of lost customers close to zero, Sandy Hills would need to increase its bike supply to over 200 bikes. This results in a very low utilization rate of bikes and a high cost. A better solution is to offer incentives and to increase the bike supply. We found that with 36 total bikes and an incentive program resulting in 20% of

customers repositioning their bikes, only 10% of customers arrived to find zero bikes available. We believe the figure of 10% is a good level for Sandy Hills to strive for.

Problem Description

Sandy Hills currently has a bike share system in place in their downtown area. The system is newly implemented and is therefore still in a trial phase. Data has been collected for analysis to determine the correct logistics for the program. During the trial phase, customers often reported that bike stations were empty when they arrived to pick up a bike, resulting in customer dissatisfaction and decreased use of the bike share program. To try and tackle this problem, the mayor has hired our team to analyze the data and come up with a plan as to how Sandy Hills should allocate the bikes at each station and whether or not they should invest in purchasing additional bikes.

Purchasing additional bikes is the quickest and easiest way to make sure there are more bikes available to customers; however, the cost of purchasing new bikes, in addition to increased maintenance costs can cause a strain on the town budget. The Mayor has said that he will offer incentives to customers if they are willing to take a bike from their current station to the station with the least amount of bikes. These incentives will be posted on a website in real-time so that customers always know which station to return their bike to in order to earn the incentive.

Our job is to use the data from the trial phase to simulate the bike share system, using different amounts of bikes and different percentages of customers taking the incentive to see what plan will be most efficient for the town. In order to limit unnecessary spending, our goal is to find a scenario with a high percentage of bike utilization and a low percentage of lost customers. Through our simulation experiments, we will determine an ideal distribution of bikes that satisfies these goals. Combined with an effective incentive plan, this distribution of bikes will allow Sandy Hills to better provide for their customers and stay within their town budget.

Data Analysis

To effectively model this bike system, we first had to determine the distributions needed to drive our model. We were given the records for 120 days of activity, consisting of 9,326 records. Each record consisted of the following fields: record no., day, pick-up time, pick-up station, drop off time, and drop off station. Although 2,675 entries had blank drop off time and location fields, this does not affect our analysis until we handle trip durations. We began our analysis by modelling customer arrivals.

Our first step was to fit a non-stationary Poisson process to the customer arrival times. In using a non-stationary process, we are able to account for the fact that customer arrival frequency is not uniformly distributed over a 24-hour day. First, we divided a day into 24 distinct one-hour time intervals. We then counted the number of customer arrivals that occurred in each time interval and summed these results over 120 days. Dividing by 120, we were able to find the expected number of customer arrivals per day for each of the 24 time intervals. The expected number of customer arrivals per day is represented by $\lambda(t)$, a non-stationary rate function which takes input parameter t (from 0 to 24 hours) and outputs the expected number of arrivals per day. We find that the peak hour of activity for the Sandy Hills bike share system is from 1:00-2:00 PM, which expects to see 6.33 customer arrivals per hour. The expected number of hourly customer arrivals is less than 2 for every hour between 10 PM and 8 AM. The non-stationary rate function is shown in Figure 1 below, and the complete function is given in the Appendix.

The second step of our data analysis was to determine the fraction of customers that arrive at each of the six stations. Because each record in our data coincided with a customer arrival, we had the pick-up station data for every arrival in our sample. Therefore, we were able to count the number of customers who picked up a bike at station 1 upon arriving, for example. We counted the number of arrivals at each of the six stations in this manner, and divided by the total number of arrivals to find the fraction of arrivals at each station. This fraction represents the expected probability that, given a customer arrival, the customer picks up a bike at a given station (or more simply, the customer arrives at that station). The expected probabilities for each station are shown in Figure 2 below. Because these expectations were estimated based on sample data, we provide 95% confidence intervals with each estimate. The corresponding calculations can be found in the Appendix.

Time	Rate (per hour)
0:00-1:00	1.18
1:00-2:00	1.00
2:00-3:00	1.06
3:00-4:00	1.15
4:00-5:00	1.22
5:00-6:00	1.24
6:00-7:00	1.77
7:00-8:00	1.72
8:00-9:00	4.11
9:00-10:00	3.97
10:00-11:00	3.99
11:00-12:00	6.03
12:00-13:00	6.06
13:00-14:00	6.33
14:00-15:00	5.76
15:00-16:00	5.63
16:00-17:00	4.16
17:00-18:00	3.82
18:00-19:00	4.33
19:00-20:00	3.74
20:00-21:00	3.52
21:00-22:00	3.45
22:00-23:00	1.56
23:00-24:00	1.58

Figure 1. *Customer Arrival Rates*

Station	Percentage
1	15.08%
2	17.26%
3	18.21%
4	16.53%
5	16.59%
6	16.33%

Figure 2

After analyzing the pick-up location data to determine the percentage of arrivals at each station, we analyzed the drop-off location data to determine where customers were going. Just as the frequency of customer arrivals was dependent on the time of day, customers' drop-off locations are dependent on their pick-up locations. Therefore, given that a customer picks up a bike at station i, we estimated the conditional probability that the customer drops the bike off at station j (for i and $j = 1,...,6$). To do this, we first counted the total number of customers who picked a bike up at station 1 and dropped it off at station 2, for example. We then divided this count by the total number of customers who picked a bike up at station 1. The result gave us the fraction of customers who dropped a bike off at station 2, given they picked the bike up at station 1. We repeated this calculation for all pairs *i* and *j*, thus completing our analysis of customers' drop-off location data. The resulting conditional probabilities are shown in Figure 3 below.

	Probability of going to station					
Pick up station	1	2	3	4	5	6
1	13.34%	12.34%	19.46%	24.17%	18.25%	12.44%
2	13.17%	11.39%	20.20%	24.56%	18.33%	12.37%
3	13.02%	13.02%	19.24%	24.30%	16.33%	14.10%
4	13.38%	12.39%	21.36%	22.53%	17.68%	12.66%
5	11.38%	11.19%	21.74%	24.95%	16.70%	14.04%
6	10.98%	12.59%	20.45%	22.23%	18.13%	15.63%

Figure 3. *Conditional Probabilities of going from pick-up station to drop-off station*

Now that we had estimates for the fraction of customers that drop bikes off at station *j* given they started at station *i*, as well as estimates for the fraction of total customers who start at station *i*, we needed to estimate the duration of each bike trip. To do this, we first computed each customer's trip duration by subtracting their pick-up time from their drop-off time. However, to prevent the computation of negative trip durations, we first checked to see whether a customer's drop-off time was larger than his pick-up time. If not, the trip duration was computed as follows: *(24 - start_time) + (end_time).* To account for the customers with no drop-off data, we assumed their trip durations to be 0 and did not account for their trips when computing the average trip duration. The observed maximum and sample average and standard deviation for customers' trip durations are shown in Figure 4 below. Figure 5 shows a plot that fits the normal distribution with sample mean and sample standard deviation to a histogram of customers' trip duration data. Although the fit appears to be fairly close, the QQ-Plot in Figure 6 shows that the normal distribution does not fit the data well over the entire range of observed trip durations.

Rather than modelling total customers' trip durations as normally distributed, we attempted to model trip durations as a condition of customers' pick-up location in hopes of finding a better fit.

