Operational Plan for the Amazonoff Company

52086

A report submitted The Amazonoff Company

April 25, 2023

Contents

Chapter 1

Executive Summary

Objective: This report's objective is to document the modelling and simulation completed on behalf of the Amazonoff Company. This report also outlines changes that we feel could improve operating efficiency, whilst minimising costs.

Findings: Section 2.1 describes the cost-minimising assignment of customers to facilities so that we satisfy each customer's demand. We find the optimal operating cost to be 210.64, even after we account for the capacity of each facility. We also suggest methods which allow the company to balance loads between facilities. In Section 2.2 we construct optimal routes for the delivery vehicles to follow. We show that the two processes discussed above can be modelled separately, and therefore are easily actionable. Our routing implementation offers an additional cost of just 88.86, which remains the same using just one delivery vehicle or three. We believe it is important to consider uncertainty in our research and in Section 2.3, we offer insights into how we can factor this into the internal running of the depot. Finally, in Section 2.4, we simulate the next 12 months for your insurance subsidiary. It should be noted that we estimate the probability of legal issues for the company to be roughly 38%. To help remedy this, we further estimate the value of your capital holdings to a high degree of certainty and find it to be in the region of 32585.55. We, therefore, suggest a small increase in the monthly rate you charge customers, or sourcing a capital injection before the next year begins. In this section, we also offer a method by which we can be more certain about our estimate.

Conclusion: Following our research, if our suggestions are implemented, we find a short-term operations cost of 299.5. In the longer term, we suggest following the advice laid out in the rest of the report so as to avoid legal issues.

Chapter 2

Management Report

In this chapter, we lay out a more detailed analysis of your operations, and offer distinct guidelines on how to fulfil your warehousing, routing and e-commerce requirements.

2.1 Warehousing

The Amazonoff Company operates in the space of a 21×21 grid, in which exist the entire customer base and potential facilities. We provide a map for reference in Figure 2.1 We suggest only opening a subset of these facilities, as we give a satisfactory assignment of all 60 customers with just facilities 2 to 15, allowing the firm to save expenditure on unnecessary rent. Outlined in Table 2.1 are the assignments, and their respective facilities we consider to be optimal, and we refer the reader to Appendices A.2.1 and A.2.2 for a technical discussion of our modelling process.

We find that even upon factoring in the demands of each client, the facilities are easily able to hold the required stock, and as such our allocation remains optimal. It may seem odd to the reader we suggest opening so many facilities, but upon consideration of the costs for opening facilities, the motivation becomes clear. As the index of the facility increases, the cost to open it decreases exponentially, and as such it incurs a minimal cost to open an extra facility. With this in hand, it makes sense to open many facilities, so we can truly minimise the distance between them and the clients. Upon review of Figure 2.1, we can visualise this idea.

In Table 2.1, we see the demand faced by each facility. It is clear that the distribution is extremely unbalanced, with facility 8 facing 145 units, and facility 14 facing just 14. As such, we refer the reader to Appendix A.2.3 for a modelling approach that allows the firm to control this difference.

We calculate the cost of this approach to be 210.64 if implemented correctly.

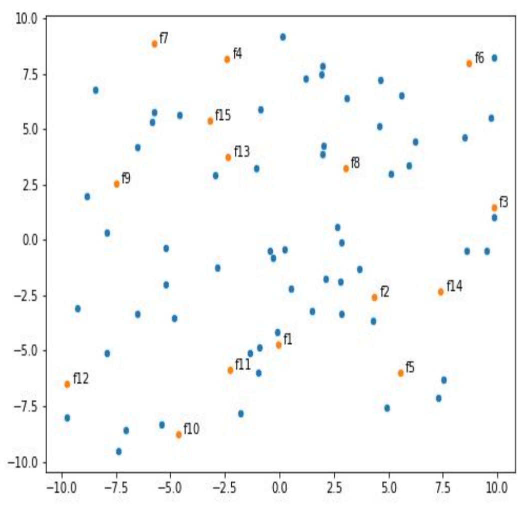

Figure 2.1: Map of facilities (orange) and customers (blue)

Facility	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Customers Assigned	10	14	40	4	5	42	3	1	17	8	26	12	45	7
	27	21	46	58	19		6	2	25	23	34	18		13
	30		47	60	22		9	28	39	24	43	29		15
	33		50		31		11	57		32	59			38
	37				49		15			36				
	41						20			53				
	44						35			54				
	48						52			56				
	51						55							

Table 2.1: Assignment of customers to facilities

Facility	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Demand Faced	104	34	50	47	54	17	145	35	44	91	53	52	14	65
Capacity (100×)	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}

Table 2.2: Demands faced by each facility

2.1.1 Further Considerations

It is important to consider limitations in our work. Firstly, we have modelled this scenario as an Integer Program, and as such, we were unable to perform sensitivity analysis. Sensitivity analysis would allow us to more accurately understand how a change in the environment would affect our suggestions. For example, if there was a change in the demand of a given customer, which is likely, we could analyse how that would affect the assignment.

This speaks to a wider limitation of our model, as in this particular scenario, we have not considered uncertainty at all. It is easy to imagine a situation in which demands are not so rigid but despite this, we can still be confident in our result given the sheer size of the facilities.

2.2 Vehicle Routing

We next tackle the routing operation. The company has 25 *composite* clients whose deliveries are to be prioritised. There is a single depot and the delivery vehicles have no capacity limit. In our modelling, we considered using one or three delivery vehicles and found that the total cost of using either is the same, that being 88.86. This was due to the assumption of our model that it was not necessary to use all three vehicles, just that it was available if necessary. We found however that it was still optimal to use just one either way. This is to be expected since all routes must return to the same depot, and if we used more vehicles, we would pay unnecessary costs since there may be some overlap on the routes. In Figure 2.2 we outline the optimal route, where the depot is highlighted in red and labelled `d1`. Each composite client is then labelled by their index and prefixed with a `c`. For a technical discussion of our models, please refer to Appendices B.2.1 and B.2.2

2.2.1 Further Considerations

A key insight of our result is that the route and cost are the same using one vehicle or three since it is always optimal to just use one, under our model. This result isn't very robust, however, since upon factoring in capacity limits for the vehicles, it is more likely that it is necessary to use multiple vehicles. This is true also if we wanted to factor in distance limits for the vehicles. As outlined in Appendix B.2.2 however, it would not be hard to expand our model to incorporate this due to the way we have modelled the problem, namely as a Shortest Path Integer Program. Simply altering and adding some parameters in the constraints, would allow us to model a sturdier program.

