A heterogeneous vehicle routing problem with drones and multiple depots

Panagiotis Zachos

June 2024

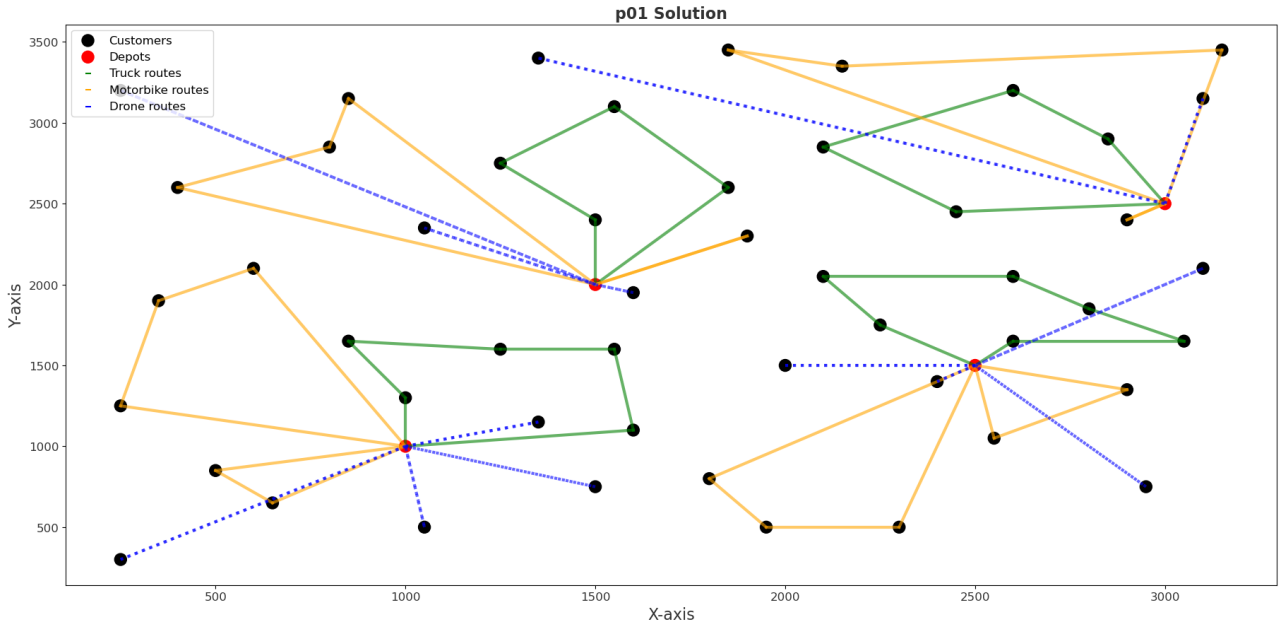
1 Στόχος

Επέκταση του mixed fleet capacitated multiple TSP (mfcmTSP) (Oikonomou et al. 2019) όπου ένα φορτηγό με απεριόριστη χωρητικότητα, μία μοτοσυκλέτα με αυθαίρετη χωρητικότητα και ένα drone με χωρητικότητα 1, πρέπει να επισκεφτούν πελάτες, οι οποίοι έχουν ζήτηση 1 δέμα ο καθένας, στον μικρότερο δυνατό χρόνο.

Κύρια χαρακτηριστικά του νέου προβλήματος **(Multi-Depot mfcmTSP):**

- Ανάθεση χωρητικότητας στα φορτηγά (έναντι απεριόριστης).
- Οι μοτοσυκλέτες μπορούν να επισκεφτούν όλους τους πελάτες.
- Τα φορτηγά μπορούν να επισκεφτούν μερικούς πελάτες μόνο, λόγω μεγάλου μεγέθους (στενοί δρόμοι, σοκάκια) (~90%).
- Τα drones μπορούν να επισκεφτούν μερικούς πελάτες μόνο, λόγω ακατάλληλων συνθηκών προσγείωσης ή λόγω προτίμησης πελατών (~85%).
- Πολλαπλές αποθήκες (>1), με την κάθε αποθήκη να διαθέτει τον δικό της στόλο οχημάτων. Συγκεκριμένα, η κάθε αποθήκη μπορεί να είναι εξοπλισμένη με [1,∞] φορτηγά, [0,∞] μοτοσυκλέτες και [0,∞] drones. Ωστόσο, τουλάχιστον μία αποθήκη θα πρέπει να εμπεριέχει στον στόλο της τουλάχιστον μία μοτοσυκλέτα έτσι ώστε να μπορούν να εξυπηρετηθούν όλοι οι πελάτες.

Figure 1: Solution example with 50 customers and 4 depots



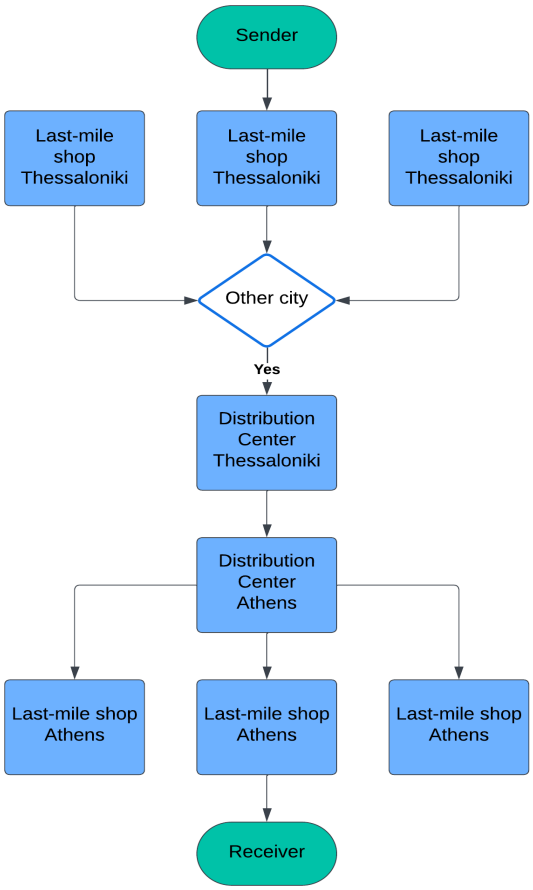
2 Περιεχόμενα

- Υλοποίηση του mfcmTSP (2019) heuristic , με προσθήκη υποστήριξης πολλαπλών οχημάτων κάθε είδους σε κάθε αποθήκη, και χρήση τεχνικής cluster first-route second για την προσαρμογή του αλγορίθμου στην εκδοχή του προβλήματος με πολλαπλές αποθήκες.
- Βελτίωση του αλγορίθμου και σύγκριση αποτελεσμάτων.
- Χρήση local search optimization για την περαιτέρω βελτίωση αποτελεσμάτων.
- Παρουσίαση ενός υβριδικού metaheuristic που κάνει χρήση του Ant Colony Optimization σε συνδυασμό με το heuristic.
- Σύγκριση των αλγορίθμων.

2.1 Motivation / Use case

Consider a parcel delivery company operating in Greece, with depots located in two major cities: Athens and Thessaloniki. The company manages numerous last-mile delivery shops in these cities, which serve as depots in the routing problem. For instance, a customer in Thessaloniki sends a parcel to a friend in Athens by visiting a nearby shop. After the shop stops receiving parcels for the day, a truck collects the parcels and delivers them to the Thessaloniki distribution center. Parcels destined for Athens are then transported overnight to the Athens distribution center. Upon arrival, parcels are sorted based on their delivery areas and sent to the appropriate last-mile shops in Athens. These shops, acting as depots in our problem, dispatch vehicles (Trucks, Motorbikes, Drones) to deliver the parcels to their final destinations. **The MD-mfcmTSP addresses how these vehicles can be optimally routed to minimize delivery time, while simultaneously solving the problem of parcel allocation in each last-mile shop**.

Figure 2: Life of a parcel



3 Objective

The objective of the MD-mfcmTSP is to minimize the total makespan in which all customers are visited/served i.e min M_{total}.

Notation

- m : Number of depots
- D : Set of depots
- R_T^i : Truck route of depot $i \in D$
- R_M^i : Motorbike route of depot $i \in D$
- R_D^i : Drone route of depot $i \in D$
- M_T^i : Time of the Truck's route(s) ($N_T^i = 1$) or Makespan of the Trucks' route(s) ($N_T^i > 1$) of depot $i \in D$
- M_M^i : Time of the Motorbike's route(s) ($N_M^i = 1$) or Makespan of Motorbikes' route(s) ($N_M^i > 1$) of depot $i \in D$
- M_D^i : Time of the Drone's route(s) ($N_D^i = 1$) or Makespan of Drones' route(s) ($N_D^i > 1$) of depot $i \in D$
- $M_{total}^i = max(M_T^i, M_M^i, M_D^i)$: Makespan of depot $i \in D$
- $M_T = max(M_T^i, M_T^{i+1}, ..., M_T^m)$: Makespan of Trucks
- $M_M = max(M_M^i, M_M^{i+1}, ..., M_M^m)$: Makespan of Motorbikes
- $M_D = max(M_D^i, M_D^{i+1}, ..., M_D^m)$: Makespan of Drones
- $M_{total} = max(M_T, M_M, M_D)$: Total makespan
- $M_{total} = max(M_{total}^i, M_{total}^{i+1}, ..., M_{total}^m)$: Total makespan

Table 1: Example

Depot	M_T	M_M	M_D	**Depot makespan**
1	3	2	1	3
2	4	3	2	4
3	1	5	2	5
4	2	6	1	6
M_{total}	4	**6**	2	**6**

3.1 Makespan calculations

In the case where a depot is equipped with more than 1 vehicle of any type (excluding vehicles with capacity = 1 i.e drones), and two or more routes need to be assigned to them, a **job scheduling problem** arises, where the routes are the jobs that need to be assigned to machines (vehicles). Since the vehicles are identical, it's more specifically an **identical-machines scheduling problem**. Minimizing the maximum completion time (makespan) is NP-hard. Many exact and approximation algorithms are known. We use the **longest-processing-time-first (LPT)** greedy algorithm, which orders the jobs by descending order of their cost and then schedules each job in this sequence into the machine of which the current load is smallest. An example is given below. Assume 4 routes that need to be assigned to two trucks. Each route has a time cost associated with it. The route assignment needs to be done in such a way that the trucks' makespan is minimized. We first sort the routes based on their cost in a descending order. Then, starting from the route with the maximum cost, we assign it to the truck with the currently minimum makespan. This is done iteratively until all routes have been assigned to a vehicle.

