Cost Minimization of a Truck-Drone Combination Package Delivery System

Abstract

Formulating the truck-drone combination package delivery system as a mixed integer programming problem with the objective of minimizing cost subject to various constraints and solving it using Xpress-MP.

Authors

Ashutosh Singh | M13433470
Harsh Singhal | M13480322
Himanshu Chhabra | M13469297
Jagruti Joshi | M1348019
Rasesh Garg | M13437578

Contents

Introduction

An innovative solution for last-mile delivery of packages is to use a truck and drone combination package delivery system wherein a truck operates with a fleet of drones to deliver packages. The truck travels to a customer location which can be used as a launch site for the drones to reach other customers that are in vicinity of the visited customer. This gives rise to a new variant of the Travelling Salesman Problem (TSP) called the Flying Sidekick Travelling Salesman Problem (FSTSP). In TSP, we make sure that each customer is visited exactly once by the truck. In FSTSP, we make sure that each customer is visited exactly once by either the truck or the drone. We formulate the problem as Mixed Integer Programming (MIP) problem. The truck has a higher cost per mile cost compared to the drone and our objective is to minimize the total package delivery cost.

Problem Statement

A transportation company is evaluating a prototype system that combines trucks and drones for last-mile delivery services. The test run considers a set of orders that must be delivered to known locations. A delivery truck starts from a depot and visits "launch sites" corresponding to the customers locations. From each launch site, the truck deploys a series of drones that deliver the orders and return to meet the truck at the launch site. Once all the drones are recovered, the truck moves to the next launch site and repeats the process until all orders are delivered.

We need to write constraints that make sure that:

- The truck must start and end at the depot.
- The truck can deploy up to K drones at each stop in a launch site.
- The drones have limited cargo capacity and as a result they can only visit 1 customer (not counting the starting point) before returning to the truck.
- Each customer should be visited at least once by a drone. The customer locations used as launch sites will be visited by the truck and also will serve as the starting and ending point of a drone tour.

We are also given that:

- Assume that the depot is located at the origin (0,0).
- For each unit of distance traveled by the truck there is a cost of $10, while for each unit of distance traveled by a drone there is a cost of $3.

Our objective is to minimize the total travel cost (both by the trucks and the drones).

We've use Excel for data generation for testing our MIP formulation's code in Xpress-MP. We've solved this problem and found the minimum cost using Xpress-MP. Finally, we've used Python (Jupyter Notebook) to visualize our solution.

Our Approach

To solve the given problem, we implemented the following steps:

- **Data Generation using Excel**

  We generate random x and y coordinates between 0 and 100 for each order using Excel's *RANDBETWEEN(0,100)* function for 15, 20 and 25 orders/customers.

  We use a combination of Excel functions to calculate and round-off the Euclidean distances between a pair of points. An example formula to calculate and round-off the Euclidean distance between customers 1 and 8 in our testbed for 10 customers in Excel is:

  ROUND(SQRT((VLOOKUP($A10,$A$2:$C$13,2,FALSE)-VLOOKUP(F$1,$A$2:$C$13,2,FALSE))*(VLOOKUP($A10,$A$2:$C$13,2,FALSE)-VLOOKUP(F$1,$A$2:$C$13,2,FALSE))+(VLOOKUP($A10,$A$2:$C$13,3,FALSE)-VLOOKUP(F$1,$A$2:$C$13,3,FALSE))*(VLOOKUP($A10,$A$2:$C$13,3,FALSE)-VLOOKUP(F$1,$A$2:$C$13,3,FALSE)))2),0)