Station	Mean (hours)	SD
1	2.0071	0.5383
2	2.0134	0.5554
3	2.0272	0.5408
4	3.3998	0.4132
5	2.0296	0.5324
6	2.0228	0.5328

Figure 4. *Trip Duration Parameters*

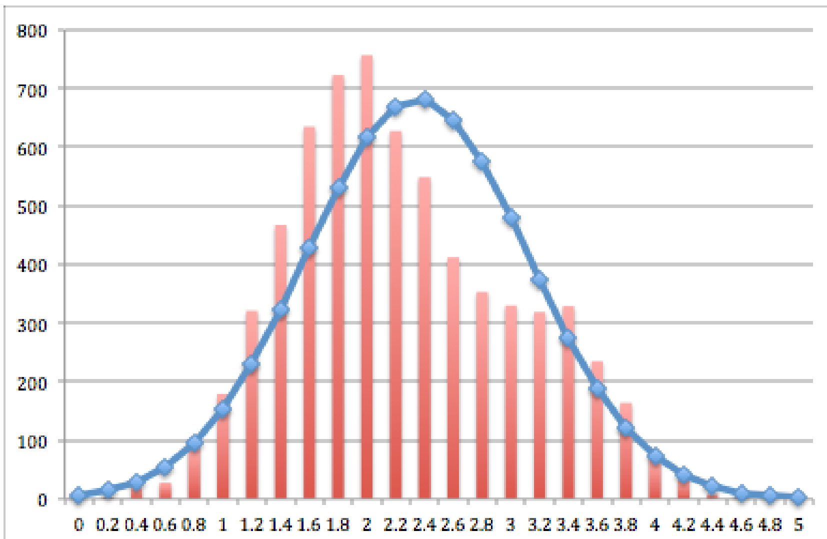

Figure 5. *Histogram of pilot study's trip duration data fitted to normal distribution*

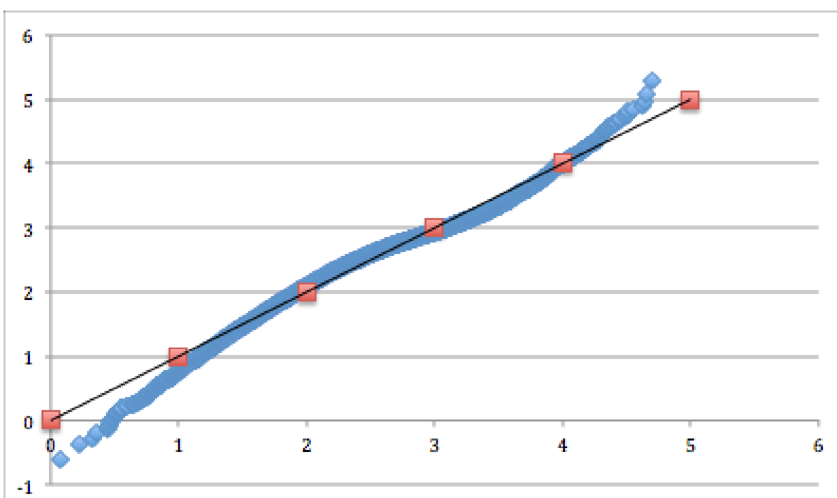

Figure 6. *QQ-Plot of Figure 5*

Modelling Approach and Assumptions

We used ProModel, a discrete event simulation software, to model and simulate Sandy Hills' bike share system. Our simulation model consists of the six bike stations and an arrival queue, which simulates customer arrivals to the system. As we have mentioned, the frequency of customer arrivals changes throughout the day. To model this behavior, we created a location called "Fake_Arrivals", to which customers arrive at the maximum calculated rate of 6.33 customers per hour. Then, for each customer arrival, we calculate the arrival rate for the current hour as a percentage of the maximum rate of 6.33 (See Figure 1 for hourly arrival rates). This percentage represents the probability that a fake arrival becomes a real arrival. For example, a fake arrival occurring between 4:00 and 5:00 PM is accepted with probability *(4.16/6.33) = 65.7%*. This allows us to account for the non-uniformity of customer arrivals. In this way, we are able to recreate the distribution of customer arrivals displayed in Figure 5.

Once customers have been accepted as real arrivals, they become real customers and go to the "Master_Queue". From here they are distributed to one of the six different stations according to the percentages given in Figure 2. When customers arrive at a given station, the model checks if there are any bikes available at this station. If there are none, then the customer leaves and is counted as a lost customer. The model updates the appropriate counters, thus signifying the loss. If, on the other hand, there are bikes available, then the model subtracts one from the variable counting the number of bikes at this station and generates the trip duration for this customer. Trip durations are based on the distributions we calculated for each station and are shown in Figure 4. We also designate a destination location for this customer based on the probabilities we calculated for each station, which are shown in Figure 3. Immediately after the generated trip duration has passed, we increase the counter for the number of bikes at the destination station by one, which simulates the return of the bike to this station. As soon as a bicycle is "dropped off" at a station, it is available for use by another customer. We keep track of bike utilization with a variable that stores the total time bikes are in use. It is increased by a customer's trip duration whenever a customer picks up a bike. At the end of our simulation, we divided this variable's value by the amount of bikes*168 hours (total amount of bike-hours available for use) to arrive at our utilization percentage for a given week.

Next, we simulate the effects of an incentive plan, which would offer vouchers for discounts at local businesses to customers who agree to drop off their bike at the current short station. In order to simulate this effect, we had to modify the way in which customers are routed from their pick-up station to their drop-off station. In the modification, whenever a customer arrives at a station, the model checks which station has the least amount of bikes, and routes a certain percentage of customers directly to this station. For example, a 10% incentive plan means that one out of ten customers are routed to the current short station, regardless of their pick-up station. A higher percentage incentive plan means that better vouchers are offered, thus providing customers with more incentive to accept the offer. A direct routing means that, instead of generating normally distributed trip durations, the model assigns trip durations deterministically. These durations were determined by dividing the distance between two stations by the speed at which a customer travels, which is assumed to be 20 km/hr. The number of bikes at the pick-up station and drop-off station are decremented and incremented in the same manner as is described above. By varying the probability of customers who accept the incentive, we can simulate the effects of different discounts that could be offered.

We repeat multiple simulations, using different numbers of bikes in the system. This allows us to measure how the system would respond if more bikes were purchased and made available for use by the customers. Ultimately, this will allow us to make a recommendation on the optimal number of bikes in the system.

Lastly, we will analyze how the system would respond to an increase in customer demand. Tourism is one of the primary sources of revenue for Sandy Hills, and an improved bike share system may increase the number of tourists to the town. Therefore, the system's ability to handle an increase in customer arrivals without a substantial decrease in customer service is crucial to the success of the town's tourism industry. To model increased customer demand, we increased the rates of customer arrivals proportionally throughout the day. This means that a 20% increase in tourism to Sandy Hills can be modeled by increasing each of the arrival rates by 20%, while keeping all other aspects of the model constant.

Model Verification

To ensure that our computer model correctly represents the Sandy Hills bike share program, we went through a lengthy verification process. The process of verification consists of testing many components, mainly by changing our input parameters and making sure the results exhibit the behavior we expect. Our first step involved verifying the simulation results when extreme input values were used. This included the following:

- Having zero bikes available in the system, resulting in all lost customers.
- Having a very high number of bikes at each station, resulting in no lost customers.
- Setting our incentive proportion to one, resulting in every customer choosing the incentive.
- Setting the incentive proportion to zero, resulting in zero customers choosing the incentive.

The next step involved varying our input parameters, observing the trends, and verifying that these trends exhibit the behavior we expected:

- By changing the total number of bikes, we observed that more bikes available in the system results in fewer customers lost, which is what we expected.
- By increasing demand, we observed that more customer arrivals resulted in a higher percentage of bike utilization and a slightly higher percentage of lost customers, which is what we expected.

Another way in which we verified our model was by making sure that the total number of bikes in the system remained constant over the course of the simulation. This would ensure that there were no bugs that would cause bikes to disappear. To test this, we kept counters for the number of bikes at each station as well as the total number of bikes in transit. By adding these counters together, we were able to verify that the total bikes in the system remained constant.

