ROCKY MOUNTAIN BAKERY

Rudi's

EST. 1976 · BOULDER, CO

Final Report

02.27.2024

Team 3

Vincent Carter, vica3049@colorado.edu

Annan He, anhe7062@colorado.edu

Akshaj Khirwadkar, akkh8908@colorado.edu

Will Peltier, wipe6186@colorado.edu

Raghav Pariti, vepa3976@colorado.edu

Spring 2024
Leeds School of Business
Boulder, CO

Business Understanding

Rudi's Bakery, with its roots in Boulder, Colorado, is the popular brand in the gluten-free and organic bread market. Rudi's success lies in its commitment towards traditional baking methods and its beliefs in simplicity, quality, and honesty.

The operational challenge Rudi's faces revolve around the optimization of its Direct Store Delivery (DSD) routes, particularly in the Colorado Springs area. The objective is to enhance operational efficiency by developing optimal driving routes for delivery drivers. This entails a careful balance of several factors, including equitable distribution of store counts among drivers, sticking to set delivery window times, and minimization of fuel and labor costs. Achieving optimal routes is expected to have significant operational and financial benefits, improving service frequency to stores and, consequently, potentially boosting sales while reducing operational costs.

Problem Statement

The primary challenge for Rudi's Bakery is to devise an optimal strategy for its DSD routes that maximizes operational efficiency and sales while minimizing costs.

Specifically, the project aims to:

- Increase the frequency of store service visits on two of the existing DSD routes in the greater Colorado Springs area from at least three to no more than five times a week. This is based on the hypothesis that an additional store visit could lead to a 5% increase in sales.
- Minimize operational costs, including time/labor and distance/fuel expenses, associated with these deliveries.

The scope of Problem is to examine 2 of the 14 DSD routes in the Colorado Springs Area, with an emphasis on identifying potential opportunities for route crossover and optimization. The expected outcome is a comprehensive methodology, and model that recommend two optimized routes.

Overarching Objectives

Efficiency Improvement: Enhance the efficiency of delivery operations by designing two optimal routes per day that minimize total travel time and distance, ensuring all 50 destinations are serviced within the constraints of the 9-hour daily delivery window and the 15-minute service time per stop.

Increased Delivery Frequency: Increase the delivery frequency to all serviced stores, with a specific focus on daily deliveries to Costco (except Wednesdays and Sundays) and three times a week to other retail stores, to meet customer demand more effectively and ensure product freshness.

Sales Growth: Drive sales growth by increasing the availability of fresh bread at retail locations, thereby enhancing customer satisfaction and loyalty.

Waste Reduction: Reduce the amount of stale bread through more frequent deliveries, aligning with sustainability goals and reducing losses.

Data Overview

Raw Data Analysis: The initial dataset provided by the Stakeholder/Client contained various errors, including incorrect formatting, invalid store addresses, missing information, and duplicates. Data processing was essential for further analysis and project work.

- **Data Preparation:** To begin the project, we prioritized data preparation. This involved meticulously cleaning address data for uniformity and compatibility.
- **Validation API:** We have validated each store's location present in the "***Routes Dataset***" using "***Validation API***" (by Google Maps API) and integrating with Python has equipped us with accurate Store Location updations.
    - Additionally, since the initial dataset lacked unique identifiers, we incorporated them to improve data formatting and consistency.
- **GeoPandas**: This open-source library enabled us to fetch the latitudes and longitudes of each store. We used this data to create precise geometrical store location points. This was a crucial step, allowing us to calculate distances and generate time matrices for further stages within the project.
- **Visualizing Store Locations:** We used the "LeafMap" library to visualize store locations spread across Colorado Springs, Pueblo, and Castle Rock. This was necessary to gain insights into the exact store positions and to confirm address validation.
- **Distance-Time Matrix**: The creation of a distance and time matrix for each store relative to the 52 others (including the depot) is crucial. This matrix is essential for segregating routes using the "***K-Nearest Neighbors algorithm***" and serves as a key input for the optimization model.

MODEL

Modelling Approach:

mTSP Considerations

- The route optimization problem presented here is modelled as a **Mixed-Integer Linear Programming** (MILP) problem, specifically addressing a variant of the vehicle routing problem.
- Since we are not considering vehicle capacity and delivery windows for now the problem becomes a **multiple Traveling Salesman Problem** (mTSP).
- The mTSP seeks to determine the optimal set of routes for multiple salesmen (vehicles, in this context) starting and ending at a depot, visiting a set of locations (stores) without repetition, and minimizing the total distance (time) travelled thereby maximizing the efficiency of deliveries.

Qualification as an NP-Hard Problem

This problem qualifies as NP-hard due to the following reasons:

- **Combinatorial Nature**: The need to select routes from a potentially exponential number of combinations of store visits for each vehicle renders the problem computationally intensive, characteristic of NP-hard problems.
- **Complexity with Constraints**: The addition of operational constraints, such as delivery frequencies, working hours, and specific scheduling requirements, further complicates the solution space, akin to the complexities that make the mTSP NP-hard.
- **Decision Variables**: The use of mixed-integer decision variables for routing and scheduling introduces discontinuities and non-linearities, characteristic challenges in solving NP-hard problems efficiently.

Therefore, the approach adopted was a combination of **k-nearest neighbour heuristic** for route selection for each vehicle. Then the daily store assignment within time and consecutive days constraints was solved for each vehicle separately using Mixed integer linear programming

STEP 1: Heuristic Approach for Route Sequencing using KNN (k – nearest neighbours)

The core of our routing optimization strategy employs a nearest neighbor clustering algorithm, designed to efficiently assign stores to delivery routes. This method is implemented in Python, utilizing the Pandas library for data manipulation, highlighting our commitment to leveraging open-source technologies for complex logistical challenges.

