A Collaborative Drone-Truck Delivery System with Memetic Computing Optimization

Ruonan Zhai [1], Yi Mei [2], Tong Guo [2], and Wenbo Du [2]

[1]Beihang University
[2]Affiliation not available

December 7, 2023

Abstract

With technological breakthroughs, drone deliveries have become increasingly popular, especially during the COVID-19 pandemic. Driven by both economical benefit and efficiency, drone-truck combined deliveries are in demand. However, it is very challenging to handle the collaboration between trucks and drones. Existing methods for truck-only routing cannot be directly applied, since their solution representations and search operators cannot consider the drone-truck collaborations effectively. In this paper, we model the system as Traveling Salesman Problem with Drones (TSP-D), and propose a new memetic algorithm named MATSP-D for solving it. Specifically, we design a new drone-truck solution representation and develop new crossover and local search operators under the new representation, which can modify the drone services effectively. MATSP-D conducts exploration by crossover, and exploitation by a variable neighborhood search process. The experimental results show that the proposed MATSP-D significantly outperforms the state-of-the-art algorithms for most test instances, especially the large instances with more complex collaborations between the truck and drone. Further analysis verifies the effectiveness of the newly developed local search operators in searching for better drone-truck collaborations.

A Collaborative Drone-Truck Delivery System with Memetic Computing Optimization

Ruonan Zhai, *Student Member, IEEE,* Yi Mei, *Senior Member, IEEE,* Tong Guo, *Student Member, IEEE,* and Wenbo Du, *Member, IEEE*

***Abstract*—With technological breakthroughs, drone deliveries have become increasingly popular, especially during the COVID-19 pandemic. Driven by both economical benefit and efficiency, drone-truck combined deliveries are in demand. However, it is very challenging to handle the collaboration between trucks and drones. Existing methods for truck-only routing cannot be directly applied, since their solution representations and search operators cannot consider the drone-truck collaborations effectively. In this paper, we model the system as Traveling Salesman Problem with Drones (TSP-D), and propose a new memetic algorithm named MATSP-D for solving it. Specifically, we design a new drone-truck solution representation and develop new crossover and local search operators under the new representation, which can modify the drone services effectively. MATSP-D conducts exploration by crossover, and exploitation by a variable neighborhood search process. The experimental results show that the proposed MATSP-D significantly outperforms the state-of-the-art algorithms for most test instances, especially the large instances with more complex collaborations between the truck and drone. Further analysis verifies the effectiveness of the newly developed local search operators in searching for better drone-truck collaborations.**

***Index Terms*—Collaborative Drone-Truck Delivery, Traveling salesman problem with drones, evolutionary computation, memetic algorithm**

I. Introduction

DRONE delivery systems have contributed to human tasks in recent years. For example, during the outbreak of Covid-19, drones have provided logistical support for the fight against the epidemic, participating in drone disinfection, publicity, patrols, and contact-free delivery of relief supplies. Commercial companies such as Zipline [1], Manna [2] and SF Express [3] used drones to provide supplies for people in Ghana, Ireland and China in 2020–2021. As drones are limited by capacity and endurance, the drone-truck combined operations (DTCO) in delivery was proposed [4], and showed greater potential than drone-only systems [5]. In DTCO, a drone can travel between trucks and customers and deliver/pick up parcels without any human intervention. The truck can deliver products at the same time or serve as a mobile hub for drones.

DTCO is a very promising operation in delivery. It can be essentially modeled as the traveling salesman problem with drones (TSP-D) [6]. Briefly speaking, TSP-D is to serve all the nodes in the given graph by a truck and a drone within the shortest time subject to the following constraints:

1) Each node is served exactly once by either the truck or the drone.
2) The drone must depart from the truck at a node, serve exact one *fly* node, and return to the truck at another node.
3) The truck/drone must wait for each other at the node where the drone returns to the truck.

The collaboration between the truck and drone in TSP-D makes a more efficient and flexible delivery system than the truck-only system (TSP). However, TSP-D is also more challenging than TSP, due to the interaction between the truck and drone routes, which results in a large and complex solution space. TSP-D is NP-hard [7]. Exact methods (e.g., [4], [6], [8]–[11]) are only applicable to small-size instances. To solve larger problem instances, meta-heuristic methods (e.g., [4], [6], [12], [13]) can obtain near-optimal solutions in a short time. Most existing search methods for TSP-D (e.g., [12]–[15]) are individual-based search, and can easily get stuck into poor local optima. They also optimize the truck and drone routes separately, thus cannot handle the complex interactions between the truck and drone routes effectively. This leads to insufficient exploration in the solution space, which inevitably misses some promising solutions. However, optimizing the truck and drone routes simultaneously can result in huge and complex search space, which makes it increasingly difficult to design effective search methods.

To not miss the complex interactions between the truck and drone, we optimize both kinds of routes simultaneously. To search in the resultant huge solution space effectively and reduce the chance of getting stuck into poor local optima, we select the Memetic Algorithm (MA) [16] as the technique to solve TSP-D. MA is a hybrid evolutionary algorithm that combines the exploration ability of genetic algorithm and the exploitation strength of local search. One of the common strategies is replacing mutation with local search [17], [18]. When solving complex combinatorial optimization problems with MA, the crossover operator can help the solutions

This work was funded in part by the National Natural Science Foundation of China (NSFC) under Grant XXX and Marsden Fund of New Zealand Government under Contract XXX. *(Corresponding authors: Yi Mei, Wenbo Du)*

Ruonan Zhai, Tong Guo and Wenbo Du are with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China (e-mail: zhairuonan1106@126.com; guotong1997@buaa.edu.cn; wenbodu@buaa.edu.cn).

Yi Mei is with School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. (email:yi.mei@ecs.vuw.ac.nz).

escape from the current local optima[1], and the local search can refine the region around the solutions. Moreover, it is easy to incorporate domain knowledge into the MA design, e.g., through the design of problem-specific local search operators. MAs have been successfully applied to solve many complex combinatorial optimization problems, such as vehicle routing problems [19], [20], scheduling [21] and knapsack problem [22].

When designing MA for TSP-D, it is important to design proper individual representation and search operators to balance exploration and exploitation. The solution space of TSP-D is very complex due to the decisions of *truck and drone routing*. Specifically, it requires to determine (1) which nodes are served by the truck and which by the drone, (2) the truck route and (3) the departure and return nodes for the drone to serve each drone node. These decisions interact with each other, making it challenging to search in the huge and complex solution space.

The existing representation and search operators are not effective enough to handle the complex interactions between the different decisions in TSP-D. For example, the variable neighborhood search algorithm in [13] is an individual-based search with a heuristically generated initial solution, and is lack of exploration ability. The hybrid genetic algorithm [23], [24] represents an individual by a giant sequence of the nodes, and decodes it into a feasible TSP-D solution by a split algorithm [23] for evaluation. The crossover operator is based on the giant sequences, while the local search operators are based on the decoded TSP-D solutions (i.e., truck and drone routes). The limitation of this algorithm is that there is an inconsistency before and after the decoding. In addition, the local search operators cannot explicit change the number of drone routes.