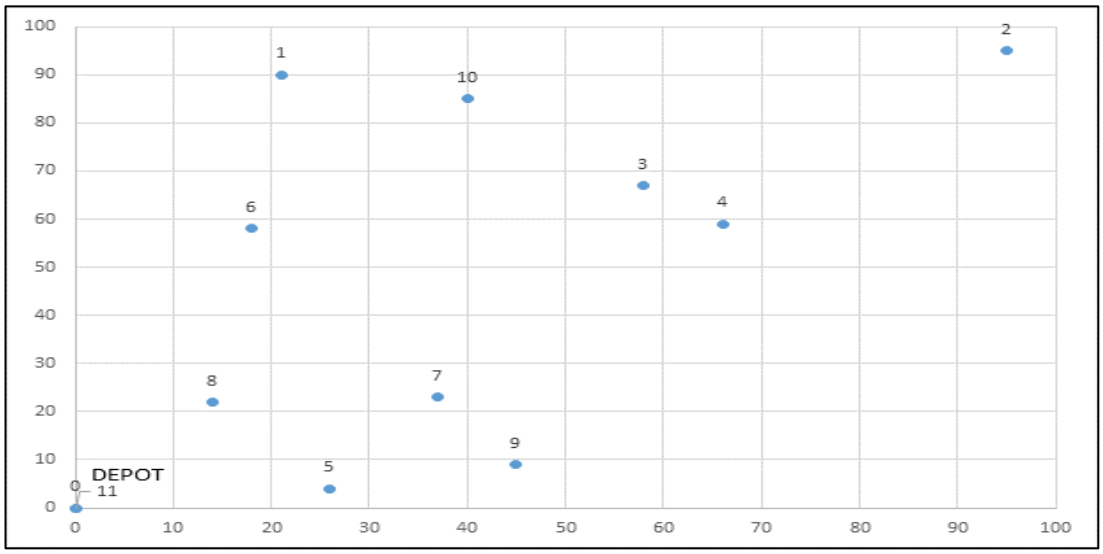
  The Euclidean distance matrix is a symmetric matrix. Once, we've generated all the testbeds, we use brute force to input this data into our distance matrix in Xpress-MP.

  An example formula to generate the first row of the Euclidean distance matrix by concatenation of cells in Excel is:

  P2&","&Q2&","&R2&","&S2&","&T2&","&U2&","&V2&","&W2&","&X2&","&Y2&","&Z2&","&AA2&","

  An example problem instance with 10 customers that was generated is shown below (Note that 0 and 11 are both depots as we need to create a dummy depot in the MTZ formulation).

Figure 1: Testbed generated for 10 customers



- **MIP Formulation**

Our MIP formulation for the FSTSP problem with 1 truck and multiple drones with all the parameters, variables, constraints, and objective is as follows:

Parameters

L : Number of datapoints/orders/customers in our dataset

K : Number of drones

M : A big number to facilitate some constraints

C : {0,1,2,…,L} be the customer index/order number

Variables

Dt : Euclidean distance matrix of size (C,C)

st : Truck switch matrix of size (C,C) where st(i,j) = 1 if truck goes from node i to node j and 0 otherwise

sd : Drone switch matrix of size (C,C) where sd(i,j) = 1 if drone goes from node i to node j and 0 otherwise

p : Position array of size (C) where if p(5) = 3 indicates that node 4 is the node that was visited 3rd by the truck

t : Truck node switch array of size(C) where t(i) = 1 if truck comes to node i and 0 otherwise

d : Drone node switch array of size(C) where d(i) = 1 if drone comes to node i and 0 otherwise

Objective Function

Minimize:

$$\sum_{i \in C} \sum_{j \in C} 10 * (Dt) * (st) + 3 * 2 * (Dt) * (sd)$$

For each unit of distance traveled by the truck there is a cost of $10, while for each unit of distance traveled by a drone there is a cost of $3. As drone is coming back to the same node from where it was launched so the objective function for cost of drone has been doubled.

Constraints

1. $\sum_{j \in C} st(L, j) = 0$
2. $\sum_{j \in C} st(j, 0) = 0$

3. $\sum_{j \in C} st(0, j) = 1$
4. $\sum_{j \in C} st(j, L) = 1$
5. $st(0,\mathrm{L}) = 0$
6. $\sum_{j \in C} st(i, j) \leq 1 \quad \forall\, i\ in\ C$
7. $\sum_{i \in C} st(i, j) \leq 1 \quad \forall\, j\ in\ C$
8. $\sum_{i \in C \mid i=j} st(i, j) = 0 \ \ \forall\, j\ in\ C$
9. $\sum_{i \in C \mid i=j} sd(i, j) = 0 \ \ \forall\, j\ in\ C$
10. $p(i) + 1 - M * \big(1 - st(i, j)\big) \leq p(j)0 \quad where\ p(j) \leq L \qquad \forall\, i, j\ in\ C$
11. $\sum_{j \in C} sd(0, j) = 0$
12. $\sum_{j \in C} sd(L, j) = 0$
13. $\sum_{j \in C} sd(j, 0) = 0$
14. $\sum_{j \in C} sd(j, L) = 0$
15. $d(i) + t(i) = 1 \quad \forall\, i\ in\ C$
16. $\sum_{j=1}^{L} st(i, j) = t(i) \ \ \forall\, i\ in\ 0 \ldots L - 1$
17. $\sum_{i=0}^{L-1} st(i, j) = t(j) \ \ \forall\, j\ in\ 1 \ldots L$
18. $\sum_{i=1}^{L-1} sd(i, j) = d(j) \ \ \forall\, j\ in\ 1 \ldots L\text{-}1$
19. $\sum_{j=1}^{L-1} sd(i, j) \leq k * t(i) \ \ \forall\, i\ in\ 1 \ldots L\text{-}1$
20. $sd(i, j) \leq 2 - t(i) - t(j) \quad \forall\, i, j\ in\ C$
21. $st(i, j) \leq 2 - d(i) - d(j) \quad \forall\, i, j\ in\ C$