- **Nearest Neighbor Heuristic**

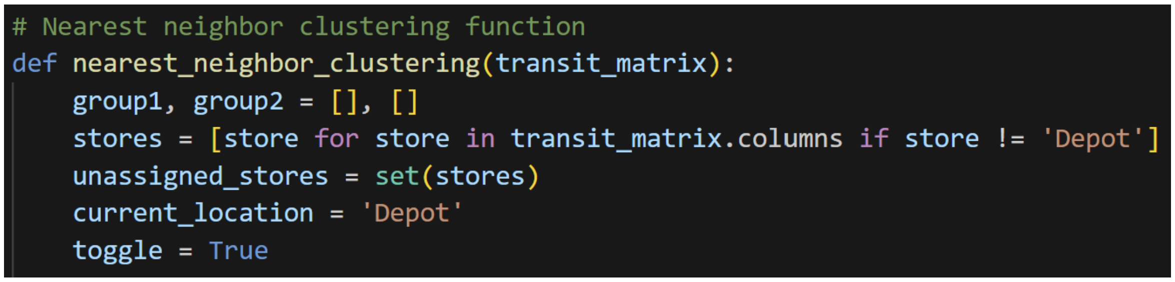
  The algorithm begins by distinguishing the depot (starting point) from the list of stores needing deliveries. This distinction is crucial for routing as all deliveries originate and conclude at the depot.

- **Routing Logic**

  The algorithm iteratively assigns stores to one of two groups. These groups represent the delivery routes for our two vehicles. The assignment is based on proximity, with the nearest unassigned store to the current location being added to a route. This process alternates between the two groups, ensuring a balanced distribution of delivery assignments.

- **Toggle Mechanism**

  A toggle mechanism ensures that assignments alternate between the two groups, promoting equitable workload distribution and route efficiency. This adaptability is key to managing real-world logistical variables and constraints.



```python
# Nearest neighbor clustering function
def nearest_neighbor_clustering(transit_matrix):
    group1, group2 = [], []
    stores = [store for store in transit_matrix.columns if store != 'Depot']
    unassigned_stores = set(stores)
    current_location = 'Depot'
    toggle = True
```

Figure 1: Code Snippet for Toggle Mechanism

- **Sequential Assignment**

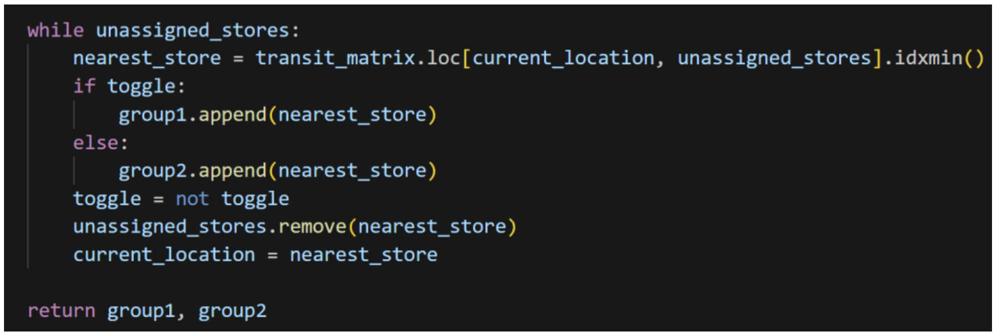
  The process starts by placing the nearest store to the depot in the first group, then alternates to the second group for the next closest store, continuing this pattern.

- **Balanced Distribution**

  This alternating pattern ensures a balanced allocation of delivery points, preventing route bias and overburdening.

- **Minimized Travel Distance**

  By assigning stores based on time, the algorithm reduces the travel time between consecutive stops, optimizing route efficiency.



```
while unassigned_stores:
    nearest_store = transit_matrix.loc[current_location, unassigned_stores].idxmin()
    if toggle:
        group1.append(nearest_store)
    else:
        group2.append(nearest_store)
    toggle = not toggle
    unassigned_stores.remove(nearest_store)
    current_location = nearest_store

return group1, group2
```

Figure 2: Algorithm for Optimizing Route Efficiency

- **Operational Efficiency Goal**

  The overarching aim is to cut down on transit times and elevate delivery operation efficiency, leveraging our resources effectively while maintaining prompt service delivery.

After applying the KNN approach to cluster and sequence the delivery routes efficiently, we now have the sequenced routes for both vehicles. These routes serve as a critical input for the subsequent phase of the project, where they will be further refined through modeling and incorporating various operational constraints using the MILP framework. This sequential approach ensures that the foundational route structure is both efficient and primed for optimization, aligning with the project's objectives and operational requirements.

STEP 2: MILP (Mixed Integer Linear Programming) Approach

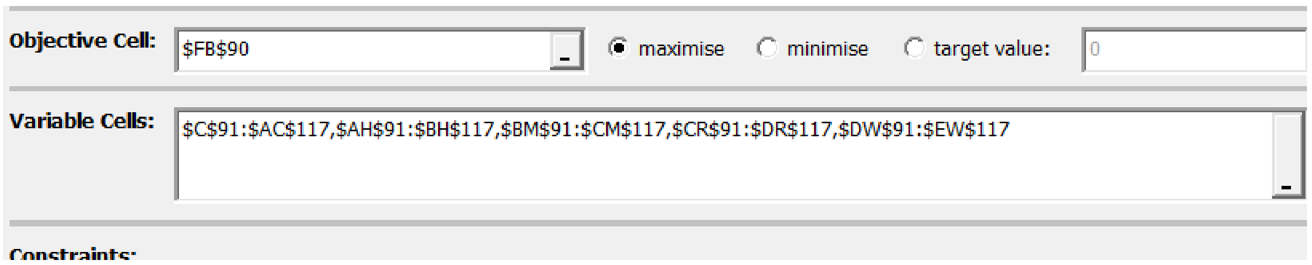
- The MILP framework is employed due to its robustness in handling the linear relationships between the decision variables and the objective, while efficiently incorporating the integer constraints that define the delivery decisions.
- This approach allows for the formulation of the delivery schedule optimization as a linear objective function, subject to linear equality and inequality constraints that capture the operational, temporal, and logistical requirements of the problem.