To address the above issues, this paper develops a new MA for solving TSP-D more effectively, by designing a new explicit individual representation and its corresponding genetic operators. The paper has the following contributions:

1) Design a new explicit individual representation that consists of both the truck and drone routes.
2) Develop a new extended crossover operator and new effective local search operators for the newly designed explicit solution representation.
3) Propose a MA based on the explicit solution representation and genetic operators.
4) Verify the effectiveness of the newly proposed MA on a wide range of TSP-D instances.

II. Background

A. Notations and Problem Definition

For quick reference, the parameters, indices, and decision variables in the problem definition are listed in Table I.

[1]This is different from conventional GA, where crossover mainly focuses on exploitation. This is because the standard crossover operators often generate invalid offspring that violate the complex constraints of the combinatorial optimization problems. After repairing the offspring, they become substantially different from both parents, making it more likely to jump out of the current local optima.

TABLE I
Parameters, Indices and decision variables of TSP-D.

Name	Description
N	The number of customers.
v_i	The ith customer vertex, $i \in \{1, ..., N\}$.
v_0	The depot vertex.
(v_i, v_j)	The edge (v_i, v_j).
$c(v_i, v_j)$	The travel time of the truck for the edge (v_i, v_j).
$c^d(v_i, v_j)$	The travel time of the drone for the edge (v_i, v_j).
o	An operation
O	Set of possible operations.
V	Set of all customers and the depot v_0.
$S \subset V$	A subset of nodes.
$O(v)$	Set of operations containing v.
$O^-(v)$	Set of operations with start node v.
$O^+(v)$	Set of operations with end node v.
$O^-(S)$	Set of operations with start node in S and end node outside S.
$O^+(S)$	Set of operations with start node outside S and end node in S.
x_o	A binary variable that indicates the use of operation o, $x_o \in \{0, 1\}, \forall\, o \in O$.
y_v	A binary variable that indicates whether node v is a start node in at least one operation of the solution, $y_v \in \{0, 1\}, \forall\, v \in V$.

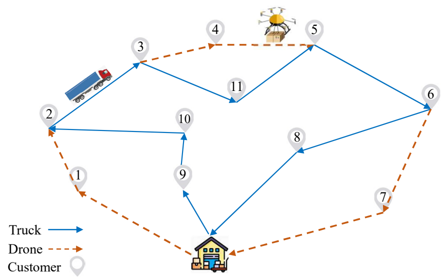
TSP-D [6] seeks for the minimum cost plan for a truck and a drone to serve customers cooperatively. Given a graph $G(V, E)$, where $V = \{v_0, v_1, ..., v_N\}$ is the vertex set, containing the depot v_0 and N customers $\{v_1, ..., v_N\}$. $E = \{(v_i, v_j) \mid v_i, v_j \in V, v_i \neq v_j\}$ is the set of edges. For $(v_i, v_j) \in E$, the travel time of the truck and drone for the edge are denoted as $c(v_i, v_j)$ and $c^d(v_i, v_j)$, respectively.

A TSP-D solution is a sequence of *operations*, each consisting of a *start node*, an *end node*, an optional *fly node* (which can be null) and a list of *truck nodes* (could be empty). Noting that the drone can only serve one customer after launching, which suggests that the *fly node* is either one customer node or null. In an operation, the truck serves and departs from the start node, serves a (possibly empty) list of truck nodes, and finally reaches the end node. If the fly node is not null, then the drone takes off the truck at the start node, serves the fly node, and returns to the truck at the end node. Without loss of generality, we only consider the *elementary* truck operations. That is, if the fly node is null, then the list of truck nodes must be empty.

Under this definition, the set of possible operations is $O \doteq \{\langle v^s, v^e, v^f, \mathbf{v}^t \rangle\}$, where $v^s \in V$, $v^e \in V$ and $v^f \in \{V \cup \textit{null}\} \setminus v_0$ represent the start node, end node, and fly node, respectively. $\mathbf{v}^t$ stands for the list of truck nodes, where $v' \in V \setminus v_0$ for each truck node $v' \in \mathbf{v}^t$. All the nodes in the operation are different from each other, i.e., $v^s \neq v^e \neq v^f \neq v', \forall\, v' \in \mathbf{v}^t$. In addition, $v^f = \textit{null} \rightarrow \mathbf{v}^t = [\ ]$.

Fig. 1 shows an example TSP-D solution serving 11 customers (index from 1 to 11, the depot is indexed 0) and its representation as a sequence of operations. The solution is illustrated in Fig. 1(a), where the solid arrows stand for the truck route and the dashed arrows are the drone routes. The solution contains 5 operations, as shown in Fig. 1(b).

As the truck and the drone serve customers simultane-



(a)

Operation	Start Node	End Node	Fly Node	Truck Node
1	0	2	1	9,10
2	2	3	null	[ ]
3	3	5	4	11
4	5	6	null	[ ]
5	6	0	7	8

(b)

Fig. 1. An example TSP-D solution and its sequence of operations.

ously, the cost of an operation is the maximal cost (e.g., travel time) induced by the truck and the drone. For example, in Fig. 1(b), the cost of operation 2 is $c(o_2) = c(2,3)$, since the drone cost is zero. On the other hand, the cost of operation 5 is $c(o_5) = \max\{c(6,8)+c(8,0), c^d(6,7)+c^d(7,0)\}$.

To formulate the problem, we first define the following notations. The set of all possible operations is denoted as O. For each operation $o \in O$, c_o denotes the operation cost. For each node $v \in V$, let $O(v), O^-(v), O^+(v) \subset O$ be the set of all operations containing v, with start node v, and with end node v, respectively. For each subset of nodes $S \subset V$, let $O^-(S) \subset O$ be the set of all operations with start node in S and end node in $V \setminus S$. Similarly, let $O^+(S) \subset O$ be the set of all operations with end node in S and start node in $V \setminus S$. The binary decision variable $x_o = 1$ if the operation o is selected in the solution, and $x_o = 0$ otherwise. The binary decision variable $y_v = 1$ if node v is a start node in at least one operation of the solution, and $y_v = 0$ otherwise.

