Mathematical Optimisation

mTSP-DS

The multiple traveling salesman problem in presence of drone- and robot-supported packet stations

Davide Panarella

Sections

mTSP-DS

Problem Description

Section 1

The Paper

European Journal of Operational Research 305 (2023) 630–643

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing, Transportation and Logistics

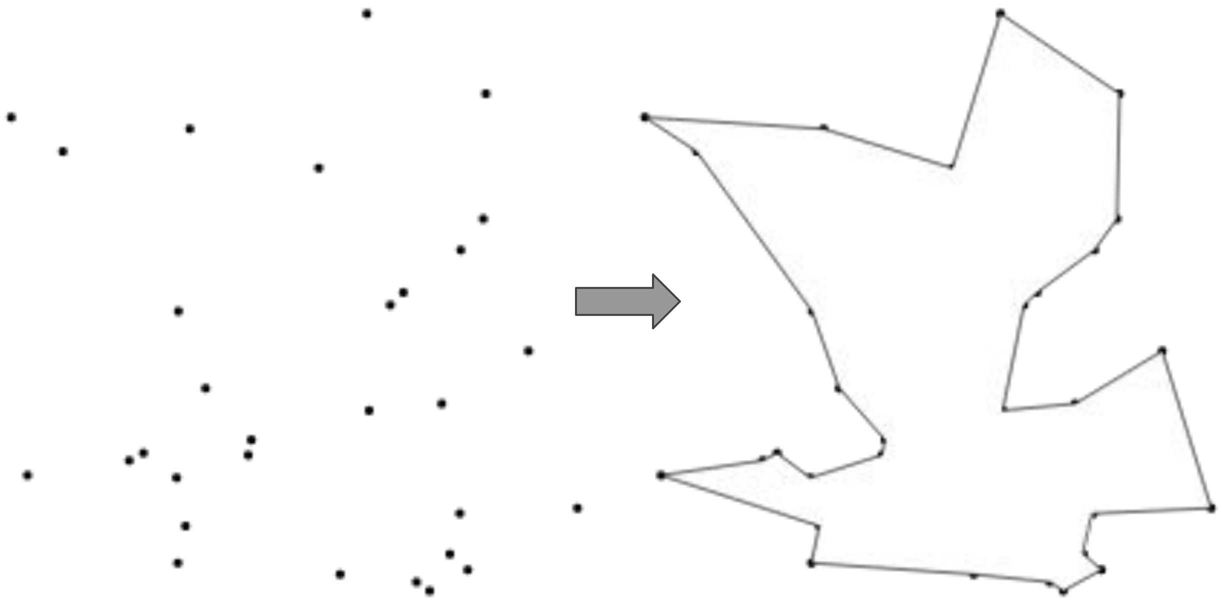
The multiple traveling salesman problem in presence of drone- and robot-supported packet stations

Konstantin Kloster[a], Mahdi Moeini[a,*], Daniele Vigo[b], Oliver Wendt[a]

[a] Chair of Business Information Systems and Operations Research (BISOR), Technische Universität Kaiserslautern, Kaiserslautern 67663, Germany
[b] Department of Electrical, Electronic, and Information Engineering "G. Marconi", University of Bologna, Bologna 40136, Italy

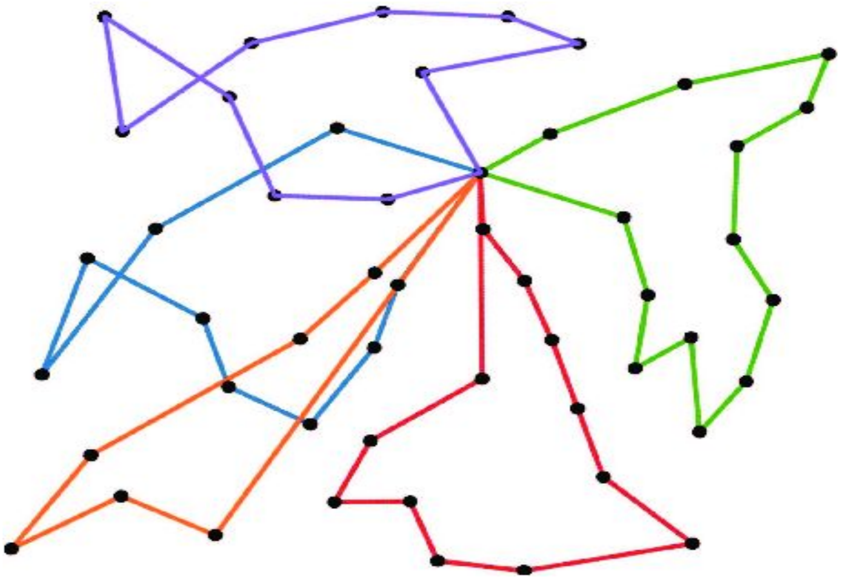
TSP Problem

Objective: Identify the Hamiltonian tour of minimum cost.



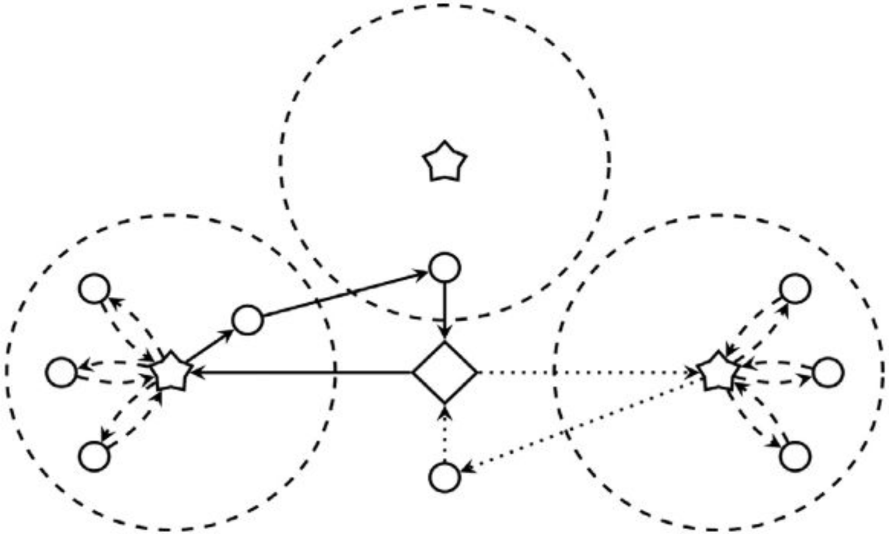
mTSP Problem

Objective: Identify the set of Hamiltonian tours with the minimum *overall* cost.