Objective Function

To maximize the total number of deliveries to 52 stores while starting from the depot and ending at the depot within a time constrained operational week, utilizing two vehicles.

Decision Variables

The deliveries to each store for the period of 5 operational days.



Objective Cell: $FB$90 [✓ maximise] minimise target value: 0

Variable Cells: $C$91:$AC$117,$AH$91:$BH$117,$BM$91:$CM$117,$CR$91:$DR$117,$DW$91:$EW$117

Constraints:

Figure 3: Decision Variables

- **Daily Delivery Requirement for Costco**

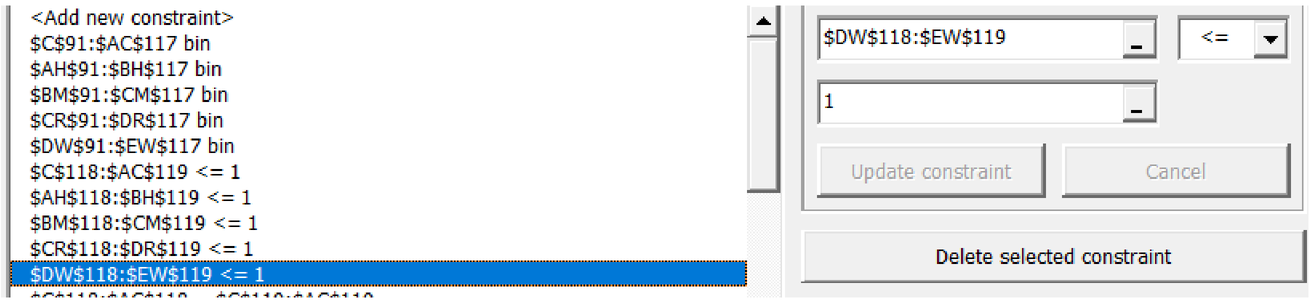
  Ensures daily deliveries to Costco stores throughout the five operational days to meet specific inventory demands.

- **Minimum Weekly Delivery Frequency**

  Guarantees that each non-Costco store receives a minimum of three deliveries per week, ensuring adequate stock levels are maintained.

- **Distribution of Delivery Days**

  Prevents scheduling deliveries on back-to-back days for stores with a three-times-a-week delivery requirement, except on Fridays and Saturdays to accommodate higher sales volume. This would avoid stale bread at the stores.



```
<Add new constraint>
$C$91:$AC$117 bin
$AH$91:$BH$117 bin
$BM$91:$CM$117 bin
$CR$91:$DR$117 bin
$DW$91:$EW$117 bin
$C$118:$AC$119 <= 1
$AH$118:$BH$119 <= 1
$BM$118:$CM$119 <= 1
$CR$118:$DR$119 <= 1
$DW$118:$EW$119 <= 1
```

$DW$118:$EW$119 <= 1

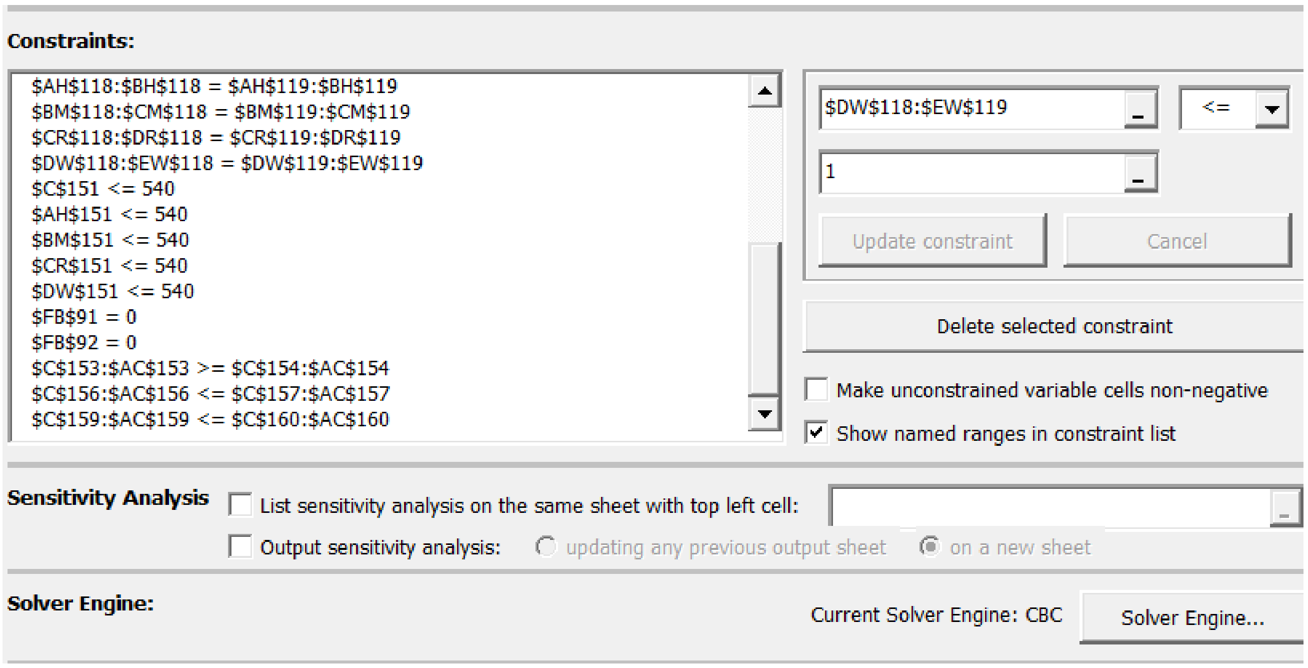
Figure 4 Constraints

- **One Delivery per Store per Operational Day**

  Enforces a policy where each store is eligible for only one delivery per operational day to ensure equitable distribution of delivery resources.