Then, TSP-D can be formulated as follows [6]:

$$\min \quad \sum_{o \in O} c_o x_o \tag{1}$$

$$\text{s.t.:} \quad \sum_{o \in O(v)} x_o \geq 1, \forall\, v \in V \tag{2}$$

$$\sum_{o \in O^+(v)} x_o \leq N \cdot y_v, \forall\, v \in V \tag{3}$$

$$\sum_{o \in O^+(v)} x_o = \sum_{o \in O^-(v)} x_o, \forall\, v \in V \tag{4}$$

$$\sum_{o \in O^+(S)} x_o \geq y_v, \forall\, S \subset V \setminus v_0, v \in S \tag{5}$$

$$\sum_{o \in O^+(v_0)} x_o \geq 1 \tag{6}$$

$$y_{v_0} = 1 \tag{7}$$

$$x_o \in \{0,1\}, \forall\, o \in O \tag{8}$$

$$y_v \in \{0,1\}, \forall\, v \in V \tag{9}$$

The objective Eq. (1) minimizes the total cost of the selected operations. Eq. (2) ensures that all the nodes are visited. Eq. (3) indicates that y_v must be 1 if at least one operation with the start node v is selected. Eqs. (4)–(6) guarantee that the operations in the solution $\{o \in O \mid x_o = 1\}$ form a feasible route, i.e., an Eulerian graph. Eq. (7) implies that the solution starts and ends at the depot. Eqs. (8) and (9) are the domain constraints of the x and y variables.

B. Related Work

Drones play an essential role in various applications [4], [25], [26]. Drone delivery routing [27]–[29] has attracted more and more research attention recently. Most studies concentrate on DTCO [30], and model it as TSP-D [6]. There are a variety of TSP-D variants, such as the min-cost TSP-D [23] that minimizes the total transportation cost and waiting time of vehicles. the TSP-D with drone battery capacity [31], and the Flying Sidekick TSP (FSTSP) [4], [32] that disallows drones to take off and land on the same customer node. It can also be extended to DTCO systems with multiple drones [33] and each drone can serve multiple customers in each flight [34]–[37]. The drone may also depart and land on the truck at any location along an arc [38], [39]. The trucks can regularly resupplied by drones [40], [41], and the truck can recharge the drone instead of replacing a new battery [42], [43]. Multi-objective models are considered in [44], [45].

The exact methods (e.g., integer linear programming) [4], [6], [8]–[10], [46], [47] can guarantee optimality, but are only applicable to small-size instances (at most 39 customers as shown in [10]). For large-scale real-world instances, (meta-)heuristic methods (e.g., [4], [6], [8], [23], [48]–[53]) are more commonly used. Murray and Chu [4] proposed a heuristic named TSP-MC. First, a TSP route is obtained by a TSP solver [54]. Then, some customers are allocated to the drone to reduce cost. Agatz et al. [6] developed a route first-cluster second heuristic. First, a truck route is obtained by a TSP solver. Then, two heuristics, i.e., a greedy partitioning heuristic and an exact partitioning algorithm, are applied to the truck route. To explore the solution space more comprehensively, multiple truck routes are obtained and partitioned. Ha et al. [23] presented two heuristics to solve the min-cost TSP-D, based on GRASP and local search, respectively. Both heuristics start from a TSP route. GRASP involves new local search operators for TSP-D. Yurek and Ozmutlu [8] presented a heuristic to reduce the computing cost of the exact method. The heuristic first applies the nearest neighbor approach to obtain the truck routes and then solves a mixed integer linear programming model to assign the customers to the drone. Ponza [12] proposed a simulated annealing algorithm that employs four move operators to improve the current solution. Freitas et al. [55] proposed a randomized variable neighborhood descent algorithm for the FSTSP that starts from a TSP solution obtained by Concorde. A Hybrid General Variable Neighborhood Search algorithm (HGVNS) was proposed in [13]. It employs seven neighborhoods in the framework of the VNS of the TSP-

Algorithm 1: Framework of MATSP-D

$\mathcal{P} = \texttt{init}(\textit{popsize}, \textit{trials})$;
for $S \in \mathcal{P}$ **do** $fit(S) = \texttt{evaluate}(S)$;
$S^* = \arg\min_{S \in \mathcal{P}} fit(S)$;
while *stopping criterion is not met* **do**
  Set the offspring population $\mathcal{P}' = \emptyset$;
  for $i = 1 \to$ *offsize* **do**
    $C = null$;
    $S_1, S_2 = \texttt{TournamentSelection}(\mathcal{P}, 2)$;
    $S_x = \texttt{crossover}(S_1, S_2)$;
    if S_x *is not a clone of any* $S \in \mathcal{P}$ **then** $C = S_x$;
    if $\texttt{rand}(0,1) < \Pr_{ls}$ **then**
      $S_m = \texttt{LocalSearch}(S_x)$;
      if S_m *is not a clone of any* $S \in \mathcal{P}$ **then** $C = S_m$;
    if $C \neq null$ **then** $\mathcal{P}' = \mathcal{P}' \cup C$;
  $\mathcal{P}$ = the top *popsize* individuals in $\mathcal{P} \cup \mathcal{P}'$, update S^*;
return S^*;

MC algorithm [4]. Other local search operators [56]–[58] also mainly modify the drone and truck routes separately. A Hybrid Genetic Algorithm (HGA) was proposed in [24]. It uses a giant TSP tour as a chromosome, which is decoded into a TSP-D solution by the split procedure. HGA applies 16 local search neighborhoods to the individuals in the education step, which change the truck route and drone route separately. Other population-based algorithm such as Artificial Bee Colony (ABC) was designed in [59].

In summary, the exact methods for TSP-D can guarantee optimality, but are restricted to small-size instances due to their high computational complexity. The meta-heuristics are more practical for large real-world instances. Most existing meta-heuristics for TSP-D optimize the routes of the truck and drone separately. Although the search space is much reduced, they ignore the complex interactions between the truck and drone, and may miss promising solutions. Moreover, the existing search operators cannot explore the solution space effectively. Specifically, they cannot effectively change the number of drone nodes, and may lead the search to poor local optima. To address these issues, we propose the new MATSP-D.

III. Memetic Algorithm for TSP-D

Algorithm 1 shows the overall framework of MATSP-D. First, a population of individuals $\mathcal{P}$ is initialized. At each generation, an offspring population $\mathcal{P}'$ is generated. For generating each offspring, two parents S_1 and S_2 are selected by the size-2 tournament selection [60]. An offspring S_x is generated by crossing over S_1 and S_2. S_x has a probability of $\Pr_{ls}$ to undergo a local search to generate S_m. If S_m or S_x has different fitness from all the individuals in $\mathcal{P}$, then it is inserted into $\mathcal{P}'$. Finally, the top *popsize* individuals in $\mathcal{P} \cup \mathcal{P}'$ are selected into the next generation. The evolution process continues until the stopping criterion is met, e.g., after a certain number of generations. Finally, the best-found solution is returned.

