Fixing Toner It Down! Inc.'s Repair Network

Elizabeth Brett

Rujuta Desai

Medhavi Gandhi

Amanda Zeitlin

Executive Summary:

This report recommends that the Toner It Down! Inc.'s (TIDInc) repair division employs a total of seven mechanics and three van operators, direct its van operators to begin copier swaps immediately after returning to dispatch, and utilize an alternate strategy for routing mechanics described as follows:

> If a mechanic finishes service and the request queue is empty, that mechanic should wait at the business center for a new request. If a new request arrives and there are multiple mechanics waiting at different business centers, the closest one should be sent. Additionally, if a mechanic finishes service and there are multiple requests in the request queue, the mechanic should go to the closest request.

The management of Toner It Down! Inc.'s repair division has been concerned about the speed of their service. The goal of this report is to analyze how TIDInc can improve its repair network in order to keep up with its competitors. This report used computer simulation to examine the effect of different operational improvements on both repair response times and installation times, and looked at the number of mechanics and vans needed to match competitors' service speeds.

Our suggested repair network will cost a total of $1,280,000 annually and is robust to both modeling error and real-world factors like traffic. Additionally, it will allow TIDInc to beat its competitors' repair speeds. Our proposed network has an average mechanic response time of 41 minutes and an average copier delivery time of 2 hours and 25 minutes.

We have extensively analyzed and tested our model and believe that our solution would allow TIDInc to provide excellent repair service at a relatively low cost. Documented in this report are the details of our research on the company, modeling approach, testing, results, and analysis.

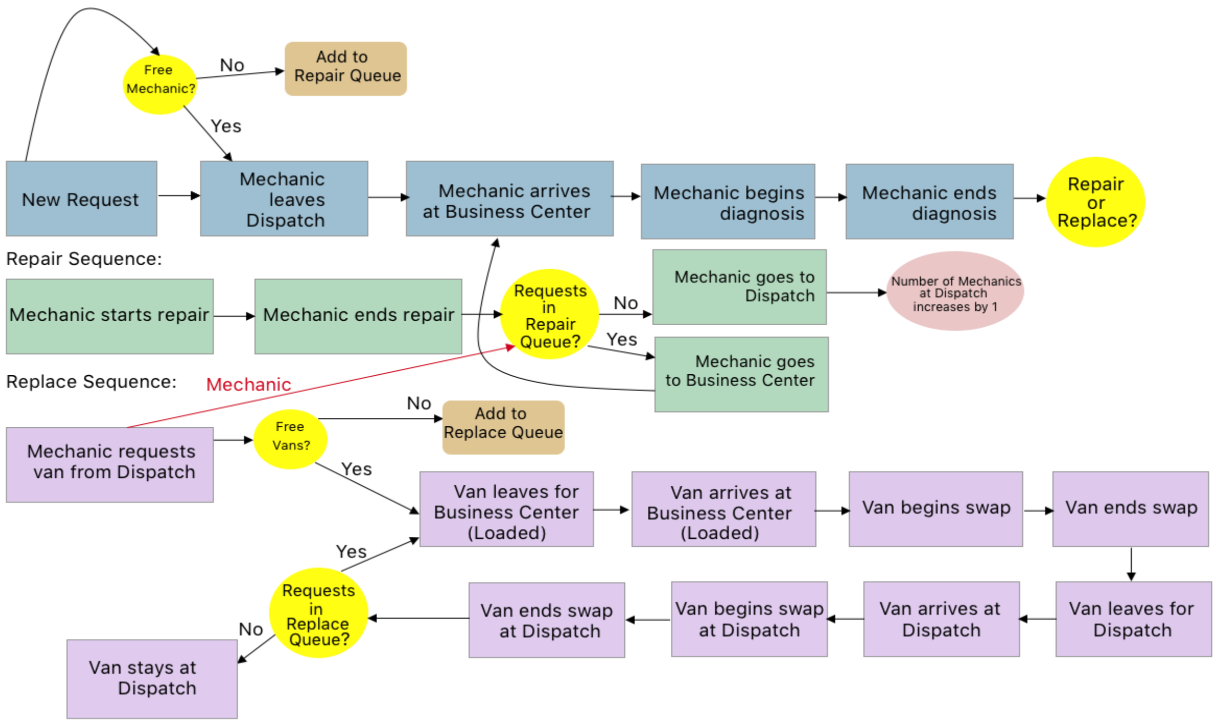
Problem Description:

Toner It Down! Inc. is a developer and manufacturer of document management systems worried about losing customers due to their inefficient repair network. Their goal is to renovate their repair network to be able to serve customers faster and more efficiently. As simulation specialists, we created a model to study their repair network, improve its performance, and help determine what resources they would need to better serve their customers and keep up with their competitors.

TIDInc is interested in improving two primary metrics. The first is the initial response time, the time taken for a mechanic to arrive on-site to service a request. The second is delivery time for replaced copiers, the time between when a request for a new copier is placed and when the new copier is delivered and fully installed.

<u>Modeling the events in the system</u>

After analyzing the data and conducting interviews with company employees to understand the flow of requests through the system, we came up with a general model of the system. This model is described in the flowchart below:



Free Mechanic?
No
Add to Repair Queue
Yes
New Request
Mechanic leaves Dispatch
Mechanic arrives at Business Center
Mechanic begins diagnosis
Mechanic ends diagnosis
Repair or Replace?
Repair Sequence:
Mechanic starts repair
Mechanic ends repair
Requests in Repair Queue?
No
Mechanic goes to Dispatch
Number of Mechanics at Dispatch increases by 1
Yes
Mechanic goes to Business Center
Replace Sequence:
Mechanic
Mechanic requests van from Dispatch
Free Vans?
No
Add to Replace Queue
Yes
Van leaves for Business Center (Loaded)
Van arrives at Business Center (Loaded)
Van begins swap
Van ends swap
Van leaves for Dispatch
Van arrives at Dispatch
Van begins swap at Dispatch
Van ends swap at Dispatch
Requests in Replace Queue?
Yes
No
Van stays at Dispatch

Both the repair queue and the replace queue are First-In, First-Out (FIFO), so an available mechanic or van will always serve the request that arrived first in the system.