- **Daily Working Hours Cap**

  Restricts the total daily working hours to 9 (540 mins) hours per vehicle, including transit, loading, and unloading times, to adhere to labour standards and promote driver well-being.



Constraints:

```
$AH$118:$BH$118 = $AH$119:$BH$119
$BM$118:$CM$118 = $BM$119:$CM$119
$CR$118:$DR$118 = $CR$119:$DR$119
$DW$118:$EW$118 = $DW$119:$EW$119
$C$151 <= 540
$AH$151 <= 540
$BM$151 <= 540
$CR$151 <= 540
$DW$151 <= 540
$FB$91 = 0
$FB$92 = 0
$C$153:$AC$153 >= $C$154:$AC$154
$C$156:$AC$156 <= $C$157:$AC$157
$C$159:$AC$159 <= $C$160:$AC$160
```

$DW$118:$EW$119 <= 1

☐ Make unconstrained variable cells non-negative
☑ Show named ranges in constraint list

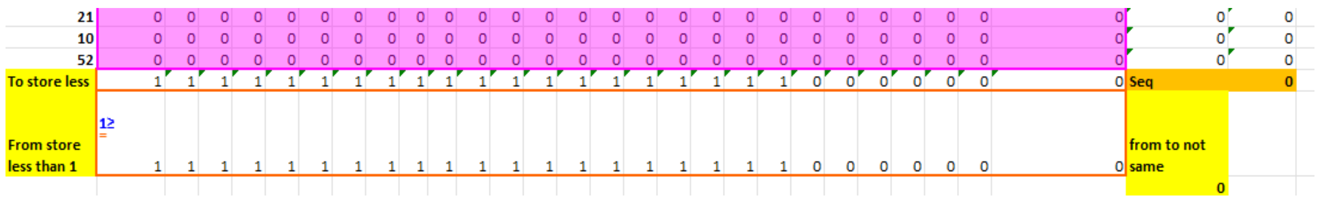
Sensitivity Analysis ☐ List sensitivity analysis on the same sheet with top left cell:
☐ Output sensitivity analysis: ○ updating any previous output sheet [✓ on a new sheet]

Solver Engine: Current Solver Engine: CBC

Figure 5: some more set of Constraints

Sequence Constraint

Since each vehicle needs to adhere to the sequence of stores as given by the route from KNN heuristics a sequence constraint was put so that the vehicle cannot visit the stores before it in the sequence.



21		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0
10		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0
52		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0
To store less		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	Seq		0
From store less than 1	1≥ =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	from to not same		
																														0	

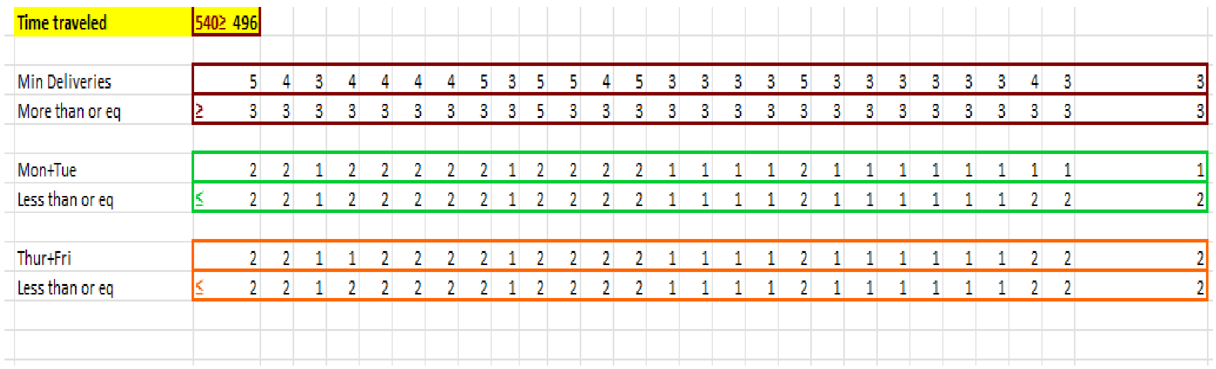
Figure 6: Sequence Constraint

Entry and Exit Equality Constraint

Ensures the number of entries into any given store equals the number of exits, maintaining consistency in vehicle routing, which should be equal to or less than 1.

Diagonal Entries Constraint

Specifies that diagonal elements in the route matrix (representing a store visiting itself) must be set to zero, reinforcing the rule that a store cannot be considered a destination from itself within the same route.



Time traveled	540≥ 496																										
Min Deliveries		5	4	3	4	4	4	4	5	3	5	5	4	5	3	3	3	3	5	3	3	3	3	3	3	4	3	3
More than or eq	≥	3	3	3	3	3	3	3	3	3	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Mon+Tue		2	2	1	2	2	2	2	2	1	2	2	2	2	1	1	1	1	2	1	1	1	1	1	1	1	1	1
Less than or eq	≤	2	2	1	2	2	2	2	2	1	2	2	2	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2
Thur+Fri		2	2	1	1	2	2	2	2	1	2	2	2	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2
Less than or eq	≤	2	2	1	2	2	2	2	2	1	2	2	2	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2

Figure 7: List of Deliveries Made

- The Costco store has been manually set to 5 deliveries a week.
- The sum of Monday and Tuesday is equal to 1 where deliveries are 3. Same constraint has been put for the sum of Thursday and Friday.