Algorithm 2: $fit(S) = \texttt{evaluate}(S)$

Input: An individual $S = (\mathbf{NS}, \mathbf{SV})$
Output: The fitness value of the individual $fit(S)$
$fit = 0$, $st = 0, ptr = 0, fly = null, truck = [\ ]$;
while $ptr < N + 2$ **do**
  $ptr \leftarrow ptr + 1$;
  if $\mathbf{SV}[ptr] = \mathsf{T}$ **then**
    $op = \langle \mathbf{NS}[st], \mathbf{NS}[ptr], fly, truck \rangle$;
    $fit = fit + c(op)$, $st = ptr, fly = null, truck = [\ ]$;
  else if $\mathbf{SV}[ptr] = \mathsf{I}$ **then**
    $truck = [truck, \mathbf{NS}[ptr]]$;
  else $fly = \mathbf{NS}[ptr]$;
return fit;

NS	0	1	9	10	2	3	4	11	5	6	7	8	0
SV	T	F	I	I	T	T	F	I	T	T	F	I	T

Fig. 2. An example of the newly designed two-level representation.

A. Individual Representation and Evaluation

We design a new explicit two-level individual representation for TSP-D. An individual consists of two parts, i.e., the *node sequence* and *state vector*. The node sequence describes the order of customers visited by the truck and drone, with the depot at the beginning and the end. Thus, it has $N + 2$ elements, where N is the number of customers. The state vector has the same dimensionality as the node sequence. At each dimension d, it describes the state of the action that arrives the dth customer in the node sequence. It can take three possible values as follows:

- T (*Terminal*): the node is served by the truck with the drone on the truck (the drone can also take off or land on the truck at the node).
- I (*Internal*): the node is served by the truck when the drone is not on the truck.
- F (*Fly*): the node is served by the drone.

Note that no service is needed for the depot at the beginning and end of the node sequence. For the sake of convenience, we define their states as T.

Fig. 2 shows an example of the two-level representation of the solution shown in Fig. 1. Each operation corresponds to the sub-sequence between two adjacent nodes with state T. For example, the first operation corresponds to the node sequence $[0, 1, 9, 10, 2]$ and state vector $[\mathsf{T}, \mathsf{F}, \mathsf{I}, \mathsf{I}, \mathsf{T}]$, which is $\langle 0, 2, 1, [9, 10] \rangle$.

Given an individual $(\mathbf{NS}, \mathbf{SV})$ under the proposed two-level representation, the evaluation process is shown in Algorithm 2. It detects the operations in the solution by scanning the node sequence and state vector. If the current state is T, then it calculates the cost of the newly detected operation. If the current state is I, then it appends the current node into the list of the truck nodes. If the current state is F, it sets the current node as the fly node. Finally, it calculates the fitness, which is the total cost of all the operations.

Algorithm 3: $S_x = \texttt{crossover}(S_1, S_2)$

Input: Two parents S_1, S_2
Output: Offspring S_x
Sample a random index $i_{cross} \in [2, \ldots, N]$;
$\mathbf{NS}'[0, .., i_{cross}] = \mathbf{NS}_1[0, .., i_{cross}]$;
$\mathbf{NS}'[i_{cross}+1, .., N+2] = \mathbf{NS}_2[i_{cross}+1, .., N+2]$;
$\mathbf{SV}'[0, .., i_{cross}] = \mathbf{SV}_1[0, .., i_{cross}]$;
$\mathbf{SV}'[i_{cross}+1, .., N+2] = \mathbf{SV}_2[i_{cross}+1, .., N+2]$;
$R = \{\}$, $M = \{\}$; // Repair process
for $i = i_{cross} \rightarrow N+2$ **do**
  if $\mathbf{NS}_2[i] \in \mathbf{NS}_1[0, .., i_{cross}]$ **then** $R = R \cup i$;
  if $\mathbf{NS}_1[i] \in \mathbf{NS}_2[0, .., i_{cross}]$ **then** $M = M \cup \mathbf{NS}_1[i]$;
for $i = 1 \rightarrow |R|$ **do** $\mathbf{NS}'[R[i]] = M[i]$;
if $\mathbf{SV}'[i_{cross}] \neq$ "T" **then**
  $\mathbf{SV}'[i_{cross}] =$ "T", $i = i_{cross} + 1$;
  while $\mathbf{SV}'[i] =$ "I" **do**
    $\mathbf{SV}'[i] =$ "T", $i = i + 1$;
return $S_x = (\mathbf{NS}', \mathbf{SV}')$;

B. Initialization

To initialize the population a TSP route is first generated by Gurobi as a high-quality initial solution. Such seeding strategy has been successfully applied in previous studies [61]. The remaining individuals are generated randomly. To ensure feasibility, all the states are set to T, i.e., all the customers are served by the truck. No individual with the same fitness is allowed in the initial population. If a newly initialized individual violates this condition, the random generation is repeated for at most *trials* times for re-initialization.

C. Crossover

We develop a new specific crossover operator for the two-level representation by extending the well-known Partial-Mapped Crossover (PMX) [62] for TSP, which is named PMX with Drone (PMX-D). First, it applies the PMX with a single crossover point on the two parents. Specifically, it exchanges the node sequence and state vector at a random crossover point, and then repairs the duplicated and missing nodes in the node sequence.

Note that after applying the PMX, the state vector may become infeasible (e.g., internal nodes between terminal nodes without fly node, and multiple fly nodes between terminal nodes). To address this issue, we design a new *state vector repair operator* to adjust the state vector accordingly. Specifically, we reset the last element of the former sub-sequence (i.e., $\mathbf{SV}[i_{cross}]$) to be T. Then, for each subsequent element, if it is I, we replace it with T, since there is no fly node in between.

Fig. 3 shows an example of the develop PMX-D operator, where the crosser point is shown in dashed lines. After swapping the sub-sequences, nodes 1 and 3 are duplicated and nodes 6 and 8 are missing in the offspring. PMX replaces node 1 at index 7 with node 6 and node 3 at index 8 with node 8. To repair the state vector, since the state of node 4 at index 4 is already T, there is no need for change. However, indices 5 and 6 have the state I. Thus, we change them to T to avoid internal nodes without fly nodes.

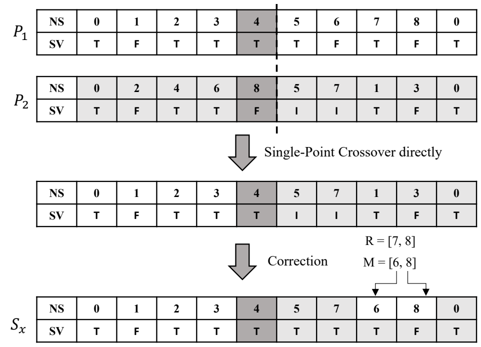


Fig. 3. An example of the proposed PMX-D crossover.

Algorithm 4: $S^* = \texttt{LocalSearch}(S_0)$

Input: The initial solution S_0
Output: The improved solution S^*
$S^* = S_0$, $[\mathcal{N}_1, .., \mathcal{N}_6] = [\mathcal{N}_{\mathbf{Ins}}, \mathcal{N}_{\mathbf{DM}}, \mathcal{N}_{\mathbf{2\text{-}opt}}, \mathcal{N}_{\mathbf{DI}}, \mathcal{N}_{\mathbf{Swap}}, \mathcal{N}_{\mathbf{DD}}]$;
for $k = 1 \rightarrow 6$ **do**
  $S' = \texttt{RandomSample}(\mathcal{N}_k(S^*))$;
  $S' = \texttt{VND}(S', [\mathcal{N}_1(\cdot), \ldots, \mathcal{N}_k(\cdot)])$;
  if $fit(S') < fit(S^*)$ **then** $S^* = S'$, $k = 1$;
return S^*;

D. Local search

After the crossover, we employ a Variable Neighborhood Search (VNS) [63] process to further improve S_x due to its strong ability to jump out of local optima.