We analyzed historical data on TIDInc.'s repair network and used our findings to model the arrival rate, as well as the probability distribution of the initial diagnosis times and repair times. We interviewed van operators to determine how to model copier swap times. Based on these interviews, we decided to model the swap time at dispatch as triangular with a minimum of 10 minutes, a maximum of 25 minutes, and a mode of 15 minutes. We decided to model the on-site swap times as triangular with a minimum of 20 minutes, a maximum of 60 minutes, and a mode of 30 minutes. In addition, we made the following assumptions:

1. Mechanics are not able to check for new requests when they are driving. So, for example, if a mechanic is going back to dispatch because there are no requests in the queue, and one arrives on their way back to dispatch, they will only be able to see the new request once they arrive.
2. Mechanics and vans travel at 60 miles per hour.
3. Mechanics and vans cannot speed up their work due to the queue size. All service and travel times follow a constant probability distribution.

Once we came up with our basic model for the system we used object-oriented programming and discrete event simulation to run it in Python. Discrete event simulation is a method of simulating real-world systems by focusing on individual events. The benefit of object-oriented programming is that it is flexible and all our events can have their own properties as well as shared properties. A detailed description of our code can be found in the appendix. We used trial-and-error to iterate over different numbers of mechanics and vans in order to determine the minimum number required to match our competitors' service. Namely, we wanted the average initial response time to be under 1 hour and the average copier delivery time to be under 3 hours.

Our basic model is described above. In addition, we tested two model variations, both of which improved the network's performance and reduced the number of mechanics needed.

In the first variation, if the request queue was empty when a mechanic finished servicing a request, he or she stayed at that business center waiting for another request rather than returning to dispatch. The request queue was again organized as a First-In, First-Out queue. Therefore, if the request queue was not empty when a mechanic finished service, we retrieved the first request in the queue and sent the mechanic to its business center as usual. If there were available mechanics when a new request entered the system, the closest available mechanic was sent. If not, the request was added to the request queue as usual.

In the second variation, the mechanics again stayed at the business center if the queue was empty. Additionally, if there were available mechanics when a new request arrived, the closest available mechanic was again sent. However, the repair queue was no longer First-In, First-Out. Instead, available mechanics always serviced the request closest to them. Therefore, if a mechanic finished servicing a request and the request queue was not empty, he or she would look at all the requests in the queue, find the closest one, and go to its business center.

In order to get reliable performance metrics, we ran each model variation 100 times. During each run, the model ran for 200 hours with a 100 hour warm-up period. During the warm-up period, the model ran but no statistics were recorded. Requests that were serviced during the warm-up period did not "count" towards our final results. This allowed us to look at the model in its long-term, steady state and prevented our results from being affected by the fact that, initially, the system is empty and the queue length is 0.

Using data to model our system:

We were provided with a historical dataset that documented customer requests over a 60 day period. This document includes information about the time of day the request was placed, which business center the request originated from, how long the initial diagnosis took, whether the copier needed to be repaired or replaced, and the repair time if the repair was done on premise.

When we first looked at the given data, we checked to ensure that it was "clean". We noticed that some requests appeared to come in after midnight - at times 24 or greater, and eliminated them. Once the data was fixed, we performed our analysis, making the assumption was that the requests were independent and identically distributed.

We first estimated the percent of copiers which needed to be replaced, which was around 18%. We also estimated the probability of a request originating from each specific business center by looking at the proportion of requests originating from each location. The results of this analysis are as follows: BC_1: 0.039509, BC_2: 0.0821, BC_3: 0.1079, BC_4: 0.1354, BC_5: 0.1184, BC_6: 0.055, BC_7: 0.1238, BC_8: 0.0579, BC_9: 0.1375, BC_10: 0.1424..

The analysis of on-site repair times was very technical in nature and is detailed in the Appendix of our report. After the analysis, we discovered that repair times follow a Beta distribution with alpha=2.6158, beta=7.4606, and the loc and scale equal to -0.0077 and 1.5579, respectively. The loc and scale adjust the beta distribution, so if T represents the initial diagnosis time and X is a beta random variable with parameters alpha and beta, T=(X - loc)/(scale)

To analyze initial diagnosis times, we noticed that some business centers, namely centers 2, 3 and 9, had higher average initial diagnosis times than the rest, so we decided to separate the business centers into two groups. Group 1 contained BC's 2, 3, and 9 while group 2 had BC's 1, 4, 5, 6, 7, 8, and 10. The probability distribution of initial diagnosis times in both groups followed a normal distribution, but with different means and standard deviations. Group 1 had a mean of 22.15 and standard deviation of 5.34. Group 2 had a mean of 16.13 and standard deviation of 2.89.

The last metric we needed to model was the frequency of requests throughout the day. After thorough testing, we decided to model request arrivals as a non-stationary poisson process. Late at night, between 10 pm and 3 am, requests occur at a rate of 1.137 per hour. Between 3am and 10 pm, the arrival rate varied according to a quadratic function peaking roughly mid-day. The specific form of this function can be found in the Appendix.

Verifying the model:

Our base model and each model variation was checked using model traces to ensure that the model was behaving as expected. Multiple individuals reviewed these traces and verified that there were no errors. Additionally, the base model was verified using queuing theory. We modified the probability distributions of our service times and travel times so that we could make analytical predictions and check that they matched our model. The details of this method are summarized below and explored in more detail in the appendix.

Queuing Theory Verification:

We first created two variations of our base model--in one model copiers were always repaired and in the other they were always replaced. This helped simplify our analysis. The model was then amended so that all the requests went to the same business center and all of the travel and service times took an exponentially distributed amount of time. This allowed us to use common queuing theory calculations. We set the number of available mechanics in each model equal to 2, the number of available vans equal to 2, and the arrival rate equal to 1/hr. We then used queuing theory to make predictions as to what our performance metric should be, and verified that the results of our simulation matched our predictions.