We also went through multiple simulations and observed the trace (text visualization of simulation through time), as well as printed text descriptions to make sure our model was behaving correctly. We observed the exact behavior we expected, allowing us to conclude that our computer model correctly represents Sandy Hill's real bike share program.

Model Analysis

During testing, we found that after 10 hours of running, changing the initial positioning of the resources no longer produced differences in the results. To ensure this stability, when collecting data for our analysis, we ran 24 hours of warm up before each repetition. In order to fairly compare the simulation data across multiple trials, we used the same stream of random numbers to generate customer arrivals, and therefore each trial had the same number of arrivals. This approach ensured that any differences in the output data were caused by the different input parameters we tested, and not by a fluctuating amount of customer arrivals. In each simulation experiment, we ran the model for 168 hours, or the equivalent of one week, not including the 24 hours of warm up time. We ran 100 replications of each experiment and averaged the results, which allowed us to obtain accurate estimations for each output parameter. Each simulation experiment tested a different set of input parameters.

Level of Resources

One of the possible solutions available to Sandy Hills is to increase the total amount of bikes available to customers, resulting in a higher level of customer service. In order to determine the benefit of purchasing more bikes, we had to consider the trade-off between customer service level and total bike utilization of the system. The graphs below summarize the effect of different resource levels on total utilization, peak hour utilization and percentage of lost customers (See Figures 7-9).

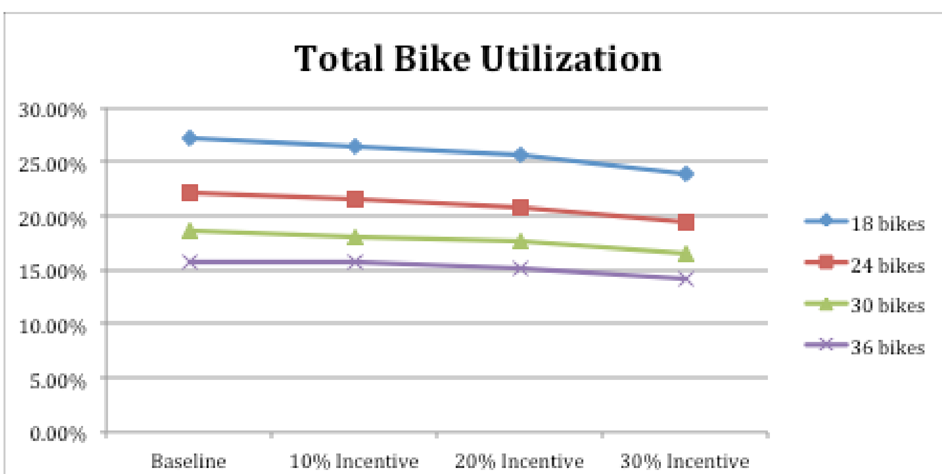

Figure 7. *Total Bike Utilization using different resource levels and incentive plans*

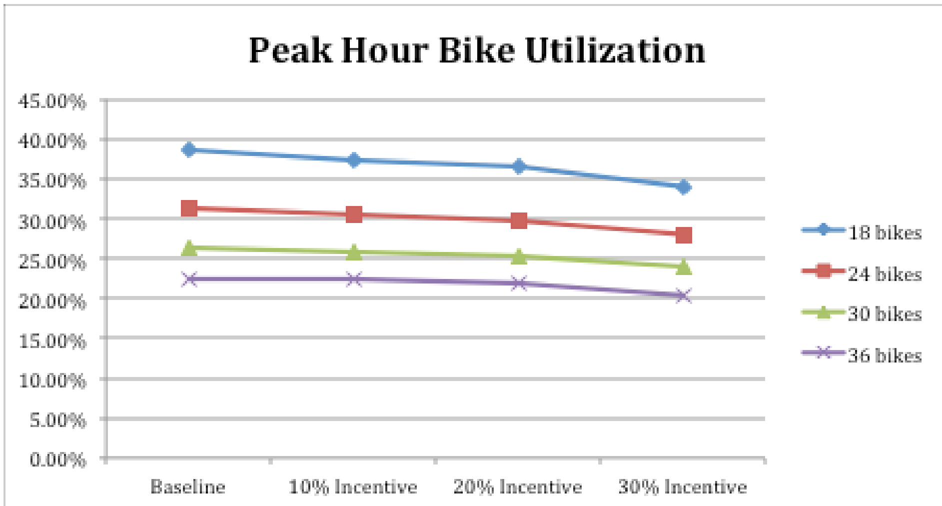

Figure 8. *Peak Hour Bike Utilization using different resource levels and incentive plans*

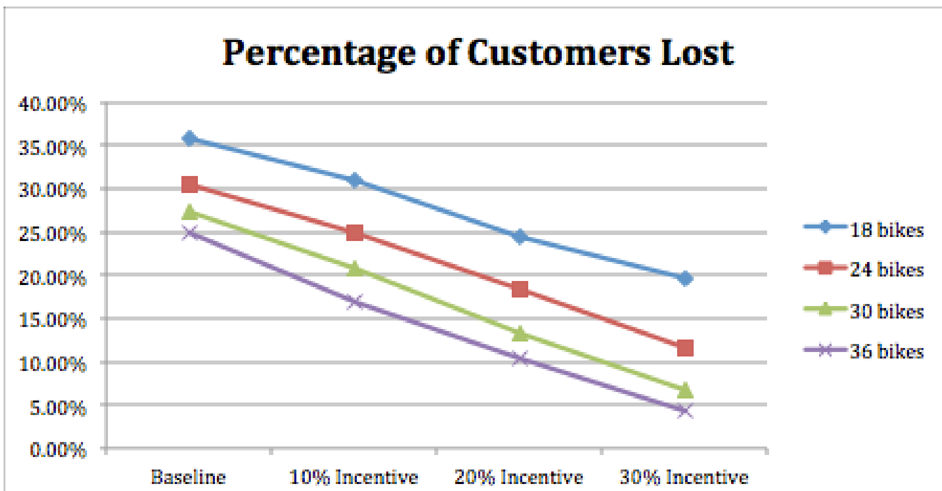

Figure 9. *Percentage of Customers Lost using different resource levels and incentive plans*

As can be seen, just increasing the amount of bikes available in the system does not completely solve the problem, especially concerning the percentage of lost customers. As the resource level increases, the percentage of lost customers begins to decrease at a declining rate, shown in Figure 10 below. The total and peak hour utilization also decreased at a rate similar to that of the percentage of lost customers. By looking at the data, we can see that doubling the resources, results in the percentage of lost customers decreasing by around 10%. Also, we compared the service offered, in terms of percentage of lost customers, between the different resource levels. Our 95% confidence interval for the decrease in percentage of lost customers when bike resources increase from 18 to 36 is (9.23%, 12.74%).

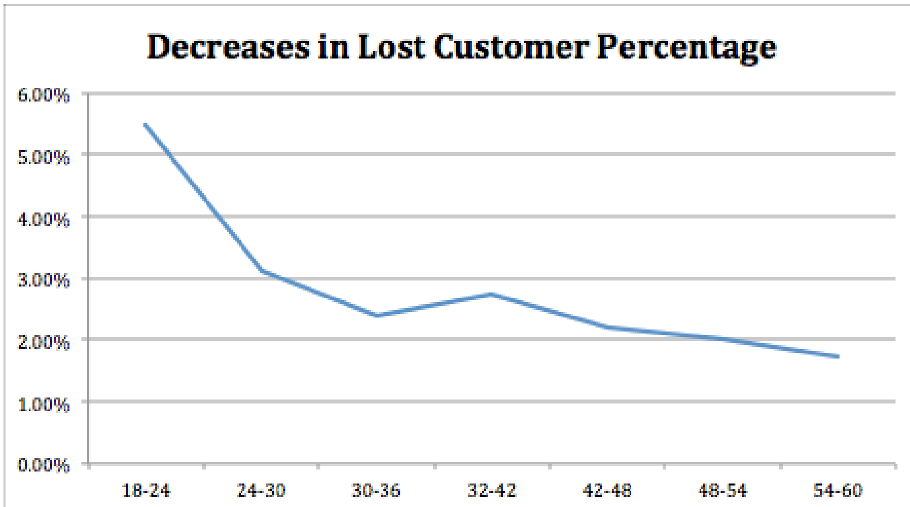

Figure 10. *Percentage Decrease in Lost Customers based on increased bike supply*

As a recommendation, we would advise the mayor to think about the trade-off between the service level shown here, when compared to the utilization and costs of purchasing extra bikes. If this solution is chosen, then Mayor Reed must decide how much money he is willing to spend in order to achieve a target percentage of lost customers.

