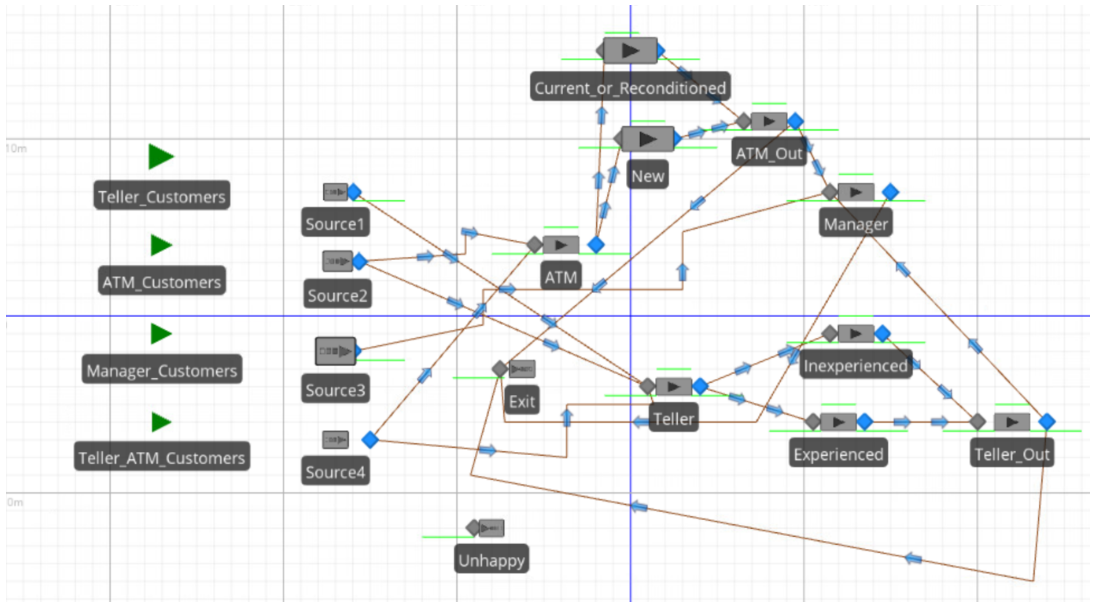
Final Project Report

Nikita Jakkam, Mihir Kandarpa, Sunny Patel, Hannah Tracy
ISYE 3044
Group BC3
Dr. Doug Bodner
April 27, 2022

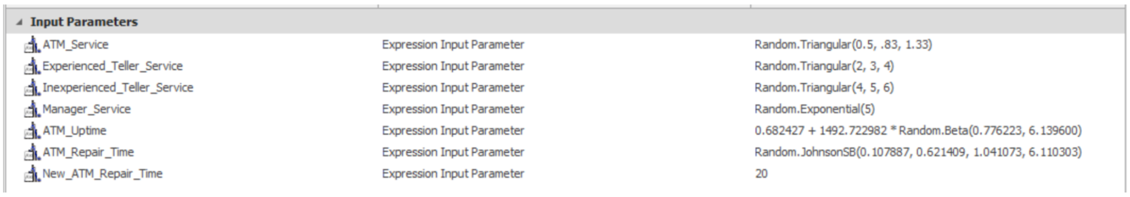
The setup of a small bank branch includes an ATM, multiple tellers, and a manager, which we represented in Simio with 4 sources for the 4 different types of customers, a manager server, and an exit sink. For the teller station, since there are 3 teller workers, we created a dummy station that branched to 2 teller stations, one for experienced and one for inexperienced, and then connected at the end.

We did the same for the 2 ATM machines. The dummy stations for teller and ATM both had processing times of 0 and infinite input buffer capacity. The 2 ATMs and the 2 teller stations that were branched out to have an input buffer of 0 in order for our queue to be at the output node of the dummy servers. To represent the manager we have a simple source, server, and sink model since there is only one manager server.

We used time paths to represent the connections between the servers, and each time path's travel time was the time in seconds to travel between the respective stations given by the table in the problem description. Some of the information in the table was unnecessary as we assumed that customers did not move between the different servers unless specified otherwise in the problem. Link weights on the time paths were 1 unless specified otherwise. The image below is the final Simio representation of the small bank branch.