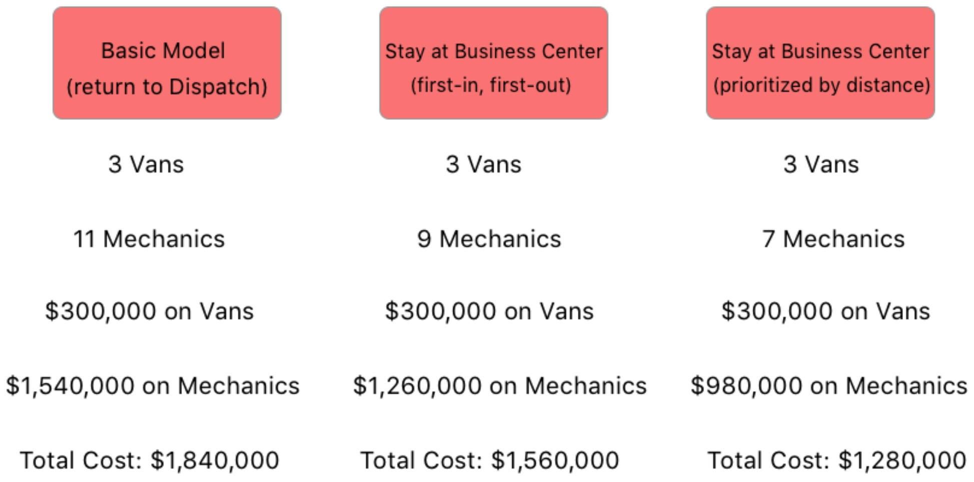
In the repair-only model, our predicted value for the average response time was 131 minutes. Based on our simulation runs, our 95% confidence interval for the average response time was [110, 134], which contains the predicted value 131, indicating that our model for repairs is functioning well.

In the replacement-only model, we continued to look at response time rather than time to installation for the sake of consistency. Here, there are two relevant queueing theory predictions used in the verification of our model. The first prediction is the average response time if the copier requests never had to wait for the dispatch copier swap. In this scenario, the average response time is 89 minutes. The second prediction is the average response time if the copier requests always had to wait for the dispatch copier swap. In this scenario, the average response time is 213 minutes. We would expect the average response time in our model to be in between these two statistics because the dispatch copier swaps occurred immediately after the vans returned to dispatch, so the requests only had to wait for the dispatch swap to occur sometimes. Our simulation runs produced a 95% confidence interval of [128, 171], safely in between the theoretical bounds.

Analyzing the model to test variations and robustness:

Model Analysis

In our basic model, once mechanics are done servicing a request, they either return to dispatch if there are no requests waiting, or go to the business center of the request that arrived earliest. We need 3 vans and 11 mechanics in this case. As described earlier, we also tested two model variations corresponding to different mechanic routing strategies in the repair network. For our first variation, we decided to test a system in which a mechanic-instead of returning to dispatch, which might be further away than a request- stays at the business center they completed a repair at while there are no more requests waiting. We predicted that this would be more efficient since it would cut out the travel time required to get to dispatch, and we were correct about this. We were able to achieve our desired time metrics with only 3 vans and 9 mechanics. For our second variation, we had the mechanics stay at business centers after they were done (like before), but instead of attending to requests in the order that they arrived, prioritized requests by how far away they were from the business center that the mechanic was at. We thought this would be a better system because it would reduce travel time, and we were right- this was our best model, and gave us an average initial response time of under an hour and copier delivery time under 3 hours, using only 3 vans and 7 mechanics. As seen below, by using our best model, we would be hiring 37% fewer mechanics and saving $560,000 or 30%.



Basic Model (return to Dispatch)	Stay at Business Center (first-in, first-out)	Stay at Business Center (prioritized by distance)
3 Vans	3 Vans	3 Vans
11 Mechanics	9 Mechanics	7 Mechanics
$300,000 on Vans	$300,000 on Vans	$300,000 on Vans
$1,540,000 on Mechanics	$1,260,000 on Mechanics	$980,000 on Mechanics
Total Cost: $1,840,000	Total Cost: $1,560,000	Total Cost: $1,280,000

The results for the average initial response time and copier delivery time and their respective 95% confidence intervals for each of the three variations are summarized in the table below:

Model Type	Mean Estimate for Initial Response Time (Minutes)	95% Confidence Interval for Initial Response Time	Mean Estimate for Copier Delivery Times (Minutes)	95% Confidence Interval for Copier Delivery Times
Basic Model	52.0145	[50.8994, 53.1296]	149.1901	[142.3262, 156.0539]
First Variation (Mechanics stay at Business Center, FIFO)	45.0393	[42.2863, 47.7923]	134.8724	[129.0018, 140.7431]
Second Variation (Mechanic finds the closest Request)	40.7557	[39.7430, 41.7684]	145.4021	[137.8938, 152.9104]

Consistent with the idea that our second model variation is optimal for the repair network, this variation, in which the mechanic goes to the request that is the closest to him if he is free, produces the best average initial response time of ~ 41 minutes. The initial response time has decreased by 11% in this variation from the first variation where the mechanic stays at the business center but the requests are in a First-In, First-Out queue. Although the average copier delivery times increased by roughly 8% from the first variation to the second variation, the average is still well under three hours, so TIDInc is still able to keep up with its competitors. In both variations, three vans are still needed to keep up with the requests, so the difference in copier delivery times between variations is not substantial. Additionally, for the two variations, we computed the 95th percentiles of the two performance measures in order to analyze the 5% worst case times for the initial response time and copier delivery time. The differences between the 95th percentile of copier delivery times in the two models was not significant, with both values around 5 hours. However, this was not the case for initial response times. In our second variation, the 95th percentile, or 5% worst case, of initial response times was around 125 minutes with a 95% CI of [120.86 , 129.46]. Consonant with the results stated earlier, the 5% worst case times for initial response time in the second variation are still better than the 5% worst case times in our first model variation where the mechanics operate using a FIFO queuing system. The 95th percentile of initial response times in the first variation was around 143 minutes with a 95% CI of [136.04, 149.89]. The fact that the 95th percentile of initial response times are still lower for the second variation than for the first variation helps alleviate the worry that the second variation, by abandoning the FIFO queueing system, would cause some customers to have extremely long waits.

