**1. Data Visualisation**

**Demand:**

From the demand data provided, we plot the mean demand (rounded up to the nearest integer) of pallets on weekdays and on Saturdays over the 4-week period. We use the mean to get an estimate of the average demand over time. We round these up to integers to ensure that we have whole-pallet demand values in our linear program and so that we slightly over-estimate demands. By tending toward over-estimation, our truck routes are more likely to hold if demand varies.

This will help us formulate our route generation model, which will generate routes which meet the mean demand for each of the stores.

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

Figure Figure

We notice that on Saturdays, pallets only need delivery to The Warehouse stores and not Noel Leeming stores, and that on Sundays there is no pallet demand at any location. Hence, we remove Sundays from our demand estimation.

On weekdays we see that the average daily pallet demand at every store is between 4 and 7 pallets. This demand decreases on Saturdays - The Warehouse stores only require an average delivery of between 2 and 4 pallets every Saturday. Since average demand nearly halves for The Warehouse stores on Saturdays, we will solve the linear program for weekdays and Saturdays separately.

**Store locations:**

Using the Warehouse Locations dataset, we plotted the approximate geographic locations of the stores by type to get a general idea of where our trucks will visit.

Map

Description automatically generated

Figure 3

Our distribution points are located due North and South of the large cluster of Noel Leeming and Warehouse stores at the centre of the map. There is overlap in the locations of some stores. This helps us decide which groups of nodes to include in routes during route generation.

**2. Route Generation**

Using the assumption that the mean demands in Figures 1 and 2 accurately represent daily demand at the stores, we generate a large selection of feasible routes for trucks to take.

**Division by geography:**

Using the plot of store locations in Figure 3, we have identified 5 clusters of stores. Thus, our algorithm’s initial step divides The Warehouse and Noel Leeming stores into these 5 geographical regions. We can reasonably assume that no optimal truck routes will include delivery to stores in different clusters. Therefore, our algorithm considers each of these clusters separately and generates many possible routes to deliver to any number of these stores within each cluster. The cluster division is shown in Figure 4. The function that achieves this split is *region\_divide* in *Generate\_Routes.py*. Map

Description automatically generated

Figure 4

Note that groups 1, 2 and 5 contain routes that begin and end at the North Distribution. Groups 3 and 4 are associated with the South Distribution supply. In the case where the North Distribution centre is closed, all routes will begin and end at the South Distribution.

**Route generation for clusters:**

With the aforementioned clusters, the function *all\_routes* creates 1000 feasible routes for every cluster and formats them for input to the linear program in *Solve\_LP.py*. These routes are generated randomly through the following algorithm:

For each cluster in Auckland (as defined by *region\_divide*):

Execute 1000 times:

1. We have an initial route of no stores.
2. A store is randomly selected from the cluster.
3. The store is appended to the route if and only if:

* the updated route beginning and ending at the associated supply and going through the stores in the route takes less than 6 hours to execute
* the updated route has a total demand of less than 20 pallets.

1. If the store was appended, go back to step 2.
2. If the store was not appended, finish the generation of the route and stash.

N.B.: if a store has a demand of 0 (all Noel Leeming stores on Saturday), then it will be excluded from the possible selections for stores in Step 2.

This way, we generate 1000 feasible routes for every sub-region. These routes, although randomly generated, are all feasible because stores were added until the demand and truck driving time exceeded the maximum truck capacity of 20 pallets, or time of 6 hours.

The cost of the route as the length of time to drive the route and unload the pallets is calculated and sent to the linear program. We find this value by considering both the travel time and an unloading rate of 10 minutes/pallet at each store.

**Why a 6-hour route time restriction?**

We have discovered that with a ‘hard’ constraint of a 4-hour shift for every truck driver, **it is not possible to deliver to all the stores should the North Distribution be discontinued**. This is another factor to consider when making the decision regarding this supply. Truck drivers will have to exceed 4 hour shifts if this change is made.

By allowing our upper limit on total route time to exceed 4 hours and range up to 6 hours, we allow for routes that may exceed 4 hours and thus make closing the North Distribution have feasible routing solutions. This route overtime is duly penalised in the objective function of our linear program.

**3. Optimisation Model Formulation**

The enumeration of many feasible routes allowed the following formulation of this problem as a vehicle routing problem (VRP).

**Decision Variables:**

For a set of routes, we have

Where *.*

**Objective Function:**

Dividing up the routes, we will have a set of routes which have , and a set of routes which have . We must charge $250/h for time these routes go over 4 hours. We charge $175/h for all routes under 4 hours and for the first 4 hours of the routes which go over 4 hours. Thus, our objective function is:

Where is the ‘cost’ of a route in seconds, calculated in *Generate\_Routes.py.*

**Constraints:**

Each store (which we henceforth describe as a `node’) must be visited exactly once. However, we only need to visit the node if its demand is greater than zero. Thus, as seen in Figure 2, when all Noel Leeming stores have zero demand, we need not visit these nodes.

For each node of the set of nodes in the network:

If the demand of a node is such that , and a set of routes visits node , then we add the constraint:

Denoting the set of nodes with as , the number of these node-visiting constraints in the formulation is equal to .