Repositioning Incentives

As previously mentioned, we modeled the effects of giving customers an incentive to reposition their bike to the station with the least amount of bikes at that time. The incentive plan helps make the system more efficient and decreases the amount of lost customers, as customers who accept the incentive have drastically shorter trip durations. We modeled such a plan and collected simulation results for 10%, 20% and 30% rates of customer acceptance of the incentive plan. The results of the incentive plan simulations were in accordance with our expectations. Increasing the incentives resulted in significant decreases in the number of lost customers. Figure 11 below shows our 95% confidence intervals for the change in percentage of lost customers as a result of different incentive proportions when compared to the baseline.

Incentive	10%	20%	30%
Lower CI	4.44%	12.22%	18.87%
Upper CI	8.36%	15.67%	22.18%

Figure 11. Lost customer decrease 95% confidence intervals

We also observed slight decreases in both total and peak utilization of the bicycles. These decreases in utilization can be explained by the fact that when a customer accepts the relocation incentive, our model deterministically generates a trip duration based on the shortest path to the short station. Because of this, customers who accepted the incentive had an average trip duration of 7.7 minutes, compared to normal customer trip durations, which had an average of 2.35 hours, or 141 minutes. Even so, the decreases in bike utilization are very small compared to the decreases in lost customers, and therefore the primary cost of implementing an incentive plan would be the cost of acquiring discount vouchers from local merchants. Figures 7-9 above show the changes in total utilization, peak hour utilization and lost customer figures based on increases in the percentage of customers who accept the reposition incentive.

As can be seen, the decreases in lost customers resulting from the repositioning incentives have a clearly defined linear trend. We could imagine that given enough incentive, the percentage of lost customers would drop to an extremely small level, especially for the system where there are 36 bicycles available. However, even 30% is already a relatively high number for the percentage of customers who would realistically take the repositioning option. The cost of acquiring vouchers with large enough discounts to incentivize more than 30% of customers would begin to outweigh the benefits of increased customer service. Furthermore, there is likely some maximum percentage of customers who would realistically accept the incentive, because many customers arrive to a pick-up station with a pre-determined drop-off station in mind. Because of this, we only test for the 10%, 20%, and 30% values, and therefore any further reduction in lost customers would have to be reached through the addition of bicycles to the system.

Service Response to Increased Demand

The town of Sandy Hills is becoming increasingly popular as a tourist destination. In order to account for this, we wanted to make sure our recommended resource levels would be able to cope with future increases in demand. We assumed that potential growth levels in demand would be 5% to 20% in the upcoming years. We looked at our system comprised of 18 bikes, with no incentives offered to test this increased demand and its effect. Figure 12 below highlights our results.

18 Bikes				
Demand Increase	0%	5%	10%	20%
Customers Lost	35.86%	36.31%	36.33%	37.72%
Total Usage	27.27%	28.46%	29.83%	31.79%
Peak Usage	38.51%	40.02%	41.86%	44.44%

Figure 12. Effects of increased demand

The increased demand led to an increase in total bike usage as well as our peak time bike usage. This behaviour is to be expected, and does not factor heavily into our analysing response to increased demand. More importantly, the customers lost only increased by 1.86% when the demand increased by 20%. We were able to obtain a 95% confidence interval for the change in percentage of lost customers between the base case and a 20% increase in demand. We are 95% confident that this change in percentage of lost customers is between 0.5% and 3.2%. This is not a very large change, so we are confident that our bike system will be able to cope with increased levels of demand. Also, extrapolating from our findings, we can be reasonably sure that both resource levels, 30 and 36 bikes, will be more than enough to cope with a future increase in tourists.

Conclusions

We have modeled the bike share system and analyzed the simulation data to evaluate the effects of potential changes that could be made to improve the system's performance. We have developed the following suggestions based on Mayor Reed's goal of achieving a high level of customer service while simultaneously minimizing the costs of purchasing new resources and implementing an incentive plan. We believe that the best results are achieved by implementing a bike share system with 36 active bicycles and an incentive program that could capture response of 20% or more of incoming customers. To achieve this, an initial investment would be required to purchase 18 additional bicycles. We believe that the cost of this initial investment is outweighed by the high level of customer service that can be expected as a result of using 36 total bikes. In addition, the investment provides the Sandy Hills bike share system with adequate resources to withstand potential increases in demand resulting from increased tourism to the town. This is not a large investment and would provide the necessary resources to produce good results in the system, even with the potential increases in demand that would come from increased tourism to Sandy Hills.

Appendix

Arrival rates for different hours of the day:

Bin		Cumulativ e	Quantit y	λ(t)
0:00-1:00	1	140	140	1.18
1:00-2:00	2	259	119	1.00
2:00-3:00	3	385	126	1.06
3:00-4:00	4	522	137	1.15
4:00-5:00	5	667	145	1.22
5:00-6:00	6	814	147	1.24
6:00-7:00	7	1025	211	1.77
7:00-8:00	8	1230	205	1.72
8:00-9:00	9	1719	489	4.11
9:00-10:00	10	2191	472	3.97
10:00-11:00	11	2666	475	3.99

11:00-12:00	12	3383	717	6.03
12:00-13:00	13	4104	721	6.06
13:00-14:00	14	4857	753	6.33
14:00-15:00	15	5543	686	5.76
15:00-16:00	16	6213	670	5.63
16:00-17:00	17	6708	495	4.16
17:00-18:00	18	7162	454	3.82
18:00-19:00	19	7677	515	4.33
19:00-20:00	20	8122	445	3.74
20:00-21:00	21	8541	419	3.52
21:00-22:00	22	8952	411	3.45
22:00-23:00	23	9138	186	1.56
23:00-24:00	24	9326	188	1.58

Probability of going to each station:

	Probability of going to station					
Pick up station	1	2	3	4	5	6
1	13.34%	12.34%	19.46%	24.17%	18.25%	12.44%
2	13.17%	11.39%	20.20%	24.56%	18.33%	12.37%
3	13.02%	13.02%	19.24%	24.30%	16.33%	14.10%
4	13.38%	12.39%	21.36%	22.53%	17.68%	12.66%
5	11.38%	11.19%	21.74%	24.95%	16.70%	14.04%
6	10.98%	12.59%	20.45%	22.23%	18.13%	15.63%

Average trip durations for each starting station:

Station	**Mean**	**SD**
1	2.0071	0.5383
2	2.0134	0.5554
3	2.0272	0.5408
4	3.3998	0.4132
5	2.0296	0.5324
6	2.0228	0.5328

Percentage of customers that start at each station:

Station	1	2	3	4	5	6
Average	15.08%	17.30%	18.23%	16.58%	16.64%	16.39%
CI Min	14.47%	16.62%	17.55%	15.90%	15.95%	15.70%

CI Max	15.69%	17.91%	18.86%	17.17%	17.22%	16.96%

Confidence Intervals for statistics gathered from pro model:

% Lost Customers

		average	CI low	CI high
18 bikes	baseline	35.86%	35.09%	36.63%
	10% incentive	30.93%	30.23%	31.63%
	20% incentive	24.44%	23.84%	25.04%
	30% incentive	19.67%	19.07%	20.26%
24 bikes	baseline	30.39%	29.53%	31.25%
	10% incentive	24.78%	23.87%	25.69%
	20% incentive	18.28%	17.35%	19.20%
	30% incentive	11.64%	11.00%	12.28%
30 bikes	baseline	27.28%	26.29%	28.27%
	10% incentive	20.88%	19.92%	21.84%
	20% incentive	13.34%	12.53%	14.14%
	30% incentive	6.76%	6.07%	7.44%
36 bikes	baseline	24.88%	23.88%	25.87%
	10% incentive	16.80%	15.96%	17.64%
	20% incentive	10.26%	9.36%	11.16%
	30% incentive	4.24%	3.61%	4.87%

Total Utilization

		average	CI low	CI high
18 bikes	baseline	0.2727	0.2693	0.3663
	10% incentive	0.2644	0.2614	0.2673
	20% incentive	0.2565	0.2540	0.2590
	30% incentive	0.2385	0.2365	0.2406
24 bikes	baseline	0.2214	0.2182	0.2245
	10% incentive	0.2156	0.2128	0.2183
	20% incentive	0.2082	0.2057	0.2107
	30% incentive	0.1945	0.1925	0.1964
30 bikes	baseline	0.1861	0.1833	0.1890
	10% incentive	0.1814	0.1791	0.1838
	20% incentive	0.1767	0.1747	0.1788
	30% incentive	0.1652	0.1637	0.1667
36 bikes	baseline	0.1582	0.1560	0.1603
	10% incentive	0.1567	0.1548	0.1586
	20% incentive	0.1510	0.1493	0.1528
	30% incentive	0.1413	0.1400	0.1427

Peak Utiliziation

		average	CI low	CI high
18 bikes	baseline	0.3851	0.3800	0.3902
	10% incentive	0.3745	0.3703	0.3787
	20% incentive	0.3642	0.3605	0.3678
	30% incentive	0.3402	0.3369	0.3435
24 bikes	baseline	0.3141	0.3091	0.3190
	10% incentive	0.3065	0.3023	0.3106
	20% incentive	0.2975	0.2939	0.3012

	30% incentive	0.2804	0.2772	0.2835
30 bikes	baseline	0.2644	0.2603	0.2686
	10% incentive	0.2580	0.2546	0.2615
	20% incentive	0.2532	0.2501	0.2563
	30% incentive	0.2395	0.2371	0.2418
36 bikes	baseline	0.2255	0.2222	0.2288
	10% incentive	0.2249	0.2662	0.2735
	20% incentive	0.2180	0.2153	0.2207
	30% incentive	0.2039	0.2019	0.2058

ProModel Text Description:

```
********************************************************************************
**
*                                                    *
*                  Formatted Listing of Model:                  *
*           C:\Users\yf265\AppData\Local\Temp\ProjectV2.MOD           *
*                                                    *
********************************************************************************
**

 Time Units:                Minutes
 Distance Units:             Meters


********************************************************************************
**
*                        Locations                        *
********************************************************************************
**

 Name        Cap     Units Stats       Rules          Cost
 ------------- -------- ----- ----------- -------------- ------------
 Station_1    INF      1    Time Series Oldest, ,
 Station_4    INF      1    Time Series Oldest, ,
 Station_6    INF      1    Time Series Oldest, ,
 Station_3    INF      1    Time Series Oldest, ,
```