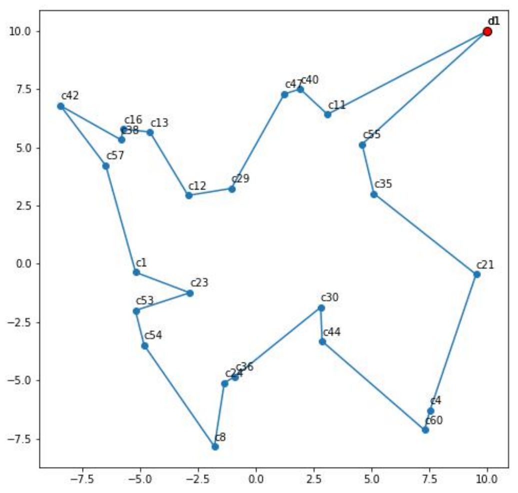

Figure 2.2: Suggested delivery route

2.3 Internal Depot Transportation with Uncertainty

Given the nature of the company, it is important to factor in uncertainty when modelling, specifically in the internal depot operations. In this section, we propose two models that allow the firm varying degrees of control. The first is a *robust* model, meaning regardless of the dimensions of each item, there will always be some way to pack the electric vehicle. The second allows the firm some control over the likelihood that the vehicle cannot be packed. We, therefore, label this model as the *non-robust* formulation. We briefly discuss both models in the following subsections, but for a more detailed description, we refer the reader to Appendices C.1 and C.2.

2.3.1 Robust Formulation

A key assumption of the robust model is that we can, prior to production, garner some information about the maximum volume and weight of each item. With this data in hand, we simply consider the problem using only these maximum weights. In doing this, we have essentially removed the uncertainty from the problem. Despite this, we can imagine a situation where the volume and weight of a given item can vary drastically, so if we pack an item using only its maximum weight, then there is a good chance there is a lot of empty space on the vehicle. As such, we are not effectively maximising the profit garnered on each vehicle. To account for this loss, we recommend implementing the second model, which we now turn our attention to.

2.3.2 Non-Robust Formulation

Our next model allows the firm to retain some control over the uncertainty. More specifically, we add a notion of risk control. Once again we assume we can garner some information about the volume and weight of each item. In this case, we want to consider the average volume and weight, along with the variation of these estimates. We suggest performing some data analysis on the items to acquire these values. Then, we model the problem as a standard Linear Program using the expected dimensions. For this situation, however, we require a further parameter to be set, that being some tolerated level of risk, which refers to the probability that we cannot fit the items on the vehicle. Using this approach, we better utilise the space on each vehicle, since we are making use of an estimate instead of a maximum. Of course, there is a small chance that the allocation of goods will not fit on the vehicle, but this can be remedied with the appropriate setting of the risk level.

The choice between these two approaches will depend largely on the needs of the company. If the consequences of allocating too many items to one vehicle are large, we recommend a robust approach. Otherwise, we suggest the second model since the profit per vehicle would certainly be higher. It is our opinion that the firm would be better off implementing the second.

2.3.3 Further Considerations

Despite the fact we have modelled uncertainty, unlike our previous two models, it is still possible to imagine ways in which we can improve our approach. Of course, the quality of both models heavily depends on the volume of good data we can source, and more data will always be better. The better data we can source, the more accurately would be able to tune our level of risk.

Additionally, both of our models are variations of Linear Programs that factor in uncertainty in some way. We could imagine using a more complex modelling approach to solve this problem, for example, by using a simulation approach as in Section 2.4. If we were to run simulations, we could better estimate values such as the total expected volume and weight of the vehicle, and this could inform investments such as expanding the capacity of the vehicle.

2.4 Product Specific Insurance

We now bring our attention to the insurance subsidiary of the company. In particular, we wish to simulate the next 12 months of your operations and deduce meaningful estimates of key values, so as to inform your decision-making in the next year. The key values we considered are the likelihood of legal issues for the firm, and the

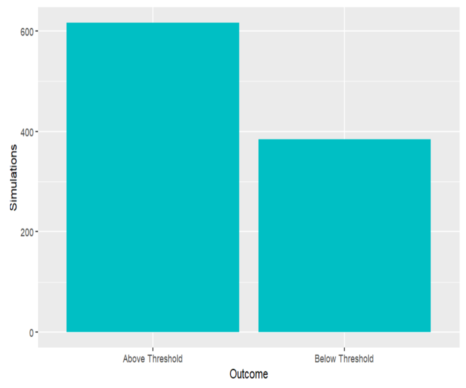

Figure 2.3: Distribution of outcomes (1000 simulations)

expected capital stock at the end of the term.

Based on your data, we run the simulation with a starting capital stock of $50,000$ and a fixed monthly incoming fee of 300. By then considering the entry and leaving rates of your service, and the rate at which customers lodge claims, we can generate estimates of these values in the long run.

2.4.1 Liklehood of Legal Issues

We first wish to estimate the probability that your firm will encounter legal issues. After running the simulation 1000 times, we return with the distribution laid out in Figure 2.3. We immediately see how even the distribution is, and precisely we found the probability of legal issues to be 0.384. We deem this figure to be far too high since it implies in the next 3 years of operations, you are very likely to encounter legal trouble. To counteract this we suggest raising the monthly fee. In Figure 2.4 we depict the results of two further simulations in which we raise the monthly fee for the service to $M = 350$ and $M = 400$. We see that with an increase to 350, the failure decreases by more than half, and in the case of 400 it halves again. Specifically, these probabilities are 0.189, and 0.069 respectively.