Since we have introduced a dummy node L to complete the route of truck to the original depot, we are ensuring the truck doesn't leave from that dummy Node and also ensuring at no point truck goes back to origin from any point (Constraints 1 and 2).

In order to get our truck running from depot we must make sure truck switch *st* is switched for exactly one of the nodes j but at the same time it should not go the dummy node (Constraints 3 and 5). Similarly, we must ensure that truck ends at our dummy node L (Constraint 4). Through Constraints 6 and 7, we are giving freedom to leave or start from any node. The constraints for dummy and origin position will automatically prohibit certain conditions imposed by these constraints as they have some constraints specific to them.

Both the truck and drone switch should not be active when i=j thereby implying that they are not staying at the same node (Constraint 8 and 9). Finally, to eliminate truck subtours, we have applied the position constraint (Constraint 10). Just like we have applied the constraints on truck, we must apply some similar constraints on drone like

- Drones should not depart from Depot and Dummy (Constraints 11 and 12)
- Drones should not come back at Depot and Dummy (Constraints 13 and 14)

Since one point can be either launch site or the drone point so exactly one switch should be active (Constraint 15). Through Constraint 16 and 17, we are ensuring that starting and ending truck node is switched on if that arc is on. A similar type of equation is written for drone (Constraint 18) and making sure that number of drones from a launch site doesn't exceed k (Constraint 19). The last three constraints ensure

- Drones don't fly between two truck nodes (Constraint 20)
- Truck doesn't travel between two drone nodes (Constraint 21)
- Drones don't fly between two drone nodes (Constraint 22)

- **Solving the MIP using Xpress-MP**
  Xpress-MP Code

  Refer Appendix -B

  Xpress-MP Output

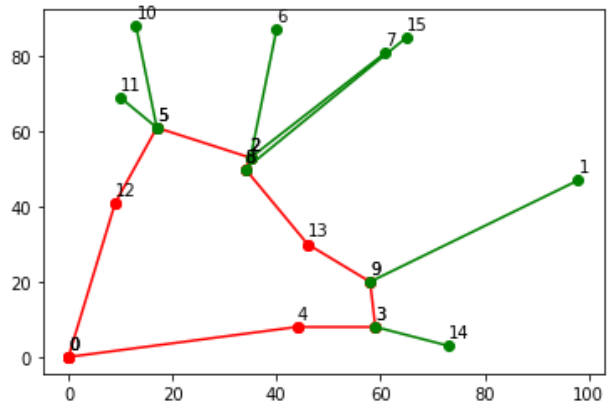
  Refer Result Section

- **Visualizing Xpress-MP Output using Python (Jupyter Notebook)**
  Refer Result Section