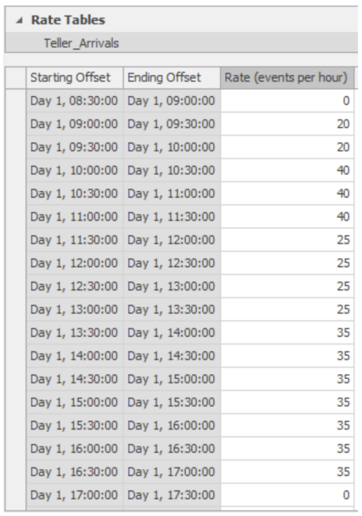


Then we created input parameters for the service times of the ATM, the experienced teller, the inexperienced teller, and the manager, and assigned the processing times of these servers to the appropriate input parameters. The screenshot below shows the input parameters for the service times of each server.



Input Parameters		
ATM_Service	Expression Input Parameter	Random.Triangular(0.5, .83, 1.33)
Experienced_Teller_Service	Expression Input Parameter	Random.Triangular(2, 3, 4)
Inexperienced_Teller_Service	Expression Input Parameter	Random.Triangular(4, 5, 6)
Manager_Service	Expression Input Parameter	Random.Exponential(5)
ATM_Uptime	Expression Input Parameter	0.682427 + 1492.722982 * Random.Beta(0.776223, 6.139600)
ATM_Repair_Time	Expression Input Parameter	Random.JohnsonSB(0.107887, 0.621409, 1.041073, 6.110303)
New_ATM_Repair_Time	Expression Input Parameter	20

Next, since customers arrive in four independent arrivals streams, we created 4 rate tables of their arrival rates in customers per hour. Each rate table starts at 12am and ends at 12am the next day, so since the bank is only open 9am to 5pm, the number of customers per hour is 0 when the bank is not open.

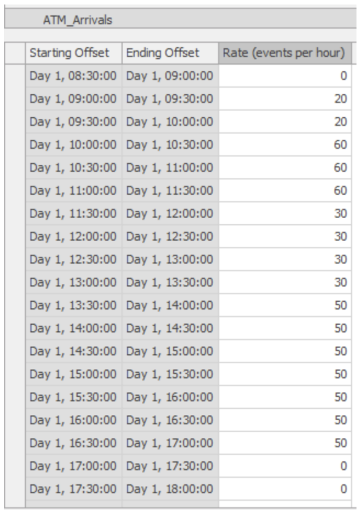


Rate Tables

Teller_Arrivals

Starting Offset	Ending Offset	Rate (events per hour)
Day 1, 08:30:00	Day 1, 09:00:00	0
Day 1, 09:00:00	Day 1, 09:30:00	20
Day 1, 09:30:00	Day 1, 10:00:00	20
Day 1, 10:00:00	Day 1, 10:30:00	40
Day 1, 10:30:00	Day 1, 11:00:00	40
Day 1, 11:00:00	Day 1, 11:30:00	40
Day 1, 11:30:00	Day 1, 12:00:00	25
Day 1, 12:00:00	Day 1, 12:30:00	25
Day 1, 12:30:00	Day 1, 13:00:00	25
Day 1, 13:00:00	Day 1, 13:30:00	25
Day 1, 13:30:00	Day 1, 14:00:00	35
Day 1, 14:00:00	Day 1, 14:30:00	35
Day 1, 14:30:00	Day 1, 15:00:00	35
Day 1, 15:00:00	Day 1, 15:30:00	35
Day 1, 15:30:00	Day 1, 16:00:00	35
Day 1, 16:00:00	Day 1, 16:30:00	35
Day 1, 16:30:00	Day 1, 17:00:00	35
Day 1, 17:00:00	Day 1, 17:30:00	0