Table 2: 4 routes that need to be assigned to a depot's 2 trucks

Route	Cost
1	5
2	6
3	2
4	3

Procedure for the assignment of 4 routes to 2 trucks of the same depot:

Table 3: Sort routes in descending order based on cost

Route	Cost
2	6
1	5
4	3
3	2

Table 4: Iteratively assign routes to Trucks

Iteration	Route	Cost	Truck	Truck 1 ms	Truck 2 ms
1	2	6	1	6	0
2	1	5	2	6	5
3	4	3	2	6	8
4	3	2	1	8	8
Total makespan = 8 (optimal)					

4 Pseudocode for the MD-mfcmTSP heuristic

Algorithm 1: MD-mfcmTSP heuristic

Input: $G_T, G_M, ..., G_D$
Output: M_{total}, $Sol = \{Sol^i = \{R_T^i, R_M^i, ..., R_D^i\}, Sol^{i+1}, ..., Sol^m\}$ for each $i \in D$

Create clusters K^i of customer nodes for each depot $d^i \in D$
by assigning each customer to the closest possible depot
for *each* $d^i \in D$ **do**
  Call $Initialization(d^i, K^i)$
  while $(M_T^i > M_M^i \parallel M_T^i > M_D^i)$ && $stop \neq true$ **do**
    $diff_M = M_T^i - M_M^i$
    $diff_D = M_T^i - M_D^i$
    if $diff_M \geq diff_D$ **then**
      $vt = M$
      $cap =$ Motorbike's capacity
    else
      $vt = D$
      $cap = 1$
    end if
    $M_{dep} = M_T^i$
    $r_{best} = \emptyset$
    for $j = 1$ *to* $|R_T^i| - cap$ **do**
      $successive\_nodes = \emptyset$
      $load = 0$
      while $load + v_j^{demand} \leq cap$ && $v_j \in G_{vt}$ **do**
        $successive\_nodes \mathrel{+}= v_j$
      end while
      if $|successive\_nodes| == cap$ **then**
        $r_{new} = R_T^i[0] + \{successive\_nodes\} + R_T^i[0]$
        $R'^i_{vt} = R_{vt}^i + r_{new}$
        $M'_{vt} = R'^i_{vt}$'s makespan
        $R'^i_T = R_T^i - \{successive\_nodes\}$
        $M'_T = R'^i_T$'s makespan
        $M_{new} = MAX(M'_T, M'_{vt})$
        if $M_{new} < M_{dep}$ **then**
          $M_{dep} = M_{new}$
          $r_{best} = r_{new}$
        end if
        $r_{new} = \emptyset$
      end if
      $j \mathrel{+}= 1$
    end for
    if $r_{best} \neq \emptyset$ **then**
      $R_T^i = R_T^i - \{r_{best}^{customers}\}$
      $M_T = R_T^i$'s makespan
      $R_{vt}^i \mathrel{+}= r_{best}$
      $M_{vt} = R_{vt}^i$'s makespan
      (+Swap)
      Call $local\_optimization(R_T^i, n_{max})$
      Call $local\_optimization(R_{vt}^i, n_{max})$
    else
      $stop = true$
    end if
  end while
  $Sol^i = \{R_T^i, R_M^i, ..., R_D^i\}$
end for
$M_T = MAX(M_T^i, M_T^{i+1}, ..., M_T^m)$
$M_M = MAX(M_M^i, M_M^{i+1}, ..., M_M^m)$
$M_D = MAX(M_D^i, M_D^{i+1}, ..., M_D^m)$
$M_{total} = MAX(M_T, M_M, ..., M_D)$
Call $optimization\_full(Sol, n_{max})$

Algorithm 2: Initialization(d^i, K^i)

while $\{K^i\} \cap \{G_T\} \neq \emptyset$ **do**
  R_T^i += $NearestNeighbour(\{K^i\} \cap \{G_T\})$
end while
$M_T^i = R_T^i$'s makespan
$v_{free} = \{K^i\} - \{G_T\}$
if $v_{free} = \emptyset$ **then**
  return R_T^i
else
  while $v_{free} \neq \emptyset$ **do**
    if $M_T - M_M \geq M_T - M_D \parallel G_D = \emptyset$ **then**
      R_M^i += $NearestNeighbour(\{K^i\} \cap \{G_M\})$
      $v_{free} = v_{free} - \{R_M^i\}$
      $M_M^i = R_M^i$'s makespan
    else
      R_D^i += $closest(\{K^i\} \cap \{G_D\})$
      $M_D^i = R_D^i$'s makespan
    end if
  end while
end if
return Sol^i

Algorithm 3: $optimization\_full(Sol, n_{max})$

do
  Call $vt\_optimization(Sol, n_{max} = 2)$
  for *each* vt **do**
    for *each* $i \in D$ **do**
      Call $local\_optimization(R_{vt}^i, n_{max} = 2)$
    end for
    Call $mutual\_depot\_optimization(R_{vt}, n_{max} = 2)$
  end for
while *any route improves*

Algorithm 4: $local\_optimization(r, n_{max})$

for $n = 1$ *to* n_{max} **do**
  for each *combination of n successive nodes on the route*
    move the node(s) to a different place on the same route
    evaluate the new route
    if *this route is better than the original and all constraints are satisfied* **then**
      replace the original route with the new one
    continue in point 3 unless all possible places in the route have been evaluated
  end for
end for
return r

Algorithm 5: $mutual\_depot\_optimization(R_{vt}, n_{max})$

for $n = 1$ *to* n_{max} **do**
  for each *possible pair of depots* $c1$ *and* $c2$
    for each *combination of n successive nodes in the route of* $c1$
      remove the nodes from the route of c1 and insert them into c2
      evaluate the newly-created routes
      if $MAX(|R'^{c1}_{vt}|, |R'^{c2}_{vt}|) < MAX(|R^{c1}_{vt}|, |R^{c2}_{vt}|)$ *and all constraints are satisfied* **then**
        replace the original routes with the new ones
      continue in point 4 unless all possible places in c2 have been evaluated
    end for
  end for
end for
return R_{VT}

Algorithm 6: $vt\_optimization(Sol, n_{max})$

for $n = 1$ *to* n_{max} **do**
  for each *depot* $i \in D$
    for each *possible pair of vehicle types* $t1, t2 \in VT$
      for each *combination of n successive nodes in* R_{t1}^i
        remove the nodes from R_{t1}^i and insert them in R_{t2}^i
        if $MAX(|R'^i_{t1}|, |R'^i_{t2}|) < MAX(|R^i_{t1}|, |R^i_{t2}|)$ *and all constraints are satisfied* **then**
          replace the original routes with the new ones
        continue in point 5 unless all possible places in R_{t2}^i have been evaluated
      end for
    end for
  end for
end for
return Sol

5 Solving the MD-mfcmTSP

In this section, we introduce the two algorithmic approaches to address the MD-mfcmTSP. The first approach which is used to tackle the problem comprises of a clustering phase, which transforms the multi-depot problem into multiple single-depot mfcmTSP problems, and a routing phase which calls the mfcmTSP heuristic for each depot. The second approach leverages a hybrid metaheuristic, combining Ant Colony Optimization (ACO) with the mfcmTSP heuristic.

5.1 MD-mfcmTSP heuristic

To manage the complexity of multi-depot routing problems, a widely used approach is their transformation to multiple single-depot routing problems. Although a naive approach, it provides feasible solutions of which the results are used as a baseline for comparison with other solution methods.

For this purpose, we use a straightforward constrained proximity clustering to assign customers to depots, and then run the mfcmTSP heuristic for each depot. Each customer is assigned to the closest *possible* depot, creating clusters where each depot serves the customers nearest to it. Specifically, vehicle availability at each depot may limit the customers that are assigned to its cluster. The reason is that the initialization phase of the heuristic, which is based on these clusters, must form routes that visit every node $c_i \in C$. Therefore, while the assignment is based on proximity, we must adjust the clustering to account for these constraints. This means that some customers may need to be assigned to a more distant depot if the nearest depot lacks the vehicle that can serve them. For example, a customer only accessible by motorbike will be assigned to a more distant depot when the closest to them doesn't have a motorbike available in its fleet.