Another performance measure we looked at was average queue length for the mechanics and van operators. Examining this metric allowed us to further analyze whether the number of mechanics and vans we suggested would be able to keep up with the requests coming in. The table below summarizes the average queue lengths and their respective confidence intervals for the three models:

Model Type	Average Request Queue Length	95% Confidence Interval for Mechanic	Average Copier Request Queue Length	95% Confidence Interval for Van
Basic Model	6.2608	[6.1720, 6.3497]	2.2088	[2.1017, 2.3159]

Stay at Business Center (FIFO)	5.8894	[5.6807, 6.0981]	1.9600	[1.8512, 2.0687]
Stay at Business Center (Prioritize Distance)	5.5566	[5.4736, 5.6340]	2.1587	[2.0497, 2.2676]

We see once again that the second variation has the best performance for average request queue length. Although the average queue length for copier requests is larger in the second variation than in the first, the difference is small. The fact that on average there are roughly around 6 people waiting in the regular request queue for a mechanic and 2 people in the copier request queue evinces that the number of mechanics and vans is able to, in the long-term, keep up with the requests coming in and not leave an excessive number of people waiting in the queue at once.

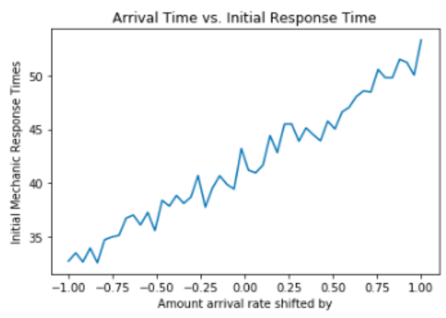
Sensitivity Analysis

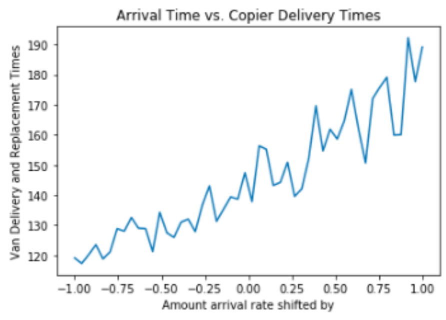
Additionally, we conducted sensitivity analysis on the optimal model (variation #2, where the request queue is prioritized by distance) in order to make sure that the model is robust and does not depend heavily on our estimated parameters. In addition to ensuring our solution will still be viable despite potential modeling error, this analysis also allows us to examine the effect that changes in request patterns could have on our performance metrics. We perturbed the parameters of the different distributions, and analyzed the effect on our performance measures.

Arrival Rate:

The arrival times for the requests from 3am-10pm are characterized by a nonstationary Poisson process in which the arrival rate is described by a quadratic function. In order to understand the effect a small change in arrival rate would have on our network, we analyzed the effect uniformly shifting the arrival rate up or down by a small amount during the quadratic time period (the overnight arrival rate from 10pm-3am stayed the same). We ran 50 simulation replications for 150 hours with 50 hours of warm-up time, and the constant we added to the quadratic function took values on the range of [-1,1] in intervals of width size 1/25. We chose -1 as a lower bound in order to ensure that the arrival rate was always greater than zero.

After running the model, we found the following relationships for how arrival rate affects initial response time and copier delivery time:

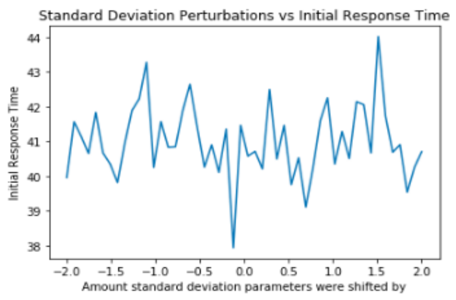


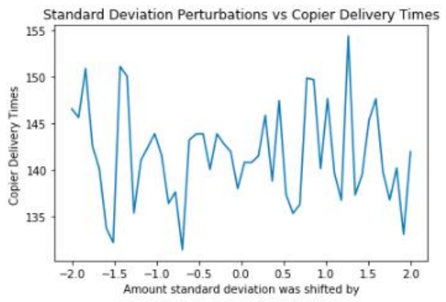


There are some fluctuations caused by random variation in the service times, but the results are pretty consistent with what we would expect. As we increase the arrival rate, both the initial response times and copier delivery times increase. This is because there are more requests in the system, so the queues will get longer and the mechanics and van operators will be busy more often. Fortunately, we do not see drastic fluctuations in our performance metrics when we change the arrival rate. Even when we add a full extra request per hour, the average initial response time is approximately around 53 minutes and average copier delivery time is approximately around 189 minutes, compared to 41 minutes and 145 minutes when there is no perturbation. The average initial response time is still under an hour and the average copier delivery time is just over 3 hours, so TIDInc. will still have a competitive repair network. If management is concerned that the average delivery time in this case will be over 3 hours, they could consider hiring an extra van operator if the request arrival rate increases. Overall, the performance measures did not change by an exorbitant amount when the arrival rate was perturbed, meaning that TIDInc will be able to keep up with fluctuations in demand

Diagnosis Time:

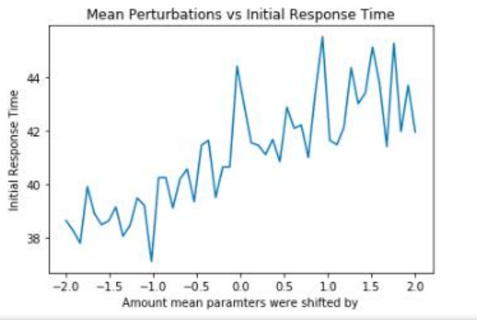
The time that a mechanic takes to diagnose an issue and determine whether the copier must be replaced is modeled by a normal distribution as mentioned earlier. To perform sensitivity analysis on the diagnosis times, we perturbed both the mean and the standard deviation separately and analyzed the effect on our two performance measures. For each value of the mean and standard deviation we looked at, we ran 50 simulation replications for 150 hours with 50 hours of warm-up time. For both parameters, we added a small perturbation that ranged from [-2,2] in interval width sizes of 2/25. Here are the results on initial response time and copier delivery time when we slightly adjusted the standard deviation of the on-site diagnosis time:

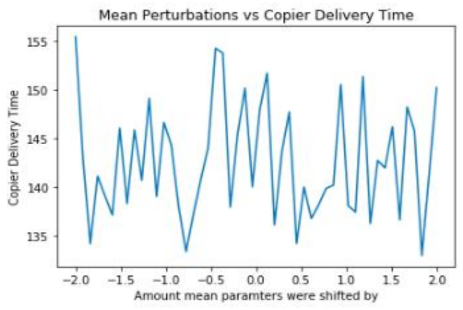




While there are fluctuations caused by random variance in the simulation runs, there is no general trend that we see in either performance measure when we increase the standard deviation. This is reasonable because increasing the standard deviation does not directly increase the amount of time it takes for either the mechanic or the van operator to respond to a request. Therefore, our model is robust to changes in the standard deviation.