A constraint has been put for each day, which sum of time taken for the whole route needs to be less than 540 mins (9 hours)

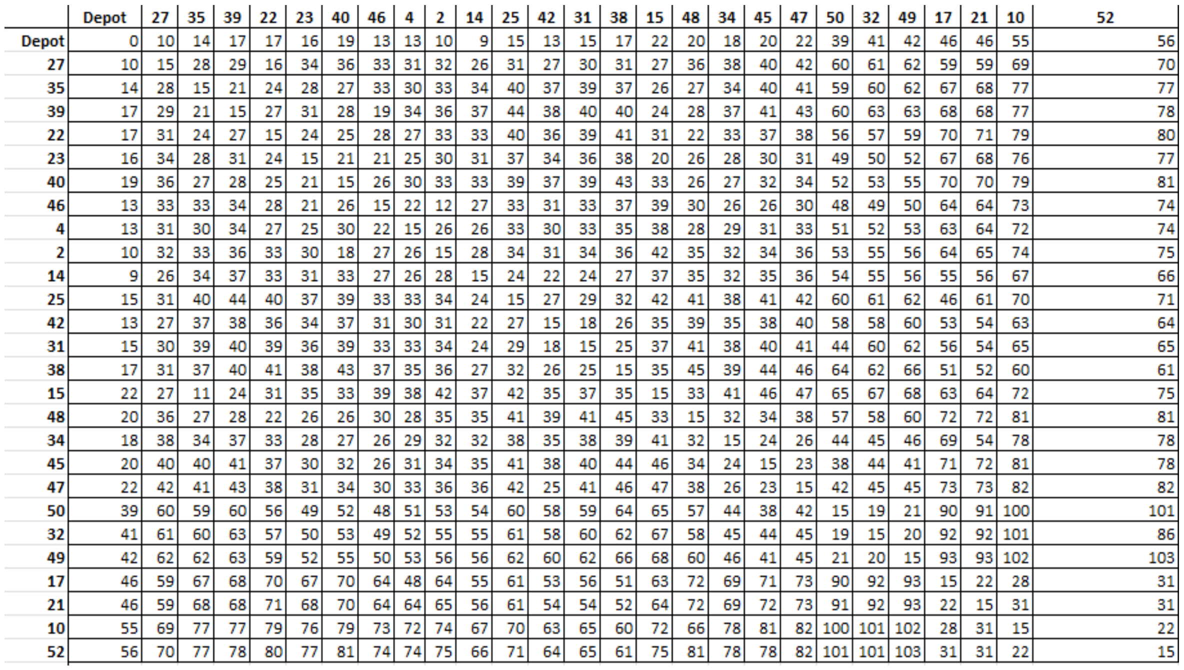
Data inputs considered along with the constraints

- **Operational Day Restrictions**

  Omits Wednesdays and Sundays from the delivery schedule, aligning with company policies and operational planning.

- **Standard Loading and Unloading Time**

  Allocates a consistent 15-minute period for loading and unloading activities at each store visit, optimizing time management across all deliveries, excluding depot operations.



	Depot	27	35	39	22	23	40	46	4	2	14	25	42	31	38	15	48	34	45	47	50	32	49	17	21	10	52
Depot	0	10	14	17	17	16	19	13	13	10	9	15	13	15	17	22	20	18	20	22	39	41	42	46	46	55	56
27	10	15	28	29	16	34	36	33	31	32	26	31	27	30	31	27	36	38	40	42	60	61	62	59	59	69	70
35	14	28	15	21	24	28	27	33	30	33	34	40	37	39	37	26	27	34	40	41	59	60	62	67	68	77	77
39	17	29	21	15	27	31	28	19	34	36	37	44	38	40	40	24	28	37	41	43	60	63	63	68	68	77	78
22	17	31	24	27	15	24	25	28	27	33	33	40	36	39	41	31	22	33	37	38	56	57	59	70	71	79	80
23	16	34	28	31	24	15	21	21	25	30	31	37	34	36	38	20	26	28	30	31	49	50	52	67	68	76	77
40	19	36	27	28	25	21	15	26	30	33	33	39	37	39	43	33	26	27	32	34	52	53	55	70	70	79	81
46	13	33	33	34	28	21	26	15	22	12	27	33	31	33	37	39	30	26	26	30	48	49	50	64	64	73	74
4	13	31	30	34	27	25	30	22	15	26	26	33	30	33	35	38	28	29	31	33	51	52	53	63	64	72	74
2	10	32	33	36	33	30	18	27	26	15	28	34	31	34	36	42	35	32	34	36	53	55	56	64	65	74	75
14	9	26	34	37	33	31	33	27	26	28	15	24	22	24	27	37	35	32	35	36	54	55	56	55	56	67	66
25	15	31	40	44	40	37	39	33	33	34	24	15	27	29	32	42	41	38	41	42	60	61	62	46	61	70	71
42	13	27	37	38	36	34	37	31	30	31	22	27	15	18	26	35	39	35	38	40	58	58	60	53	54	63	64
31	15	30	39	40	39	36	39	33	33	34	24	29	18	15	25	37	41	38	40	41	44	60	62	56	54	65	65
38	17	31	37	40	41	38	43	37	35	36	27	32	26	25	15	35	45	39	44	46	64	62	66	51	52	60	61
15	22	27	11	24	31	35	33	39	38	42	37	42	35	37	35	15	33	41	46	47	65	67	68	63	64	72	75
48	20	36	27	28	22	26	26	30	28	35	35	41	39	41	45	33	15	32	34	38	57	58	60	72	72	81	81
34	18	38	34	37	33	28	27	26	29	32	32	38	35	38	39	41	32	15	24	26	44	45	46	69	54	78	78
45	20	40	40	41	37	30	32	26	31	34	35	41	38	40	44	46	34	24	15	23	38	44	41	71	72	81	78
47	22	42	41	43	38	31	34	30	33	36	36	42	25	41	46	47	38	26	23	15	42	45	45	73	73	82	82
50	39	60	59	60	56	49	52	48	51	53	54	60	58	59	64	65	57	44	38	42	15	19	21	90	91	100	101
32	41	61	60	63	57	50	53	49	52	55	55	61	58	60	62	67	58	45	44	45	19	15	20	92	92	101	86
49	42	62	62	63	59	52	55	50	53	56	56	62	60	62	66	68	60	46	41	45	21	20	15	93	93	102	103
17	46	59	67	68	70	67	70	64	48	64	55	61	53	56	51	63	72	69	71	73	90	92	93	15	22	28	31
21	46	59	68	68	71	68	70	64	64	65	56	61	54	54	52	64	72	69	72	73	91	92	93	22	15	31	31
10	55	69	77	77	79	76	79	73	72	74	67	70	63	65	60	72	66	78	81	82	100	101	102	28	31	15	22
52	56	70	77	78	80	77	81	74	74	75	66	71	64	65	61	75	81	78	78	82	101	101	103	31	31	22	15

Figure 8: Time Matrix

- This is the matrix we built into the model which incorporates the time taken from each store to every other store plus the 15 minutes window.