mTSP-DS Problem

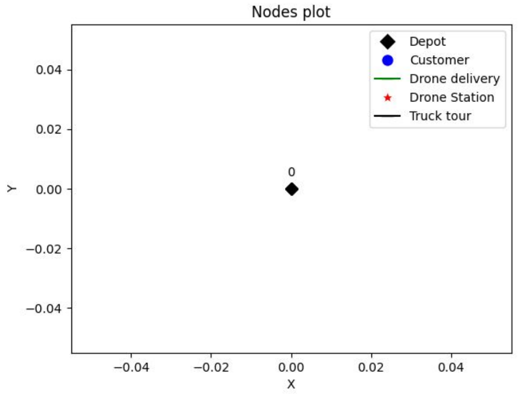
Objective: Identify the set of Hamiltonian tours with the minimum *overall* cost using trucks and/or autonomous vehicles (Drones/AGVs)



Actors

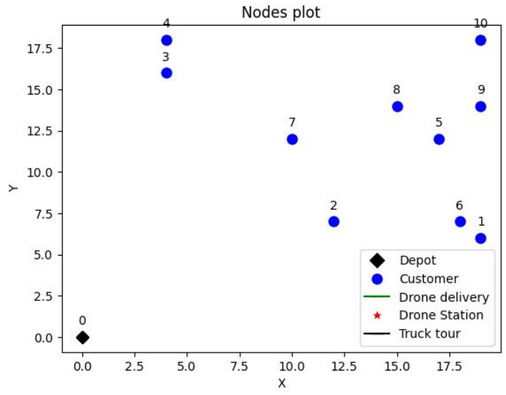
Depot

- Start and end of each tour



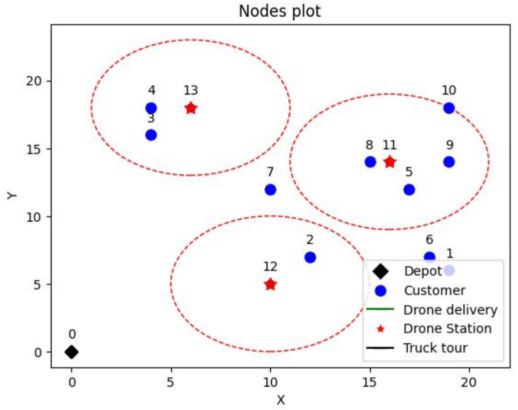
Customer

- Nodes to be served by truck or by drone



Drone Station

- Starting point for drones or AGVs



mTSP-DS

Mathematical Formulation

Section 2

Parameters

Table 1: Problem Parameters

Parameter	**Description**
$n \in Z^+$	Number of customers to be served.
$m \in Z^+$	Number of drone stations.
$K_N \in Z^+$	Total number of trucks, $K = \{1, \ldots, K_N\}$.
$C \in Z^+$	Maximum number of drone stations that can be activated, $C \leq m$.
$D_N \in Z^+$	Number of available drones per drone station, $D = \{1, \ldots, D_N\}$.
$\alpha \in R^+$	Drone velocity relative to truck speed: $\alpha > 1$: drones faster, $\alpha < 1$: drones slower, $\alpha = 1$: same speed.
$\varepsilon \in R^+$	Maximum distance covered by drones without recharging, Drone range: $\varepsilon_r = \varepsilon/2$.

mTSP-DS

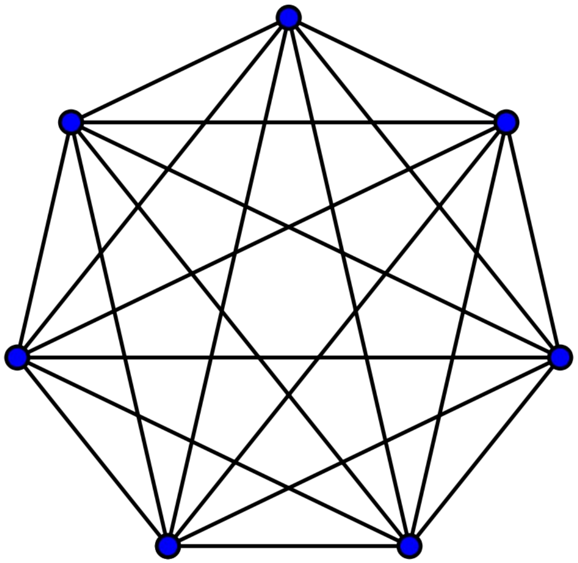
MILP Formulation

Section 2.1.1

Mathematical Description

$G = (V, E)$

V:	Set of nodes
$V_n = \{1, 2, \ldots n\}$:	Customers
$V_s = \{s1, s2, \ldots sm\}$:	Drone Stations
$\{0\}$ and $\{n+1\}$:	Depot
$V = \{0\} \cup V_n \cup V_s \cup \{n+1\}$	
$V_r = V \setminus \{0\}$:	Set of initial nodes of arcs
$V_l = V \setminus \{n+1\}$:	Set of ending nodes of arcs



Decision Variables

τ	$\in \mathbb{R}_{\geq 0}$	:	A continuous variable that represents the makespan.
x_{ij}^{k} $\forall k \in K, i \in V_L, j \in V_R$	$\in \{0, 1\}$	:	Variables that are used to indicate whether truck k traverses edge (i, j).
y_{sj}^{d} $\forall d \in D, s \in V_S, j \in V_N$	$\in \{0, 1\}$	:	Variables that specify if customer j is served by drone d from drone station s.
z_s $\forall s \in V_S$	$\in \{0, 1\}$	:	Variables that state whether a drone station is activated.
a_i^k $\forall k \in K, i \in V$	$\in \mathbb{R}_{\geq 0}$	:	Continuous variables that indicate the time at which truck k arrives at node i.