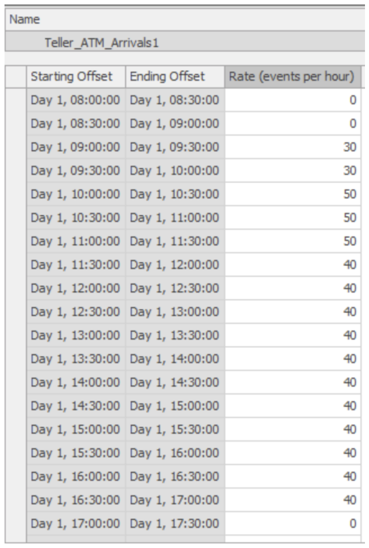
Rate Table for Teller Customer Arrivals



ATM_Arrivals

Starting Offset	Ending Offset	Rate (events per hour)
Day 1, 08:30:00	Day 1, 09:00:00	0
Day 1, 09:00:00	Day 1, 09:30:00	20
Day 1, 09:30:00	Day 1, 10:00:00	20
Day 1, 10:00:00	Day 1, 10:30:00	60
Day 1, 10:30:00	Day 1, 11:00:00	60
Day 1, 11:00:00	Day 1, 11:30:00	60
Day 1, 11:30:00	Day 1, 12:00:00	30
Day 1, 12:00:00	Day 1, 12:30:00	30
Day 1, 12:30:00	Day 1, 13:00:00	30
Day 1, 13:00:00	Day 1, 13:30:00	30
Day 1, 13:30:00	Day 1, 14:00:00	50
Day 1, 14:00:00	Day 1, 14:30:00	50
Day 1, 14:30:00	Day 1, 15:00:00	50
Day 1, 15:00:00	Day 1, 15:30:00	50
Day 1, 15:30:00	Day 1, 16:00:00	50
Day 1, 16:00:00	Day 1, 16:30:00	50
Day 1, 16:30:00	Day 1, 17:00:00	50
Day 1, 17:00:00	Day 1, 17:30:00	0
Day 1, 17:30:00	Day 1, 18:00:00	0

Rate Table for ATM Customer Arrivals

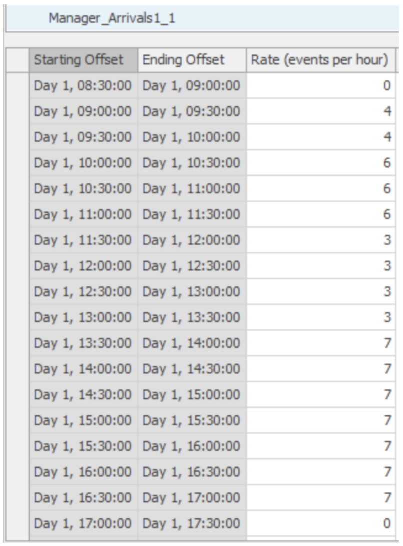


Name

Teller_ATM_Arrivals1

Starting Offset	Ending Offset	Rate (events per hour)
Day 1, 08:00:00	Day 1, 08:30:00	0
Day 1, 08:30:00	Day 1, 09:00:00	0
Day 1, 09:00:00	Day 1, 09:30:00	30
Day 1, 09:30:00	Day 1, 10:00:00	30
Day 1, 10:00:00	Day 1, 10:30:00	50
Day 1, 10:30:00	Day 1, 11:00:00	50
Day 1, 11:00:00	Day 1, 11:30:00	50
Day 1, 11:30:00	Day 1, 12:00:00	40
Day 1, 12:00:00	Day 1, 12:30:00	40
Day 1, 12:30:00	Day 1, 13:00:00	40
Day 1, 13:00:00	Day 1, 13:30:00	40
Day 1, 13:30:00	Day 1, 14:00:00	40
Day 1, 14:00:00	Day 1, 14:30:00	40
Day 1, 14:30:00	Day 1, 15:00:00	40
Day 1, 15:00:00	Day 1, 15:30:00	40
Day 1, 15:30:00	Day 1, 16:00:00	40
Day 1, 16:00:00	Day 1, 16:30:00	40
Day 1, 16:30:00	Day 1, 17:00:00	40
Day 1, 17:00:00	Day 1, 17:30:00	0