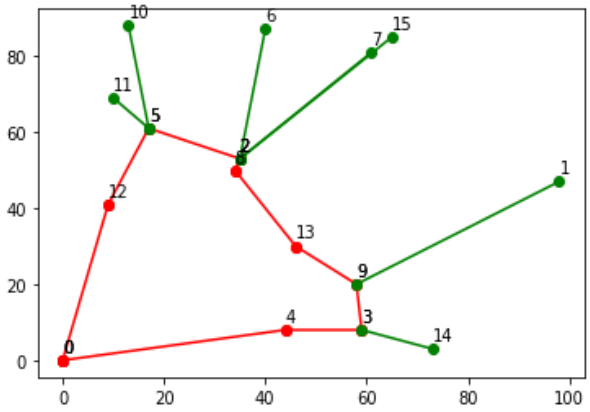
Results

- **15 Orders (2,3,4,5 Drones)**

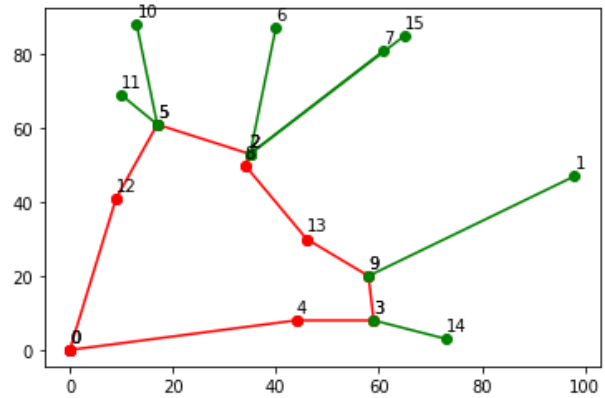
K=2



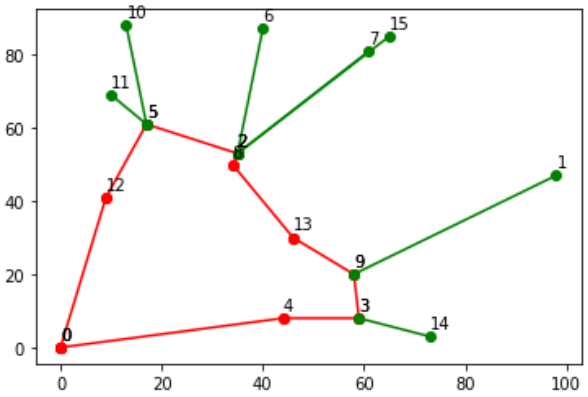
K=3



K=4



K=5

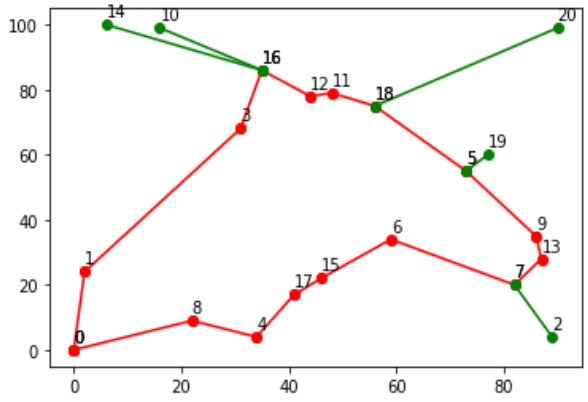


Case	Truck Route	Drone Route	Cost (in $)
15 Orders, 2 Drones	0-4-3-9-13-8-2-5-12-16	2-6-2, 2-7-2, 3-14-3, 5-10-5, 5-11-5, 8-15-8, 9-1-9	$3300
15 Orders, 3 Drones	0-4-3-9-13-8-2-5-12-16	2-6-2, 2-7-2, 2-15-2, 3-14-3 5-10-5, 5-11-5, 9-1-9	$3282

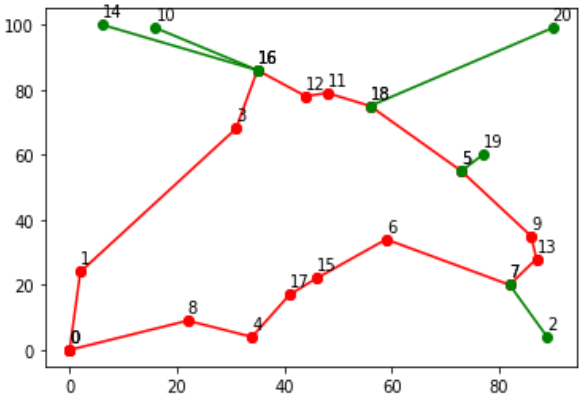
15 Orders, 4 Drones	0-12-5-2-8-13-9-3-4-16	2-6-2, 2-7-2, 2-15-2, 3-14-3, 5-10-5, 5-11-5, 9-1-9	$3282
15 Orders, 5 Drones	0-4-3-9-13-8-2-5-12-16	2-6-2, 2-7-2, 2-15-2, 3-14-3, 5-10-5, 5-11-5, 9-1-9	$3282