There is also the option of sourcing a capital injection, and we ran further experiments with this idea. However, we found that an increase of around $20,000$ in initial capital was necessary to give us comparable results to the monthly fee increases, and as such, we suggest an increase in only the monthly fee to decrease the chances of legal issues.

2.4.2 Estimation of Capital Stock

The next section is dedicated to the estimation of your capital stock after a 12-month period, which is only more justified given the analysis in the previous section. If we can gain a good estimate, we can provide a more accurate value for a suggested monthly fee.

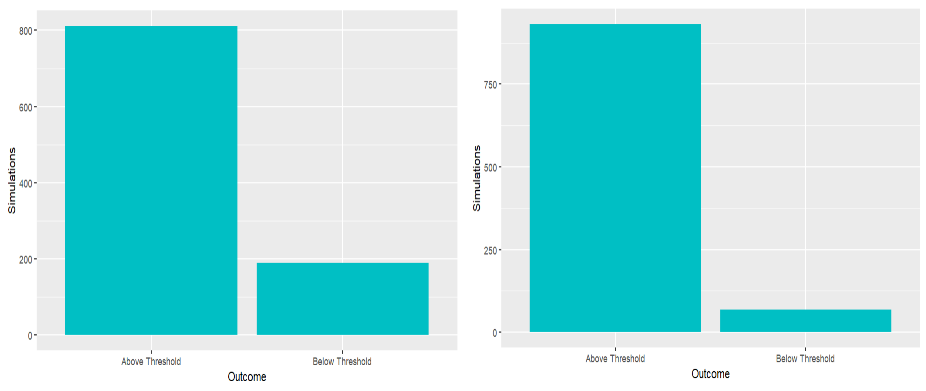

Figure 2.4: Distribution of outcomes with M = 350, 400 (1000 Simulations)

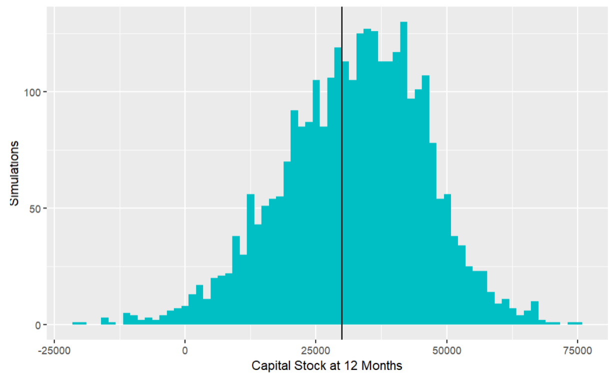

Figure 2.5: Distribution of capital stock after 12 months (3000 Simulations)

We aim to find an estimate for the capital stock after 12 months, which we denote C_{12}, that we can be 90% sure is within 500 of the true value. To build a better picture, we ran the simulation for 2000 more iterations than in the previous section, and at iteration 2034, we find 32585.55 as our estimator of capital stock. We also have the standard deviation of this estimate to be 303.86, and with this, we can give an interval to which we are 95% sure the true value lies within, and we calculate this to be $C_{12} \pm 595.56$. A detailed description of the simulation process, and how we arrived at our estimates, can be found in Appendices D.1 and D.2.

Upon reviewing Figure 2.5, we observe there is a massive variance in our estimate. In some cases, we finish with negative capital stock. This variance is due to the insurance payout formula, which can sometimes provide extortionately high payouts. In Appendix D.3, we describe measures to reduce this variance in our estimate, by considering the quantity of insurance claims we observe.

We have now presented a clear image of the 12-month forecast for your insurance subsidiary, and with this, we would like to recommend a specific monthly service charge, with the intent of reducing the likelihood you encounter legal issues. Using our analysis laid out in this section, we suggest $M = 415$, which will allow the firm to avoid legal issues 95% of the time.

2.4.3 Further Considerations

We could improve our estimates further still if we simply ran more simulations. This would increase the confidence to which we can estimate C_{12} and the likelihood of legal trouble. However, due to the certainty of our estimates, there are not many further insights we could gather from this.

Another way we could build upon our model is to allow customers to rejoin the service, which we have not modelled in our analysis. This makes our model more realistic since it's easy to imagine a situation in which a customer does so. We hypothesise that factoring this in would see added revenue at the end of each simulation, and thus decrease the estimated likelihood of legal issues.

Finally, our model is based on the idea that each item pays out the same, based on a uniform distribution. This could be more nuanced since we could assign a different probability function to each individual item type. This would no doubt garner actionable suggestions such as charging customers different monthly fees depending on what items they own. Furthermore, it may go some way to reduce the variance in our estimate of C_{12} as we would have a better idea about the size of each payout.

Appendix A

Warehousing Methodology

Contained in this appendix is a detailed, technical report of the methodology applied in the warehousing scenario.

A.1 Data Generation

A.1.1 Grid Generation

`Associated Files: grid_generation.Rmd`

Before we carried out our modelling, it was necessary to generate certain data that we could feed into our models. First, we generated uniformly random coordinates of each of the 60 clients and 15 facilities. Our method is outlined in Algorithm 1, and implemented in `grid_generation.Rmd`

We first let n_f and n_c be the number of facilities and customers respectively. Then for each axis, we randomly generate the required amount of coordinates, and transform them to be in a $[-10, 10]$ interval. Using the loop on lines 2 and 4 as an example, we multiply each random coordinate on the y axis of the facilities by 20, to get n_f random variables between $[0, 20]$. Then we subtract 10 from these values, to arrive at n_f values between $[-10, 10]$. The x axis is repeated like this, and so are the axes for the customers.

We are left with the coordinates as required, and after some wrangling to get into the right format, we are left with Table A.1 for customer locations, and A.2 for facility locations.

It is important to note that the R function we used to generate the initial values (`runif(n)`) gives the variables a high degree of accuracy. As such, we round these values to 2 decimal places.