Rate Table for Teller/ATM Customer Arrivals



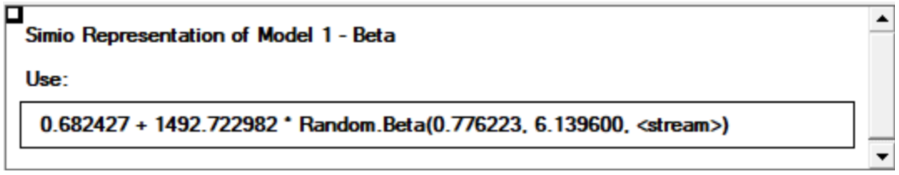
Manager_Arrivals1_1

Starting Offset	Ending Offset	Rate (events per hour)
Day 1, 08:30:00	Day 1, 09:00:00	0
Day 1, 09:00:00	Day 1, 09:30:00	4
Day 1, 09:30:00	Day 1, 10:00:00	4
Day 1, 10:00:00	Day 1, 10:30:00	6
Day 1, 10:30:00	Day 1, 11:00:00	6
Day 1, 11:00:00	Day 1, 11:30:00	6
Day 1, 11:30:00	Day 1, 12:00:00	3
Day 1, 12:00:00	Day 1, 12:30:00	3
Day 1, 12:30:00	Day 1, 13:00:00	3
Day 1, 13:00:00	Day 1, 13:30:00	3
Day 1, 13:30:00	Day 1, 14:00:00	7
Day 1, 14:00:00	Day 1, 14:30:00	7
Day 1, 14:30:00	Day 1, 15:00:00	7
Day 1, 15:00:00	Day 1, 15:30:00	7
Day 1, 15:30:00	Day 1, 16:00:00	7
Day 1, 16:00:00	Day 1, 16:30:00	7
Day 1, 16:30:00	Day 1, 17:00:00	7
Day 1, 17:00:00	Day 1, 17:30:00	0

Rate Table for Manager Customer Arrivals

To implement the ATM reliability logic we used ExpertFit to fit distributions to the given uptime and repair time data in the Excel file. The Simio representation of the uptime and downtime distributions are below:

Uptime:

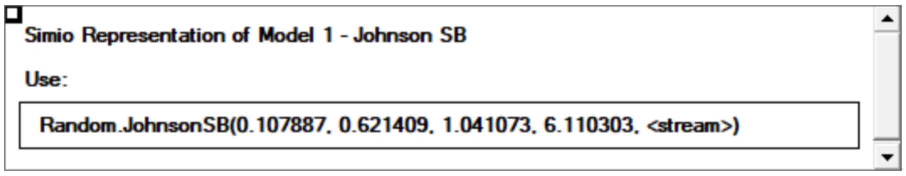


Simio Representation of Model 1 - Beta

Use:

```
0.682427 + 1492.722982 * Random.Beta(0.776223, 6.139600, <stream>)
```

Downtime:



Simio Representation of Model 1 - Johnson SB

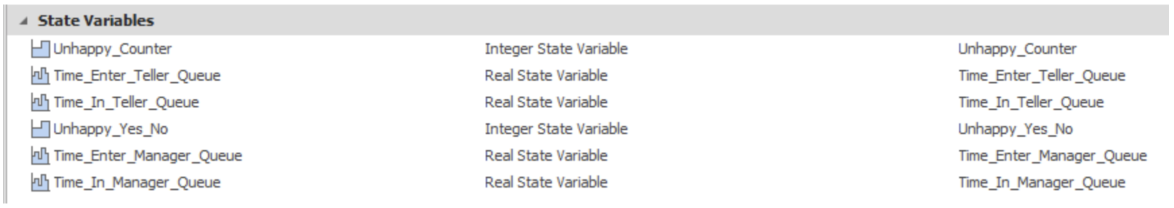
Use:

```
Random.JohnsonSB(0.107887, 0.621409, 1.041073, 6.110303, <stream>)
```

Once we had these distributions we went into Simio and specified them as input parameters. We then clicked on the ATM and inputted the distributions under reliability logic using Calendar Time Based failure logic.

All of the balking logic for this model was implemented under the balking and reneging properties on the source. An Unhappy sink was used to keep track of the customers that balked from the source. An exit sink was also used to keep track of the customers that went through the model. At any time if a customer balked and exited through the unhappy sink or became unhappy with the time spent in a certain queue, we had an unhappy_counter state variable which we incremented to keep track of the unhappy customers. If customers were switching between lines or if customers were becoming unhappy and had a percentage chance of traveling to another server, we implemented this logic using Math.If on the time paths with link weights.