Objective Function

Minimize makespan, defined as the latest arrival time of a **truck** at the *Depot* or of **Drone** at a *Drone Station*.

$$\min \quad \tau$$

Constraints: τ

Define τ through *lower bounds.*

s.t.

$$a_{n+1}^{k} \leq \tau \ : \forall k \in K, \tag{2}$$

$$a_{s}^{k} + \sum_{j \in V_N} 2 \cdot \bar{t}_{sj} \cdot y_{sj}^{d} \leq \tau \ : \forall k \in K, s \in V_s, d \in D, \tag{3}$$

Constraints: Flow conditions

$$\sum_{k \in K} \sum_{\substack{i \in V_L \\ i \neq j}} x_{ij}^k + \sum_{s \in V_s} \sum_{d \in D} y_{sj}^d = 1 : \forall j \in V_N, \tag{4}$$

$$\sum_{j \in V_N} x_{0j}^k = \sum_{i \in V_N} x_{i,n+1}^k = 1 : \forall k \in K, \tag{5}$$

$$\sum_{\substack{i \in V_L \\ i \neq h}} x_{ih}^k - \sum_{\substack{j \in V_R \\ h \neq j}} x_{hj}^k = 0 : \forall k \in K, h \in V_N \cup V_S, \tag{6}$$

$$\sum_{k \in K} \sum_{\substack{i \in V_L \\ i \neq s}} x_{is}^k \leq 1 : \forall s \in V_S, \tag{7}$$

Constraints: Activation of Drone Stations

$$\sum_{k \in K} \sum_{\substack{i \in V_L \\ i \neq s}} x_{is}^{k} = z_s : \forall s \in V_S, \tag{8}$$

$$\sum_{s \in V_S} z_s \leq C, \tag{9}$$

Constraints: Drone Operations

$$\sum_{d \in D} \sum_{j \in V_N} y_{sj}^{d} \leq n z_s : \forall s \in V_S, \tag{10}$$

$$2 \cdot \overline{d}_{sj} \cdot y_{sj}^{d} \leq \mathcal{E} : \forall s \in V_S, d \in D, j \in V_N, \tag{11}$$

Constraints: Travel Time

$$M(x_{ij}^k - 1) + a_i^k + t_{ij} \leq a_j^k : \forall k \in K, i \in V_L, j \in V_R, i \neq j, \quad (12)$$

- *M is a sufficiently large number.*
- *If all nodes have unique coordinates, subtours are eliminated.*

Lazy Constraints: Subtour Elimination

Ensure the number of selected edges in each non-empty subset $S \subset V$, excluding the depot, does not exceed $|S| - 1$.

$$\sum_{i \in \mathcal{S}} \sum_{\substack{j \in \mathcal{S} \\ i \neq j}} x_{ij}^{k} \leq |\mathcal{S}| - 1 : \forall k \in K, \mathcal{S} \subset V, \{0, n+1\} \notin \mathcal{S}, |\mathcal{S}| > 1. \tag{13}$$

Implemented as **lazy constraints** using a *callback function*.

mTSP-DS

Implementation

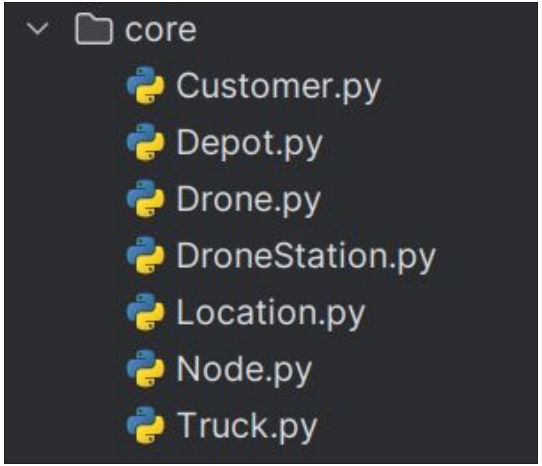
Section 2.1.2

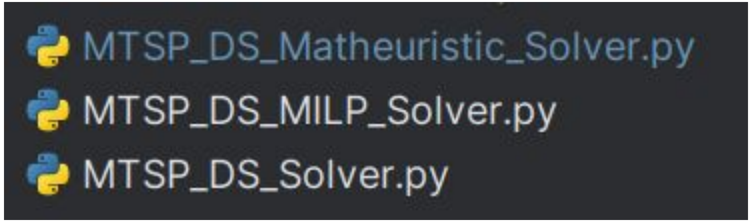
Tools

- Python (version 3.10.12)
- Gurobi (version 11.0.0 build v11.0.0rc2)
- Github

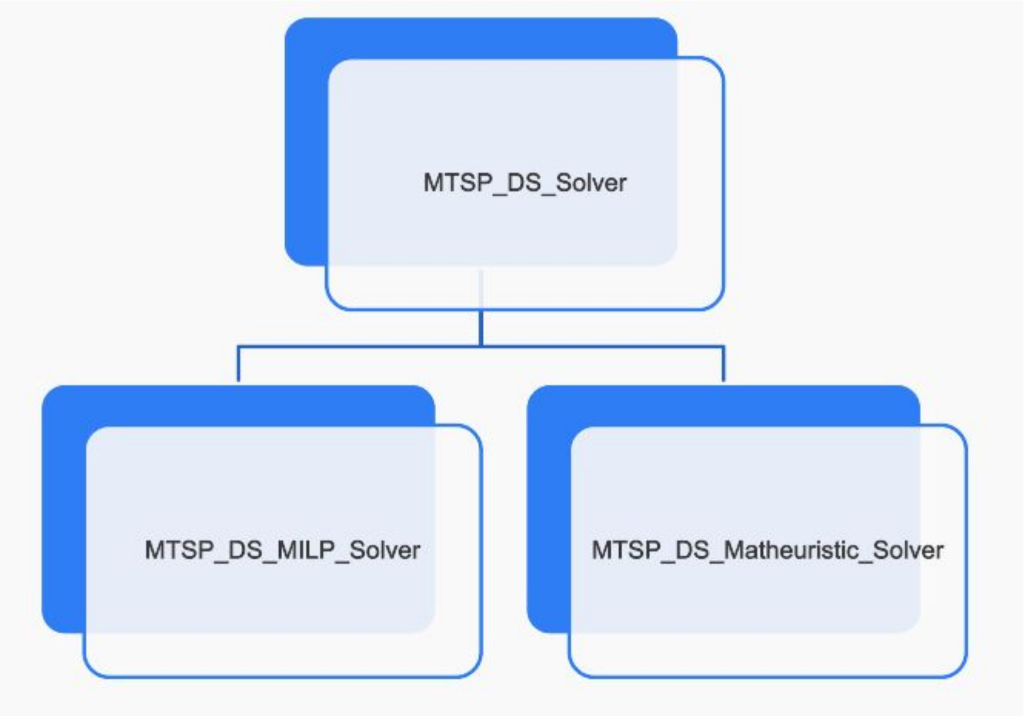
Code Structure

- **Core elements**
  - Location
  - Node
    - Customer
    - Depot
    - DroneStation
  - Truck & Drone
- **Solvers**
- **TourUtils**
- **Analysis Notebooks**