The VNS process is described in Algorithm 4. It is based on 6 pre-defined neighborhoods. Given an initial solution S_0, it examines the neighborhoods one by one. For each neighborhood, to escape from the current local optimum, it first *shakes* the current solution S^*, i.e., randomly samples a neighbor S' from its current neighborhood $\mathcal{N}_k(S^*)$. Then, it improves S' through a Variable Neighborhood Descent (VND) process (shown in Algorithm 5). If the VND improves the current solution S^*, then it replaces S^* and the search starts from the first neighborhood again. Otherwise, it continues to explore the next neighborhood. The entire VNS process stops when no improvement can be found.

Algorithm 5 shows the VND process. Given a solution S and a list of neighborhoods $[\mathcal{N}_1(\cdot), \ldots, \mathcal{N}_k(\cdot)]$, the VND examines each neighborhood in turn. For each neighborhood, if the best neighbor within the neighborhood is better than the current solution, then it replaces the current solution, and the search starts from the first neighborhood again. The VND stops when no neighborhood makes improvements.

1) Neighborhood Structures: The neighborhood structure is determined by the move operators. Specifically, the neighborhood of a solution S determined by the operator op can be written as $\mathcal{N}_{\texttt{op}}(S) = \{S' \mid S' = \texttt{op}(S)\}$.

As shown in Algorithm 4, we design 6 different neighborhoods using 6 operators. $\mathcal{N}_1(\cdot)$, $\mathcal{N}_3(\cdot)$, $\mathcal{N}_5(\cdot)$ are traditional operators that modify the node sequence as follows.

- **Insert**(i, j): Move a node i to a new position j.
- **2-opt**(i, j): Reverse the direction of a sub-sequence between positions i and j ($i < j$).

Algorithm 5: $S^* = \texttt{VND}(S, [\mathcal{N}_1(\cdot), \ldots, \mathcal{N}_k(\cdot)])$

Input: Solution S, neighborhoods $[\mathcal{N}_1(\cdot), \ldots, \mathcal{N}_k(\cdot)]$
Output: Improved solution S^*

$S^* = S$;
for $i = 1 \to k$ **do**
  $S' = null$, $fit' = \infty$;
  for $S'' \in \mathcal{N}_i(S^*)$ **do**
    $fit(S'') = \texttt{IncrEvaluate(S'')}$;
    if $fit(S'') < fit'$ **then** $S' = S''$, $fit' = fit(S'')$;
  if $fit(S') < fit(S^*)$ **then** $S^* = S'$, $i = 1$;
return S^*;

NS	0	1	2	3	4	5	6	7	8	0
SV	T	F	T	T	F	I	I	T	F	T

(a)

NS	0	2	3	4	1	5	6	7	8	0
SV	T	F	T	T	F	I	I	T	F	T

(b)

NS	0	1	5	4	3	2	6	7	8	0
SV	T	F	T	T	F	I	I	T	F	T

(c)

NS	0	1	4	3	2	5	6	7	8	0
SV	T	F	T	T	F	I	I	T	F	T

(d)

Fig. 4. Examples of (a) the original solution, and the solutions obtained by the (b) Insert$(1, 4)$, (c) 2-opt$(2, 5)$ and (d) Swap$(2, 4)$ operators.

- **Swap**(i, j): Swap the nodes at position i and j $(i < j)$.

The above operators change the node sequence only. When applied to a solution, the state vector remains unchanged. Fig. 4 shows an example of the three traditional operators.

The traditional operators cannot change the number of drone nodes. To address this issue, we design three new operators $\mathcal{N}_2(\cdot)$, $\mathcal{N}_4(\cdot)$, $\mathcal{N}_6(\cdot)$ to insert, delete and move drone nodes, by modifying the state vector as follows.

- **Drone-Move**(s, d, f): It pushes $\mathbf{NS}[s]$ forward or backward, depending on d and f, where $d \neq 0$ denotes the move distance, and $f \in \{-1, 1\}$ indicates whether $\mathbf{NS}[s]$ is a fly node or end node. If $f = -1$, the start node $\mathbf{NS}[s-1]$ and fly node $\mathbf{NS}[s]$ are moved to positions $s + d - 1$ and $s + d$. Otherwise, the end node $\mathbf{NS}[s]$ is moved to position $s + d$. If $d * f > 0$, the state of the nodes from s to $s + d$ are changed to I. Otherwise, their states become T. To ensure feasibility, the operator can only be applied when $d * f \leq 0$ or the state of the nodes from positions s to $s + d$ are all T.
- **Drone-Insert**(s, e): It selects a truck node $\mathbf{NS}[s + 1]$ and serve it by drone. The drone departs from $\mathbf{NS}[s]$, serves $\mathbf{NS}[s + 1]$, and lands on $\mathbf{NS}[e]$. The state of $\mathbf{NS}[s + 1]$ is turned into F. The state of the nodes between $\mathbf{NS}[s+1]$ and $\mathbf{NS}[e]$ (exclusive) are changed to I. To ensure feasibility, this operator can only be applied under the following conditions: (1) the drone is available from positions s to e, i.e., there is no node between $\mathbf{NS}[s]$ and $\mathbf{NS}[e]$ with F or I state; (2) $\mathbf{NS}[s+1]$ is served by the truck, i.e., its state is T or I.
- **Drone-Delete**(f): It turns a drone node $\mathbf{NS}[f]$ into truck-served. For each node from $\mathbf{NS}[f]$ to the next terminal node, if its state is I, we change it to T.

NS	0	1	2	3	4	5	6	7	8	0
SV	T	F	T	T	T	T	F	I	T	T

NS	0	1	2	3	4	5	6	7	8	0
SV	T	F	T	F	I	T	F	I	T	T

(a)

NS	0	1	2	3	4	5	6	7	8	0
SV	T	F	T	T	T	T	T	T	T	T

(b)

NS	0	1	2	3	4	5	6	7	8	0
SV	T	F	T	T	T	F	I	I	T	T

(c)

NS	0	1	2	3	4	5	6	7	8	0
SV	T	F	T	T	T	T	F	T	T	T

(d)

Fig. 5. Examples of (a) Drone-Insert$(2, 5)$, (b) Drone-Delete(6), (c) Drone-Move$(6, -1, -1)$, (d) Drone-Move$(8, -1, 1)$.