A.1.2 Demand Generation

`Associated Files: demand_generation.Rmd`

Algorithm 1: Coordinate Generation

```
   Data: $n_f = 15, n_c = 60$
   Result: Coordinates on a $[-10, 10] \times [-10, 10]$ grid
1  $f_y \leftarrow n_f \times 1$ vector of random uniform(0,1) variables;
2  for $f \in f_y$ do
3      $f \leftarrow 20 \cdot f$ ;
4      $f \leftarrow f - 10$ ;
5  end
6  $f_x \leftarrow n_f \times 1$ vector of random uniform(0,1) variables;
7  for $f \in f_x$ do
8      $f \leftarrow 20 \cdot f$ ;
9      $f \leftarrow f - 10$ ;
10 end
11 $c_y \leftarrow c_f \times 1$ vector of random uniform(0,1) variables;
12 for $c \in c_y$ do
13     $c \leftarrow 20 \cdot c$ ;
14     $c \leftarrow c - 10$ ;
15 end
16 $c_x \leftarrow c_f \times 1$ vector of random uniform(0,1) variables;
17 for $c \in c_y$ do
18     $c \leftarrow 20 \cdot c$ ;
19     $c \leftarrow c - 10$ ;
20 end
```

Customers	**Coordinates**	**Customers**	**Coordinates**	**Customers**	**Coordinates**
c1	(-5.21, -0.36)	**c26**	(-9.74, -8.01)	**c51**	(3.66, -1.31)
c2	(-8.82, 1.99)	**c27**	(4.31, -3.67)	**c52**	(2.03, 4.25)
c3	(2.85, -0.13)	**c28**	(-7.94, 0.37)	**c53**	(-5.22, -2.0)
c4	(7.53, -6.28)	**c29**	(-1.07, 3.24)	**c54**	(-4.84, -3.49)
c5	(5.58, 6.55)	**c30**	(2.8, -1.86)	**c55**	(4.59, 5.14)
c6	(5.95, 3.37)	**c31**	(9.84, 8.26)	**c56**	(-0.95, -5.95)
c7	(-0.89, 5.88)	**c32**	(-0.09, -4.13)	**c57**	(-6.5, 4.22)
c8	(-1.8, -7.84)	**c33**	(-0.31, -0.82)	**c58**	(4.93, -7.57)
c9	(6.22, 4.47)	**c34**	(-6.53, -3.35)	**c59**	(-7.9, -5.09)
c10	(2.1, -1.77)	**c35**	(5.1, 3.02)	**c60**	(7.29, -7.13)
c11	(3.09, 6.42)	**c36**	(-0.92, -4.84)		
c12	(-2.94, 2.94)	**c37**	(0.22, -0.43)		
c13	(-4.59, 5.66)	**c38**	(-5.85, 5.33)		
c14	(9.85, 1.06)	**c39**	(-5.43, -8.32)		
c15	(2.67, 0.59)	**c40**	(1.91, 7.51)		
c16	(-5.74, 5.79)	**c41**	(1.5, -3.22)		
c17	(-7.41, -9.53)	**c42**	(-8.46, 6.79)		
c18	(-0.44, -0.46)	**c43**	(-9.29, -3.07)		
c19	(8.48, 4.65)	**c44**	(2.86, -3.32)		
c20	(1.98, 3.85)	**c45**	(8.57, -0.47)		
c21	(9.52, -0.45)	**c46**	(1.96, 7.84)		
c22	(4.64, 7.22)	**c47**	(1.22, 7.29)		
c23	(-2.87, -1.24)	**c48**	(0.52, -2.2)		
c24	(-1.37, -5.1)	**c49**	(9.7, 5.55)		
c25	(-7.04, -8.59)	**c50**	(0.15, 9.21)		

Table A.1: Customer Coordinates

Facility	**Coordinates**
f1	(-0.05, -4.69)
f2	(4.35, -2.56)
f3	(9.84, 1.46)
f4	(-2.4, 8.16)
f5	(5.55, -5.97)
f6	(8.69, 7.97)
f7	(-5.76, 8.89)
f8	(3.03, 3.22)
f9	(-7.49, 2.58)
f10	(-4.66, -8.76)
f11	(-2.28, -5.88)
f12	(-9.73, -6.47)
f13	(-2.35, 3.74)
f14	(7.39, -2.32)
f15	(-3.19, 5.4)

Table A.2: Facility Coordinates

Upon expansion of our model, it was necessary to generate the demand of each of the 60 clients. We were given that each demand is between 1 and 25, and was uniformly distributed. The method for calculating these random variables is outlined in Algorithm 2 and implemented in `demand_generation.Rmd`

Algorithm 2: Demand Generation

```
  Data: Set of customers $C$
  Result: Set of demands for each customer $c$
1 $U \leftarrow 60 \times 1$ vector of random uniform(0,1) variables ;
2 for $u \in U$ do
3     $u \leftarrow 25 \cdot u$ ;
4     $u \leftarrow \text{upper}(u)$ ;
5 end
6 Let each index of $U$ represent the corresponding customer in $C$ ;
```

In the first line, we initialise a 60×1 vector of random uniform(0,1) variables. Then the loop on lines 2-4 firstly multiplies those values by 25, to give us 25 random variables between 0 and 25, and then returns the upper integer of that value, which gives us 25 random integers between 1 and 25, as required. The exact demands are described in Table A.3.

Customers	**Demand**	**Customers**	**Demand**	**Customers**	**Demand**
c1	7	**c26**	10	**c51**	12
c2	10	**c27**	1	**c52**	22
c3	15	**c28**	10	**c53**	11
c4	23	**c29**	22	**c54**	7
c5	6	**c30**	9	**c55**	2
c6	23	**c31**	13	**c56**	3
c7	24	**c32**	15	**c57**	8
c8	17	**c33**	13	**c58**	13
c9	16	**c34**	5	**c59**	17
c10	2	**c35**	21	**c60**	11
c11	6	**c36**	17		
c12	5	**c37**	20		
c13	18	**c38**	3		
c14	10	**c39**	19		
c15	20	**c40**	11		
c16	13	**c41**	21		
c17	18	**c42**	17		
c18	25	**c43**	20		
c19	10	**c44**	14		
c20	20	**c45**	14		
c21	24	**c46**	20		
c22	6	**c47**	1		
c23	17	**c48**	12		
c24	4	**c49**	19		
c25	7	**c50**	18		