Code Structure - Solver Inheritance



Inheritance for solver's common methods:

- Node and index initialization
- Distance matrix
- Load and save node configurations
- Gurobi model utilities

Project Features

- ❏ Random initialization
- ❏ VRP loader for custom setups
- ❏ Custom mTSP-DS plot
- ❏ Valid inequalities to enhance performance
- ❏ Lazy constraints for subtour elimination

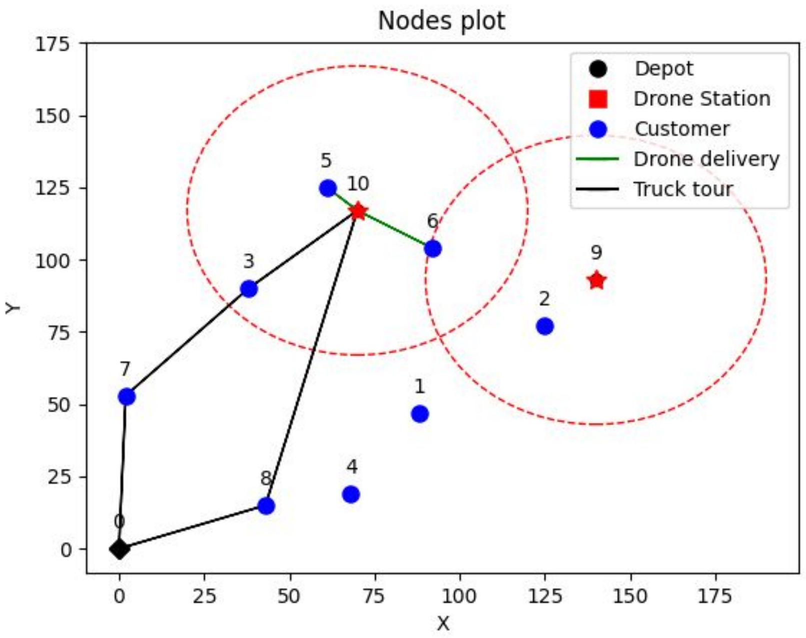
Lazy Constraints: Subtour Elimination

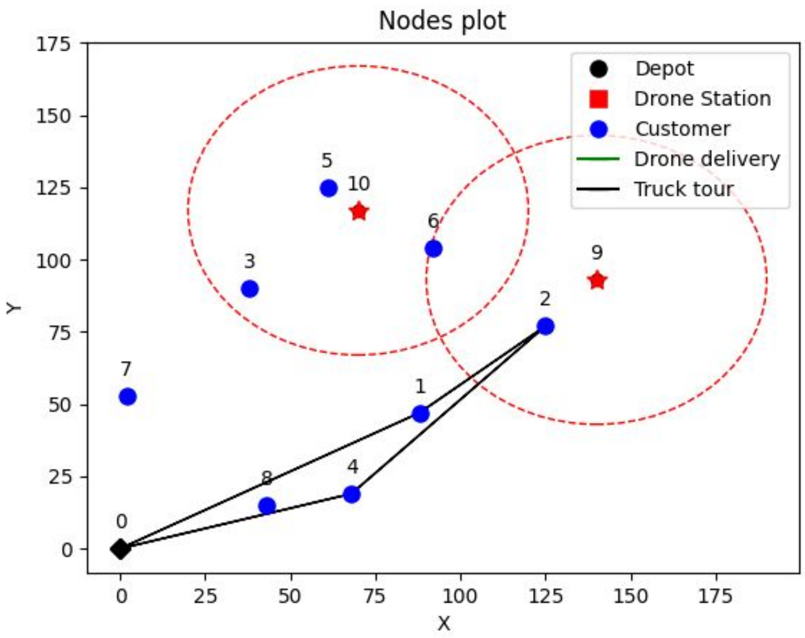
Implemented as **lazy constraints** using a *callback function*.

```
1 usage   Spopoi
def subtourelim(self, model, where):
    if where == GRB.Callback.MIPSOL:
        tours = getTrucksTour_callback(model)
        truck_index = 1
        x_k_ij = model._edges
        for truck_tour in tours:
            node_indexes = getVisitedNodesIndex(truck_tour)
            sub_tours_indexes = generate_sub_tours_indexes(node_indexes[1:-1])
            # Constraint (13)
            for S in sub_tours_indexes:
                model.cbLazy(
                    gp.quicksum(gp.quicksum(x_k_ij[truck_index, i, j] for j in S if i != j) for i in S)
                    <= len(S) - 1)
                model.update()
            truck_index += 1

2 usages   Spopoi
def solve(self):
    self.model.optimize(self.subtourelim)
```

Plots


Nodes plot
Depot
Drone Station
Customer
Drone delivery
Truck tour
175
150
125
100
75
50
25
0
0
25
50
75
100
125
150
175
X
Y
0
1
2
3
4
5
6
7
8
9
10


Nodes plot
Depot
Drone Station
Customer
Drone delivery
Truck tour
175
150
125
100
75
50
25
0
0
25
50
75
100
125
150
175
X
Y
0
1
2
3
4
5
6
7
8
9
10

mTSP-DS

Matheuristic Formulation

Section 2.2.1

Matheuristic Decomposition

Decompose the mTSP-DS into *easier-to-handle* **subproblems**:

- Drone Station Location Problem
- Allocation Problem
- Sequencing Problem
- Assignment Problem
- Scheduling Problem

Routing Problem

Matheuristic Algorithm

Algorithm 1: mTSP-DS Matheuristic.

```
init solution_pool
d_station_combos = get_all_d_station_combos(V_S, C)
foreach d_station_combo ∈ d_station_combos do
    solution = get_mtsp_tours(depot, V_N, d_station_combo)
    foreach d_station ∈ d_station_combo do
        solution = solve_local_dasp(solution, d_station)
    solution_pool.add(solution)
best_solution = get_best_solution(solution_pool)
best_solution = post_processing(best_solution)
return best_solution
```