Figure 3: Initialization example

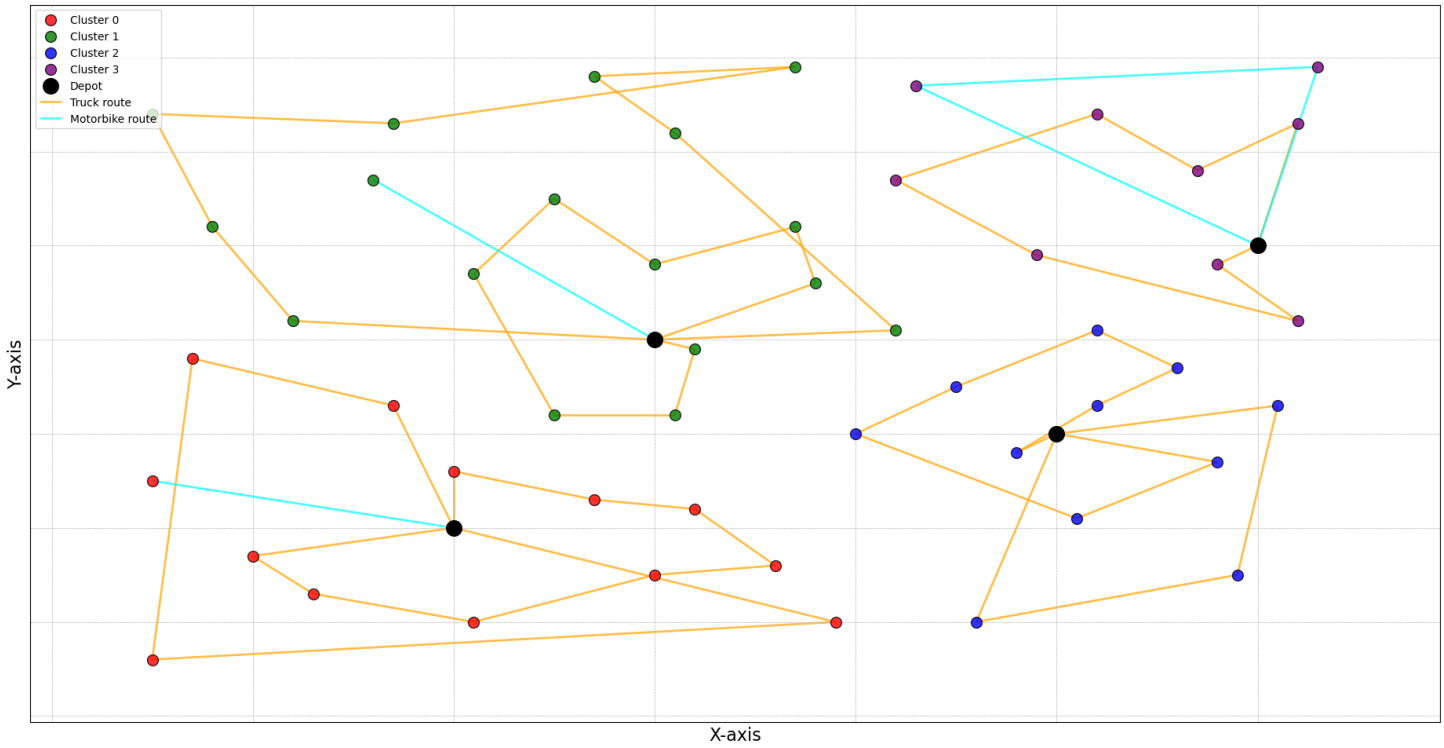


Figure 4: Initialization example

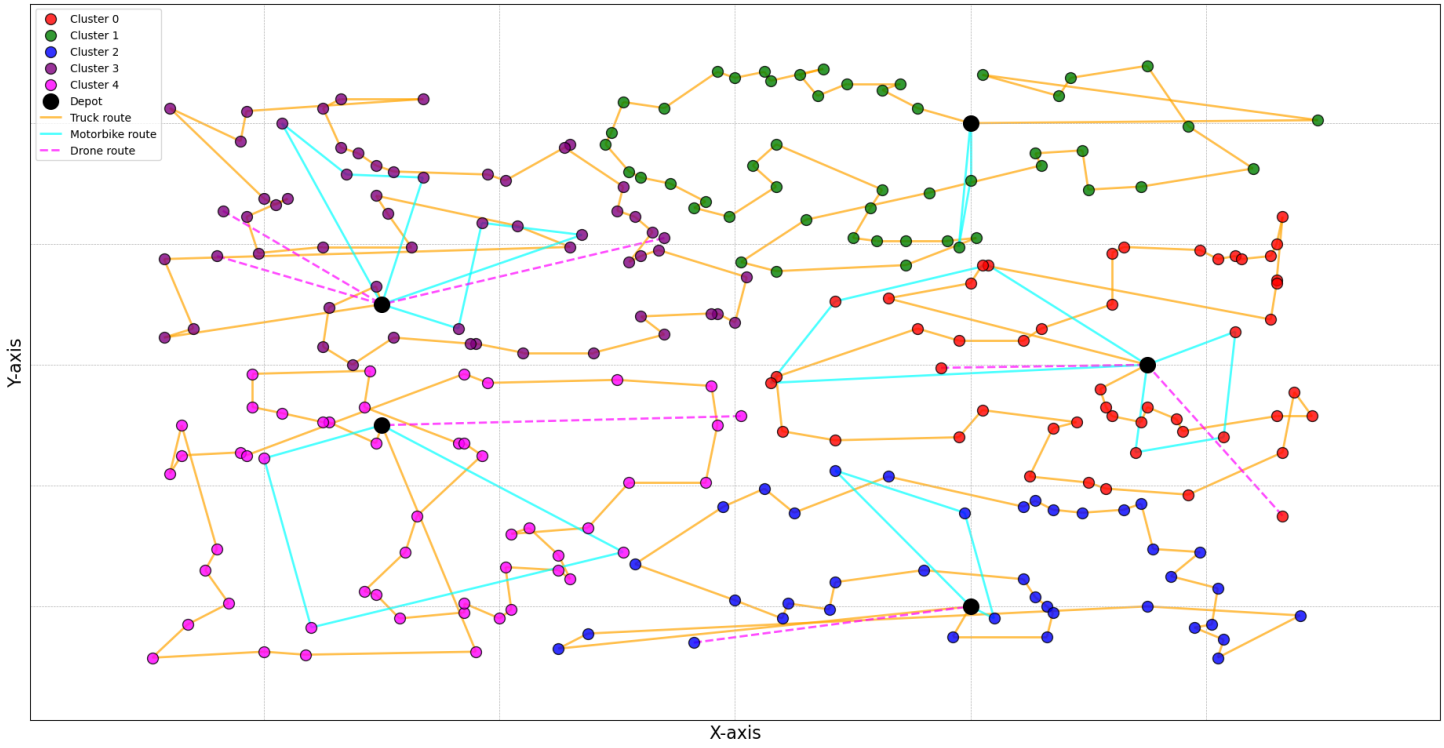
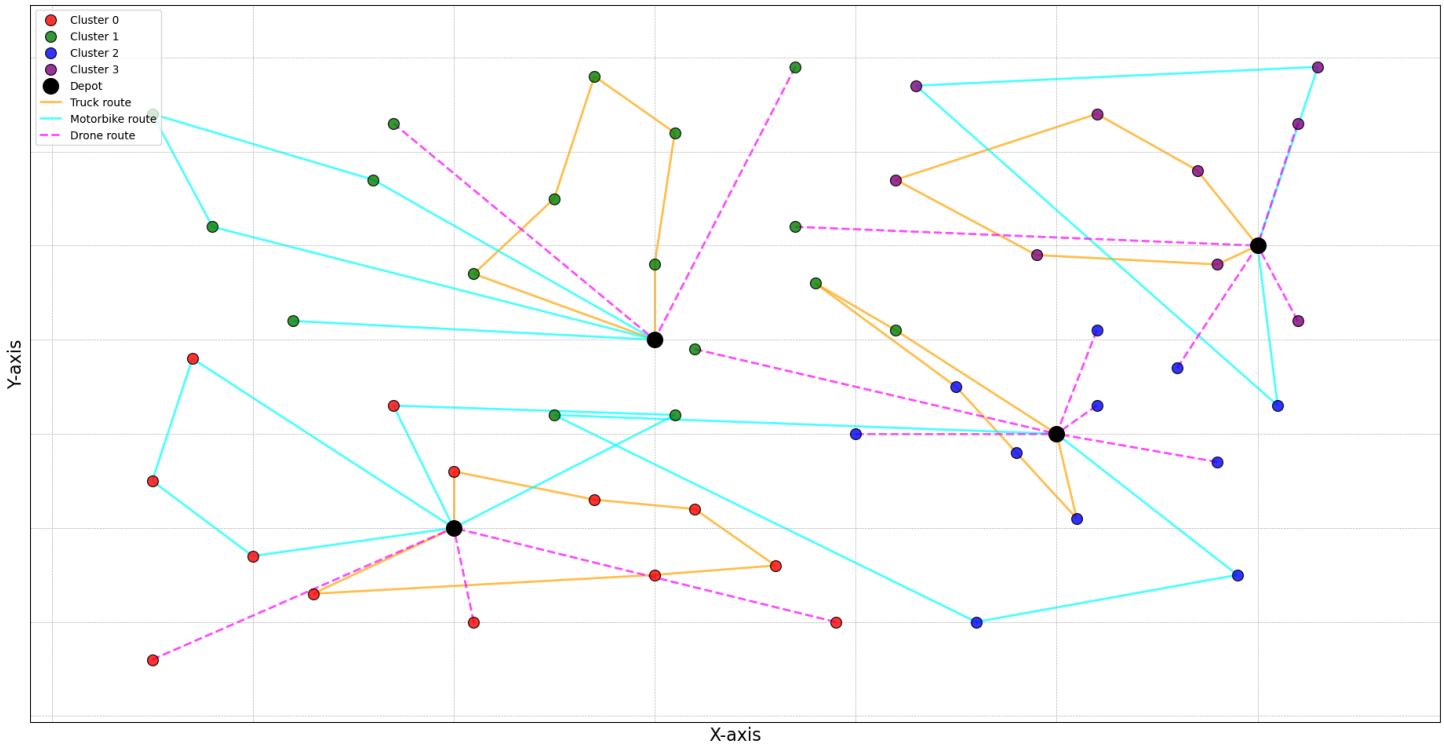


Figure 5: Solution after mutual depot optimization



5.2 Hybrid Ant Colony Optimization

For the second approach to solve the MD-mfcmTSP we introduce a novel hybrid metaheuristic based on the principles of Ant Colony Optimization combined with the mfcmTSP heuristic. Specifically, we employ a modified version of the Adaptive Ant Colony Optimization with Node Clustering (AACONC) (Stodola et al. 2022) which was developed for the Multi-Depot Vehicle Routing Problem and showed promising results. From now on, we refer to this algorithm as the H-AACONC.

5.2.1 H-AACONC Algorithm

The Ant Colony part of the algorithm disregards accessibility constraints and assumes that all customers must be served by trucks such as to minimize the total time (makespan). As such, the pheromone matrix, node clustering and the AntSolution function remain the same as in AACONC and are used to create an initial trucks-only solution which is stored in Rt. Then, with frequency n_{freq}, which is one of the AACONC control parameters, the single and mutual colony local optimization procedures may be called. In the resulting solution Rt, customers that are not accessible by trucks are then offloaded to the motorbike(s) of the same depot if possible, else to the closest depot with atleast one motorbike in its fleet. The resulting routes are inserted into R_{best} and the MD-mfcmTSP heuristic is called, while skipping the clustering and initialization phase.