In order to have customers choose the smallest queue, we used Node Lists with the selection goal as the smallest value. This logic was used for ATMs and Tellers as customers would choose the ATM or teller that would have the smallest line. Node Lists were helpful anytime we had a dummy server branching out to the actual servers. We also used state variables to calculate the time spent in the Teller and Manager queue which were later used to increment the unhappy_counter if a customer spent long enough in the respective queue to become unhappy.



State Variables

Unhappy_Counter	Integer State Variable	Unhappy_Counter
Time_Enter_Teller_Queue	Real State Variable	Time_Enter_Teller_Queue
Time_In_Teller_Queue	Real State Variable	Time_In_Teller_Queue
Unhappy_Yes_No	Integer State Variable	Unhappy_Yes_No
Time_Enter_Manager_Queue	Real State Variable	Time_Enter_Manager_Queue
Time_In_Manager_Queue	Real State Variable	Time_In_Manager_Queue

Verification

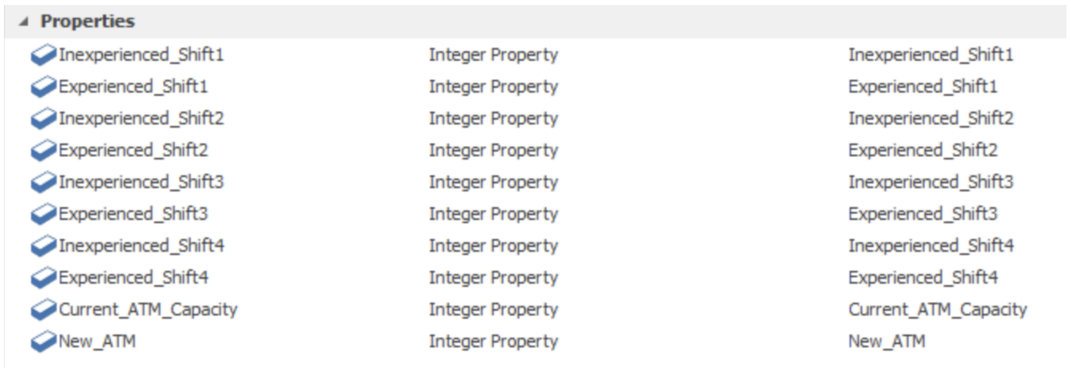
Over the course of this project we have been to Siddhant Singhania and Dr. Bodner's office hours to clear up any confusion about the way we implemented the different aspects of the model. To verify our model, we used the trace function to ensure the model does not exhibit any unusual behavior and made sure the customers were traveling to the right servers. We made changes when we saw entities traveling to the wrong servers. While running the experiment we paused the simulation at certain times to examine values of key variables. Moreover, we examined the model output for total cost to determine if it was giving us the correct cost based on the number of unhappy customers and the number of experienced and inexperienced tellers.

We were able to build a model with high face validity. We followed all the instructions given in the project description. We made a few assumptions such as having one teller in the system at all times in order for the system to keep running.

The one thing we did notice with the ATM is that when we ran our model starting at 12am, the ATM would keep crashing even before the bank was open. In order to get around this we tried to implement a work schedule for the ATM however we still noticed that the ATM kept crashing. After talking to Dr.Bodner about this he recommended we leave this as it is and to make it an assumption in our model since we are using the Calendar Time Based reliability logic.

Once we implemented and verified the base model provided in the project description, it was time to test multiple scenarios and optimize the number of each type of teller for each shift. We set up OptQuest in our experiment and set multiple controls, constraints, and an objective function. The controls to be optimized include the number of each type of teller for each 2-hour shift and the number of new and refurbished ATMs to be purchased. The constraints ensure that there are at most 3 tellers total for each shift and no more than 2 ATMs in the bank. The objective function is minimizing the total daily cost for the bank. The objective function, the constraints, and the decision variables are shown below.