- **20 Orders (2,3,4,5 Drones)**

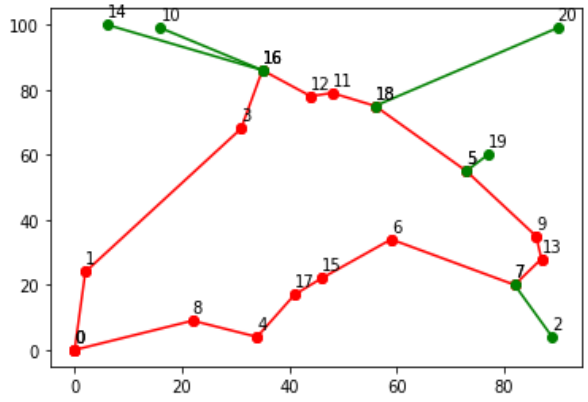
K=2



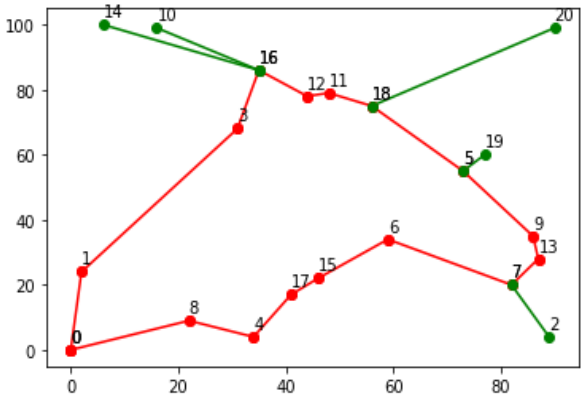
K=3



K=4



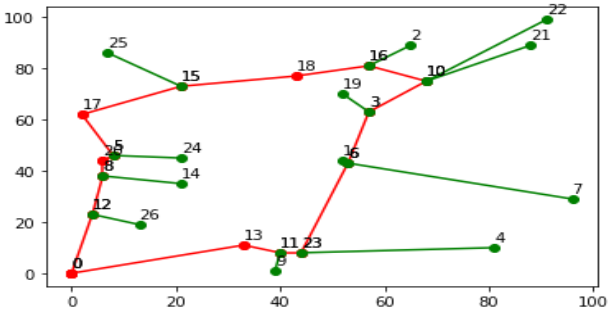
K=5



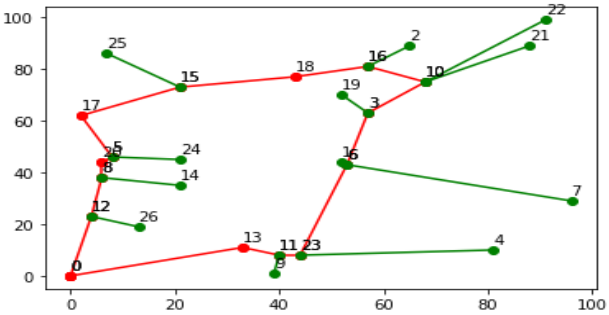
Case	Truck Route	Drone Route	Cost (in $)
20 Orders, 2 Drones	0-1-3-16-12-11-18-5-9-13-7-6-15-17-4-8-21	5-19-5, 7-2-7, 16-10-16, 16-14-16, 18-20-18	$3620
20 Orders, 3 Drones	0-8-4-17-15-6-7-13-9-5-18-11-12-16-3-1-21	5-19-5, 7-2-7, 16-10-16, 16-14-16, 18-20-18	$3620
20 Orders, 4 Drones	0-1-3-16-12-11-18-5-9-13-7-6-15-17-4-8-21	5-19-5, 7-2-7, 16-10-16, 16-14-16, 18-20-18	$3620
20 Orders, 5 Drones	0-8-4-17-15-6-7-13-9-5-18-11-12-16-3-1-21	5-19-5, 7-2-7, 16-10-16, 16-14-16, 18-20-18	$3620