Matheuristic Algorithm

Algorithm 1: mTSP-DS Matheuristic.

```
init solution_pool
d_station_combos = get_all_d_station_combos(V_S, C)
foreach d_station_combo ∈ d_station_combos do
    solution = get_mtsp_tours(depot, V_N, d_station_combo)
    foreach d_station ∈ d_station_combo do
        solution = solve_local_dasp(solution, d_station)
    solution_pool.add(solution)
best_solution = get_best_solution(solution_pool)
best_solution = post_processing(best_solution)
return best_solution
```

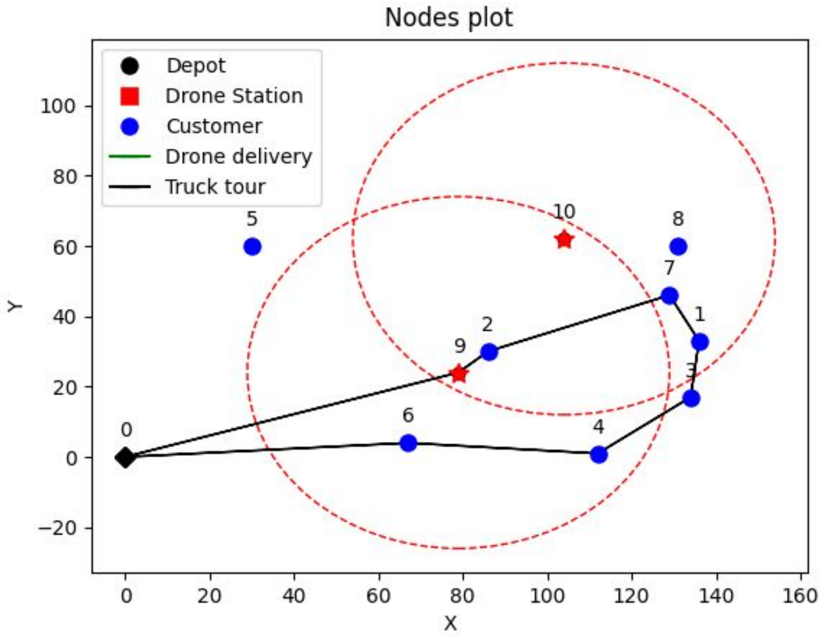
Drone Station Location Problem

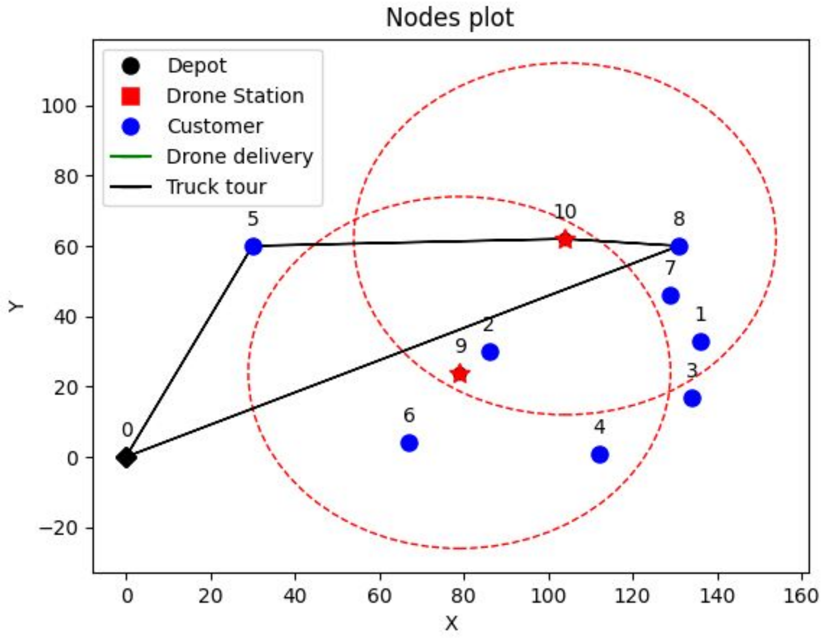
Routing Problem

Drone Assignment and Scheduling Problem (DASP)

Matheuristic - Routing Problem

MILP Solver to find optimal truck tours.