Costs Using Technologies

Google Maps API: Utilizing the Google Maps API for accurate transit times and distances between locations introduces cost based on the volume of requests made. While offering unparalleled accuracy and up-to-date information, this API operates on a pay-as-you-go pricing model, where costs can escalate with increased usage, especially in applications requiring a large number of route calculations or real-time data requests.

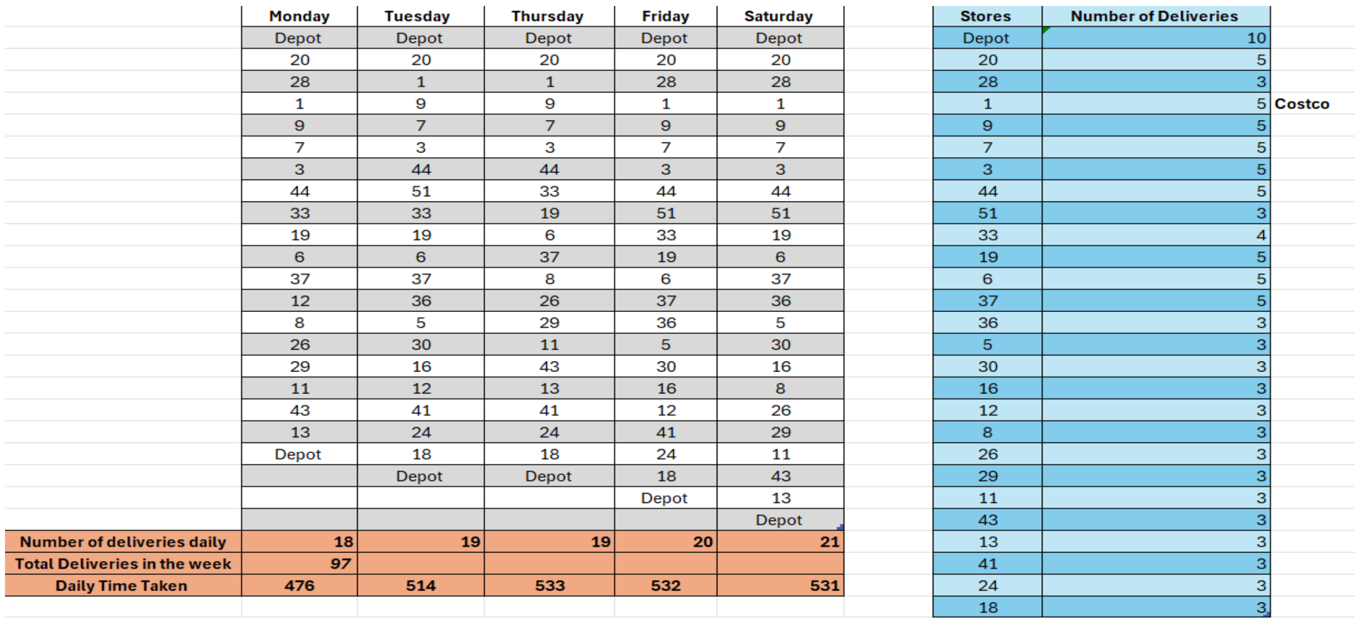
Python for KNN: Employing Python, an open-source programming language, for implementing the K-Nearest Neighbors (KNN) algorithm is highly cost-effective in terms of licensing, as there are no direct costs. However, the indirect costs may include the time and resources needed for development, testing, and maintenance of the algorithm, as well as potential hardware or cloud computing resources required for processing large datasets or complex calculations. As we have used part heuristic part MILP approach, we expect this model to run on a desktop itself even as the number of stores grows, without incurring any cloud costs

Open Solver: The use of Open Solver, an open-source optimization add-on for Excel, represents a cost-effective solution for solving complex linear and nonlinear problems. While the software itself is free, ensuring optimal performance may require investments in training for personnel to effectively utilize its capabilities, alongside potential upgrades to computing hardware to handle large-scale optimization models efficiently.