Station_2 INF 1 Time Series Oldest, ,
Station_5 INF 1 Time Series Oldest, ,
Corner_1 1 1 Time Series Oldest, ,
Corner_2 1 1 Time Series Oldest, ,
Corner_3 1 1 Time Series Oldest, ,
Corner_4 1 1 Time Series Oldest, ,
Corner_5 1 1 Time Series Oldest, ,
Corner_6 1 1 Time Series Oldest, ,
Master_Queue INFINITE 1 Time Series Oldest, FIFO,
Fake_Arrivals 1 1 Time Series Oldest, ,

```
****************************************************************************
**
*                        Entities                        *
****************************************************************************
**

 Name      Speed (mpm)  Stats      Cost
 ---------- ------------ ----------- ------------
 Customer   50          Time Series


****************************************************************************
**
*                        Processing                        *
****************************************************************************
**

                  Process              Routing

Entity   Location     Operation          Blk  Output   Destination  Rule                 Move Logic
-------- ------------- ------------------   ---- -------- ------------ ------------------------  ------------
Customer Master_Queue  Total_Bikes = S1_Bikes + S2_Bikes + S3_Bikes + S4_Bikes +
S5_Bikes + S6_Bikes + Moving_Bikes
                                  1    Customer Station_1    0.150800 1
                                       Customer Station_2    0.172600
                                       Customer Station_3    0.182100
                                       Customer Station_4    0.165300
                                       Customer Station_5    0.165900
```

```
                              Customer Station_6    0.163300
Customer Fake_Arrivals #Call method defining non-stationary Poisson arrivals
          is_real_arrival = accept_arrival( arrival_time )
                         1    Customer Master_Queue IF is_real_arrival = 1, 1
                              Customer EXIT        IF is_real_arrival = 0
Customer Station_1     #Check if there are bikes in the station and if not,
          #mark as lost customer
          IF( S1_Bikes < 1 ) THEN
          {
               INC S1_Lost
               INC Tot_Lost
          }

          ELSE IF ( S1_Bikes >= 1 )THEN
          {
          #If the customer is willing to take the incentive, then
          #have him move a bike to the short station
          IF(Take_Incentive()=1)THEN
          {REAL MinS = Find_Min_Station()

          IF(MinS=1)THEN
               {INC S1_Use
               INC Tot_Use
               INC Inc_Use
               }
          ELSE IF(MinS=2)THEN
               {INC S1_Use
               INC Tot_Use
               INC Inc_Use
               Dec S1_Bikes
               INC Moving_Bikes
               WAIT 0.15 HR
               Bike_Use = Bike_Use + 0.15
               IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.15
               DEC Moving_Bikes
               INC S2_Bikes
               }
          ELSE IF(MinS=3)THEN
               {INC S1_Use
               INC Tot_Use
```

```
            INC Inc_Use
            Dec S1_Bikes
            INC Moving_Bikes
            WAIT 0.15 HR
            Bike_Use = Bike_Use + 0.15
            IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.15
            DEC Moving_Bikes
            INC S3_Bikes
            }
    ELSE IF(MinS=4)THEN
            {INC S1_Use
            INC Tot_Use
            INC Inc_Use
            Dec S1_Bikes
            INC Moving_Bikes
            WAIT 0.2 HR
            Bike_Use = Bike_Use + 0.2
            IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.2
            DEC Moving_Bikes
            INC S4_Bikes
            }
    ELSE IF(MinS=5)THEN
            {INC S1_Use
            INC Tot_Use
            INC Inc_Use
            Dec S1_Bikes
            INC Moving_Bikes
            WAIT 0.3 HR
            Bike_Use = Bike_Use + 0.3
            IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.3
            DEC Moving_Bikes
            INC S5_Bikes
            }
    ELSE IF(MinS=6)THEN
            {INC S1_Use
            INC Tot_Use
            INC Inc_Use
            Dec S1_Bikes
            INC Moving_Bikes
            WAIT 0.3 HR
```

```
		Bike_Use = Bike_Use + 0.3
		IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.3
		DEC Moving_Bikes
		INC S6_Bikes
		}
}

#If the customer doesn't take the incentive, then generate
#his travel time and update the counters to match the
#transfer of the bike
ELSE{
DEC S1_Bikes
INC Moving_Bikes
INC S1_Use
INC Tot_Use
REAL Tm = N(2.0071, 0.5383)
WAIT Tm HR
DEC Moving_Bikes
Bike_Use = Bike_Use + Tm
IF (is_peak = 1)THEN Peak_Use = Peak_Use + Tm

REAL rn
rn = U(0.5,0.5)
IF( rn < 0.1334 ) THEN
{
		INC S1_Bikes
}
ELSE IF( rn < 0.2568 ) THEN
{
		INC S2_Bikes
}
ELSE IF( rn < 0.4514 ) THEN
{
		INC S3_Bikes
}
ELSE IF( rn < 0.6931 ) THEN
{
		INC S4_Bikes
}
ELSE IF( rn < 0.8756 ) THEN
```

```
                {
                        INC S5_Bikes
                }
                ELSE IF( rn <= 1 ) THEN
                {
                        INC S6_Bikes
                }
                }
                }               1    Customer EXIT         FIRST 1
Customer Station_2     IF( S2_Bikes < 1 ) THEN
                {
                        INC S2_Lost
                        INC Tot_Lost
                }

                ELSE IF ( S2_Bikes >= 1 )THEN
                {IF(Take_Incentive()=1)THEN
                {REAL MinS = Find_Min_Station()

                IF(MinS=1)THEN
                        {INC S2_Use
                        INC Tot_Use
                        INC Inc_Use
                        Dec S2_Bikes
                        INC Moving_Bikes
                        WAIT 0.15 HR
                        Bike_Use = Bike_Use + 0.15
                        IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.15
                        DEC Moving_Bikes
                        INC S1_Bikes
                        }
                ELSE IF(MinS=2)THEN
                        {INC S2_Use
                        INC Tot_Use
                        INC Inc_Use
                        }
                ELSE IF(MinS=3)THEN
                        {INC S2_Use
                        INC Tot_Use
                        INC Inc_Use
```

```
            Dec S2_Bikes
            INC Moving_Bikes
            WAIT 0.1 HR
            Bike_Use = Bike_Use + 0.1
            IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.1
            DEC Moving_Bikes
            INC S3_Bikes
            }
    ELSE IF(MinS=4)THEN
            {INC S2_Use
            INC Tot_Use
            INC Inc_Use
            Dec S2_Bikes
            INC Moving_Bikes
            WAIT 0.15 HR
            Bike_Use = Bike_Use + 0.15
            IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.15
            DEC Moving_Bikes
            INC S4_Bikes
            }
    ELSE IF(MinS=5)THEN
            {INC S2_Use
            INC Tot_Use
            INC Inc_Use
            Dec S2_Bikes
            INC Moving_Bikes
            WAIT 0.25 HR
            Bike_Use = Bike_Use + 0.25
            IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.25
            DEC Moving_Bikes
            INC S5_Bikes
            }
    ELSE IF(MinS=6)THEN
            {INC S2_Use
            INC Tot_Use
            INC Inc_Use
            Dec S2_Bikes
            INC Moving_Bikes
            WAIT 0.25 HR
            Bike_Use = Bike_Use + 0.25
```

```
        IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.25
        DEC Moving_Bikes
        INC S6_Bikes
        }
}

ELSE{
DEC S2_Bikes
INC Moving_Bikes
INC S2_Use
INC Tot_Use
REAL Tm = N(2.0134,0.5554)
WAIT Tm HR
DEC Moving_Bikes
Bike_Use = Bike_Use + Tm
IF (is_peak = 1)THEN Peak_Use = Peak_Use + Tm
REAL rn
rn = U(0.5,0.5)
IF( rn < 0.1317 ) THEN
{
        INC S1_Bikes
}
ELSE IF( rn < 0.2456 ) THEN
{
        INC S2_Bikes
}
ELSE IF( rn < 0.4475 ) THEN
{
        INC S3_Bikes
}
ELSE IF( rn < 0.6931 ) THEN
{
        INC S4_Bikes
}
ELSE IF( rn < 0.8763 ) THEN
{
        INC S5_Bikes
}
ELSE IF( rn <= 1 ) THEN
{
```

```
                INC S6_Bikes
        }
        }
        }               1    Customer EXIT        FIRST 1
Customer Station_3    IF( S3_Bikes < 1 ) THEN
        {
                INC S3_Lost
                INC Tot_Lost
        }

        ELSE IF ( S3_Bikes >= 1 )THEN
        {IF(Take_Incentive()=1)THEN
        {REAL MinS = Find_Min_Station()

        IF(MinS=1)THEN
                {INC S3_Use
                INC Tot_Use
                INC Inc_Use
                Dec S3_Bikes
                INC Moving_Bikes
                WAIT 0.15 HR
                Bike_Use = Bike_Use + 0.15
                IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.15
                DEC Moving_Bikes
                INC S1_Bikes
                }
        ELSE IF(MinS=2)THEN
                {INC S3_Use
                INC Tot_Use
                INC Inc_Use
                Dec S3_Bikes
                INC Moving_Bikes
                WAIT 0.1 HR
                Bike_Use = Bike_Use + 0.1
                IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.1
                DEC Moving_Bikes
                INC S2_Bikes
                }
        ELSE IF(MinS=3)THEN
                {INC S3_Use
```

```
            INC Tot_Use
            INC Inc_Use
            }
ELSE IF(MinS=4)THEN
            {INC S3_Use
            INC Tot_Use
            INC Inc_Use
            Dec S3_Bikes
            INC Moving_Bikes
            WAIT 0.05 HR
            Bike_Use = Bike_Use + 0.05
            IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.05
            DEC Moving_Bikes
            INC S4_Bikes
            }
ELSE IF(MinS=5)THEN
            {INC S3_Use
            INC Tot_Use
            INC Inc_Use
            Dec S3_Bikes
            INC Moving_Bikes
            WAIT 0.15 HR
            Bike_Use = Bike_Use + 0.15
            IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.15
            DEC Moving_Bikes
            INC S5_Bikes
            }
ELSE IF(MinS=6)THEN
            {INC S3_Use
            INC Tot_Use
            INC Inc_Use