- **26 Orders (2,3,4,5 Drones)**

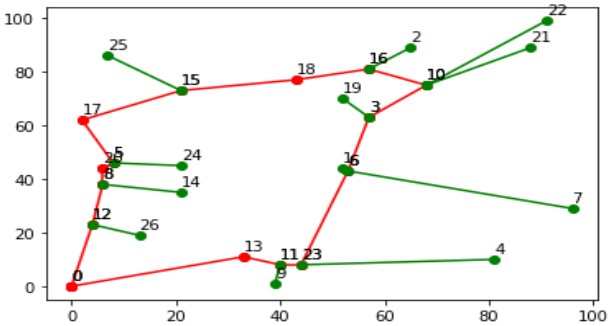
K=2



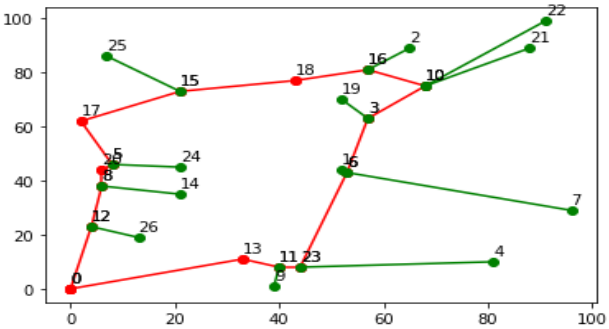
K=3



K=4



K=5



Case	Truck Route	Drone Route	Cost (in $)
26 Orders, 2 Drones	0-13-11-23-6-3-10-16-18-15-17-5-20-8-12-27	3-19-3, 5-24-5, 6-1-6, 6-7-6, 8-14-8, 10-21-10, 10-22-10, 11-9-11, 12-26-12, 15-25-15, 16-2-16, 23-4-23	$3894
26 Orders, 3 Drones	0-13-11-23-6-3-10-16-18-15-17-5-20-8-12-27	3-19-3, 5-24-5, 6-1-6, 6-7-6, 8-14-8, 10-21-10, 10-22-10, 11-9-11, 12-26-12, 15-25-15, 16-2-16, 23-4-23	$3894
26 Orders, 4 Drones	0-12-8-20-5-17-15-18-16-10-3-6-23-11-13-27	3-19-3, 5-24-5, 6-1-6, 6-7-6, 8-14-8, 10-21-10, 10-22-10, 11-9-11, 12-26-12, 15-25-15, 16-2-16, 23-4-23	$3894
26 Orders, 5 Drones	0-12-8-20-5-17-15-18-16-10-3-6-23-11-13-27	3-19-3, 5-24-5, 6-1-6, 6-7-6, 8-14-8, 10-21-10, 10-22-10, 11-9-11, 12-26-12, 15-25-15, 16-2-16,	$3894

		23-4-23	

Additional Experiments for Sensitivity Analysis

Analyzing the difference in objective function upon reducing the cost to half ($1.5) –

Nodes	K	$3	$1.5
15	2	$3300	$2586
	3	$3282	$2577
	4	$3282	$2576
	5	$3282	$2576
20	2	$3620	$3137
	3	$3620	$3123
	4	$3620	$3080
	5	$3620	$3093

Conclusion

- Upon analyzing the sensitivity relation, it has been observed that on decreasing the drone cost per distance unit, more drones are being released when the value of K is increased as opposed to the case when the cost of drones was $3 where increasing the drone limit was not affecting the number of drones being released from a particular node.

References

- Chase C. Murray and Amanda G. Chu, “The Flying Sidekick Traveling Salesman Problem: Optimization of Drone-assisted Parcel Delivery,” Transportation Research Part C: Emerging Technologies, 54, 86-109, 2015 – (http://www.chasemurray.com/wp-content/uploads/2016/04/The_Flying_Sidekick_TSP.pdf)

Appendices

- **Appendix A: Dataset co-ordinates used**

Type	15-Nodes		20-Nodes		26-Nodes	
Axis	**X**	**y**	**X**	**y**	**X**	**y**
Depot Point	0	0	0	0	0	0
Customer	98	47	2	24	52	44
Customer	35	53	89	4	65	89
Customer	59	8	31	68	57	63
Customer	44	8	34	4	81	10
Customer	17	61	73	55	8	46
Customer	40	87	59	34	53	43
Customer	61	81	82	20	96	29
Customer	34	50	22	9	6	38
Customer	58	20	86	35	39	1
Customer	13	88	16	99	68	75
Customer	10	69	48	79	40	8
Customer	9	41	44	78	4	23
Customer	46	30	87	28	33	11
Customer	73	3	6	100	21	35
Customer	65	85	46	22	21	73
Customer	--	--	35	86	57	81
Customer	--	--	41	17	2	62
Customer	--	--	56	75	43	77
Customer	--	--	77	60	52	70
Customer	--	--	90	99	6	44
Customer	--	--	--	--	88	89
Customer	--	--	--	--	91	99
Customer	--	--	--	--	44	8
Customer	--	--	--	--	21	45
Customer	--	--	--	--	7	86
Customer	--	--	--	--	13	19