The following Properties were added to support the constraints and controls of the OptQuest Model:



Properties

Inexperienced_Shift1	Integer Property	Inexperienced_Shift1
Experienced_Shift1	Integer Property	Experienced_Shift1
Inexperienced_Shift2	Integer Property	Inexperienced_Shift2
Experienced_Shift2	Integer Property	Experienced_Shift2
Inexperienced_Shift3	Integer Property	Inexperienced_Shift3
Experienced_Shift3	Integer Property	Experienced_Shift3
Inexperienced_Shift4	Integer Property	Inexperienced_Shift4
Experienced_Shift4	Integer Property	Experienced_Shift4
Current_ATM_Capacity	Integer Property	Current_ATM_Capacity
New_ATM	Integer Property	New_ATM

OptQuest Model:

Decision Variables:

$t_{i,j}$ represents the number of teller type i for shift j where i $\epsilon\{1 - experienced, 2 - inexperienced\}$ and j$\epsilon\{1,2,3,4\}$

a_i represents the number of ATM type i purchased where i $\epsilon\{1 - new,\ 2 - reconditioned\}$ and a_i$\epsilon\{0,1\}$

Objective Funtion:

$$\text{Minimize Cost} = 130 * \sum_{j=1}^{4} t_{1,j} + 90 * \sum_{j=1}^{4} t_2, j + 100 * unhappy\_customers + 500 * a_1 + 350 * a_2$$

Constriants:

$$s.t.$$
$$1 \le t_{1,j} + t_{2,j} \le 3 \ \ \forall j\epsilon\{1,2,3,4\} \ (1)$$
$$a_1 + a_2 \le 1 \ (2)$$
$$t_{i,j} \ge 0 \ \ \forall_{i,j}$$
$$a_i\epsilon\{0,1\} \ \ \forall_i\epsilon\{1,2\}$$

The optimization model above was formulated to minimize the overall daily costs for the bank. The model incorporates decisions for which tellers and how many of each type to include for each 2-hr shift and if an additional ATM (and what type) should be purchased. OptQuest ran the default maximum of 300 scenarios and the default 6 replications of each scenario. Each scenario evaluated different teller staffings during the day and the implications of purchasing an additional ATM.

Two decision variables were used in the model. The first decision variable, $t_{i,j}$ was used to determine the number of teller type i (experienced or inexperienced) assigned to shift j (each of the 2hr shifts from 9am-5pm). The second decision variable, a_i, is a binary variable and was used to determine if each ATM type i (new or reconditioned) would be purchased.

The objective of this model was to minimize the overall daily costs for the bank. The bank could incur daily costs for paying each teller at $130/shift for experienced tellers and $90/shift for inexperienced tellers, $100 per unhappy customer, $500 if a new ATM was purchased, and $350 if a reconditioned ATM was purchased.

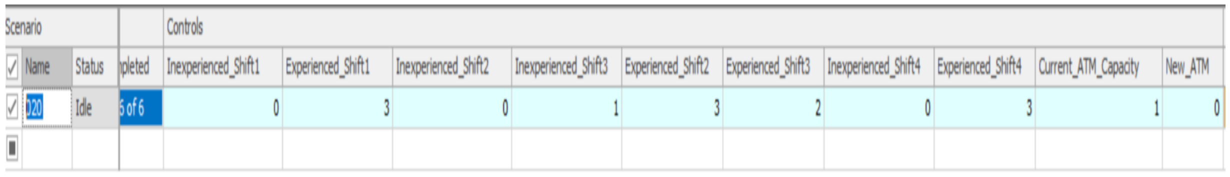
The equation below was used for OptQuest to represent total cost:

minimize *130 * (Experienced_Shift1 + Experienced_Shift2 + Experienced_Shift3 + Experienced_Shift4) + 90 * (Inexperienced_Shift1 + Inexperienced_Shift2 + Inexperienced_Shift3 + Inexperienced_Shift4) + (100 * Unhappy_Counter) + (500 * New_ATM) + (350 * (Current_ATM_Capacity - 1))*

There were two main constraints added to this model. Constraint (1) ensured there's no more than 3 tellers (experienced and inexperienced combined) during each shift. This was implemented in OptQuest by creating a Shift_capacity constraint for each of the four shifts that summed up the number of experienced and inexperienced tellers assigned to each shift.