            Dec S3_Bikes
            INC Moving_Bikes
            WAIT 0.15 HR
            Bike_Use = Bike_Use + 0.15
            IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.15
            DEC Moving_Bikes
            INC S6_Bikes
            }
}
```

```
ELSE{

DEC S3_Bikes
INC Moving_Bikes
INC S3_Use
INC Tot_Use
REAL Tm = N(2.0272,0.5408)
WAIT Tm HR
DEC Moving_Bikes
Bike_Use = Bike_Use + Tm
IF (is_peak = 1)THEN Peak_Use = Peak_Use + Tm
REAL rn
rn = U(0.5,0.5)
IF( rn < 0.1302 ) THEN
{
	INC S1_Bikes
}
ELSE IF( rn < 0.2604 ) THEN
{
	INC S2_Bikes
}
ELSE IF( rn < 0.4527 ) THEN
{
	INC S3_Bikes
}
ELSE IF( rn < 0.6957 ) THEN
{
	INC S4_Bikes
}
ELSE IF( rn < 0.8590 ) THEN
{
	INC S5_Bikes
}
ELSE IF( rn <= 1 ) THEN
{
	INC S6_Bikes
}
}
}		1	Customer EXIT		FIRST 1
```

```
Customer Station_4     IF( S4_Bikes < 1 ) THEN
                {
                        INC S4_Lost
                        INC Tot_Lost
                }

                ELSE IF ( S4_Bikes >= 1 )THEN
                {IF(Take_Incentive()=1)THEN
                {REAL MinS = Find_Min_Station()

                IF(MinS=1)THEN
                        {INC S4_Use
                        INC Tot_Use
                        INC Inc_Use
                        Dec S4_Bikes
                        INC Moving_Bikes
                        WAIT 0.2 HR
                        Bike_Use = Bike_Use + 0.2
                        IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.2
                        DEC Moving_Bikes
                        INC S1_Bikes
                        }
                ELSE IF(MinS=2)THEN
                        {INC S4_Use
                        INC Tot_Use
                        INC Inc_Use
                        Dec S4_Bikes
                        INC Moving_Bikes
                        WAIT 0.15 HR
                        Bike_Use = Bike_Use + 0.15
                        IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.15
                        DEC Moving_Bikes
                        INC S2_Bikes
                        }
                ELSE IF(MinS=3)THEN
                        {INC S4_Use
                        INC Tot_Use
                        INC Inc_Use
                        Dec S4_Bikes
                        INC Moving_Bikes
```

```
		WAIT 0.05 HR
		Bike_Use = Bike_Use + 0.05
		IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.05
		DEC Moving_Bikes
		INC S3_Bikes
		}
	ELSE IF(MinS=4)THEN
		{INC S4_Use
		INC Tot_Use
		INC Inc_Use
		}
	ELSE IF(MinS=5)THEN
		{INC S4_Use
		INC Tot_Use
		INC Inc_Use
		Dec S4_Bikes
		INC Moving_Bikes
		WAIT 0.2 HR
		Bike_Use = Bike_Use + 0.2
		IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.2
		DEC Moving_Bikes
		INC S5_Bikes
		}
	ELSE IF(MinS=6)THEN
		{INC S4_Use
		INC Tot_Use
		INC Inc_Use
		Dec S4_Bikes
		INC Moving_Bikes
		WAIT 0.2 HR
		Bike_Use = Bike_Use + 0.2
		IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.2
		DEC Moving_Bikes
		INC S6_Bikes
		}
	}

	ELSE{

	DEC S4_Bikes
```

```
                INC Moving_Bikes
                INC S4_Use
                INC Tot_Use
                REAL Tm = N(3.3998,0.4132)
                WAIT Tm HR
                DEC Moving_Bikes
                Bike_Use = Bike_Use + Tm
                IF (is_peak = 1)THEN Peak_Use = Peak_Use + Tm
                REAL rn
                rn = U(0.5,0.5)
                IF( rn < 0.1338 ) THEN
                {
                        INC S1_Bikes
                }
                ELSE IF( rn < 0.2576 ) THEN
                {
                        INC S2_Bikes
                }
                ELSE IF( rn < 0.4713 ) THEN
                {
                        INC S3_Bikes
                }
                ELSE IF( rn < 0.6966 ) THEN
                {
                        INC S4_Bikes
                }
                ELSE IF( rn < 0.8734 ) THEN
                {
                        INC S5_Bikes
                }
                ELSE IF( rn <= 1 ) THEN
                {
                        INC S6_Bikes
                }
                }
                }               1    Customer EXIT        FIRST 1
Customer Station_5    IF( S5_Bikes < 1 ) THEN
                {
                        INC S5_Lost
                        INC Tot_Lost
```

```
}

ELSE IF ( S5_Bikes >= 1 )THEN
{IF(Take_Incentive()=1)THEN
{REAL MinS = Find_Min_Station()

IF(MinS=1)THEN
        {INC S5_Use
        INC Tot_Use
        INC Inc_Use
        Dec S5_Bikes
        INC Moving_Bikes
        WAIT 0.3 HR
        Bike_Use = Bike_Use + 0.3
        IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.3
        DEC Moving_Bikes
        INC S1_Bikes
        }
ELSE IF(MinS=2)THEN
        {INC S5_Use
        INC Tot_Use
        INC Inc_Use
        Dec S5_Bikes
        INC Moving_Bikes
        WAIT 0.2 HR
        Bike_Use = Bike_Use + 0.2
        IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.2
        DEC Moving_Bikes
        INC S2_Bikes
        }
ELSE IF(MinS=3)THEN
        {INC S5_Use
        INC Tot_Use
        INC Inc_Use
        Dec S5_Bikes
        INC Moving_Bikes
        WAIT 0.15 HR
        Bike_Use = Bike_Use + 0.15
        IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.15
        DEC Moving_Bikes
```

```
            INC S3_Bikes
            }
ELSE IF(MinS=4)THEN
            {INC S5_Use
            INC Tot_Use
            INC Inc_Use
            Dec S5_Bikes
            INC Moving_Bikes
            WAIT 0.2 HR
            Bike_Use = Bike_Use + 0.2
            IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.2
            DEC Moving_Bikes
            INC S4_Bikes
            }
ELSE IF(MinS=5)THEN
            {INC S5_Use
            INC Tot_Use
            INC Inc_Use
            }
ELSE IF(MinS=6)THEN
            {INC S5_Use
            INC Tot_Use
            INC Inc_Use
            Dec S5_Bikes
            INC Moving_Bikes
            WAIT 0.1 HR
            Bike_Use = Bike_Use + 0.1
            IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.1
            DEC Moving_Bikes
            INC S6_Bikes
            }
}

ELSE{

DEC S5_Bikes
INC Moving_Bikes
INC S5_Use
INC Tot_Use
REAL Tm = N(2.0296,0.5324)
```

```
                    WAIT Tm HR
                    DEC Moving_Bikes
                    Bike_Use = Bike_Use + Tm
                    IF (is_peak = 1)THEN Peak_Use = Peak_Use + Tm
                    REAL rn
                    rn = U(0.5,0.5)
                    IF( rn < 0.1138 ) THEN
                    {
                         INC S1_Bikes
                    }
                    ELSE IF( rn < 0.2257 ) THEN
                    {
                         INC S2_Bikes
                    }
                    ELSE IF( rn < 0.4431 ) THEN
                    {
                         INC S3_Bikes
                    }
                    ELSE IF( rn < 0.6927 ) THEN
                    {
                         INC S4_Bikes
                    }
                    ELSE IF( rn < 0.8596 ) THEN
                    {
                         INC S5_Bikes
                    }
                    ELSE IF( rn <= 1 ) THEN
                    {
                         INC S6_Bikes
                    }
                    }
                    }          1    Customer EXIT       FIRST 1
Customer Station_6    IF( S6_Bikes < 1 ) THEN
                    {
                         INC S6_Lost
                         INC Tot_Lost
                    }

                    ELSE IF ( S6_Bikes >= 1 )THEN
                    {IF(Take_Incentive()=1)THEN
```

```
{REAL MinS = Find_Min_Station()

IF(MinS=1)THEN
	{INC S6_Use
	INC Tot_Use
	INC Inc_Use
	Dec S6_Bikes
	INC Moving_Bikes
	WAIT 0.3 HR
	Bike_Use = Bike_Use + 0.3
	IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.3
	DEC Moving_Bikes
	INC S1_Bikes
	}
ELSE IF(MinS=2)THEN
	{INC S6_Use
	INC Tot_Use
	INC Inc_Use
	Dec S6_Bikes
	INC Moving_Bikes
	WAIT 0.2 HR
	Bike_Use = Bike_Use + 0.2
	IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.2
	DEC Moving_Bikes
	INC S2_Bikes
	}
ELSE IF(MinS=3)THEN
	{INC S6_Use
	INC Tot_Use
	INC Inc_Use
	Dec S6_Bikes
	INC Moving_Bikes
	WAIT 0.15 HR
	Bike_Use = Bike_Use + 0.15
	IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.15
	DEC Moving_Bikes
	INC S3_Bikes
	}
ELSE IF(MinS=4)THEN
	{INC S6_Use
```

```
            INC Tot_Use
            INC Inc_Use
            Dec S6_Bikes
            INC Moving_Bikes
            WAIT 0.2 HR
            Bike_Use = Bike_Use + 0.2
            IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.2
            DEC Moving_Bikes
            INC S4_Bikes
            }
ELSE IF(MinS=5)THEN
            {INC S6_Use
            INC Tot_Use
            INC Inc_Use
            Dec S6_Bikes
            INC Moving_Bikes
            WAIT 0.1 HR
            Bike_Use = Bike_Use + 0.1
            IF (is_peak = 1)THEN Peak_Use = Peak_Use + 0.1
            DEC Moving_Bikes
            INC S5_Bikes
            }
ELSE IF(MinS=6)THEN
            {INC S6_Use
            INC Tot_Use
            INC Inc_Use
            }
}

ELSE{

DEC S6_Bikes
INC Moving_Bikes
INC S6_Use
INC Tot_Use
REAL Tm = N(2.0228,0.5328)
WAIT Tm HR
DEC Moving_Bikes
Bike_Use = Bike_Use + Tm
IF (is_peak = 1)THEN Peak_Use = Peak_Use + Tm
```

```
REAL rn
rn = U(0.5,0.5)
IF( rn < 0.1098 ) THEN
{
        INC S1_Bikes
}
ELSE IF( rn < 0.2357 ) THEN
{
        INC S2_Bikes
}
ELSE IF( rn < 0.4402 ) THEN
{
        INC S3_Bikes
}
ELSE IF( rn < 0.6625 ) THEN
{
        INC S4_Bikes
}
ELSE IF( rn < 0.8438 ) THEN
{
        INC S5_Bikes
}
ELSE IF( rn <= 1 ) THEN
{
        INC S6_Bikes
}
}
}              1    Customer EXIT        FIRST 1
```

```
**************************************************************************
**
*                    Arrivals                        *
**************************************************************************
**

 Entity   Location      Qty Each   First Time Occurrences Frequency    Logic
 -------- ------------- ---------- ---------- ----------- ------------ ------------
 Customer Fake_Arrivals 1          0          INF         e(9.47867,2) arrival_time=CLOCK(MIN)
                                                          is_peak = Calc_Peak(arrival_time)
```