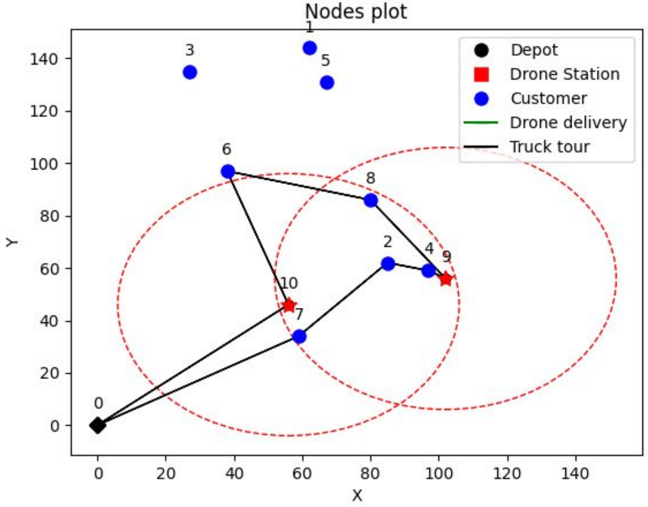
mTSP-DS

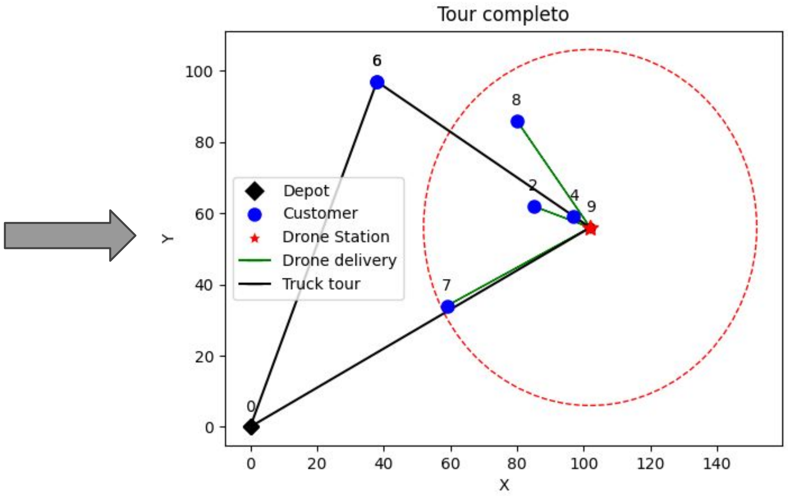
Local DASP

Section 2.2.2

Local DASP

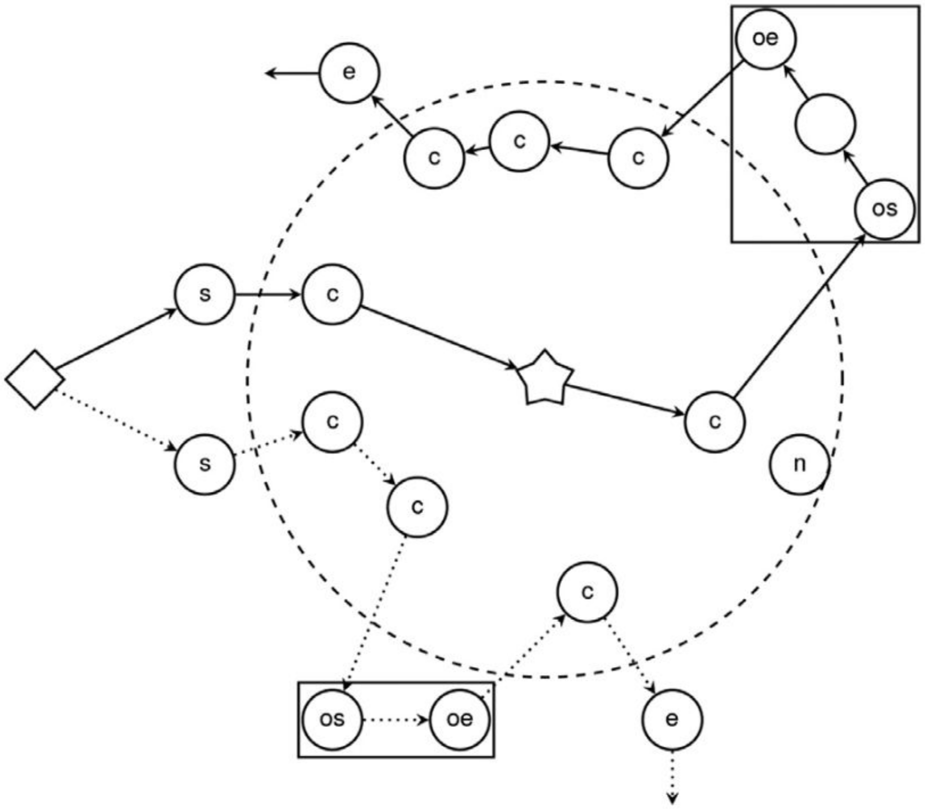
- Determine which customers within a *single drone station's range* should be served by **trucks** and which by **drones**.
- Schedules the drone deliveries.





Local DASP - Nodes

- Start Nodes (Vstart)
- End Nodes (Vend)
- Outliers (O)
  - Start (V_OS)
  - End (V_OE)
- Customer (Vn)



Local DASP - Nodes

- *First* nodes of an *arc*:

  $Vl = Vstart \cup Vn \cup \{ds\} \cup V\_OE$

- *Second* nodes of an *arc*:

  $Vr = Vn \cup \{ds\} \cup V\_OS \cup Vend$

- *Nodes*:

  $V = Vstart \cup Vn \cup \{ds\} \cup V\_OS \cup V\_OE \cup Vend$

Local DASP - Decision Variables

$$\overline{\tau} \in \mathbb{R}_{\geq 0} \quad : \quad \text{A continuous variable representing the makespan.}$$

$$\underset{\substack{\forall k \in K, i \in V_L, j \in V_R \\ \wedge \forall k \in K, (os, oe) \in O}}{x_{ij}^k} \in \{0, 1\} \quad : \quad \text{Variables that indicate whether truck } k \text{ traverses arc } (i, j).$$

$$\underset{\forall d \in D, j \in V_N}{y_j^d} \in \{0, 1\} \quad : \quad \text{Variables that specify if customer } j \text{ is served by drone } d \text{ from drone station } ds.$$

$$\underset{\forall k \in K, i \in V}{a_i^k} \in \mathbb{R}_{\geq 0} \quad : \quad \text{Continuous variables that indicate the time at which truck } k \text{ arrives at node } i.$$

Local DASP - Objective Function

Minimize the *makespan*, defined as the latest arrival time of a **truck** at the *Depot* or of **Drone** at a *Drone Station*.

$$\min \quad \overline{\tau}$$

Local DASP - τ Constraints

Define τ through *lower bounds*.

$$a^k_{end_k} \leq \overline{\tau} : \forall k \in K, \tag{16}$$

$$a^k_{ds} + \sum_{j \in V_N} 2 \cdot \bar{t}_{ds,j} \cdot y^d_j \leq \overline{\tau} : \forall k \in K, d \in D, \tag{17}$$

Local DASP - Tour Constraints

$$\sum_{k \in K} \sum_{\substack{i \in V_L \\ i \neq j}} x_{ij}^k + \sum_{d \in D} y_j^d = 1 : \forall j \in V_N, \tag{18}$$

$$\sum_{\substack{j \in V_N \cup ds \\ \cup V_{OS} \cup end_k}} x_{start_k, j}^k = \sum_{\substack{i \in V_N \cup ds \\ \cup V_{OE} \cup start_k}} x_{i, end_k}^k = 1 : \forall k \in K, \tag{19}$$

$$\sum_{k \in K} \sum_{\substack{i \in V_L \\ i \neq ds}} x_{i,ds}^k = \sum_{k \in K} \sum_{\substack{j \in V_R \\ j \neq ds}} x_{ds,j}^k = 1, \tag{20}$$

Local DASP - Outliers Constraints

$$\sum_{k \in K} \sum_{\substack{i \in V_L \\ i \neq oe}} x^k_{i,os} = 1 : \forall (os, oe) \in O, \tag{21}$$

$$\sum_{k \in K} \sum_{\substack{j \in V_R \\ j \neq os}} x^k_{oe,j} = 1 : \forall (os, oe) \in O, \tag{22}$$

$$\sum_{i \in V_L} x^k_{i,os} - x^k_{os,oe} = 0 : \forall k \in K, (os, oe) \in O, \tag{23}$$

Local DASP - Arrival Time Constraints

$$cost\_to\_start\_k = a^k_{start_k} : \forall k \in K, \tag{25}$$

$$M(x^k_{ij} - 1) + a^k_i + t_{ij} \leq a^k_j : \begin{array}{l} \forall k \in K, i \in V_L, \\ j \in V_N \cup ds \cup V_{OS}, i \neq j, \end{array} \tag{26}$$

$$\begin{array}{l} M(x^k_{os,oe} - 1) + a^k_{os} + \\ traversal\_cost(os, oe) \end{array} \leq a^k_{oe} : \forall k \in K, (os, oe) \in O, \tag{27}$$

$$\begin{array}{l} M(x^k_{i,end_k} - 1) + a^k_i + t_{i,end_k} + \\ cost\_after\_end_k \end{array} \leq a^k_{end_k} : \forall k \in K, i \in V_L, \tag{28}$$

Local DASP - Issues

Not **all** possible *outlier node configurations* are handled.