Final Suggestion

Suggested Routes

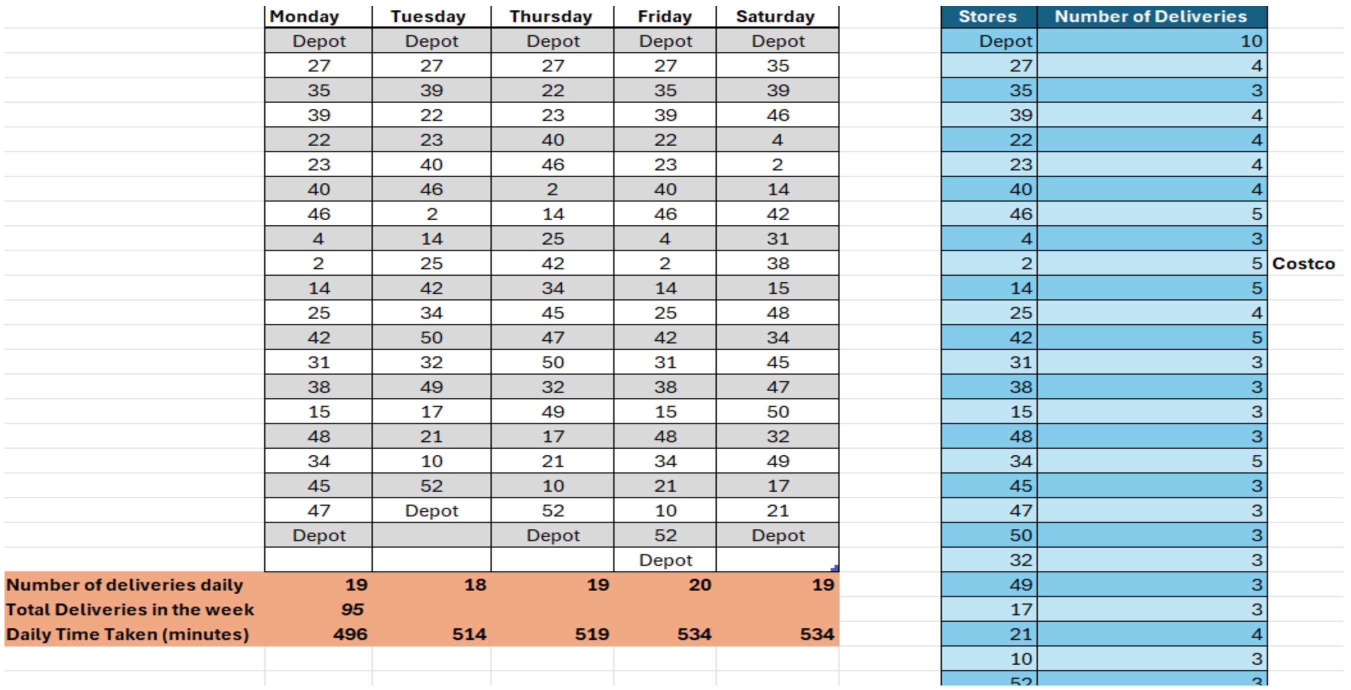
- The following two summary matrices encompass our final route suggestions.
- The columns are ordered in the sequence of the stores



	Monday	Tuesday	Thursday	Friday	Saturday
	Depot	Depot	Depot	Depot	Depot
	20	20	20	20	20
	28	1	1	28	28
	1	9	9	1	1
	9	7	7	9	9
	7	3	3	7	7
	3	44	44	3	3
	44	51	33	44	44
	33	33	19	51	51
	19	19	6	33	19
	6	6	37	19	6
	37	37	8	6	37
	12	36	26	37	36
	8	5	29	36	5
	26	30	11	5	30
	29	16	43	30	16
	11	12	13	16	8
	43	41	41	12	26
	13	24	24	41	29
	Depot	18	18	24	11
		Depot	Depot	18	43
				Depot	13
					Depot
Number of deliveries daily	18	19	19	20	21
Total Deliveries in the week	97				
Daily Time Taken	476	514	533	532	531

Stores	Number of Deliveries	
Depot	10	
20	5	
28	3	
1	5	Costco
9	5	
7	5	
3	5	
44	5	
51	3	
33	4	
19	5	
6	5	
37	5	
36	3	
5	3	
30	3	
16	3	
12	3	
8	3	
26	3	
29	3	
11	3	
43	3	
13	3	
41	3	
24	3	
18	3	

Figure 9: Vehicle 1 Route



	Monday	Tuesday	Thursday	Friday	Saturday
	Depot	Depot	Depot	Depot	Depot
	27	27	27	27	35
	35	39	22	35	39
	39	22	23	39	46
	22	23	40	22	4
	23	40	46	23	2
	40	46	2	40	14
	46	2	14	46	42
	4	14	25	4	31
	2	25	42	2	38
	14	42	34	14	15
	25	34	45	25	48
	42	50	47	42	34
	31	32	50	31	45
	38	49	32	38	47
	15	17	49	15	50
	48	21	17	48	32
	34	10	21	34	49
	45	52	10	21	17
	47	Depot	52	10	21
	Depot		Depot	52	Depot
				Depot	
Number of deliveries daily	19	18	19	20	19
Total Deliveries in the week	95				
Daily Time Taken (minutes)	496	514	519	534	534

Stores	Number of Deliveries	
Depot	10	
27	4	
35	3	
39	4	
22	4	
23	4	
40	4	
46	5	
4	3	
2	5	Costco
14	5	
25	4	
42	5	
31	3	
38	3	
15	3	
48	3	
34	5	
45	3	
47	3	
50	3	
32	3	
49	3	
17	3	
21	4	
10	3	
52	3	