The effect of perturbations on the mean parameter on the two performance measures is shown in the graphs below:

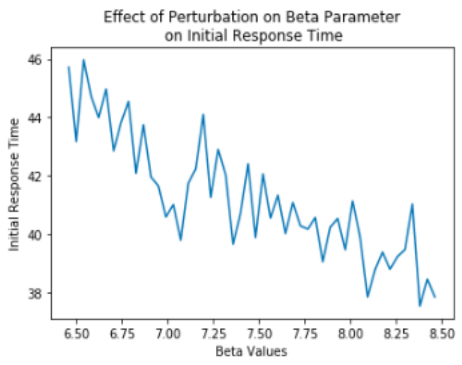


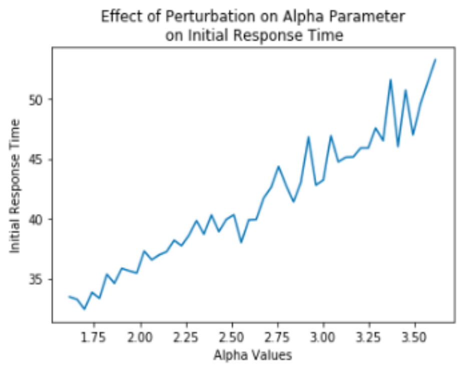


As we can see from the graph on the left, increasing the mean diagnosis time increases the initial response time because the mechanics spend longer on each request and can service a smaller number of requests in a given amount of time. As a result, the mechanics are busy more often and the queue length increases. While we again see random fluctuations in both charts, the copier delivery times do not show a trend when the mean is perturbed because the mechanic diagnosis times do not affect the copier delivery times. Once again, perturbing the values does not prevent the repair network from meeting its desired performance metrics. Even when we increase the mean diagnosis time by two entire minutes, the average initial response time is approximately 42 minutes and the copier delivery time is around 150 minutes. The initial response time barely increased from the variation with the original parameters which had a mean initial response time of roughly 41 minutes. The fact that small increases in the mean diagnosis time failed to significantly alter the initial response shows that the model is robust and does not heavily depend on the original parameters we input into the distributions. It also demonstrates that the mechanic is still able to keep up with requests on average even if diagnosis takes a little longer than usual.

Repair Time:

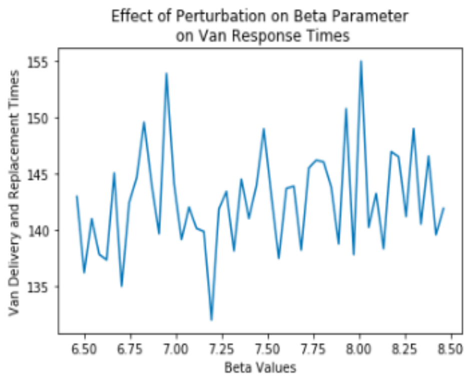
The on-site repair time for a mechanic is described by a beta distribution, with 4 (alpha, beta, loc, scale) parameters. We wanted to analyze the effect of perturbing both the alpha and beta parameters to see how they would affect the initial response time and time to install a new copier. We changed the alpha and beta parameters in the range of [-1,1], with an interval width of 1/50, and ran separate experiments for both. The effect on initial response times is pictured below:

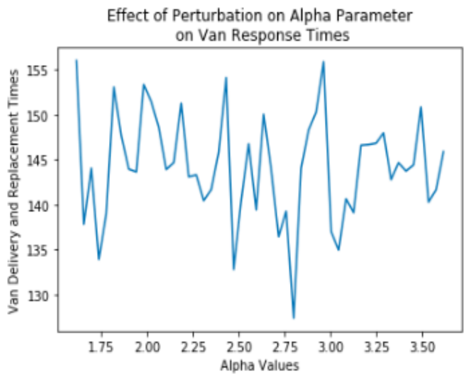




We observed that increasing the alpha parameter by 1 unit, the farthest right of the range, led to an increase in the initial response time to around 53 minutes - well within our goal of keeping the initial response time under an hour. When we decreased the beta parameter by 1 unit, the most extreme case, the initial response time increased to roughly 46 minutes (the beta parameter and initial response time have a decreasing relationship as shown in the graph). This is also a small difference from the original average initial response time of 41 minutes.

We also examined the effect on copier delivery times-these results are pictured in the graph below:



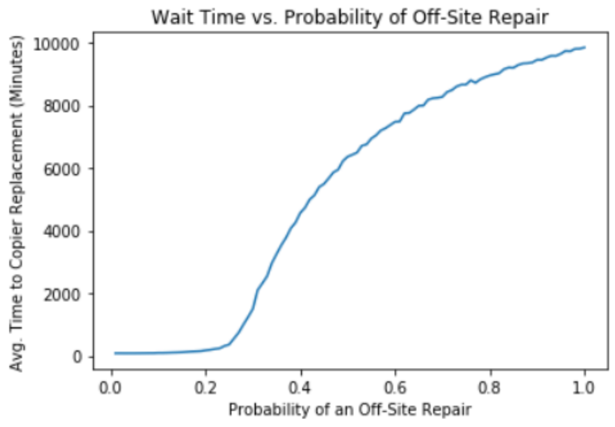


Again we see random fluctuations in the graphs, but, as we expected, changing the parameters for repair time had no effect on the copier delivery time, since it is only directly related to mechanics and not vans. We can see in the plots that there is no trend, and we found no statistically significant differences in van response time caused by varying repair time.