It is allowed for a truck to enter the outlier_start node and for *another one* to exit from the outlier_end node.

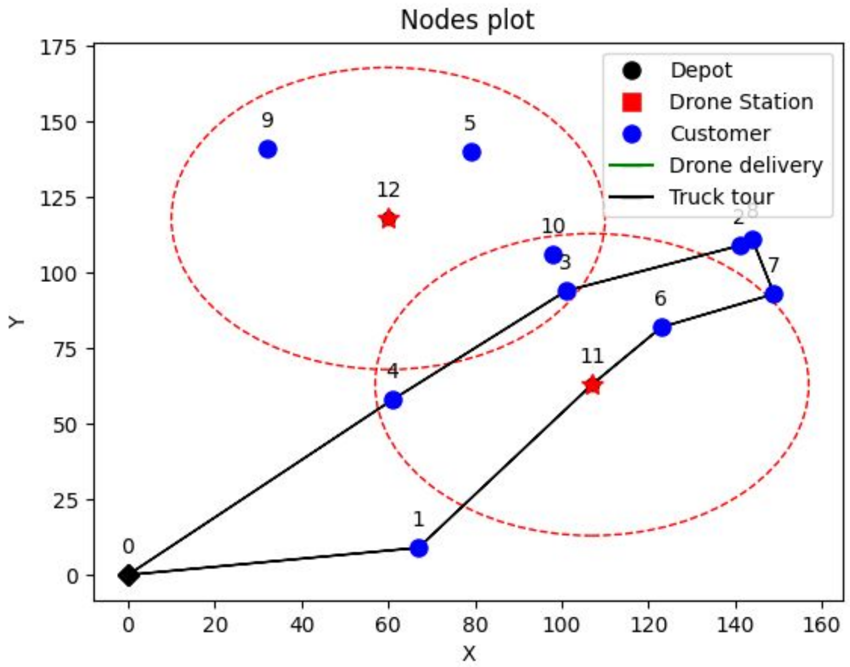
$$\sum_{k \in K} \sum_{\substack{i \in V_L \\ i \neq oe}} x^k_{i,os} = 1 : \forall (os, oe) \in O, \tag{21}$$

$$\sum_{k \in K} \sum_{\substack{j \in V_R \\ j \neq os}} x^k_{oe,j} = 1 : \forall (os, oe) \in O, \tag{22}$$

$$\sum_{i \in V_L} x^k_{i,os} - x^k_{os,oe} = 0 : \forall k \in K, (os, oe) \in O, \tag{23}$$

$$\sum_{\substack{i \in V_L \\ i \neq h}} x^k_{ih} - \sum_{\substack{j \in V_R \\ j \neq h}} x^k_{hj} = 0 : \forall k \in K, h \in V_N, \tag{24}$$

Local DASP - Issues




```
V =  [Customer 1, Depot 0, Customer 6, Customer 3, C
V_Start =  [Customer 1, Depot 0]
V_End =  [Depot 13, Customer 5]
O_index:  [(7, 8)]
range(1, 3)
Tours:  [[<gurobi.Var x_k_ij[1,0,7] (value 1.0)>, <g
tour: [<gurobi.Var x_k_ij[1,0,7] (value 1.0)>, <guro
 k = 0
tuple_tour:  [(0, 7), (6, 9), (7, 8)]
Traceback (most recent call last):
  File "/home/davide/Units/MathematicalOptimisation/
    solver.solve()
```

mTSP-DS

Results & Conclusions

Section 3

mTSP vs mTSP-DS

Experimental conditions:

- Random node locations
- num_of_setups = 20
- n = 8
- m = 2
- Dn = 2
- C = 2
- Kn = 2
- eps = 100 (m)
- alpha = 1.2
- MILP Solver

Table 1: mTSP vs mTSP-DS

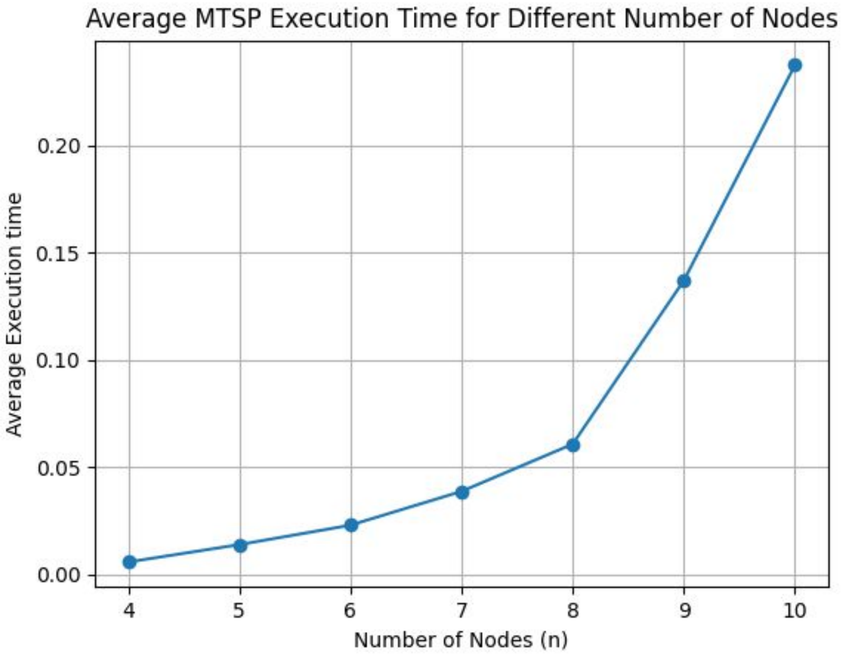
	mTSP	mTSP-DS
1	420.82365032010443	418.7741225953306
2	400.93675857161344	391.7104546718001
3	379.4398844791006	351.09715653424246
4	375.4667356586564	342.28642976314427
5	423.9105548342422	369.88954414698765
6	347.05378214592406	340.27048061325434
7	422.8527726976424	394.85279141289845
8	370.6590338829192	343.68590311503897
9	385.79845471001374	347.99861143120427
10	351.83460757223014	345.16399246050753
11	339.91175325369375	289.94482233694043
12	265.18665392823823	224.53756705674016
13	359.9195248905625	321.17907777437813
14	301.53099720593315	281.69993997222946
15	371.55962492593704	371.5596249259371
16	369.47040746219966	317.04258389055565
17	327.4848505589603	327.4848505589603
18	366.89508037039684	317.55788420847233
19	374.09678587917006	360.53697776771054
20	309.835862470817	275.13632984395207
Mean	309.835862470817	275.13632984395207
	Percentage of improvement of tau: 7.460779434090523	