****
**
* Attributes *
****
**

ID	Type	Classification
arrival_time	Real	Entity
is_real_arrival	Integer	Entity
where_to	Integer	Entity
is_peak	Integer	Entity

****
**
* Variables (global) *
****
**

ID	Type	Initial value	Stats
S1_Bikes	Integer	3	Time Series
S2_Bikes	Integer	3	Time Series
S3_Bikes	Integer	3	Time Series
S4_Bikes	Integer	3	Time Series
S5_Bikes	Integer	3	Time Series
S6_Bikes	Integer	3	Time Series
S1_Use	Integer	0	Time Series
S2_Use	Integer	0	Time Series
S3_Use	Integer	0	Time Series
S4_Use	Integer	0	Time Series
S5_Use	Integer	0	Time Series
S6_Use	Integer	0	Time Series
S1_Lost	Integer	0	Time Series
S2_Lost	Integer	0	Time Series
S3_Lost	Integer	0	Time Series
S4_Lost	Integer	0	Time Series

```
 S5_Lost     Integer     0           Time Series
 S6_Lost     Integer     0           Time Series
 Tot_Lost    Integer     0           Time Series
 Tot_Use     Integer     0           Time Series
 Inc_Use     Integer     0           Time Series
 Bike_Use     Real       0           Time Series
 Total_Bikes  Integer     0           Time Series
 Moving_Bikes Integer     0           Time Series
 Peak_Use     Real       0           Time Series


*****************************************************************************
**
*                          Subroutines                          *
*****************************************************************************
**

 ID            Type        Parameter  Type        Logic
 ---------------- ------------ ---------- ------------ ------------------
 Accept_Arrival   Integer     Arr_Time   Real        #Accept arrivals at the appropriate rates for
the time of arrival
                                         REAL arr_time_in_hours
                                         REAL rate_to_use
                                         REAL hour

                                         arr_time_in_hours = arr_time / 60
                                         hour=arr_time_in_hours MOD 24

                                         IF ( hour < 1 ) THEN
                                           rate_to_use = 1.18
                                         ELSE IF ( hour < 2 ) THEN
                                           rate_to_use = 1.00
                                         ELSE IF ( hour < 3 ) THEN
                                           rate_to_use = 1.06
                                         ELSE IF ( hour < 4 ) THEN
                                           rate_to_use = 1.15
                                         ELSE IF ( hour < 5 ) THEN
                                           rate_to_use = 1.22
                                         ELSE IF ( hour < 6 ) THEN
                                           rate_to_use = 1.24
```

```
ELSE IF ( hour < 7 ) THEN
   rate_to_use = 1.77
ELSE IF ( hour < 8 ) THEN
   rate_to_use = 1.72
ELSE IF ( hour < 9 ) THEN
   rate_to_use = 4.11
ELSE IF ( hour < 10 ) THEN
   rate_to_use = 3.97
ELSE IF ( hour < 11 ) THEN
   rate_to_use = 3.99
ELSE IF ( hour < 12 ) THEN
   rate_to_use = 6.03
ELSE IF ( hour < 13 ) THEN
   rate_to_use = 6.06
ELSE IF ( hour < 14 ) THEN
   rate_to_use = 6.33
ELSE IF ( hour < 15 ) THEN
   rate_to_use = 5.76
ELSE IF ( hour < 16 ) THEN
   rate_to_use = 5.63
ELSE IF ( hour < 17 ) THEN
   rate_to_use = 4.16
ELSE IF ( hour < 18 ) THEN
   rate_to_use = 3.82
ELSE IF ( hour < 19 ) THEN
   rate_to_use = 4.33
ELSE IF ( hour < 20 ) THEN
   rate_to_use = 3.74
ELSE IF ( hour < 21 ) THEN
   rate_to_use = 3.52
ELSE IF ( hour < 22 ) THEN
   rate_to_use = 3.45
ELSE IF ( hour < 23 ) THEN
   rate_to_use = 1.56
ELSE IF ( hour < 24 ) THEN
   rate_to_use = 1.58
ELSE
   DISPLAY "rate function is not availale " $
arr_time_in_hours
```

```
REAL rn
rn = U(0.5,0.5)
IF ( 6.33 * rn < rate_to_use ) THEN
   RETURN 1
ELSE
   RETURN 0
```

Find_Min_Station Integer #Calculate the station with the least amount of bikes

```
REAL Min_Station
REAL Min_Amount

Min_Station = 1
Min_Amount = S1_Bikes

IF(S2_Bikes<Min_Amount)THEN
{Min_Station = 2
Min_Amount = S2_Bikes
}

IF(S3_Bikes<Min_Amount)THEN
{Min_Station = 3
Min_Amount = S3_Bikes
}

IF(S4_Bikes<Min_Amount)THEN
{Min_Station = 4
Min_Amount = S4_Bikes
}

IF(S5_Bikes<Min_Amount)THEN
{Min_Station = 5
Min_Amount = S5_Bikes
}

IF(S6_Bikes<Min_Amount)THEN
{Min_Station = 6
Min_Amount = S6_Bikes
}

RETURN Min_Station
```

Take_Incentive	Integer			#Provide incentive for taking bike to short station
				REAL Incentive
				
				Incentive = 0.3
				
				REAL rn
				rn = U(0.5,0.5)
				IF (rn < Incentive) THEN
				    RETURN 1
				ELSE
				    RETURN 0

Calc_Peak	Integer	Arr_Time	Real	REAL arr_time_in_hours REAL hour arr_time_in_hours = arr_time / 60 hour = arr_time_in_hours MOD 24 IF (hour < 8) THEN RETURN 0 ELSE IF (hour > 22) THEN RETURN 0 ELSE RETURN 1