Table A.3: Demands for each customer

A.2 Modelling

A.2.1 Base-Line Model

`Associated Files: model(1.1).mod, f_loc(1.1).dat, c_loc(1.1).dat, data(1.1).dat, run(1.1).run, out(1.1).txt`

With our data in hand, we can now move to model our problems, the first of which can be modelled in the form of an integer program. The formal definition of our model is outlined below:

Sets

C	set of Customers
F	set of Facilities

Parameters

c_{ij}	$(i,j) \in C \times F$	cost of assigning customer i to facility j
k_f	$f \in F$	cost of opening facility f

Variables

$x_{ij} \in \{0,1\}$	$(i,j) \in C \times F$	Assigning customer i to facility j
$b_f \in \{0,1\}$	$f \in F$	Opening facility f

Objective

$$\min \sum_{(i,j)\in C\times F} c_{ij}x_{ij} + \sum_{f\in F} k_f b_f \tag{A.1}$$

Constraints

$$\sum_{j\in F} x_{ij} = 1 \qquad \forall i \in C \tag{A.2}$$

$$x_{ij} \le b_j \qquad \forall (i,j) \in C \times F \tag{A.3}$$

We first define sets C and F to denote customers and facilities respectively. For our model parameters, we define c_{ij} to denote the cost of assigning customer i to facility j, which is given by the l_1 norm between the two corresponding coordinates. Here we also define k_f as the cost of opening facility f, which is given explicitly by $100 \cdot 3^{-f}$. For decision variables we let x_{ij} be a binary variable that is equal to 1 if we assign customer i to j. Similarly, b_f is another binary variable that is equal to 1 if we open facility f.

We then wish to minimise the cost of the operation. We model the objective function (A.1) as the cost of assigning customer i to j, multiplied by the decision variable. We then add this to the cost of opening the facilities, constructed similarly. The problem is constrained by (A.2), which ensures that every customer is assigned to at least one facility, and (A.3), which ensures that only facilities that are open can have customers assigned to them.

We find that solving this model gives an objective value of 210.64, and an assignment as laid out in Section 2.1.

A.2.2 Capacitated Model

`Associated Files: model(1.2).mod, f_loc(1.2).dat, c_loc(1.2).dat, data(1.2).dat, run(1.2).run, out(1.2).txt`

We then wish to expand the model, by factoring in the demands we generated earlier. We do this first by adding two parameters. d_c defines the demand of customer c, and l_f defines the capacity limit of facility f. The parameter l_f is more specifically given as $100 \cdot 2^f$. We must then add one more constraint, namely:

$$\sum_{i \in C} d_i x_{ij} \leq l_j b_j \qquad \forall j \in F \tag{A.4}$$

This constraint (A.4) ensures that, summed over all customers assigned to facility f, the demand must not exceed the capacity of the facility. The binary variables here ensure the constraint holds only for open facilities. Other than this, our model remains unchanged from the previous section. We find that the solution to this model is the same as in the previous section, 210.64.

A.2.3 Balanced Loads

We are now presented with the situation in which the firm wishes to retain some control over the demands faced by each facility. In Section 2.1, we learn that the optimal solution is indeed unbalanced, and there are numerous reasons why we may want to control this, one being to protect against shocks in demand.

For this, we let there be some L that defines the greatest allowable difference between the smallest demand faced, and the greatest. We define two new positive decision variables, u_{min} and u_{max}, to describe these values specifically. Finally, we add 3 more constraints to our current model, which are outlined below:

$$\sum_{i \in C} d_i x_{ij} \geq u_{min} \qquad \forall j \in F \tag{A.5}$$

$$\sum_{i \in C} d_i x_{ij} \leq u_{max} \qquad \forall j \in F \tag{A.6}$$

$$u_{max} - u_{min} \leq L \tag{A.7}$$

Constraints (A.5) and (A.6) ensure the demands faced do not break the bounds of the limits described by u_{min} and u_{max}, and then constraint (A.7) ensures these two limits do not exceed our predefined limit L. This proposed model forces the largest and smallest demands to adhere to the limit L in any solution, as required.

Appendix B

Vehicle Routing Methodology

Contained in this appendix is a detailed, technical report of the methodology applied in the vehicle routing scenario.

B.1 Data Generation

B.1.1 Composite Client Generation

`Associated Files: client_generation.Rmd`

As in the previous problem, it was first necessary to generate data to be used in our modelling. In this case, it was necessary to generate 25 composite clients, to who we prioritise the delivery service. We outline the process used below in Algorithm 3, and the implementation can be found in `client_generation.Rmd`:

Algorithm 3: Composite Client Generation

```
Result: Set of composite clients $C_{comp}$
1 while all $c \in C_{comp}$ are not unique do
2     $C_{comp} \leftarrow 25 \times 1$ vector of random uniform(0,1) variables ;
3     for $c \in C_{comp}$ do
4         $c \leftarrow 60 \cdot c$ ;
5         $c \leftarrow \text{upper}(c)$ ;
6     end
7 end
```

Our process is very similar to that laid out in Algorithm 2, except we have added a loop enclosing the whole process that ensures the list we produce contains entirely unique numbers, otherwise, we may not have 25 composite customers. The other change is multiplying the random variables by 60, since there are 60 potential customers. The exact list of composite clients is described in Table B.1, along with the optimal route.