5.2.2 Pheromone matrix update

One key distinction between the AACONC and the H-AACONC is the pheromone matrix update procedure. The pheromone matrix update procedure uses the same logic as the AACONC. The Simulated Annealing principle in combination with the Metropolis criterion probabilistically decide which solution will be used for the update. Although in the H-AACONC, while solutions R and R_{best} are used to make this decision, it is then truck solutions Rt and Rt_{best} respectively that are used to update the pheromone matrix.

$$p(R^{best}) = 1 - p(R) = \begin{cases} e^{-\frac{(|R^{best}|-|R|)/|R|}{T_{update}}} & \text{for } |R^{best}| > |R|, \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

$$T_{\text{update}}(iter+1) = \alpha_{\text{update}} \cdot T_{\text{update}}(iter) \quad (2)$$

The solution for updating the pheromone matrix R^{update} is selected based on the calculated probabilities: $R^{update} = R^{best}$ and $Rt^{update} = Rt^{best}$with $p(R^{best})$ or $R^{update} = R$ and $Rt^{update} = Rt$ with $p(R) = 1 - p(R^{best})$.The update itself is then conducted using (3); the pheromone trails lying on the routes are increased in proportion to the pheromone updating coefficient δ and the quality of the updating route (a ratio of R to R^{update}).

$$\tau_{ij}^{k} = \tau_{ij}^{k} + \chi_{ij} \cdot \delta \cdot \frac{|R|}{|R^{update}|} \text{ for all } v_i, v_j \in V \text{ and } d_k \in D \quad (3)$$

$$\chi_{ij} = \begin{cases} 1 & \text{if there is an edge between } v_i \text{ and } v_j \text{ in } Rt^{update}, \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

Algorithm 7: Hybrid AACONC

Input: $G_T, G_M, ..., G_D$
Output: M_{total}, $Sol = \{Sol^i = \{R_T^i, R_M^i, ..., R_D^i\}, Sol^{i+1}, ..., Sol^m\}$ for each $i \in D$

$iter = 0$
Initialize pheromone matrices τ
for each $v_i \in V$
  $K^{(v_i)}$ = Call CreateClusters
end for
$Rt = \infty, R = \infty$
while *NOT TERMINATED* **do**
  $Rt_{best} = \infty, R_{best} = \infty$
  for $\alpha = 1$ *to* n_{ants} **do**
    R_a = **Call AntSolution**
    //which construct routes, using Trucks, for every $c \in C$
    if $|R_a| < |Rt_{best}|$ **then**
      $Rt_{best} = R_a$
    end if
  end for
  if $iter \ mod \ n_{freq}$=0 **then**
    Rt_{best} = Call $local\_optimization$
    Rt_{best} = Call $mutual\_depot\_optimization$
  end if
  Assign customers in Rt_{best} which do not belong to G_T to R_{best}^M, R_{best}^D
  $R_{best}^T = Rt_{best} - \{R_{best}^M, R_{best}^D\}$
  R_{best} = Call MD-mfcmTSP heuristic
  if $|R_{best}| < |R|$ **then**
    $R = R_{best}$
    $Rt = Rt_{best}$
  end if
  Update pheromones
  Evaporate pheromones
  $iter$ += 1
end while

6 Instances

Table 5: MD-mfcmTSP-C Instances

Instance	Customers	Depots	Dimensions	Truck capacity	Layout
p01	50	4	3150m x 3450m	8	Random
p02	50	4	3150m x 3450m	16	Random
p03	75	5	3500m x 3800m	14	Random
p04	100	2	3350m x 3850m	10	Random
p05	100	2	3350m x 3850m	20	Random
p06	100	3	3350m x 3850m	10	Random
p07	100	4	3350m x 3850m	10	Random
p08	249	2	9900m x 9800m	50	Random
p09	249	3	9900m x 9800m	50	Random
p10	249	4	9900m x 9800m	50	Random
p11	249	5	10500m x 5000m	50	Random

Table 6: MD-mfcmTSP new generated Instances

Instance	Customers	Depots	Dimensions	Truck capacity	Layout
x01	500	4	7500m x 7500m	100	Random
x02	500	4	7500m x 7500m	100	Random
x03	500	5	7500m x 7500m	100	Random
x04	1000	4	10000m x 10000m	100	Random
x05	1000	6	10000m x 10000m	100	Random

7 Results

7.1 New heuristic vs original heuristic

mfcmTSP heuristic (original) : Finds and swaps the minimum cost route in each iteration

MD-mfcmTSP heuristic (v1) : Finds and swaps the route which minimizes the depot's makespan in each iteration

Results show that the new (v1) heuristic performs better than the original.

Table 7: Comparison of the two algorithms ($S_T = 15, S_M = 25, S_D = 30$)

Instance	Best	new (v1)						original					
		+Swap	gap	Standard	gap	No Local	gap	+Swap	gap	Standard	gap	No Local	gap
p01	215.15	217.42	1.06	**215.15**	0.00	334.04	55.26	243.72	13.28	232.34	7.99	356.47	65.68
p02	218.40	219.20	0.37	**218.40**	0.00	308.83	41.41	243.72	11.59	221.55	1.44	372.28	70.46
p03	202.43	214.84	6.13	**202.43**	0.00	253.13	25.07	277.50	37.08	218.73	8.05	347.94	71.88
p04	668.81	**668.81**	0.00	711.80	6.43	779.34	16.53	711.00	6.31	740.45	10.71	924.31	38.20
p05	630.78	**630.78**	0.00	655.40	3.90	726.62	15.19	713.20	13.07	713.39	13.08	874.80	38.69
p06	431.32	435.42	0.95	**431.32**	0.00	650.37	50.77	481.20	11.56	471.61	9.34	751.05	74.13
p07	309.48	**309.48**	0.00	349.19	12.83	350.77	13.34	322.29	4.15	359.18	16.06	589.22	90.39
p08	3079.58	3333.93	8.26	**3079.58**	0.00	3321.63	7.86	3448.84	11.99	3380.67	9.78	4009.10	30.18
p09	1917.82	**1917.82**	0.00	2102.98	9.68	2429.24	26.67	2201.39	14.79	2157.10	12.48	2732.05	42.46
p10	1493.13	**1493.13**	0.00	1525.37	2.16	1864.23	24.85	1630.94	9.23	1614.27	8.11	2437.73	63.26
p11	1096.74	1145.75	4.47	1101.33	0.42	1239.36	13.00	1213.34	10.45	**1096.74**	0.00	1583.09	44.35
Average	933.05	962.41	**1.93%**	962.99	3.21%	1114.32	26.35%	1044.08	13.04%	1018.73	8.82%	1361.64	57.24%
x01	1454.47	1550.64	6.61%	**1454.47**	0.00%	1886.98	29.74%	1542.40	6.05%	1651.89	13.57%	2411.57	65.80%
x02	1514.61	**1514.61**	0.00%	1525.87	0.74%	1705.48	12.60%	1644.94	8.60%	1705.10	12.58%	2099.51	38.62%
x03	1306.20	1313.16	0.53%	1352.27	3.53%	1556.10	19.13%	**1306.20**	0.00%	1338.43	2.47%	1985.68	52.02%
x04	3031.00	**3031.00**	0.00%	3128.87	3.23%	3436.31	13.37%	3176.20	4.79%	3199.83	5.57%	3967.73	30.90%
x05	2046.21	**2046.21**	0.00%	2099.98	2.63%	2421.32	18.33%	2292.28	12.03%	2228.69	8.92%	3061.06	49.60%
Average	1870.50	1891.12	**1.43%**	1912.29	2.03%	2201.24	18.63%	1992.40	6.29%	2024.79	8.62%	2705.11	47.39%

Table 8: Comparison of the two algorithms ($S_T = 15, S_M = 20, S_D = 30$)

Instance id	Best	new (v1)						original					
		+Swap	gap	Standard	gap	No Local	gap	+Swap	gap	Standard	gap	No Local	gap
p01	221.54	265.62	19.90%	**221.54**	0.00%	361.73	63.28%	231.04	4.29%	258.24	16.57%	410.50	85.29%
p02	224.38	**224.38**	0.00%	230.97	2.94%	314.64	40.23%	246.38	9.80%	259.42	15.62%	382.31	70.39%
p03	231.85	232.55	0.30%	**231.85**	0.00%	275.80	18.96%	277.50	19.69%	247.39	6.70%	349.68	50.82%
p04	743.39	771.46	3.78%	**743.39**	0.00%	833.11	12.07%	769.70	3.54%	768.52	3.38%	995.79	33.95%
p05	699.49	701.25	0.25%	**699.49**	0.00%	785.30	12.27%	737.73	5.47%	733.75	4.90%	927.64	32.62%
p06	473.32	**473.32**	0.00%	523.44	10.59%	715.92	51.25%	545.35	15.22%	493.64	4.29%	808.94	70.91%
p07	339.27	367.53	8.33%	**339.27**	0.00%	417.10	22.94%	352.11	3.78%	440.15	29.73%	469.11	38.27%
p08	3330.48	3476.45	4.38%	**3330.48**	0.00%	3521.95	5.75%	3586.21	7.68%	3646.71	9.50%	4312.54	29.49%
p09	2064.72	**2064.72**	0.00%	2408.35	16.64%	2583.64	25.13%	2206.32	6.86%	2436.95	18.03%	2935.01	42.15%
p10	1601.68	1605.77	0.26%	1655.83	3.38%	1888.36	17.90%	1625.21	1.47%	**1601.68**	0.00%	2437.73	52.20%
p11	1140.61	1195.53	4.81%	**1140.61**	0.00%	1345.70	17.98%	1274.27	11.72%	1274.53	11.74%	1590.39	39.43%
Average	1006.43	1034.42	3.82%	1047.75	**3.05%**	1185.75	26.16%	1077.44	8.14%	1105.54	10.95%	1419.97	49.59%
x01	1578.55	1597.25	1.18%	**1578.55**	0.00%	1976.45	25.21%	1669.33	5.75%	1716.93	8.77%	2631.68	66.72%
x02	1655.28	**1655.28**	0.00%	1673.69	1.11%	1792.14	8.27%	1726.75	4.32%	1798.78	8.67%	2346.33	41.75%
x03	1361.22	1406.07	3.29%	**1361.22**	0.00%	1690.65	24.20%	1485.35	9.12%	1459.32	7.21%	2043.82	50.15%
x04	3226.24	3238.88	0.39%	**3226.24**	0.00%	3615.71	12.07%	3325.25	3.07%	3390.14	5.08%	4073.03	26.25%
x05	2201.89	**2201.89**	0.00%	2262.24	2.74%	2587.20	17.50%	2342.16	6.37%	2397.81	8.90%	3063.97	39.15%
Average	2004.64	2019.87	0.97%	2020.39	**0.77%**	2332.43	17.45%	2109.77	5.73%	2152.60	7.72%	2831.77	44.80%