Figure 10: Vehicle 2 Route

New Route Visualization

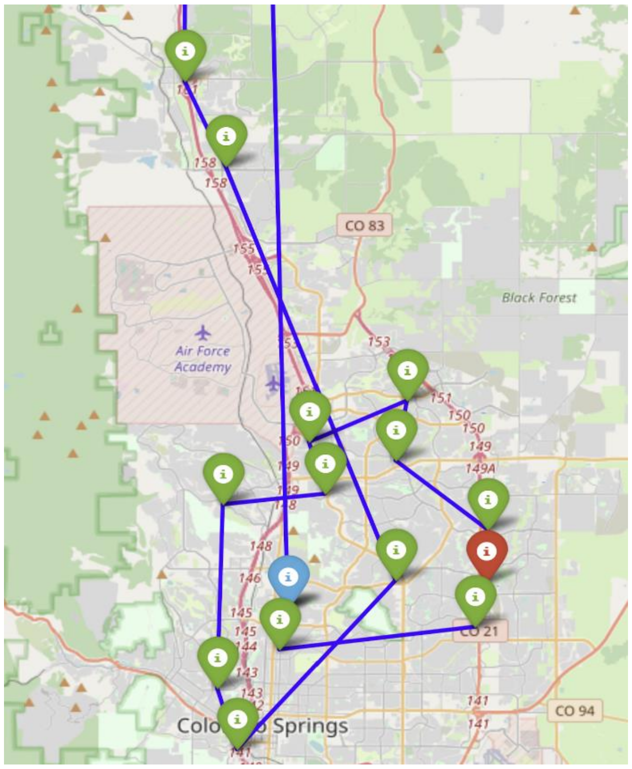


Figure 11: Monday's Route for Vehicle 1

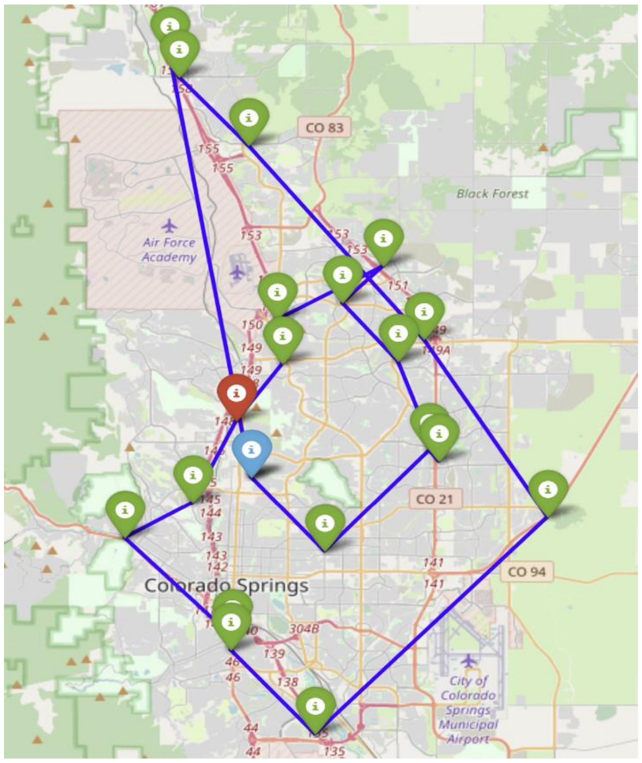


Figure 12: Monday's Route for Vehicle 2

Parameter	Original Schedule	Optimized Schedule
Weekly Deliveries	113	192
Store Delivery Frequency	33 stores fewer than 3x a week	All stores at least 3x a week
Deliveries to Specific Store	10 stores only once a week	Costco's 5x a week
Average Route Time	Not specified	8.6 hours

Figure 13: Comparison between Original Schedule and Optimized Schedule

Justification of suggested Routes and delivery days

Incorporating both the KNN approach and the MILP (Mixed Integer Linear Programming) model in the delivery maximization project offers a comprehensive and robust solution for delivery scheduling and routing. Below are bullet points that justify the suggested route as derived from these methodologies

- **Combination of Efficiency and Precision**: The KNN (K-Nearest Neighbors) approach serves as an efficient clustering method to initially group stores based on proximity, reducing the complexity of the routing problem. This step ensures that the subsequent optimization via MILP focuses on a set of already sensibly pre-clustered routes, enhancing overall computational efficiency without sacrificing the quality of solutions.

- **Optimized Resource Allocation**: The MILP model fine-tunes the initial clustering by precisely allocating delivery frequencies and sequencing the visits to meet all specified constraints (e.g., delivery windows, vehicle capacity). This two-step process ensures that resources are utilized optimally, leading to cost-effective and time-efficient delivery schedules.

- **Adherence to Operational Constraints**: By using MILP for the second step, the model rigorously adheres to all operational constraints, such as the minimum number of visits to each store, specific requirements for Costco stores, working hours, and vehicle capabilities. This adherence guarantees that the suggested route is not only optimal but also practical and implementable within the real-world operational framework.

- **Scalability and Flexibility**: The combination of KNN for preliminary clustering and MILP for detailed optimization offers scalability, making the solution adaptable to varying numbers of stores, vehicles, and other changing operational parameters. This scalability ensures that the solution remains relevant and effective even as the business grows or operational conditions evolve.

- **NP-Hard Problem handling**: Addressing the NP-hard nature of the multi-traveling salesman problem (mTSP), this hybrid approach leverages the strengths of both heuristic and exact methods and ensures that an efficient solution is found within reasonable time with minimal computing resources. KNN efficiently reduces the problem space, while MILP provides a precise optimization within that reduced space, offering a balanced approach to tackling computationally intensive routing problems.

- **Enhanced Delivery Performance**: Ultimately, the suggested route maximizes deliveries within the constraints of working hours and operational rules, ensuring that all stores, especially high-priority locations like Costco, receive timely and consistent service. This leads to improved customer satisfaction and operational reliability. Also adhering to consecutive days constraint minimizes the quantity of stale bread at stores

Future Directions

There are many potential future directions after this project. Here are some that we think are the most interesting:

Frontend:

While we did provide a strong model for route optimization, we did not have the time to provide an easy-to-use frontend for the model. If one wanted to change the list of addresses, or customize the route, they would have to have some level of technical knowledge to do so. This may limit the scalability of our model, and require the user to spend valuable time customizing the model, when it could be more easily facilitated through a frontend.

Greater Area of Service:

Similar to our last point, another avenue to continue this project would be to handle an area bigger than the one we were given, whether that be more stores, or just a greater geographical area. This would test the scalability of the model, as well as give us an opportunity to limit test the model, and see if any slowdowns happen at a certain number of stores.

More Constraints:

While we did hit the most important constraints in our model, such as hitting each store at least 3 times a week and Costco everyday, the client did mention many things in our meetings that could be micro adjusted to improve our model even further. Including these constraints moving forward could easily improve our model, or highlight areas that it may lack in.

Sales:

The second part of the project, including sales data, was not something that we got around to. Including this as a consideration in our model building could make the routes that we provided even more monetarily efficient, and it seems like it would be the next step, were we to continue. However, our MILP model design supports selection of high sales stores for more visits and lowering the visits to comparatively lower sales stores.