Our model responded well to changes in repair time, and was able to meet our target performance metrics despite these perturbations.

<u>Probability of an on-site repair:</u>

We analyzed the effect that altering the probability that a request needed an off-site repair would have on the average time to install a new copier at these sites. We found that in our solution with three vans, this time stayed under three hours as long as the probability of an off-site repair was under 21%. This was well outside our 99% confidence interval for this probability, which was [16.9%, 19.2%]. As the probability of an off-site repair increased, the average wait time for a new copier increased quickly as the vans struggled to keep up with demand. We ran our model for 300 hours, with 150 hours of warm-up time, and found the following relationship between off-site repair probability and wait time for a new copier:



Therefore, while we can be confident that our solution will not produce explosive behavior in the new-copier queue under present conditions, we can see that our proposed system is quite sensitive to changes in copier repair patterns. Management should keep this result in mind for the future and exercise caution when evaluating proposed changes to copier design that could increase the probability that broken copiers will need off-site repair. Additionally, should the proportion of copiers needing replacement suddenly increase, management should consider adding more vans to keep up with demand.

<u>Other Considerations: Traffic</u>:

A significant amount of traffic would slow down both vans and mechanics. Since traffic is a common inconvenience on modern roads, we wanted to analyze how our model would respond to it. Our results are summarized below.

Increase in travel time due to traffic (%)	Initial Response Time (average)	Copier Delivery Time (average)
10%	46 minutes	168 minutes
15%	47 minutes	177 minutes
20%	50 minutes	191 minutes

As we can see, even with a 20% increase in travel time, the initial response time stays under 1 hour and the copier delivery time is only 11 minutes over 3 hours, a small increase from our desired time. Therefore, our model is well-equipped to handle real-life variables such as traffic.

Conclusion:

Our findings show that TIDInc still has the potential to re-establish itself as a competitive business simply by renovating its copier repair network to be more efficient, and can even outperform its competitors' initial response times and copier delivery times. They can do this by hiring 7 mechanics and having 3 vans on hand and utilizing our suggested operational improvements, keeping mechanics at business centers when they are done repairing (instead of sending them back to dispatch), and prioritizing pending requests by distance from mechanics' locations. By following our recommendations, TIDInc can achieve an average initial response time of 41 minutes and an average copier delivery time of around 145 minutes (2 hours 25 minutes)- better than their competitors. Since we verified our model, tested its robustness, and tested its ability to handle real-life factors such as traffic, we know it delivered the correct recommendations and metrics. TIDInc's managers can rest assured that they will be able to meet their goals by following our model and recommendations.

Appendix:

In order to perform the simulation, we utilized an object-oriented framework within python to model the situation described in the problem statement. Our main approach to building the simulation was employing a discrete-event-simulation in which we modeled every action as an event, and every event triggers another event. For example, we model requests arriving at the dispatch center as event which will either trigger a mechanic leaving the dispatch center to go to the business center that the request came from or if there are no available mechanics left at dispatch, then we will add the request to a request queue, and once a mechanic frees up we will send him over to complete the request. Within the simulation, we model every event as a class that has its own attributes. In addition to that, we also model business centers, vans, mechanics, and the dispatch center as their own class. Below we list each class and its attributes:

Object Classes:

These are our classes that are not events that are triggered during the simulation, and instead instantiate objects that we use throughout the simulation

1. Request: an object of class Request has attributes:
    - Center: what business center the request originated at
    - Time entered: the time the request originated at
    - Time responded: the amount of time it took for a mechanic to arrive at the business center to resolve the request (Initial Response Time)
2. Business Center: an object of class Business Center has attributes:
    - Name: something from the list [“BC1”....”BC10”] to identify the center
    - Number of mechanics: number of mechanics waiting at the business center
3. Dispatch: an object of class Dispatch has attributes:
    - Number of available vans: how many vans are loaded and available to go replace copiers
    - Number of mechanics: how many mechanics are available to go to business centers and work

Super Class: Event

Attributes: simulation clock- our counter for time for the simulation, which is inherited by all sub-classes of Event

Subclasses of Event:

These are all events that are triggered in our simulation

1. NewRequest: the event class for when a new request arrives

   Handling this event:

   - We use our getBC() function to randomly pick a center from which the request originates
   - We create an object of class Request, with the center we got, and the time (simulation clock), with time responded = 0 (initially)
   - We add the NewRequest to the EventList at the current time
   - We use the closest_mechanic() function to find out if there is a mechanic available
   - If there are no free mechanics, we add the request to the RequestQueue
   - If the closest mechanic is at Dispatch, we decrease the number of mechanics at the dispatch center, and add the MechanicArrives event to our EventList, at a time that equals current time + travel time from Dispatch to the center
   - If the closest mechanic is at another Business Center, we decrease the number of mechanics at that center, and add the MechanicArrives event to our EventList, at a time that equals current time + travel time from that center to the center that the request is at

2. MechanicArrives: the event class for when a mechanic arrives at a business center

   Attributes: business center (that the request is at), request

   Handling:

   - Increase the number of mechanics at the business center
   - Get the Initial Response Time (the current time minus the time the request entered the system)
   - Add the Initial Response Time to the array of Initial Response time
   - Use the diagnosis_time() function to get diagnosis time
   - Add the DiagnosisEnds event to the EventList, at a time that equals current time + diagnosis time

3. DiagnosisEnds: the event class for when the mechanic is done diagnosing the issue at the business center

   Attributes: business center (of the request)

   Handling:

- Create an array that stores our two outcomes (repair and replacement)
- Use the probability distribution given for whether a request is a repair or replacement to store the probabilities in an array
- Use Numpy's random choice function to get an outcome for this diagnosis
- If the outcome is repair, then add the RepairStart event to EventList, at the current time
- If the outcome is replace, add the CallVan event to the EventList at the current time

4. RepairStart: the event class for the mechanic beginning the on-site repair process
   Attributes: business center (of the request)
   Handling:
   - Use the repair_time() function to get the time to complete a repair
   - Add the RepairEnd function to the EventList at a time that equals current time + time to complete a repair
5. RepairEnd: the event class for when a mechanic is done repairing a copier
   Attributes: business center (of the request)
   Handling:
   - Add the MechLeavesBusinessCenter event to EventList at the current time