8 MD-mfcmTSP Results Analysis

8.1 Local optimization impact on the heuristic

8.1.1 Proximity clustering

Table 9: Local optimization impact on heuristic using proximity clustering

Instance	Best	+Swap	gap(%)	No Local Opt	gap(%)	Standard	gap(%)	Only Swap	gap(%)
p01-C	215.15	217.42	1.06	334.04	55.26	**215.15**	0.00	173.96	27.33
p02-C	218.4	219.20	0.37	308.83	41.41	**218.40**	0.00	289.58	32.59
p03-C	202.43	214.84	6.13	253.18	25.07	**202.43**	0.00	255.52	26.23
p04-C	668.81	**668.81**	0.00	779.34	16.53	711.80	6.43	691.62	3.41
p05-C	630.78	**630.78**	0.00	726.62	15.19	655.4	3.90	697.86	10.63
p06-C	431.32	435.42	0.95	650.32	50.77	**431.32**	0.00	564.79	30.94
p07-C	309.48	**309.48**	0.00	350.77	13.34	349.19	12.83	366.51	18.43
p08-C	3079.58	3333.93	8.26	3321.63	7.86	**3079.58**	0.00	3428.15	11.32
p09-C	1917.82	**1917.82**	0.00	2429.24	26.67	2102.98	9.65	2303.70	20.12
p10-C	1493.13	**1493.13**	0.00	1864.23	24.85	1525.37	2.16	1706.14	14.27
p11-C	1101.33	1145.75	4.03	1239.36	12.53	**1101.33**	0.00	1276.59	15.91
AVG	933.48	962.42	**1.89**	1114.32	26.32	963.00	3.18	1077.67	19.20

9 New Hybrid AACONC algorithm

Algorithm 8: Hybrid AACONC

Input: $G_T, G_M, ..., G_D$
Output: M_{total}, $Sol = \{Sol^i = \{R_T^i, R_M^i, ..., R_D^i\}, Sol^{i+1}, ..., Sol^m\}$ for each $i \in D$

$iter = 0$
Initialize pheromone matrices τ
for each $v_i \in V$
  $K^{(v_i)}$ = Call CreateClusters
end for
$Rt = \infty, R = \infty$
while *NOT TERMINATED* **do**
  $Rt_{best} = \infty, R_{best} = \infty$
  for $\alpha = 1$ *to* n_{ants} **do**
    R_a = **Call AntSolution**
    //which construct routes, using Trucks, for every $c \in C$
    if $|R_a| < |Rt_{best}|$ **then**
      $Rt_{best} = R_a$
    end if
  end for
  if $iter \ mod \ n_{freq}$=*0* **then**
    Rt_{best} = Call $local\_optimization$
    Rt_{best} = Call $mutual\_depot\_optimization$
  end if
  Assign customers in Rt_{best} which do not belong to G_T to R_{best}^M, R_{best}^D
  $R_{best}^T = Rt_{best} - \{R_{best}^M, R_{best}^D\}$
  R_{best} = Call MD-mfcmTSP heuristic
  if $|R_{best}| < |R|$ **then**
    $R = R_{best}$
    $Rt = Rt_{best}$
  end if
  Update pheromones
  Evaporate pheromones
  $iter$ += 1
end while

9.1 Pheromone matrix & update

Στον αλγόριθμο AACONC (Stodola, 2022), ο πίνακας φερομόνων χρησιμοποιείται για την εύρεση μιας ολοκληρωμένης λύσης για το πρόβλημα MDVRP. Στον αλγόριθμο (Hybrid AACONC) ο πινακας φερομόνων χρησιμοποιείται για την εύρεση λύσης Rt_{best}, που δεν αποτελεί τελική λύση του προβλήματος, αλλά μια αρχική (συνήθως ανέφικτη) λύση. Η ενημέρωση του πίνακα φερομόνων χρησιμοποιεί την αρχή της προσομοίωσης ανόπτησης (simulated annealing). Συγκεκριμένα, χρησιμοποιείται το Metropolis criterion για να αποφασιστεί αν θα χρησιμοποιηθεί η καλύτερη λύση που έχει βρεθεί ως τώρα (R) ή η λύση της τωρινής γενιάς (R_{best}). Αν η τωρινή γενιά βελτιώνει το αποτέλεσμα, τότε αυτή πάντα αντικαθιστά την καλύτερη λύση $(R = R_{best})$ και χρησιμοποιείται για την ενημέρωση. Όμως υπάρχει πιθανότητα να επιλεχθεί ακόμα και αν είναι χειρότερη από την μέχρι τώρα καλύτερη λύση. Αυτό βοηθάει στη διατήρηση της ποικιλομορφίας του πληθυσμού, επεκτείνοντας έτσι τον χώρο αναζήτησης (search space) και αποτρέποντας τον αλγόριθμο από το να βρεθεί σε τοπικό βέλτιστο (local optima). Είναι σημαντικό να σημειωθεί, οτι ενώ η επιλογή της λύσης βάση της οποίας θα ενημερωθεί ο πίνακας φερομόνων εξαρτάται από τα R, R_{best}, τα δρομολόγια τα οποία χρησιμοποιούνται για την ενημέρωση είναι αυτά των Rt, Rt_{best} αντίστοιχα. Επομένως, στο AACONC(MDVRP) καθώς και σε αυτόν τον αλγόριθμο, ο πίνακας φερομόνων έχει σημαντικό ρόλο, στον τελευταίο χρησιμοποιείται για την παραγωγή μίας αρχικής λύσης ενώ στον πρώτο για την παραγωγή μιας ολοκληρωμένης λύσης.

9.2 Results

9.2.1 Motorbike speed = 20

Table 10: H-AACONC Results (Standard) ($S_T = 15, S_M = 20, S_D = 30$)

Instance	Best	Average	gap(%)	Worst	gap(%)	Average time(s)
p01	191.03	193.74	1.42	195.01	2.08	72
p02	195.01	201.24	3.20	218.32	11.95	71
p03	185.41	201.22	8.53	249.92	34.79	155
p04	677.51	684.18	0.99	692.39	2.20	361
p05	615.89	619.84	0.64	629.49	2.21	352
p06	404.05	411.80	1.92	423.33	4.77	240
p07	290.67	295.93	1.81	299.85	3.16	328
p08	3010.57	3043.91	1.11	3062.63	1.73	3453
p09	1860.04	1896.84	1.98	1938.61	4.22	2640
p10	1382.34	1416.46	2.47	1472.13	6.50	2958
p11	1043.54	1077.332	3.24	1101.95	5.60	2113
Average	896.01	912.95	2.48	934.88	7.20	1159

Table 11: H-AACONC Results (No Local Opt) ($S_T = 15, S_M = 20, S_D = 30$)

Instance	Best	Average	gap(%)	Worst	gap(%)	Average time(s)
p01	209.36	209.85	0.23	210.86	0.72	48
p02	206.64	213.29	3.22	226.27	9.50	46
p03	203.68	210.12	3.16	221.88	8.94	92
p04	712.50	729.33	2.36	754.53	5.90	223
p05	645.66	659.93	2.21	672.15	4.10	144
p06	428.15	438.03	2.31	444.57	3.84	190
p07	313.57	321.02	2.38	330.76	5.48	242
p08	3081.17	3118.92	1.23	3140.97	1.94	976
p09	1922.65	1962.08	2.05	1989.88	3.50	1088
p10	1444.00	1468.61	1.70	1492.83	3.38	951
p11	1113.63	1193.22	7.15	1377.87	23.73	938
Average	934.64	956.76	2.54	987.51	6.46	449

Table 12: H-AACONC Results (+Swap) ($S_T = 15, S_M = 20, S_D = 30$)

Instance	Best	Average	gap(%)	Worst	gap(%)	Average time(s)
p01	190.83	195.01	2.19	198.39	3.96	63
p02	197.83	211.72	7.02	229.95	16.24	70
p03	187.61	188.60	0.53	190.79	1.70	154
p04	657.93	673.26	2.33	683.14	3.83	330
p05	613.88	621.53	1.25	627.75	2.26	297
p06	402.58	409.63	1.75	413.40	2.69	313
p07	294.72	299.11	1.49	311.37	5.65	308
p08	3023.23	3045.90	0.75	3087.61	2.13	3534
p09	1827.94	1864.04	1.98	1904.51	4.19	3096
p10	1383.94	1412.37	2.05	1434.05	3.62	2664
p11	1031.64	1059.53	2.70	1096.97	6.33	2132
Average	892.01	907.34	2.19	925.27	4.78	1178

9.2.2 Motorbike speed = 25

Table 13: Comparison of the different local optimization methods

Instance	Best	Standard Best	gap(%)	+Swap Best	gap(%)	No Local Opt Best	gap(%)
p01	190.83	191.03	0.10	190.83	0.00	209.36	9.71
p02	195.01	195.01	0.00	197.83	1.45	206.64	5.96
p03	185.41	185.41	0.00	187.61	1.19	203.68	9.85
p04	657.93	677.51	2.98	657.93	0.00	712.50	8.29
p05	613.88	615.89	0.33	613.88	0.00	645.66	5.18
p06	402.58	404.05	0.37	402.58	0.00	428.15	6.35
p07	290.67	290.67	0.00	294.72	1.39	313.57	7.88
p08	3010.57	3010.57	0.00	3023.23	0.42	3081.17	2.35
p09	1827.94	1860.04	1.76	1827.94	0.00	1922.65	5.18
p10	1382.34	1382.34	0.00	1383.94	0.12	1444.00	4.46
p11	1031.64	1043.54	1.15	1031.64	0.00	1113.63	7.95
Average	889.89	896.01	0.61	892.01	**0.41**	934.64	6.65