Fig. 5 shows the examples of (a) Drone-Insert$(2, 5)$, (b) Drone-Delete(6), (c) Drone-Move$(6, -1, -1)$, (d) Drone-Move$(2, 2, 1)$. In Fig. 5(a), the state of the node 3 is changed F and the state of the subsequent node 4 is changed to I. As Fig. 5(b) shows, the state of the fly node 6 is changed to T. Then, for the two nodes 7 and 8 between the node 6 and the next terminal node 0, since their states are I, we change their state to T. In Fig. 5(c), the state $[\mathsf{T}, \mathsf{F}]$ at index $(5, 6)$ are moved to index $(4, 5)$. As $d * f > 0$, the states from index 5 to 6 are changed to I. In Fig. 5(d), as $d * f < 0$, the states from index 7 to 8 are changed to T.

The order of the neighborhoods is decided based on the following considerations. Firstly, $\mathcal{N}_1(\cdot)$, $\mathcal{N}_3(\cdot)$ and $\mathcal{N}_5(\cdot)$ change the node sequence only. Therefore, it may be more efficient to separate them from the newly designed operators that change the state vector as well. The order of $\mathcal{N}_1(\cdot)$, $\mathcal{N}_3(\cdot)$, $\mathcal{N}_5(\cdot)$ is less important. Secondly, in the VND process, the search reverts to the first neighborhood once a better solution is found. Therefore, the neighborhood that tends to bring more improvement should be in the front. To this end, we place the traditional operators $\mathcal{N}_1(\cdot)$, $\mathcal{N}_3(\cdot)$, $\mathcal{N}_5(\cdot)$ before the new operators $\mathcal{N}_2(\cdot)$, $\mathcal{N}_4(\cdot)$, $\mathcal{N}_6(\cdot)$, since they are more likely to bring more improvement. Among the new operators, we put Drone-Move in the front, since it is the most likely to make improvements. Then, we put Drone-Insert before Drone-Delete, since Drone-Insert shortens the truck route and tends to bring more improvement.

2) Incremental Evaluation: To speed up evaluation during the local search, we can calculate the fitness of the neighbors *incrementally* [20] by only calculating the difference caused by the changed parts.

In the following, we describe the designed incremental evaluation of the operators used in MATSP-D:

- **Insert**(i, j): We recalculate the links between i and j. We change the link for the drone nodes between i and j, and the two drone nodes immediately before and after i and j. The complexity is usually $O(F)$, where F is the number of fly nodes between i and j.
- **Drone-Move**(s, d, f): If $f = -1$, since the end node in this operation is e, we replace the drone links $[s-1, s]$ and $[s, e]$ with $[s + d - 1, s + d]$ and $[s + d, e]$, and replace the truck links $[s - 2, s - 1]$ and $[s - 1, s + 1]$

with $[s+d-2, s]$ and $[s, s+1]$. The complexity is $O(1)$. If $f = 1$, since fly node in this operation is f, we replace the drone link $[f, s]$ with $[s, s+d]$. The complexity is also $O(1)$.
- **2-opt**(i, j): We recalculate the links associated with i and j, the drone nodes between i and j, and immediately before i and after j. The complexity is $O(F)$.
- **Drone-Insert**(s, e): We replace the truck links $[s, s+1]$ and $[s+1, s+2]$ with $[s, s+2]$, and add the drone links $[s, s+1]$ and $[s+1, e]$. The complexity is $O(1)$.
- **Swap**(i, j): We recalculate the links associated with i and j, the drone nodes immediately before and after i, and the drone nodes immediately before and after j. The complexity is $O(1)$.
- **Drone-Delete**(f): Let the next terminal node be e, we remove the drone links $[f-1, f]$ and $[f, e]$, and add the truck links $[f-1, f]$ and $[f, f+1]$. The complexity is $O(1)$.

For each operator, the recalculation of the links is the same as that for TSP, except that the cost of the links depend on their corresponding states (truck or drone cost). Note that F is usually much smaller than the number of customers N. With the above incremental evaluation, all the operators during the local search has much lower complexity than the full evaluation, which is $O(N)$.

IV. Experimental Studies

To evaluate the effectiveness of MATSP-D, two sets of experiments have been carried out. In the first experiment (Exp 1), we compare MATSP-D with the optimal solutions obtained by CPLEX on the small-scale benchmark instances provided by [6]. In the second experiment (Exp 2), we compare MATSP-D with several state-of-the-art algorithms (TSP-EP [6], TSP-GP [6], TSP-MC [4], HGVNS [13] and HGA [24]) on two larger realistic datasets. The first dataset contains 270 random instances provided by [6], with 50$\sim$100 customers following different types of distributions and varying drone speeds. The second dataset includes 24 instances extended from the TSPLib instances [13] derived from real-world graphs with 51$\sim$200 nodes.

Table II shows the parameter setting of MATSP-D in the experiments. The algorithm stops when either the maximal number of generations or runtime is reached. For the population size and offspring size, we set them to 500 for Exp 1, and 50 for Exp 2. Since the instances in Exp 1 are small, we can afford to set a sufficiently large population and offspring sizes to enhance exploration. However, such a large population size will make the convergence too slow for the larger instances in Exp 2. Thus, we set them to 50, based on the recommendation in [64], which used MA to solve problems with similar sizes. *trials* $= 10$ and $\Pr_{ls} = 0.3$ are set by rule of thumb. The maximal number of generations G_m is set to balance the algorithm convergence and computational time. Among the compared algorithms, TSP-EP, TSP-GP, TSP-MC and HGVNS are implemented without parameters. The result of HGA is obtained directly from the original paper. For all the compared algorithms, we ensured that the best available results are considered.

TABLE II
Parameter setting of MATSP-D.

Name	**Description**	**Value**
popsize	Population size	500 (Exp 1) / 50 (Exp 2)
offsize	Offspring population size	500 (Exp 1) / 50 (Exp 2)
trials	#trials in initialization	10
$\Pr_{ls}$	Local search probability	0.3
G_m	Maximal #generations	500 (Exp 1) / $2N$ (Exp 2)

TABLE III
The results of CPLEX and MATSP-D on the small dataset.

	Avg.Cost		Time(s)		
N	CPLEX	MATSP-D	CPLEX	MATSP-D	W-D
10	217.8	218.0	45	5	3-7
11	226.3	226.6	540	6	2-8
12	229.7	230.2	6318	7	3-7

For each instance, MATSP-D and HGVNS [13] were run 30 times independently. The compared deterministic algorithms (i.e. TSP-EP [6], TSP-GP [6] ,TSP-MC [4]) were run once. The experiments were executed on 64-bit version of Windows 10 Pro, with an Intel(R) Xeon(R) Gold 6240R (2.4GHz) CPU and 64GB RAM.