- **Appendix B: Xpress-MP Code**

Example Code For 15 customers, 2 drones

```
!@encoding CP1252

model ModelName

uses "mmxprs"; !gain access to the Xpress-Optimizer solver

setparam("XPRS_MAXTIME", 120)
```

```
!sample declarations section
declarations
L = 16 !Number of points in our dataset
C = 0..L !Number of Locations
k = 2 !Max Drones allowed from a launch site
M = 1000 !Big number to faciliate some constraints
Dt: array(C,C) of integer !Distance Matrix
st: array(C,C) of mpvar !switch if truck goes from i to j
sd: array(C,C) of mpvar !switch if drone goes from i to j
p: array(C) of mpvar !position nodes
t: array(C) of mpvar !nodes denoting truck point
d: array(C) of mpvar !nodes denoting truck point
end-declarations
!Brute Force
Dt:: [0,109,64,60,45,63,96,101,60,61,89,70,42,55,73,107,0,
109,0,63,55,67,82,70,50,64,48,94,91,89,55,51,50,109,
64,63,0,51,46,20,34,38,3,40,41,30,29,25,63,44,64,
60,55,51,0,15,68,81,73,49,12,92,78,60,26,15,77,60,
45,67,46,15,0,59,79,75,43,18,86,70,48,22,29,80,45,
63,82,20,68,59,0,35,48,20,58,27,11,22,42,81,54,63,
96,70,34,81,79,35,0,22,37,69,27,35,55,57,90,25,96,
101,50,38,73,75,48,22,0,41,61,49,52,66,53,79,6,101,
60,64,3,49,43,20,37,41,0,38,43,31,27,23,61,47,60,
61,48,40,12,18,58,69,61,38,0,82,69,53,16,23,65,61,
89,94,41,92,86,27,27,49,43,82,0,19,47,67,104,52,89,
70,91,30,78,70,11,35,52,31,69,19,0,28,53,91,57,70,
42,89,29,60,48,22,55,66,27,53,47,28,0,39,74,71,42,
55,55,25,26,22,42,57,53,23,16,67,53,39,0,38,58,55,
73,51,63,15,29,81,90,79,61,23,104,91,74,38,0,82,73,
107,50,44,77,80,54,25,6,47,65,52,57,71,58,82,0,107,
```

```
0,109,64,60,45,63,96,101,60,61,89,70,42,55,73,107,0]
forall (i in C, j in C) do
        st(i,j) is_binary
        sd(i,j) is_binary
end-do
forall (i in C) do
        t(i) is_binary
        d(i) is_binary
end-do
forall (i in C) do
        p(i) is_integer
end-do
!Define Objective
obj:= sum(i in C)sum(j in C)Dt(i,j)*st(i,j)*10 + sum(i in C)sum(j in C)Dt(i,j)*sd(i,j)*3

p(0) = 0
p(L) = L
!Ensuring truck doesn't go from node 11
sum(j in C) st(L,j) = 0
!Ensuring truck doesn't enter node 0
sum(i in C) st(i,0) = 0
!Ensuring truck surely leaves depot
sum(j in C) st(0,j) = 1
!Ensuring truck ends at dummy point
sum(i in C) st(i,L) = 1
!Truck doesn't go to dummy from depot
st(0,L) = 0
!Any node should have maximum of 1 truck entering and exiting
forall(i in C) sum(j in C) st(i,j) <= 1
forall(i in C) sum(j in C) st(j,i) <= 1
```

```
!Ensuring truck and drone doesnot stay at the same node
forall (j in C) do
sum(i in C| i = j)st(i,j) = 0
sum(i in C| i = j)sd(i,j) = 0
end-do
!assigning positions to nodes
forall(i in C,j in C) do
p(i)+ 1 - M*(1 - st(i,j)) <= p(j)
p(j) <= L
end-do
!Ensuring drone doesn't go from node 0 and 11
forall(j in C)sd(0,j) = 0
forall(j in C)sd(L,j) = 0
!Ensuring drone doesn't go to node 0 and 11
forall(j in C)sd(j,0) = 0
forall(j in C)sd(j,L) = 0
!Ensure either position is truck or drone
forall(i in C) do
d(i) + t(i) = 1
end-do
!Ensuring the ending truck node is switched on if that arc is on
forall(i in 0..L-1) do
sum(j in 1..L) st(i,j) = t(i)
end-do
!Ensuring the starting truck node is switched on if that arc is on
forall(j in 1..L) do
sum(i in 0..L-1) st(i,j) = t(j)
end-do
!Ensuring the drone node is switched on if that arc is on
forall(j in 1..L-1) do
```

```
sum(i in 1..L-1) sd(i,j) = d(j)
end-do
!Ensuring only limited number of drones go from any truck node
forall(i in 1..L-1) do
sum(j in 1..L-1) sd(i,j) <= k*t(i)
end-do
!Ensuring no drone goes between truckpoints
forall (i in C, j in C) do
sd(i,j) <= 2- t(i) - t(j)
end-do
!Ensuring no drone goes between drone points
forall (i in C, j in C) do
sd(i,j) <= 2-d(i) - d(j)
end-do
!Ensuring no truck between 2 drone supplied points
forall (i in C, j in C) do
st(i,j) <= 2-d(i) - d(j)
end-do
!forall(i in C) do
!sum(j in C| i<>j) (sd(i,j) + sd(j,i)) >= 1
!end-do
!Ensuring drone comes back to the same node -- Not necessary as it will make the return j point as d(j) =
1
!forall(i in C,j in C) sd(i,j) = sd(j,i)
!Truck final points to be beginning pt of drone
!forall (i in C) do
!sum(j in 1..L-1) st(i,j) >= sum(j in 1..L-1)sd(i,j)
!end-do
!if truck enters a node it must leave that node  -- Not necessary as t(i) + d(i) = 1 is ensuring this
!forall(i in 1..L-1) sum(j in 1..L-1) st(i,j) = sum(j in 1..L-1) st(j,i)
```

```
!Any node should have maximum of 1 drone entering and exiting
!forall(i in C) sum(j in C)sd(i,j) <= 1
!forall(i in C) sum(j in C)sd(j,i) <= 1
minimize(obj)
writeln("Begin running model")
writeln("Optimal Objective: ",getobjval)
forall(i in C, j in C|getsol(st(i,j))>0) writeln("st_",i,",",j,": ",getsol(st(i,j)))
forall(i in C, j in C|getsol(sd(i,j))>0) writeln("sd_",i,",",j,": ",getsol(sd(i,j)))
forall(i in C) writeln("p_",i,": ",getsol(p(i)))
forall(i in C|getsol(d(i)) > 0) writeln("d_",i,": ",getsol(d(i)))
forall(i in C|getsol(t(i)) > 0) writeln("t_",i,": ",getsol(t(i)))
writeln("End running model")
end-model
```