9.2.3 Comparison of algorithms

Table 14: H-AACONC Results (Standard)

Instance	Best	Average	gap(%)	Worst	gap(%)	Average time(s)
p01	173.99	177.12	1.80	183.97	5.74	121
p02	178.37	185.19	3.82	193.80	8.65	130
p03	172.24	175.30	1.78	181.28	5.25	329
p04	628.26	638.38	1.61	650.05	3.47	596
p05	589.68	595.86	1.05	598.88	1.56	480
p06	381.72	385.05	0.87	389.24	1.97	498
p07	278.07	282.09	1.44	284.58	2.34	517
p08	2842.74	2947.65	3.69	3141.88	10.52	3600
p09	1768.27	1813.98	2.59	1874.42	6.00	3519
p10	1293.53	1343.61	3.87	1368.75	5.82	3573
p11	979.30	1000.54	2.17	1017.26	3.88	3485
Average	844.20	867.71	2.24	898.56	5.02	1532

Table 15: H-AACONC Results (+Swap)

Instance	Best	Average	gap(%)	Worst	gap(%)	Average time(s)
p01	180.03	183.10	1.71	185.16	2.85	105
p02	179.90	192.79	7.16	218.94	21.70	103
p03	174.47	179.53	2.90	186.63	6.97	248
p04	628.73	635.60	1.09	648.20	1.98	589
p05	583.01	589.42	1.10	593.97	1.88	501
p06	369.07	377.72	2.34	381.94	3.49	489
p07	271.82	275.79	1.46	279.73	2.91	506
p08	2809.48	2893.66	3.00	2926.91	4.18	3600
p09	1753.60	1774.02	1.16	1815.32	3.52	3600
p10	1259.76	1305.86	3.66	1349.10	7.09	3446
p11	985.47	1004.46	4.51	1029.90	4.51	2803
Average	835.94	855.63	2.74	874.16	5.65	1462

Table 16: H-AACONC Best Results comparison

Instance	Best Standard	Best +Swap	gap(%)	avg time Standard	avg time +Swap	gap(%)
p01	**173.99**	180.03	3.47	121	105	-13.22
p02	**178.37**	179.90	0.86	130	103	-20.46
p03	**172.24**	174.47	1.29	329	248	-24.65
p04	**628.26**	628.73	0.07	596	589	-1.31
p05	589.68	**583.01**	-1.13	480	591	23.06
p06	381.72	**369.07**	-3.31	498	489	-1.69
p07	278.07	**271.82**	-2.25	517	506	-2.13
p08	2842.74	**2809.48**	-1.17	3600	3600	0.00
p09	1768.27	**1753.60**	-0.83	3519	3600	2.30
p10	1293.53	**1259.76**	-2.61	3573	3446	-3.54
p11	**979.30**	985.47	0.63	3485	2803	-19.56
Average	844.20	835.94	-0.45	1532	1462	-4.55

Table 17: Performance Comparison (Standard Local Optimization)

Instance	Hybrid	New (v1)	gap(%)	Original	gap(%)
p01	173.99	215.15	23.66	232.34	33.54
p02	178.37	218.40	22.44	221.55	24.21
p03	172.24	202.43	17.53	218.73	26.99
p04	628.26	711.80	13.30	740.45	17.86
p05	589.68	655.40	11.15	713.39	20.98
p06	381.72	431.32	12.99	471.61	23.55
p07	278.07	349.19	25.58	359.18	29.17
p08	2842.74	3079.58	8.33	3380.67	18.92
p09	1768.27	2102.98	18.93	2157.10	21.99
p10	1293.53	1525.37	17.92	1614.27	24.80
p11	979.30	1101.33	12.46	1096.74	11.99
Avg	844.20	963.00	16.75	1018.73	23.09

Figure 6: Error percentage of the two versions to the best result found by H-AACONC (Standard version of Local Optimization)

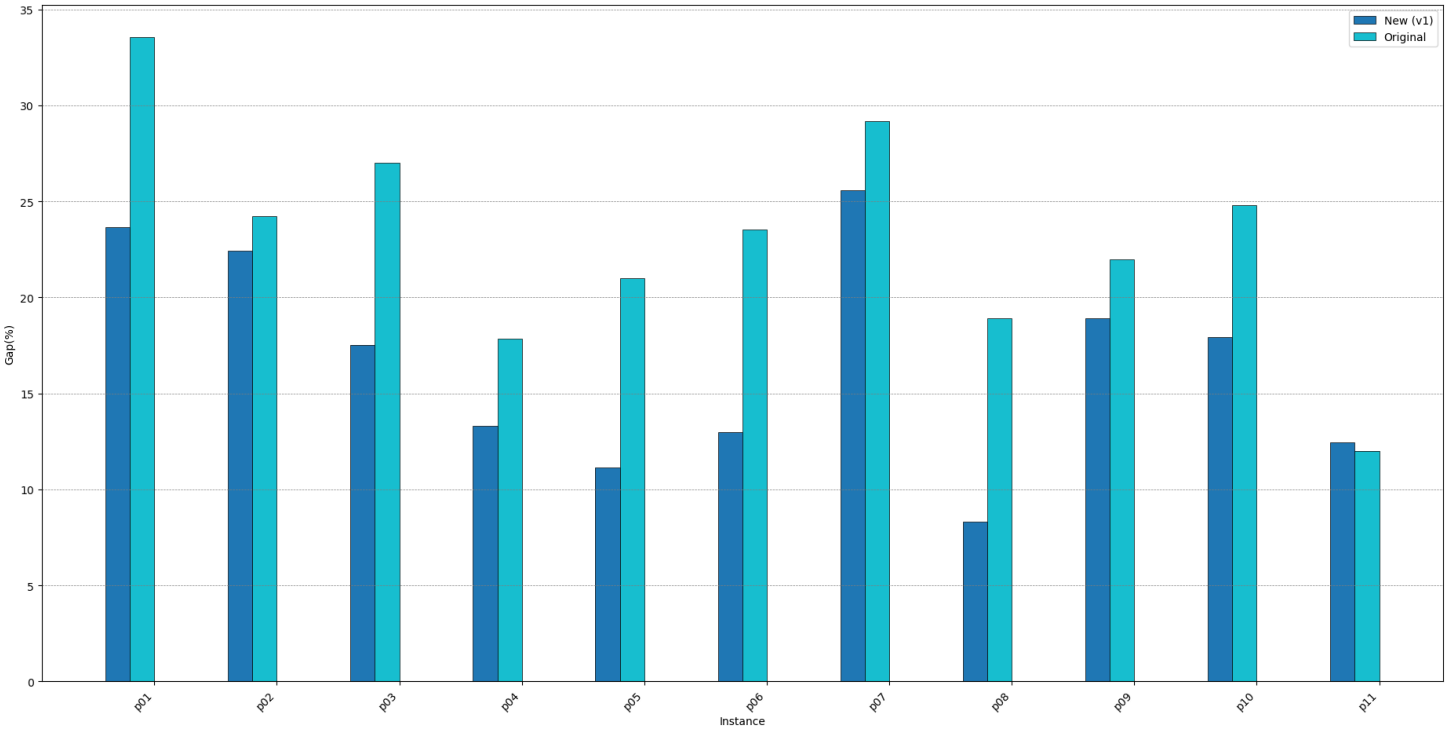


Table 18: Performance Comparison (+Swap Local Optimization)

Instance	**Hybrid**	**New (v1)**	**gap(%)**	**Original**	**gap(%)**
p01	180.03	217.42	20.77	243.72	35.38
p02	179.90	219.20	21.85	243.72	35.48
p03	174.47	214.84	23.14	277.50	59.05
p04	628.73	668.81	6.37	711.00	13.09
p05	583.01	630.78	8.19	713.00	22.30
p06	369.07	435.42	17.98	481.20	30.38
p07	271.82	309.48	13.85	322.29	18.57
p08	2809.48	3333.93	18.67	3448.84	22.76
p09	1753.60	1917.82	9.36	2201.39	25.54
p10	1259.76	1493.13	18.52	1630.94	29.46
p11	985.47	1145.75	16.26	1211.33	22.92
Avg	835.94	962.42	15.91	1044.08	28.63

Figure 7: Error percentage of the two versions to the best result found by H-AACONC (+Swap version of Local Optimization)