6. MechLeavesBusinessCenter: the event class for when a mechanic leaves a business center
   Attributes: business center (of the request)
   - For our basic model, if there is a request at a business center, leave to that business center and add MechanicArrives at a time that equals current time + travel time to the center to EventList
   - For our basic model, if there are no requests in the RequestQueue, go back to dispatch and add MechGoesToDispatch to the EventList at a time that equals current time + travel time to dispatch
   - For our variation where the mechanics stay at the business center and process requests in a First-In, First-Out way: if there are no requests in the queue, increase the number of mechanics at the business center by 1
   - For our variation where the mechanics stay at the business center and process requests in a First-In, First-Out way: get the first request from the RequestQueue,

    and go to that business center. Add the MechanicArrives event to the EventList, at a time that equals current time + travel time to the center

- For our variation where the mechanics stay at the business center and process requests by distance: if there are no requests in the queue, increase the number of mechanics at the business center by 1
- For our variation where the mechanics stay at the business center and process requests by distance: use the closest_request() function to get the request at the business center closest to us, and then go to that center. Add the MechanicArrives event to the EventList, at a time that equals current time + travel time to the center

7. MechGoesToDispatch: the event (only in our basic model) for when the mechanic returns to dispatch
   Handling:
   - If there are requests in the RequestQueue, get the first request and travel to the center where the request originated. Add the MechanicArrives event to the EventList, at a time that equals current time + travel time to the center
   - If there are no requests in the RequestQueue, increase the number of mechanics at dispatch by 1
8. CallVan: the event for when a mechanic at a business center calls dispatch and requests a van loaded with a copier
   Attributes: business center (of the request)
   Handling:
   - Create a new object of class request, called copier_request, at the current business center and time (with time responded = 0 initially)
   - If the dispatch center has an available van, send it to the business center. Add the VanArrivesAtBusinessCenter event to the EventList, at a time that equals current time + travel time to the center
   - If there are no available vans, add the copier_request to a queue called CopierQueue
   - We now need the MechLeavesBusinessCenter event to help us decide what the mechanic should do. Add this event to the EventList at the current time
9. VanArrivesAtBc: the event for when the requested van arrives at the business center to swap out the broken copier

   Attributes: business center (of the copier request), copier request

   Handling:

   - Get the time that it takes to swap a printer at the business center using the swaptime_customer() function
   - Add the VanReturnsToDispatch event, at a time that equals current time + the time that it takes to swap a printer at the business center

10. VanReturnsToDispatch: the event for when the van, after swapping a copier at a business center, returns to dispatch

    Attributes: business center (of the copier request), copier request

    Handling:

    - Update the copier request’s time responded field, by setting it to the current time minus the copier request’s time entered
    - Add the VanSwapsCopiers event to EventList, at a time that equals current time plus the travel time from the center to dispatch

11. VanSwapsCopiers: the event for when a van returns to dispatch and loads a working copier onto itself

    Handling:

    - Use the swaptime_dispatch() function to get the time it takes to swap a copier at dispatch
    - Add the VanFinishesSwap event to the EventList, at a time that equals current time plus the time it takes to swap the copier at dispatch

12. VanFinishesSwap: the event for when a van is loaded with a working copier

    Handling:

    - If there are no requests in the CopierQueue, increase the number of vans at dispatch by 1
    - If there are requests in the CopierQueue, get the first request from the queue. Add the VanArrivesAtBusinessCenter event to EventList, at a time that equals current time plus the travel time between dispatch and the business center

Functions Used:

1. arrival_time()

   Uses information from data analysis to return the arrival rate of requests

- Get time_of_day by using the current time % 24
- If time_of_day is between 3 and 22, set *(-0.067*(time_of_day**2)) + 1.581*(time_of_day) - 1.289* as the quadratic function that gives the arrival rate
- Else, set arrival rate = 1.137
- Return Numpy's random exponential distribution with parameter = 1/ arrival rate

2. diagnosis_time()

   Uses information from data analysis to return the time needed to diagnose a problem
   - Use Numpy's random normal distribution with parameters (16, 2) if the request comes from business centers 2, 3 or 9, return 1/60 * numpy.random.normal(16, 2)
   - If the request comes from business centers other than the 3 above, return 1/60 * numpy.random.normal(22, 2)

3. repair_time()

   Uses information from data analysis to return the amount of time it takes to repair a copier
   - Use Scipy's Beta.rvs function with parameters 2.6158007964976218, 7.460626334186162, -0.0077049330279426054, 1.557915557823419, and size=1 to get repair time
   - Return repair time
4. swaptime_customer()

   Uses information from interviews to return the time it takes to swap a copier at a business center
   - Use Numpy's random.triangular distribution with parameters (10/60,15/60, 25/60) to get swap time
   - Return swap time
5. swaptime_dispatch()

   Uses information from interviews to return the time it takes to swap a copier at dispatch
   - Use Numpy's random.triangular distribution with parameters (20/60,30/60, 1) to get swap time
   - Return swap time
6. closest_available_mechanic()

Used in our variations to get the closest available mechanic when a new request enters the system

- Make a list of every business center where a mechanic was available
- If there is an available mechanic at dispatch, compare how far away the current request's business center is from dispatch and the other business centers.
- Whichever center is closest to the request's center and has a free mechanic is our closest_center
- Return this closest_center and the distance from the request to the closest_center
- If there are no available mechanics, return None for both values

7. closest_request()

   Used in our second variation (distance prioritized) to get the request closest to a mechanic when the mechanic is done repairing at a business center, and there are requests in the queue

   - Go through our RequestQueue (which, in this variation, is in fact a list) and return the closest request
   - The closest request is the request whose business center is closest to the mechanic's current business center

8. distances.py

   This was a python file we created, which stored all the distances between every business center and dispatch. We then created a pandas dataframe to store all these values, and imported this file into our jupyter notebook, so distances between all the centers could be easily accessed.