A. Results and Discussions

1) Small dataset: Table III shows the mean and standard deviation of the 30 runs of MATSP-D and the optima obtained by CPLEX on the small instances [6]. The instances contain 10, 11 and 12 uniformly distributed customers, and the drone has the same speed as the truck. We conducted Wilcoxon rank sum test to compare the results of MATSP-D with the optimum. The results are summarized in the "W-D" column, where "W" indicates the number of instances on which MATSP-D performed significantly worse than the optimal solutions obtained by CPLEX, and "D" the number of instances on which MATSP-D achieved the optimal solution (the same as CPLEX).

From the table, we can see that MATSP-D performs competitively compared with the optimum. MATSP-D consistently reached the optimum on 22 out of the 30 instances. The average gap from the optimum over all the instances is 0.1%. The runtime of CPLEX increases very rapidly with the increase of problem size. It is less than 1 minute for 10 customers, increases to about 10 minutes for 11 customers, and more than 2 hours for some instances with 12 customers. This greatly restricts the applicability of CPLEX for solving real-world large scale instances. In comparison, the runtime of MATSP-D is always within 10 seconds.

Overall, MATSP-D can consistently obtain promising solutions that are very close to the optimum on small instances within a very short time.

2) First large dataset: The 270 instances are randomly generated from three problem sizes (50, 75 and

TABLE IV
THE AVERAGE PERFORMANCE AND RUNTIME OF THE COMPARED ALGORITHMS ON THE **FIRST** LARGE DATASET [6].

		TSP-EP [6]		TSP-GP [6]		TSP-MC [4]		HGVNS [13]		MATSP-D		
	N	Avg.Cost	Time(s)	Avg.Cost	Time(s)	Avg.Cost	Time(s)	Avg.Cost	Time(s)	Avg.Cost	Std	Time(s)
	50	490.2	14	552.9	1	554	1	528.5	2	488.2	3.4	132
U-1	75	566.7	219	646.5	4	646.1	6	620	7	565.3	5.3	352
	100	645.4	845	730.3	12	739.8	23	706	26	646.7	7.4	1078
W-D-L		12-9-9	-	0-0-30	-	0-0-30	-	0-0-30	-	-	-	-
	50	640.8	14	753.4	1	711.6	2	688.9	2	631.9	5	141
S-1	75	859.3	179	1013.2	4	980.1	8	927.7	9	862.8	10.4	384
	100	1037.6	846	1219.9	12	1173.7	26	1128.3	28	1035.8	12.5	1136
W-D-L		7-10-13	-	0-0-30	-	0-0-30	-	0-0-30	-	-	-	-
	50	985.3	13	1152.4	1	1128.3	2	1062.1	2	984.6	7	88
D-1	75	1214.1	174	1447.5	5	1391.1	8	1325.9	9	1214.3	10.6	400
	100	1360.5	1200	1610	16	1544.5	26	1496.4	28	1366.3	13.6	1181
W-D-L		13-7-10	-	0-0-30	-	0-0-30	-	0-0-30	-	-	-	-
gap		-0.11%	-	-13.51%	-	-10.69%	-	-7.87%	-	-	-	-
	50	409.9	34	428.3	1	532.5	2	459	2	400.3	2.8	96
U-2	75	475.2	397	495.5	6	623.8	9	549.5	11	461.3	6.1	465
	100	548.6	2072	567.2	19	720.5	27	618.1	32	534.4	7.5	1499
W-D-L		2-4-24	-	0-0-30	-	0-0-30	-	0-0-30	-	-	-	-
	50	506.4	29	554.8	2	678	2	599.1	2	497.7	6.7	150
S-2	75	686.2	340	741.3	8	928	9	799.4	10	679.4	11.4	481
	100	833.9	1821	894.7	21	1114.4	28	952.6	32	827.7	15.5	1550
W-D-L		8-2-20	-	0-0-30	-	0-0-30	-	0-0-30	-	-	-	-
	50	804.6	26	866.1	2	1082.5	2	971.9	2	786.9	9.4	92
D-2	75	980.4	409	1075.3	9	1319.1	9	1147	11	966	12.6	426
	100	1107.6	2027	1202.2	25	1477.1	27	1288.6	32	1104.1	14	1369
W-D-L		6-8-16	-	0-0-30	-	0-0-30	-	0-0-30	-	-	-	-
gap		-1.64%	-	-7.79%	-	-24.22%	-	-16.18%	-	-	-	-
	50	380.3	35	393.7	2	520.6	2	435.3	2	358.7	3	121
U-3	75	442.5	481	451.8	10	612.3	8	521.7	10	412.1	5.7	624
	100	512.1	2465	527.7	30	715.6	25	578.2	31	478	8.8	2065
W-D-L		0-0-30	-	0-0-30	-	0-0-30	-	0-0-30	-	-	-	-
	50	439.1	47	474.1	2	656.5	2	520.8	2	423.4	6.1	109
S-3	75	608.6	587	641.1	10	905.8	9	705	11	589.4	10.8	536
	100	736.4	2590	773.7	30	1068.1	28	884.2	33	715.6	15.5	1773
W-D-L		2-4-24	-	0-0-30	-	0-0-30	-	0-0-30	-	-	-	-
	50	742.4	37	760.7	2	1062	2	929.3	2	695.4	7	102
D-3	75	871.4	601	928.5	11	1273.7	10	1034.6	13	842.6	10.9	502
	100	996.9	2686	1050.1	34	1429.4	29	1186.9	33	967.5	15.4	1579
W-D-L		0-3-27	-	0-0-30	-	0-0-30	-	0-0-30	-	-	-	-
gap		-4.38%	-	-8.40%	-	-31.19%	-	-21.58%	-	-	-	-

100 nodes), three distributions (uniform, single-center and double-center), and three ratios between truck and drone speeds ($\alpha = c^d/c = 1, 2, 3$), leading to $3\times3\times3 = 27$ groups. For each group, 10 instances are randomly generated.

We compare MATSP-D with four state-of-the-art heuristic algorithms: TSP-EP [6], TSP-GP [6], TSP-MC [4] and HGVNS [13]. The source code of TSP-EP, TSP-GP and TSP-MC are publicly available. The source code of HGVNS is not available, and we re-implemented it. All the compared algorithms were run on the same computational platform. All the compared results are the best results published in the original papers.

Table IV summarizes the results of the compared algorithms on the 27 groups of large instances in [6], where the notations "U", "S" and "D" in the first column stand for the "Uniform", "Single-center" and "Double-center" distributions, and the following numbers (1, 2 or 3) indicate the α value. Each row shows the average cost and runtime of the compared deterministic algorithms over the 10 instances of that type. As MATSP-D was run 30 times independently for each instance, we first calculate the mean and standard deviation over the 30 runs for each instance, and then present the average of the mean cost and standard deviation over the 10 instances in the table. We further conducted Wilcoxon rank-sum test between MATSP-D and each compared algorithm on each instance with the significance level of 0.05 and Bonferroni correction. For each type of instances, the "W-D-L" row summarizes the number of instances where the compared algorithm performs statistically significantly better than, comparable with, and significantly worse than MATSP-D, respectively. The gap row shows the average gap between MATSP-D and each compared algorithm, calculated as $\frac{\mathtt{cost}_{\text{MATSP-D}} - \mathtt{cost}_{\text{other}}}{\mathtt{cost}_{\text{other}}}$.