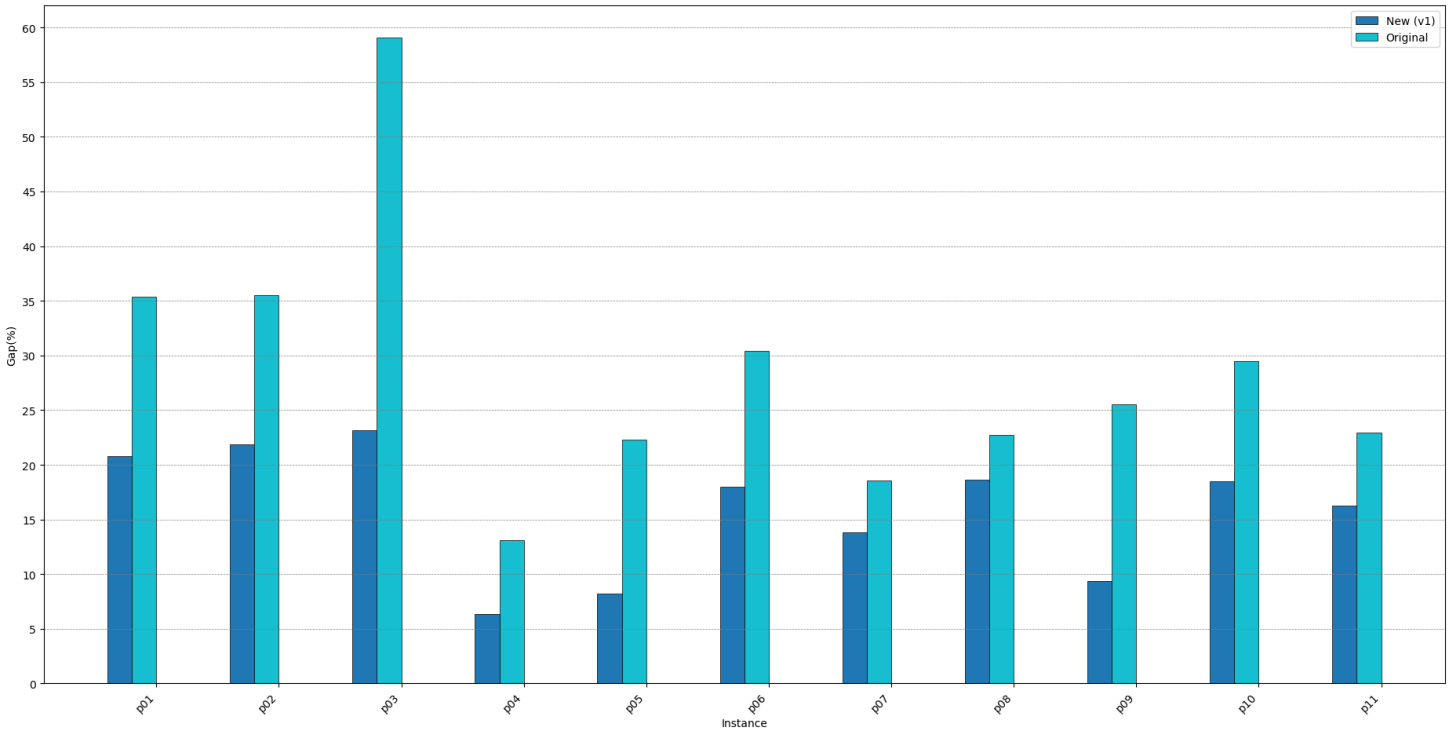


Figure 8: Results (Standard Local Optimization)

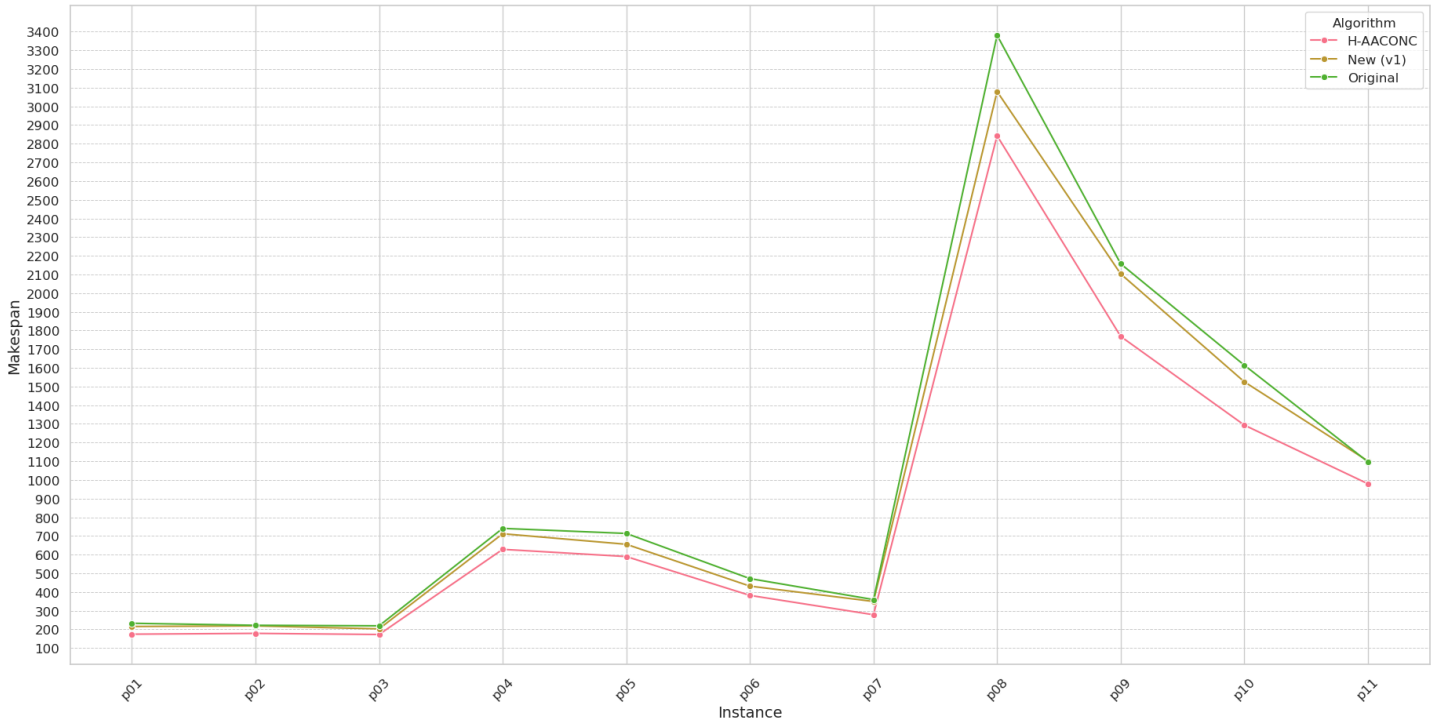
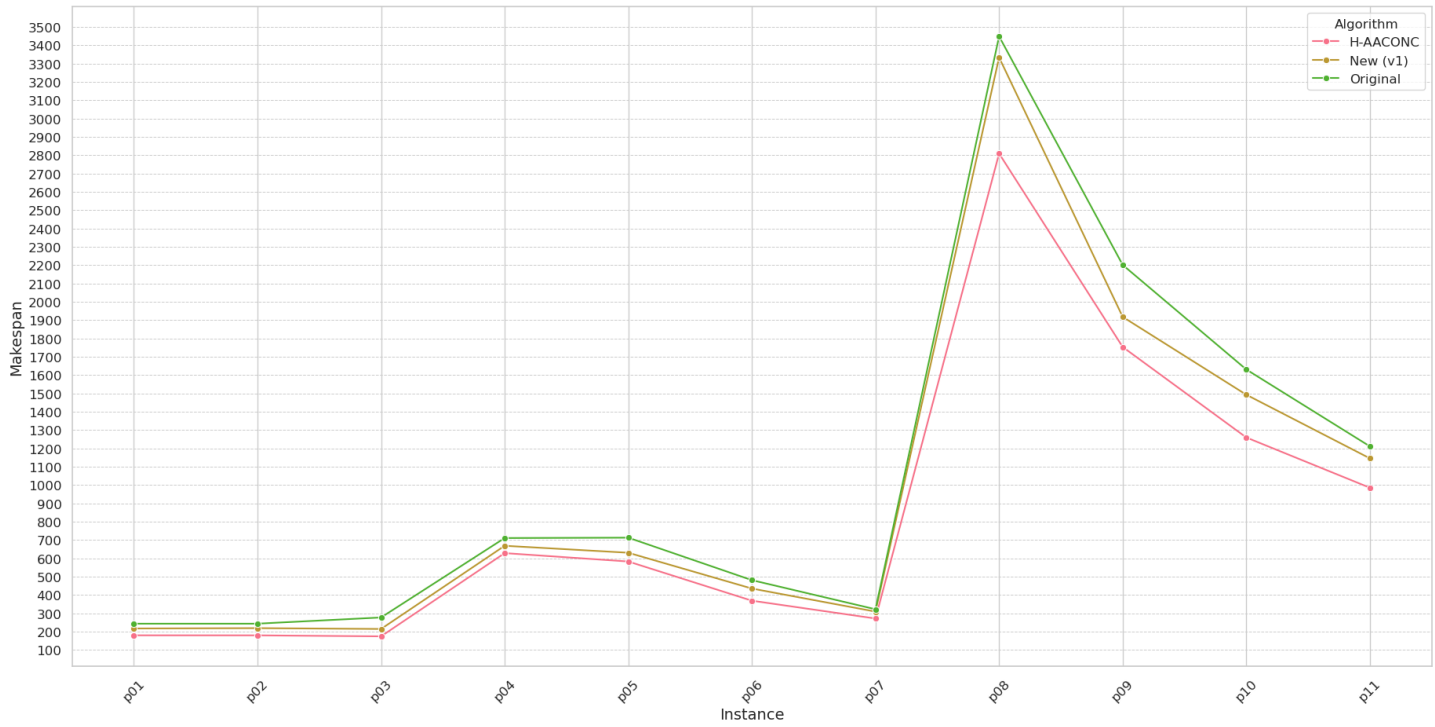


Figure 9: Results (+Swap Local Optimization)



10 Related Literature

At the same time, it is important that we address flight range limitations. Such limitations have important roles in much of the existing drone routing research. However, even he first drones developed by Amazon and JD.com ahad a flight range of 15 to 20 miles [33, 56]. Such a range is suitable to allow out and back travel in the medium-sized cities in which the authors live, Braunschweig, Germany, and Iowa City, United States. In fact, it is suitable for many larger cities such as Hamburg, Munich, and Paris. Thus, in contrast to other drone applications, in this work, we do not consider flight range as a limiting factor (Ulmer & Thomas (2018)).