The Simulation:

- Bc_coords = A list of all the coordinates of all the business centers
- Bc_names = A list of all the names of the business centers
- Function getBC() = a function that randomly picks a business center based on the probability distribution of requests from each business center
- Mec_avg = An array that holds the average Initial Response Times of every simulation

- Van_avg = An array that holds the average Copier Delivery Time for every simulation
- Warmup time t
- Simulation time T
- Number of iterations B

Within each simulation:

- Set a number of mechanics and a number of vans
- Initialize an empty list that records the Initial Response Times for every request in the simulation
- Initialize an empty list that records the Copier Delivery Times for every delivery in the simulation
- Initialize simulation clock to be = 0
- Initialize an empty list called RequestQueue
- Initialize an empty queue called CopierQueue
- Initialize a BC_List, to which we append 10 Business Center objects with the correct names and coordinates, and no mechanics or vans
- Initialize a Dispatch Center with the correct coordinates, and the number of vans and mechanics
- Initialize a priority queue called EventList, with entries sorted by a custom method **lt()**

The method lt() exists in every event sub-class, takes in (self, other), compares the time of the self event object to the other event object, and is used to sort our priority queue EventList so events are executed in the correct order.

- Create an object of class NewRequest, and add it to our EventList
- While we are within our time T, get the first element from EventList, and handle it
- Calculate mech_average, the average Initial Response Time for the simulation and copier_times_average, the average Copier Delivery Time for the simulation.

<u>Data Analysis:</u>

In order to determine the distribution of repair times, we first looked at a histogram of the data and tested the fit of a few different distributions: Beta, Gamma, Lognormal, Exponential and Rayleigh. Using a statistical analysis tool called the Kolmogorov-Smirnov tests we identified the Beta and Rayleigh distributions to be the most effective at fitting the data. The p-values for these two distributions respectively were 0.6979 and 0.5268. We decided to use the Beta distribution to depict the on-site repair times due to the higher p-value with parameters (alpha=2.6158, beta=7.4606, loc=-0.46229, scale=93.4751).

Utilizing the same KS test as before, we see that Group 1 is best modeled by a normal distribution with parameters (mu=22.14603, sigma=5.34413). Testing this fit gave us a p-value of 0.8547. Group 2 is best fit with a normal distribution using the parameters (mu=16.13003, sigma=2.89199). Testing this fit gave a p-value of 0.84999. Grouping by busines center allows us to more accurately represent the initial diagnosis times without looking at each business center individually.

After doing thorough testing on the number of arrivals between the hours of 3 am and 10 pm, we discovered that they resemble a non-stationary Poisson process with the number of requests per hour depending on the time of day. The arrival rate is therefore characterized by the quadratic function $□(□) = -0.067□^2 + 1.581□ - 1.289$ where i represents the hour of the day.

<u>Queuing Theory Verification:</u>

To simplify our model created two variations of our base model--in one model copiers were always repaired and in the other they were always replaced. We ran 100 replications of each variation for 300 hours, with 150 hours of warm-up time. We increased the warm-up time because capturing the true steady-state behavior was particularly important, since we were comparing our model to queueing theory predictions. The model was also amended so that all of the travel and service times took an exponentially distributed amount of time. This allowed us to model the full repair or replacement times (including travel time) as approximately gamma distributed, since the sum of random exponentially distributed variables is approximately gamma. It also allowed us to model the arrivals of copier-replacement requests as a poisson process, since applying exponentially distributed service times to a poisson arrival process creates an independent poisson process, provided the servers can keep up with demand. We routed all requests to the same business center to ensure that the travel times and initial diagnosis times all had the same distribution, since having multiple distributions at play could make the model result different from the queueing theory prediction. We set the number of available mechanics in each model equal to 2, the number of available vans equal to 2, and the arrival rate equal to 1/hr.

In the repair-only model, our predicted value for the average time in the queue was 106 minutes, meaning that the predicted value for the average response time was (avg. time in queue) + (avg. travel time) = 106 + (25km to BC) / (1km/min speed) = 131 minutes. We obtained this value by modeling the system as a M/G/c queue with the service time equal to (total travel time both ways)+(diagnosis time)+(repair time) which had an average value of (2*25) + (22.15) + (0.39667*60) = 96 minutes. The variance was then $(2*25)^2 + (22.15)^2 + (60*0.39667)^2 = 2,307$ minutes. Our 95% confidence interval for the average response time was [110, 134], which contained the predicted value 131.

We decided that there was no way to adjust the replacement-only model to perfectly match up with a common queueing model, since the swaps occurred immediately after the vans returned to dispatch, so requests only sometimes had to wait for it. Instead, we looked at two relevant queueing theory predictions and looked for our model to be in between the two values. The first prediction is the average response time if the copier requests never had to wait for the dispatch copier swap. In this scenario, the mean service time is 1.444 hours and its variance is 0.720679 hours, resulting in an average response time of 89 minutes. We found this number, again, by adding the average time in queue to the travel time. The second prediction is the average response time if the copier requests always had to wait for the dispatch copier swap. In this scenario, the mean service time is 1.7222 hours, and its variance is 0.7978395 hours, giving a predicted average response time of 213 minutes. The model prediction does, in fact, lie between these two numbers, with mean 150 minutes and the confidence interval was [128, 171].

Sensitivity Analysis:

In order to perform sensitivity analysis on the model, we decided to write a script that- within the actual simulation- runs code that loops through all the perturbation values. Inside of our main block of code, we created functions that would output the arrival times, repair times, and diagnosis times. When we performed sensitivity analysis, we would comment out the function in the main block of code where all the events are defined, and instead redefine that function within the block of code that actually triggers the beginning of our simulation. We create a linspace vector in python and set the num value to 50, so we loop over 50 perturbed values of the parameters within the range we defined. We run the simulation for each perturbed value for 150 hours in which 50 hours were warm-up time, and we run 50 replications of each simulation for each of the perturbed values. We calculate what the average initial response time and copier delivery time for each new perturbed parameter value. If the distribution has more than one parameter, we run the simulation for each parameter separately but with the same amount of runtime and replications in order to analyze the effect of each individual parameter on the performance measures.

Our .zip submission includes our 3 models, all of our sensitivity analysis models- labeled clearly to indicate that they're for sensitivity analysis, and our model verification file (labeled as such).