B.2 Modelling

B.2.1 Base-Line Model

`Associated Files: model(2.1).mod, data(2.1).dat, locations(2.1).dat, run(2.1).run, out(2.1).txt`

Now we have the set of composite clients, we can move to model the problem. In the first situation, we essentially have an instance of the Travelling Salesman Problem. All demand is satisfied by the depot, and the vehicle has no capacity limit, therefore, we aim simply to minimise the travelling cost, which here is given as the l_2 norm between two nodes. A definition of our model is given below, and the implementation can be found in `model(2.1).mod`:

Sets

C_{comp}	set of Composite Customers
D	set of Depots
$V = C_{comp} \cup D$	set of Nodes

Parameters

c_{ij}	$(i,j) \in V$	cost of travelling from node i to node j
N		The cardinality of set V

Variables

$x_{ij} \in \{0,1\}$	$(i,j) \in V$	Travelling from i to facility j on a given route
$u_i \in \mathbb{Z}_+$	$i \in V$	Indication of where node i is on the tour (MTZ Variable)

Objective

$$\min \sum_{i \in V} \sum_{j \neq i \in V} c_{ij} x_{ij} \tag{B.1}$$

Constraints

$$\sum_{j \neq i \in V} x_{ij} = 1 \qquad \forall i \in V \tag{B.2}$$

$$\sum_{i \neq j \in V} x_{ij} = 1 \qquad \forall j \in V \tag{B.3}$$

$$u_i - u_j + N x_{ij} \leq N - 1 \qquad \forall i, j \in V, \quad \text{where} \quad j \neq i, 1. \tag{B.4}$$

We first define sets C_{comp} and D to represent the composite customers generated, and the depot set, i.e $D = \{d_1\}$. We then define the union of these two sets to be V, the set of all nodes. For parameters, we then have c_{ij} which represents the cost as defined above from travelling from i to j, and N is the number of nodes. We now define two decision variables: x_{ij} is a binary variable indicating whether a given i, j connection is on the route; and u_i is the sub-tour elimination constraint, which forces the nodes to appear in some order on the route. In the objective function (B.1) we define the objective function, that is, the cost of travelling from node i to j summed over the route. The constraints (B.2) and (B.3) ensure only one route goes in and out of every node, and (B.4) enforces the ordering of u_i. It is also important to note that all demand is inherently satisfied here, due to the vehicle's unlimited capacity.

The result of our model will be a route connecting all of our composite clients, minimising the distance cost of the route. We laid out a visual solution in Figure 2.2, but in Table B.1, we give the precise routes. This solution accrues a cost of 88.86.

B.2.2 Multiple Vehicles

`Associated Files: model(2.2).mod, data(2.2).dat, locations(2.2).dat, run(2.2).run, out(2.2).txt`

We are now presented with the opportunity of multiple vehicles. As before, all vehicles have unlimited capacity, and they all must leave and return to the original depot d_1. We can expand our previous model to facilitate this, by first adding a new set K, with cardinality $k = 3$ to represent the vehicles. Then, we add two more constraints:

$$\sum_{i \in V} x_{d_1,i} = k \tag{B.5}$$

$$\sum_{i \in V} x_{i,d_1} = k \tag{B.6}$$

These constraints (B.5) and (B.6) ensure that the 3 vehicles all start and finish with the depot d_1. Finally, we relax the constraints (B.2) and (B.3) from the previous formulation to only hold for the composite customer nodes.

Upon solving our problem, we find the same route as in Table B.1 and Figure 2.2, and therefore the same objective value. The other vehicles are not to be used. This is to be expected, since all vehicles return to the same node, and there are no restrictions on the capacity of the vehicle. We can imagine that if there were capacity restrictions, we would need to limit the number of nodes each vehicle could visit, which can be done by changing N in constraint (B.4), to reflect this. We find the objective value is the same as in Section B.2.1 since it is the identical route.

Node	**Coordinates**
d1	(10, 10)
c55	(4.59, 5.14)
c35	(5.1, 3.02)
c21	(9.52, -0.45)
c4	(7.53, -6.28)
c60	(7.29, -7.13)
c44	(2.86, -3.32)
c30	(2.8, -1.86)
c36	(-0.92, -4.84)
c24	(-1.37, -5.1)
c8	(-1.8, -7.84)
c54	(-4.84, -3.49)
c53	(-5.22, -2)
c23	(-2.87, -1.24)
c1	(-5.21, -0.36)
c57	(-6.5, 4.22)
c42	(-8.46, 6.79)
c38	(-5.85, 5.33)
c16	(-5.74, 5.79)
c13	(-4.59, 5.66)
c12	(-2.94, 2.94)
c29	(-1.07, 3.24)
c47	(1.22, 7.29)
c40	(1.91, 7.51)
c11	(3.09, 6.42)
d1	(10, 10)

Table B.1: Route Coordinates, and list of composite clients

Appendix C

Internal Depot Transportation with Uncertainty Methodology

Contained in this appendix are the modelling suggestions for the internal operations of the depot. We suggest two formulations, a robust approach, and a non-robust approach. As outlined in Section 2.3, the choice between them depends on the consequences of an infeasible solution, which in this case refers to an electric vehicle that cannot be packed. We still recommend the non-robust formulation, mainly because it offers a more effective maximisation of the expected value of the vehicle. In both of our formulations, we model the problem considering only one vehicle. If implemented these models would be solved individually every time a vehicle is packed.

C.1 Robust Formulation

The robustness of the formulation comes from the fact that no matter what size of item eventually comes out, the problem will always be feasible. The exact program is defined below:

Sets

I	set of Items

Parameters

w_i^{max}	$i \in I$	max weight of item i
v_i^{max}	$i \in I$	max volume of item i
p_i	$i \in I$	profit gained from item i
V		volume capacity of vehicle
W		weight capacity of vehicle
L		item capacity of vehicle

Variables

$x_i \geq 0$	$i \in I$	quantity of good i allotted to the vehicle

Objective

$$\max \quad \sum_{i \in I} p_i x_i \tag{C.1}$$

Constraints

$$\sum_{i \in I} w_i^{max} x_i \leq W \tag{C.2}$$

$$\sum_{i \in I} v_i^{max} x_i \leq V \tag{C.3}$$

$$\sum_{i \in I} x_i \leq L \tag{C.4}$$

We assume we can garner some information about the items i in set I. Specifically, the volume and weight of each item i, and these values are denoted v_i^{max} and w_i^{min} respectively. We then define p_i as the certain profit from each item i. Then for the vehicle, we have parameters W for the weight capacity, V for the volume capacity, and L for the item capacity. Then for decision variables, we denote positive x_i for the quantity of good i allocated for the vehicle.