Table 19: Drone-related Routing literature

Reference	Problem	Scale	Tandem	Flight endurance	Capacitated*	Solution Method
Murray & Chu (2015)	FSTSP	1-Truck 1-Drone 1-Depot	Yes	Yes	No	MILP, Heuristics
	PDSTSP	1-Truck n-Drone 1-Depot	No	Yes	No	MILP, Heuristics
Ha et al. (2015)	FSTSP	1-Truck 1-Drone 1-Depot	Yes	Yes	No	MIP, Heuristics
Freitas & Penna (2018)	FSTSP	1-Truck 1-Drone 1-Depot	Yes	Yes	No	IP, Heuristics
Boccia et al. (2021)	FSTSP/PDSTSP	1-Truck 1-Drone 1-Depot	Yes	Yes	No	MILP, Heuristics
Dell'Amico et al. (2021)	FSTSP	1-Truck 1-Drone 1-Depot	Yes	Yes	No	MILP
Dell'Amico et al. (2021)	FSTSP	1-Truck 1-Drone 1-Depot	Yes	Yes	No	Branch and bound, Heuristic
Kuroswiski et al. (2023)	FSTSP	1-Truck 1-Drone 1-Depot	Yes	Yes	No	MILP, Metaheuristics
Pilcher (2023)	FSTSP	1-Truck 1-Drone 1-Depot	Yes	Yes	No	Self-adaptive GA
Mbiadou Saleu et al. (2018)	PDSTSP	1-Truck n-Drone 1-Depot	No	Yes	No	MILP, Heuristics
Dinh et al. (2021)	PDSTSP	1-Truck n-Drone 1-Depot	No	Yes	No	Metaheuristics
Nguyen et al. (2022)	PDSTSP	1-Truck n-Drone 1-Depot	No	Yes	No	MILP
Mbiadou Saleu et al. (2022)	PDSMTSP	m-Truck n-Drone 1-Depot	No	Yes	No	MILP, Metaheuristics
Montemanni et al. (2023)	PDSTSP	1-Truck n-Drone 1-Depot	Yes	Yes	No	Constraint Programming
Nguyen et al. (2023)	PDSTSP-c	1-Truck n-Drone 1-Depot	No	Yes	No	MILP, Metaheuristics
Montemanni et al. (2024)	PDSTSP-c	1-Truck n-Drone 1-Depot	No	Yes	No	MILP, Constraint Programming
Ham (2018)	PDSTSP^{+DP}	m-Truck n-Drone 2-Depot	No	Yes	No	Constraint Programming
Agatz et al. (2018)	TSP-D	1-Truck 1-Drone 1-Depot	Yes	Yes	No	IP, Heuristics
Yurek et al. (2018)	TSP-D	1-Truck 1-Drone 1-Depot	Yes	Yes	No	MIP, Heuristics
Ha et al. (2018)	min-cost TSP-D	1-Truck 1-Drone 1-Depot	Yes	Yes	No	MILP, GRASP, TSP-LS
Dorling et al. (2016)	DDPs	0-Truck n-Drone 1-Depot	No	Yes	Yes	MILP, SA
Ulmer & Thomas (2018)	SDDPHF	m-Truck n-Drone 1-Depot	No	No	No	Adaptive dynamic programming
Salama & Srinivas (2020)	JOCR	1-Truck n-Drone 1-Depot	Yes	Yes	No	IP, MILP, Heuristics
Lu et al. (2022)	FDTSP	1-Truck n-Drone 1-Depot	Yes	Yes	No	Heuristics, Metaheuristics
Lan (2024)	TSPTWD	1-Truck 1-Drone 1-Depot	Yes	Yes	No	Metaheuristics
Wang et al. (2017)	VRPD	m-Truck n-Drone 1-Depot	Yes	Yes	Yes	Problem formulation, theoretical study
Schermer et al. (2018)	VRPD	m-Truck n-Drone 1-Depot	Yes	Yes	Yes	Heuristics
Sacramento et al. (2019)	VRP-D	m-Truck m-Drone 1-Depot	Yes	Yes	Yes*	MIP, ALNS
Schermer et al. (2019)	VRPDERO	m-Truck n-Drone 1-Depot	No	Yes	No	MILP, VNS, TS
Nguyen et al. (2022)	PDSVRP	m-Truck n-Drone 1-Depot	No	Yes	Yes	MILP, Metaheuristics
Schermer et al. (2019)	VRPD	m-Truck n-Drone 1-Depot	Yes	Yes	No	MILP, Matheuristic
Wang & Sheu (2019)	VRPD	m-Truck n-Drone 1-Depot	No	Yes	Yes	MIP, branch-and-price
Euchi & Sadok (2021)	VRP-D	m-Truck m-Drone 1-Depot	Yes	Yes	Yes	MILP, HGA
Lei et al. (2022)	VRPD	m-Truck m-Drone 1-Depot	Yes	Yes	Yes	Dynamical Artificial Bee Colony
Karak et al. (2019)	HVDRP	m-Truck n-Drone 1-Depot	Yes	Yes	Yes(Drones)	MIP, Heuristics
Kuo et al. (2022)	VRPDTW	m-Truck m-Drone 1-Depot	Yes	Yes	Yes	MIP, VNS
Stodola et al. (2024)	MDVRP-D	m-Truck m-Drone m-Depot	Yes	Yes	Yes	ACO
Oikonomou et al. (2019)	mfcmTSP	1-Truck 1-Drone 1-Depot	No	No	No	Heuristics
This paper	MD-mfcmTSP	m-Truck n-Drone k-Depot	No	No	Yes	Metaheuristics, Heuristics

11 Initial approach to AACONC+

Algorithm 9: AACONC+ Algorithm

Function *AACONC+*$(V, n_{ants}, n_{freq}, n_{size}, n_{sect}, n_{prim}, T_{update}, \alpha, \beta, \rho_{min}, \rho_{max}, \delta)$
- $|R| = \infty$
- $iter = 0$
- Initialize pheromone matrices τ
- **for** *each* $v_i \in V$ **do**
  - $K(v_i) =$ CreateClusters$(V, v_i, n_{\text{size}}, n_{\text{sect}})$
- **end for**
- **while** *not terminated* **do**
  - $|R_{\text{best}}| = \infty$
  - $iter = iter + 1$
  - **for** $a = 1$ *to* n_{ants} **do**
    - $R_a =$ AntSolution$(V, K, \tau, \alpha, \beta)$
    - **if** $|R_a| < |R_{best}|$ **then**
      - $R_{\text{best}} = R_a$
    - **end if**
  - **end for**
  - **if** $iter \mod n_{freq} = 0$ **then**
    - $R_{\text{best}} =$ LocalOptimization(V, R_{best})
  - **end if**
  - **if** $|R_{best}| < |R|$ **then**
    - $R = R_{\text{best}}$
  - **end if**
  - Update pheromone matrices τ
  - Calculate evaporation coefficient ρ
  - Evaporate pheromone matrices τ using ρ
- **end while**
- **return** R

Algorithm 10: antSolution

Function *antSolution*$(V = \{D, C\}, K, \tau, \alpha, \beta)$
- $V_{free} = C$
- **while** $V_{free} \neq \emptyset$ **do**
  - $d =$ selectDepot$(vt, V_{free}, K^{(vt)}, \tau)$
  - $vt =$ selectVehicleType(V_{free}, K, τ)
  - $pos =$ vehicle's position
  - $k =$ selectCluster$(vt, d, v, V_{free}, K^{(pos)(vt)}, \tau, \alpha, \beta)$
  - $V_{candidates} = V_{\text{free}} \cap K_k^{(pos)(vt)}$
  - $c =$ selectCustomer$(vt, d, pos, V_{candidates}, \tau, \alpha, \beta)$
  - **if** $v_{load} < c^{(demand)}$ **then**
    - $R_d^{(vt)} = R_d^{(vt)} + \{d\}$
    - $v_{load} = vt_{capacity}$
  - **end if**
  - $R_d^{vt} = R_d^{vt} + \{c\}$
  - $v_{load} = v_{load} - c^{(demand)}$
  - $V_{\text{free}} = V_{\text{free}} - \{c\}$
- **end while**
- **for each** $d \in D$ *and* $vt \in VT$ //Vehicles return to their depots
  - $R_d^{vt} = R_d^{vt} + \{d\}$
- **end for**
- **return** $R = \{R_1^1, R_2^1, ..., R_2^3, R_3^3, R_D^{VT}\}$

Algorithm 11: selectVehicleType

Function *selectVehicleType*$(vt, d, V_{free}, K^{(vt)}, \tau)$
- **for** *each* $v_i \in V^{(vt)(d)}$ **do**
  - $V_{\text{cand}} = \emptyset$
  - $pos \leftarrow$vehicle's current location
  - **for** $k = 1$ *to* n_{prim} **do**
    - $V_{cand} = V_{cand} + V_{free} \cap K_k^{(pos)(vt)}$
  - **end for**
  - $p(v_i) = \sum_{v_j \in V_{cand}} \tau_{v_{pos} v_j}^{(vt)(d)}$
- **end for**
- $p_{sum} = \sum_{v_i \in V^{(vt)(d)}} p(v_i)$
- **return** $rouletteWheel(p(V^{(vt)(d)}), p_{sum})$

11.1 AACONC+ Results and Comparison to heuristics

Table 20: AACONC+ Results

Instance	Best	Average	gap(%)	Worst	gap(%)	Average time(s)
p01-C	178.56	196.74	10.18	225.60	26.34	141
p02-C	172.73	196.57	13.8	234.78	35.92	123
p03-C	173.82	187.26	7.73	210.15	20.90	350
p04-C	629.88	644.84	2.37	675.63	7.26	1203
p05-C	573.03	613.47	7.06	692.44	20.84	884
p06-C	379.10	390.34	2.96	403.39	6.41	1316
p07-C	288.24	297.91	3.35	314.78	9.21	798
p08-C	3023.77	3100.23	2.53	3265.97	8.01	1635
p09-C	1705.03	1774.86	4.10	1846.92	8.32	3391
p10-C	1286.28	1319.78	2.60	1349.47	4.91	3549
p11-C	992.16	1035.63	4.38	1061.7	7.01	3600
Average	854.78	887.06	5.55	934.62	14.10	1544.55

Table 21: AACONC+ best and heuristic results

Instance	AACONC+	heuristic (prox.)	gap(%)	heuristic (k-means)	gap(%)
p01-C	**178.56**	217.42	21.76	191.03	6.98
p02-C	**172.73**	219.2	26.90	188.60	9.19
p03-C	**173.82**	214.84	23.60	185.82	6.90
p04-C	**629.88**	668.81	6.18	694.96	10.33
p05-C	**573.03**	630.78	10.08	624.33	8.95
p06-C	**379.10**	435.42	14.86	416.25	9.80
p07-C	**288.24**	309.48	7.37	314.41	9.08
p08-C	**3023.77**	3333.93	10.26	3186.02	5.37
p09-C	**1705.03**	1917.82	12.48	1964.49	15.22
p10-C	**1286.28**	1493.13	16.08	1500.38	16.64
p11-C	**992.16**	1145.75	15.48	1031.09	3.92
Average	**854.78**	962.42	15.00	936.13	9.31