We impose a GUB constraint to express the number of trucks we have available to assign to each of these routes:

Finally, we have non-negativity and binary constraints:

In solving this linear program in *Solve\_LP.py*, we found that the GUB constraint for the number of trucks is non-binding. Thus, no additional variables for the purchase of extra trucks was added to the formulation. Furthermore, no wet-leasing of trucks from Mainfreight is necessary, so no ‘wet-leasing’ variables were included in the formulation.

**4. Results**

**Map

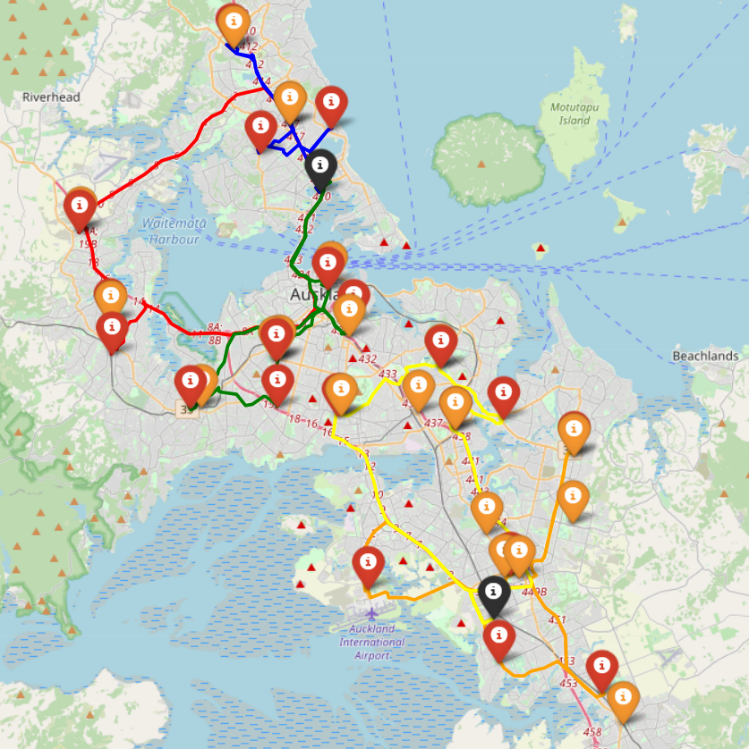
Description automatically generated**

Figure : Weekday routes with both distribution centres open. Figure : Saturday routes with both distribution centres open.

**Map

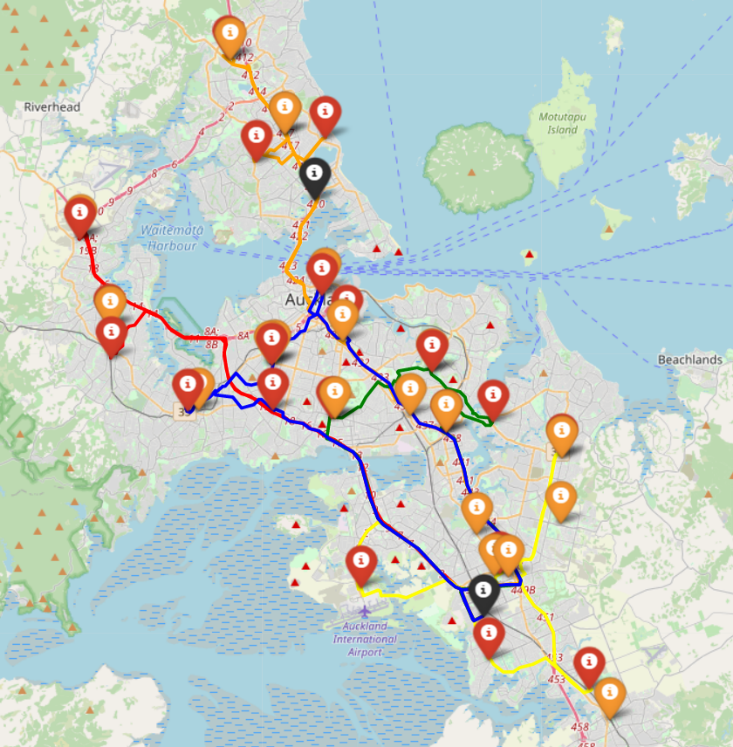
Description automatically generated**

Figure : Weekday routes with North Distribution closed.Figure 8: Saturday routes with North Distribution closed.

**Route costs:**

|  |  |
| --- | --- |
| **Using both Northern and Southern Distribution Points** | |
| **Single Weekday Cost** | $9249.35 |
| **Single Saturday Cost** | $2958.22 |
| **Total Weekly Cost** | $49,204.97 |
| **Removal of Northern Distribution Point** | |
| **Single Weekday Cost** | $10198.46 |
| **Single Saturday Cost** | $3311.56 |
| **Total Weekly Cost** | $54,303.86 |

**Commentary:**

Closing the northern distribution centre will increase weekly costs by $5098.89. This means an extra $20,400 a month. If The Warehouse Group wishes to maximise their savings, they should close the Northern Distribution Centre, which will save them $400,000 a month and make a total saving of $379,600 a month. However, many truckers will then have to drive routes longer than their 4-hour shifts allow – they may resent this added skew in their work-life balance.

1. Find all routes with total demand > 20 2. For each of these routes: • Find smallest-demand store with demand > (route demand – 20) • Get distribution centre closest to store • Create route from nearest centre to this store only • Calculate cost of new route • Delete store from old route, update cost of old route • Add new route to list of best routes 3. Output updated list of best routes and routing plan.