Scalability Analysis - n

Experimental conditions:

- Random node locations
- num_of_setups = 4
- m = 2
- Dn = 2
- C = 2
- Kn = 2
- eps = 100 (m)
- alpha = 1.2
- MILP Solver

MILP

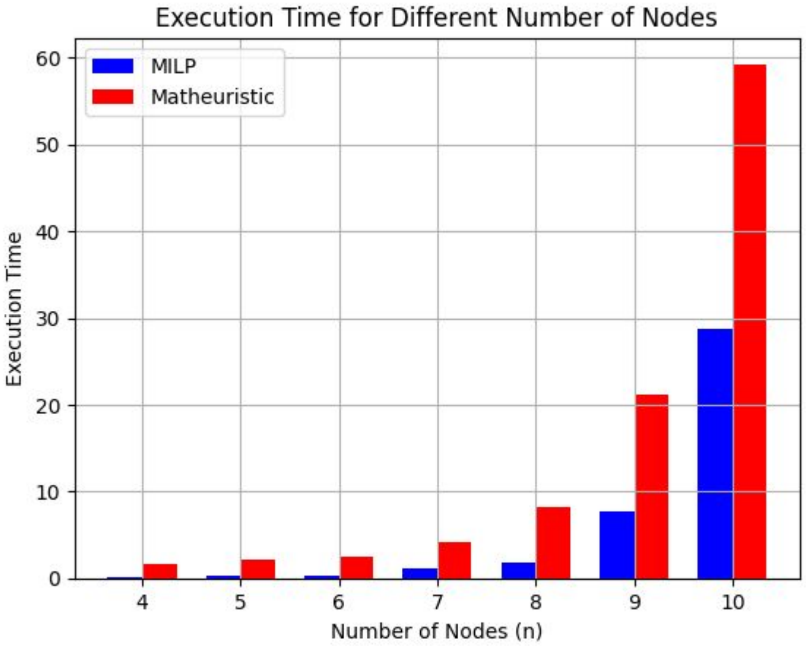


Scalability Analysis - n

Experimental conditions:

- Random node locations
- num_of_setups = 4
- m = 2
- Dn = 2
- C = 2
- Kn = 2
- eps = 100 (m)
- alpha = 1.2
- MILP Solver

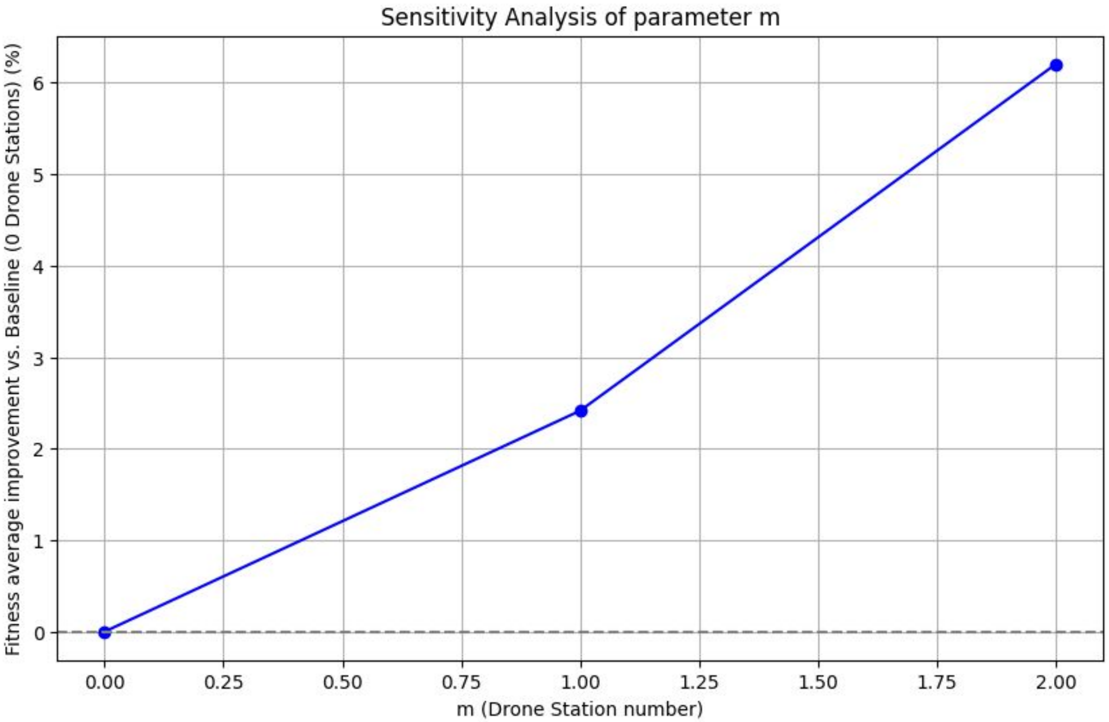
MILP vs Matheuristic



Sensitivity Analysis - m

Experimental conditions:

- *Same* random customer locations, *random* drone station placement
- num_of_setups = 3,
  ds configurations = 3
- n = 8
- Dn = 2, C = 2, eps = 100
- Kn = 2
- eps = 100 (m)
- alpha = 1.2
- MILP Solver



Conclusions

- mTSP-DS outperform mTSP
- The positioning of the drone stations is critic

References

- Source Code: https://github.com/Spopoi/mTSP-DS.git
- Paper: https://www.sciencedirect.com/science/article/pii/S0377221722004593

mTSP-DS

Thank you for your attention!

Davide Panarella