In the objective function (C.1) we wish to maximise the value of the vehicle, by summing up the profits incurred from each item in the vehicle. In constraints (C.2) and (C.3) we limit the allocation by the maximum weight and volume of each item, this ensures no matter what dimensions of the item come out, there will be a feasible solution. Finally, in constraint (C.4) we limit the number of items we can put in the vehicle.

C.2 Non-Robust Formulation

We now outline the second proposed model. Once again we assume we can garner some data about the item, in this case, we want to find the mean and variance in the dimensions of each item i. This could be done by analysing previous data on each item. With this, we define a model based on these values:

Sets

I	set of Items

Parameters

w_i	$i \in I$	expected weight of item i
v_i	$i \in I$	expected volume of item i
σ^2_{wi}	$i \in I$	variance in weight of item i
σ^2_{vi}	$i \in I$	variance in volume of item i
p_i	$i \in I$	profit gained from item i
ρ		Some tolerated variance threshold
V		volume capacity of vehicle
W		weight capacity of vehicle
L		item capacity of vehicle

Variables

$x_i \geq 0$	$i \in I$	quantity of good i allotted to the vehicle

Objective

$$\max \quad \sum_{i \in I} p_i x_i \tag{C.1}$$

Constraints

$$\sum_{i \in I} w_i x_i \leq W \tag{C.2}$$

$$\sum_{i \in I} v_i x_i \leq V \tag{C.3}$$

$$\sum_{i \in I} x_i \leq L \tag{C.4}$$

$$\sum_{i \in I} x_i \left(\sigma^2_{wi} + \sigma^2_{vi}\right) \leq \rho \tag{C.5}$$

Here we define V, W, L as before, and we work in the mean and variances of the volumes and weight of each item i. A key addition to this model is the ρ value. This can be thought of as a level of risk tolerance.

We define the objective function (C.1) as before and the same with constraints (C.2), (C.3) and (C.4), except we now use the expected value of each dimension instead of the maximum values. We then use ρ to set some acceptable level of variance for the items in the vehicle. If we were more risk-tolerant, we would use a higher ρ, and see a more effective maximisation of space. However, we would increase the probability that once the dimensions are realised, the problem will be infeasible. Here we see a trade-off between the accepted level of risk and the quality of the solution.

It is important to consider the level at which we set ρ. In order to set the right level, we could carry out simulations to see what gives us the best balance between profitability and infeasibility. We could also run Monte-Carlo Markov-Chain simulations to better determine the expected weight and volume, and their variances, of each item.

Appendix D

Product Specific Insurance Methodology

Contained in this appendix is a technical review of the simulations carried out for the insurance subsidiary. We focus on the determination of P, the probability of legal issues, and C_{12}, an estimate of the capital stock at the end of a 12-month period.

D.1 Base-Line Model

`Associated Files: baseline.Rmd`

We first wish to simulate operations to determine P. For this, we run the simulation $K = 1000$ times and then divide the number of times we encounter issues by K.

We now build the simulation model. For this, we must first define two subroutines for the arrival and departure rates of customers to the service, which are a homogeneous (Algorithm 4) and non-homogeneous (Algorithm 5) Poisson process respectively.

```
Algorithm 4: Homogeneous Poisson Process
  Data: Current time $s$, process intensity $\lambda$
  Result: Next event time $t$ in Poisson Process
1 $U \leftarrow$ Random uniform(0,1) variable;
2 $t \leftarrow s - (\frac{1}{\lambda} \cdot \log(U)$;
```

In Algorithm 5 we use $\mu(s) = \frac{1}{12+s}$ for our simulation. We also define some limit μ to which we divide the rate by, which we set to some upper bound of the rate $\mu(s)$. We also must factor in the rate at which the customers make a claim, which also follows a Homogeneous Poisson Process for each customer. We again use Algorithm 4 but with a different rate, namely $\frac{\alpha}{12}$, since our data only gives us a yearly rate.

```
Algorithm 5: Non-Homogeneous Poisson Process
  Data: Current time $s$, limit $\mu$, rate $\mu(s)$
  Result: Next event time $t$ in Non-Homogeneous Poisson Process
1 $U_1 \leftarrow$ Random uniform(0,1) variable;
2 $t \leftarrow s - (\frac{1}{\mu} \cdot \log(U))$;
3 $U_2 \leftarrow$ Random uniform(0,1) variable;
4 if $U_2 \geq \frac{\mu(t)}{\mu}$ then
5     Go to step 1
6 end
```

From here we can define our variables, events, event lists and output variables. For variables we let t be the current time, n_A be the number of arrivals by time t. We also let *cost* be the current cost accrued through claims, and *rev* denotes the revenue generated from the monthly subscriptions. To track this monthly fee, we keep a list *months* that tracks what months have elapsed. For the state variable, we let n_0 be the number of customers currently in the service.

Now for the event variables, we let t_A and t_D denote the times after t where the next arrival and departure occurs. We implement t_D as a list since the process by which customers leave is defined with each customer. It is also required each customer has a unique claim time, and thus we define t_C as a list, indexed by the customer, that holds the times after t until the next claim.

Finally, we collect a binary output variable B that is equal to 1 if we finish below the legal floor C, and 0 otherwise.

At last, we can define the pseudo-code for the simulation. We first initialise $t = n_A = 0$, as well as $n_0 = 0$. We then let $cost = rev = 0$. We also initialise our B and *months* lists to be empty. Then to begin the simulation, we generate t_A using Algorithm 4, and set the lists $t_D = t_C = \infty$, with that as the only item. We enclose the following sub-routines in a while loop, that for runs 12 months, denoted $T_{total} = 12$. Each algorithm is executed depending on the values t_A, t_D, t_C, which are stated in the caption:

```
Algorithm 6: $t_A = \min(t_A, t_D, t_C, T_{total})$
1 $t \leftarrow t_A$;
2 $n_A \leftarrow n_A + 1$;
3 $n_0 \leftarrow n_0 + 1$;
4 Generate new $t_A$ with Algorithm 4 ;
5 Generate new $t_C$ for this customer with Algorithm 4 ;
6 Generate new $t_D$ for this customer with Algorithm 5 ;
```

This first sub-routine (6) is executed if there is first an arrival, lines 1 to 3 set the current time to the time of the next arrival, and counts the arrival on n_A and n_0. We then generate the next arrival time and the time of the next claim for this arrival on lines 4 and 5. Finally, on line 6, we generate the departure time of this customer

Algorithm 7: Any $t_D = \min(t_A, t_D, t_C, T_{total})$

```
1 $t \leftarrow t_D$;
2 $n_0 \leftarrow n_0 - 1$;
3 Set $t_D$ for this customer to $\infty$;
4 Set $t_C$ for this customer to $\infty$;
```

Algorithm 7 describes the situation in which a departure is the next event. Line 1 changes the current time to the time of the next departure, and line 2 makes the necessary adjustments to the counter. In lines 3 and 4 we set the departure times and claim times for this customer to be ∞, for they have left the service.