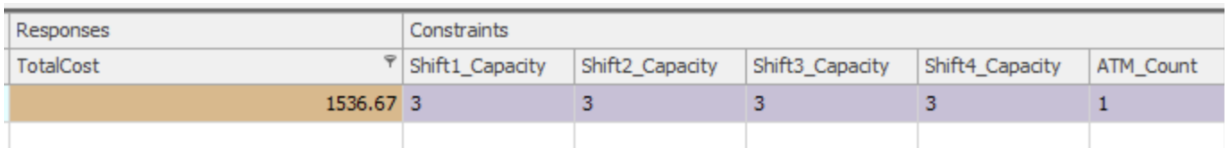
Constraint (2) was included to ensure that a maximum of 1 ATM (new or reconditioned) could be purchased. This was implemented by creating an ATM_Count constraint that limited the sum of Current_ATM_Capacity (which would have a value of 2 if a reconditioned ATM is purchased) and New_ATM (which has a value of 0 or 1) to 2. The equation below represents Constraint (2):
(lower bound)1 ≤ Current_ATM_Capacity + New_ATM ≤ 2 (upper bound)

OptQuest output for the optimal scenario:



Scenario				Controls									
✓	Name	Status	pleted	Inexperienced_Shift1	Experienced_Shift1	Inexperienced_Shift2	Inexperienced_Shift3	Experienced_Shift2	Experienced_Shift3	Inexperienced_Shift4	Experienced_Shift4	Current_ATM_Capacity	New_ATM
✓	020	Idle	6 of 6	0	3	0	1	3	2	0	3	1	0

The values above indicate the optimal number of each type of teller for each shift and if an additional ATM needed to be purchased.



Responses	Constraints				
TotalCost	Shift1_Capacity	Shift2_Capacity	Shift3_Capacity	Shift4_Capacity	ATM_Count
1536.67	3	3	3	3	1

The TotalCost represents the objective value for the optimal solution. The other values represent the constraints for the shift capacities and the ATM count (ATM_Count is 1 because it is the sum of Current_ATM_Capacity and New_ATM).

Answers to Questions

1. **Should the bank purchase an additional ATM machine? If so, should it be new or reconditioned?**

   The bank does not need to purchase an additional ATM machine. We formulated an optimization model to minimize overall daily cost. We found that we can minimize cost by using the current ATM and without purchasing any additional ones.

2. **Find the number of each classification of tellers that the bank should schedule for each of the four time blocks given in the problem description (i.e., a daily schedule for the bank of tellers).**

   The table below shows the proposed daily schedule for the bank of tellers.

	# of Inexperienced Tellers	# of Experienced Tellers
Time Block 1: 9:00 a.m. – 11:00 a.m.	0	3
Time Block 2: 11:00 a.m. – 1:00 p.m.	0	3
Time Block 3: 1:00 p.m. – 3:00 p.m.	1	2
Time Block 4: 3:00 p.m. – 5:00 p.m.	0	3

3. **What are the bank's expected daily operating costs with respect to the cost metrics provided in the problem description?**

   The total expected daily operating cost is $1536.37. This consists of the hourly costs associated with the experienced and inexperienced tellers of $65 and $45, respectively and the cost of having unhappy customers at $100 for each occurrence.

4. **What is the expected cost per day associated with "unhappy customers?"**
   After running 6 replications, we found the average expected cost per day for unhappy customers to be $16.67.

5. **How often does the ATM machine jam? How long does it take to repair it? (The answers to both of these questions should be accompanied by the method you used to obtain them.)**

   The number of occurrences of the ATM machine jamming is 3.5. It takes on average 0.0659 hours to repair it. Both of these results are visible in the PivotGrid under Server > Resource > ResourceState > TimeFailed. The average here is the average failure duration, or the time for repair.

6. **On average, how many customers is the bank serving per day? This value should include all customers who enter the bank and are involved in some sort of transaction (this should not include customers who leave immediately after entering the bank as a result of a line being too long.)**

   On average, 951.3 customers are served in the bank on a given day. Since we are only including customers who are involved in some sort of transaction in this calculation, we used the throughput of our Exit sink to determine this number.

7. **What proportion of the total arriving customers are considered to be "unhappy customers?"**

   The total number of arriving customers is 955.166 and of those, 0.16667 are "unhappy" customers, so unhappy customers make up about 0.017% of the total arriving customers on a given day. This number is slightly lower than expected but considering we have 3 experienced tellers working most of the shifts, this might contribute to a lower count of unhappy customers.