- **Appendix C: Python Code**

Example Code for 15 Customers, 2 Drones

```
import numpy as np
import matplotlib.pyplot as plt
x=[0, 98, 35, 59, 44, 17, 40, 61, 34, 58, 13, 10, 9, 46, 73, 65]
y=[0, 47, 53, 8, 8, 61, 87, 81, 50, 20, 88, 69, 41, 30, 3, 85]
path_truck = [0, 4, 13, 8, 2, 5, 12, 0]
path_drone = [2, 6, 2, 15, 4, 3, 4, 14, 5, 10, 5, 11, 8, 7, 13, 1, 13, 9]
x_truck=[]
y_truck=[]
for i in path_truck:
   x_truck = x_truck+[x[i]]
   y_truck = y_truck+[y[i]]
x_drone=[]
y_drone=[]
for i in path_drone:
   x_drone = x_drone+[x[i]]
```

```
    y_drone = y_drone+[y[i]]
for i in range(0,len(x_truck)):
    plt.plot(x_truck[i:i+2],y_truck[i:i+2],'ro-')
    label = path_truck[i]
    plt.annotate(label,(x_truck[i],y_truck[i]),textcoords="offset points",xytext=(0,5))
for i in range(0,len(x_drone),2):
    plt.plot(x_drone[i:i+2],y_drone[i:i+2],'ro-',color='green')
for i in range(0,len(x_drone),1):
    label = path_drone[i]
    plt.annotate(label,(x_drone[i],y_drone[i]),textcoords="offset points",xytext=(0,5))
plt.show()
```