Algorithm 8: Any $t_C = \min(t_A, t_D, t_C, T_{total})$

```
1 $t \leftarrow t_C$;
2 Generate new $t_C$ for this customer with Algorithm 4;
3 $U_1 \leftarrow$ Random uniform(0,1) variable ;
4 if $U_1 \leq 0.6$ then
5     $U_2 \leftarrow$ Random uniform(0,1) variable ;
6     $X \leftarrow 10 \cdot \text{upper}(U_2)$ ;
7     $payout \leftarrow 300 \cdot X + 500$ ;
8     $cost \leftarrow cost + payout$ ;
9 end
```

Algorithm 8 describes the only other possible situation, where the next event is an insurance claim. This sub-routine executes if and only if there exists a value in t_C such that it is smallest in t_C, and smaller than t_A and all values in t_D. In line 1 we first set the time to the time of the next claim, and then in line 2, we generate the next claim time for that customer. Lines 4-8 describe the eventuality that there is a claim. We start by generating a random uniform variable, and if it is less than 0.6, we calculate the payout. In line 6 we generate another random uniform variable, and in line 7 we turn this into a random variable between 1 and 10. Finally, in lines 8 and 9, we use the given formula to calculate the *payout* and add it to the total running cost.

At each time step, we check if we have entered a new month. We do this by finding the lower integer of t, and if it does not exist in our *months* list, we must have entered a new month so we add it to the list. We then calculate $M \cdot n_0$ at that time and add it to *rev*. If the month does exist in the list, we do nothing.

Finally, if $t > T_{total}$, we stop the simulation and perform a calculation to determine if our capital holdings are beneath the legal floor, namely $C_0 + rev - cost$, where C_0 is our capital stock at the start of the period. If it is, we append a 1 to B, and 0 otherwise.

As outlined earlier, we run the simulation 1000 times with the parameters $n_0 = 0, c_0 = 50,000, M = 300, C = 30,000, \lambda = 3, \alpha = 3.5$. We also set μ to be $\frac{1}{12}$, since this is the max value that $\frac{1}{12+t}$ can take. Finally, we calculate P, and we came out with a value of 0.384. This is a rather high probability, and as such we carried out further simulations. We experimented with a higher M, and found that a small increase drastically decreases the P, as discussed in Section 2.4.

D.2 Confidence

`Associated Files: confidence.Rmd`

We now move to estimating C_{12}. We can largely use the same model as in the previous section, however, we now use an output variable of C_{12} at the end of each simulation. Instead of looping 1000 times, we run the loop until we can be confident we have an estimate we are satisfied with, say 3000. We also append a new subroutine at the end of each simulation, this is designed to update the sample mean $\bar{C_{12}}$ and variance S^2 of C_{12}. It's process is outlined in Algorithm 9.

```
Algorithm 9: Update $\bar{C_{12}}$ and $S^2$
  Data: New $C_{12}$, Current $\bar{C_{12}}$, Current $S^2$, Iteration $k$
1 New $\bar{C_{12}} \leftarrow \bar{C_{12}} + \frac{C_{12} - \bar{C_{12}}}{k+1}$;
2 New $S^2 \leftarrow \left(1 - \frac{1}{k}\right) S_k^2 + (k+1)\left(\text{New } \bar{C_{12}} - \bar{C_{12}}\right)^2$
```

Then, when these values satisfy

$$\frac{S}{\sqrt{k}} < \frac{500}{1.645}$$

We record the $\bar{C_{12}}$ and use this as our estimate for C_{12}, and record S^2 and k the first iteration this inequality was satisfied. We use the values 500 since this is the interval to which we would like to be 90% sure we are within, and 1.645 since this is a 5% deviation from the standard normal distribution mean. It is worth noting that we assume the Central Limit Theorem applies here. In our simulation, we attained estimates of $C_{12} = 32585.55, k = 2034, S = 303.86$, where we calculated the last value by taking the square root of S^2

Finally, we wish to estimate an interval to which we are 95% sure the true value is within, and for this we can use S from earlier. We take 1.96, which is a 2.5% deviation from the standard normal distribution mean, and calculate $1.96 \cdot S = 1.96 \cdot 303.86 = 595.56$. So our interval is $C_{12} \pm 595.56$

D.3 Variance Reduction

Lastly, we wish to outline specifically a measure to reduce variance in the estimates attained in Section 2.4. We will employ a variance reduction method using a Control Variate. We propose using the number of insurance claims at the end of 12 months. This is a suitable estimator since it is likely highly correlated with C_{12}, and easy to observe, since we can just add a counter variable for claims. Further, since these claims occur at a Poisson rate, they will be regular and therefore will not vary so greatly. We can then run the simulations and compare these estimates, to determine a variance decrease in C_{12}

Appendix E

Further Notes

It should be noted that Algorithms 4, 5 and 9 are based heavily on algorithms found in lectures. They are also implemented based on the code provided on the course page.

Python scripts, namely `map_gen.ipynb`, `path_gen(1).ipynb` and `path_gen(2).ipynb` were developed in order to plot Figures 2.1 and 2.2. They were also used to manipulate data found in Tables A.1, A.2 and B.1.

In the respective `/Python(Figure Generation)/` directories found in the files `1 Warehousing` and `2 Vehicle Routing`, there exist `.csv` files that were manipulated in Microsoft Excel to further aid the developments of tables and figures.