From the table, we can clearly see that MATSP-D performs better than the compared algorithms in most cases. The average gap is always negative, indicating that the cost obtained by MATSP-D is lower than that of the other compared algorithms. TSP-EP performs the second best, and much better than the other three compared algorithms. This is consistent with [6].

The advantage of MATSP-D becomes more obvious as α increases. The gap between MATSP-D and TSP-EP starts with -0.11% when $\alpha = 1$, reaching -1.64% when $\alpha = 2$ and -4.38% when $\alpha = 3$. This indicates that MATSP-D

TABLE V
THE AVERAGE PERFORMANCE AND RUNTIME OF THE COMPARED ALGORITHMS ON THE **SECOND** LARGE DATASET [13] WITH $\boldsymbol{\alpha = 1}$.

	TSP-EP [6]		TSP-GP [6]		TSP-MC [4]		HGVNS [13]		HGA [24]		MATSP-D		
	Cost	Time(s)	Cost	Time(s)	Cost	Time(s)	Avg.Cost	Time(s)	Avg.Cost	Time(s)	Avg.Cost	Std	Time(s)
berlin52	173.1(+)	12	186.9(-)	2	204.4(-)	4	220.2(-)	7	199.8(-)	14	174.7	0.9	78
eil51	9.9(=)	13	10.8(-)	1	11.7(-)	3	13.7(-)	12	13.5(-)	11	9.9	0.1	70
eil76	12.1(=)	110	13.2(-)	6	15(-)	10	16.7(-)	27	16.9(-)	27	12.1	0.1	303
kroA100	486.2(+)	783	523.5(-)	20	589.6(-)	34	609.7(-)	31	541.4(-)	98	493.7	6	1007
kroC100	498.3(+)	549	539.1(-)	17	611.7(-)	27	660.9(-)	37	547.4(-)	79	505.2	4.3	1037
kroD100	501.3(=)	817	539(-)	21	597.9(-)	35	652.3(-)	40	547.2(-)	65	502.3	2.9	1027
kroE100	521.6(+)	815	563.9(-)	20	653.9(-)	26	659.5(-)	49	581.9(-)	69	526.9	7	1075
rat99	29.1(+)	386	37.4(-)	5	36.2(-)	22	37.3(-)	35	37.5(-)	55	29.5	0.3	1004
rd100	190.5(-)	672	204.3(-)	28	224.1(-)	52	243.8(-)	34	219.4(-)	85	189.1	2.2	1020
st70	15.6(=)	101	16.8(-)	6	19.1(-)	6	21(-)	4	21(-)	22	15.6	0.2	217
bier127	2619(+)	2996	2892.2(-)	54	2893.5(-)	84	3587.9(-)	54	3506.4(-)	64	2679.8	40.4	2854
ch130	143.3(+)	3456	154.3(-)	62	182.3(-)	78	180.4(-)	44	182.9(-)	76	145.4	2.2	3466
d198	409.2(+)	56808	426.1(-)	438	448.3(-)	1123	461.8(-)	68	461.2(-)	114	411.3	3	19296
kroA150	626.4(+)	9445	670.6(-)	91	767.9(-)	187	780.9(-)	41	693.6(-)	145	636.2	7.3	6570
kroA200	702.4(+)	44877	751.8(-)	311	866.4(-)	657	874(-)	47	820.9(-)	170	718.9	7.7	22776
kroB150	603.4(+)	7899	671.8(-)	113	745.4(-)	256	773.7(-)	50	676.1(-)	146	615.2	10.2	6454
kroB200	673.5(+)	51596	922.9(-)	68	861.7(-)	714	838.4(-)	32	801.4(-)	152	698.4	13.4	22792
lin105	331.2(+)	1002	351.2(-)	25	390.6(-)	35	380.4(-)	40	381.7(-)	91	334.4	2.6	1351
pr107	979.9(+)	765	1044.2(-)	26	1072.5(-)	127	1224.4(-)	33	1038.1(-)	79	993.3	6.3	1589
pr124	1428.7(-)	1083	1490.8(-)	30	1594.7(-)	99	1996.6(-)	25	1618.1(-)	47	1418.9	12.3	2671
pr136	2116.8(+)	3227	2839.5(-)	32	2568.2(-)	110	2789(-)	45	2474.3(-)	142	2135.2	35.8	3883
pr144	1544(=)	7100	1600.7(-)	96	1675.9(-)	267	1675.8(-)	43	1675.8(-)	176	1544.6	14.7	5715
pr152	1840.1(=)	9745	1897.7(-)	163	2038.1(-)	235	2128.5(-)	61	1973.7(-)	119	1844.9	11.8	6778
rat195	56.3(+)	23351	72.1(-)	62	68.3(-)	469	71.9(-)	45	71.5(-)	169	57.4	0.5	18760
W-D-L	16-6-2	-	0-0-24	-	0-0-24	-	0-0-24	-	0-0-24	-	-	-	-
gap	1.00%	-	-8.70%	-	-15.02%	-	-20.19%	-	-14.03%	-	-	-	-

can perform better for instances with faster drones. When the drone speed is higher, the optimal solutions require to use the drone more often, and the interactions between truck and drone route are more complex. In this case, MATSP-D performs better due to its stronger ability to handle the complex interactions between the truck and drone routes. The superior performance of MATSP-D for large α demonstrates its effectiveness for solving TSP-D with more complex collaborations between the truck and drone.

The "W-D-L" rows show that MATSP-D significantly outperforms TSP-GP, TSP-MC and HGVNS for all 270 instances. Compared with TSP-EP, the performance is comparable when $\alpha = 1$ (32 wins and 32 loss). When $\alpha = 2$, MATSP-D significantly outperforms TSP-EP on 60 instances, while loses on only 16 instances. When $\alpha = 3$, MATSP-D shows significantly better performance than TSP-EP on 81 out of the 90 instances.

The runtime of MATSP-D increases with a similar trend as the increase of the TSP-EP runtime. MATSP-D starts with longer runtime than TSP-EP when $N = 50$, but with slower increase as the problem size increases. When $N = 100$, MATSP-D has even shorter runtime than TSP-EP. Note that TSP-GP, TSP-MC and HGVNS have much shorter runtime, but their performance are much worse than TSP-EP. As existing studies [6] already show that TSP-EP is state-of-the-art, we mainly focus on comparing with TSP-EP.

3) Second large dataset: The second dataset [13] contains 24 instances extended from the TSPLib instances with $\alpha = 1$. In addition, we create another 24 instances with a higher drone speed by setting $\alpha = 3$, which requires more collaborations between drone and truck.

Tables V and VI show the performance of each algorithm on the instances with $\alpha = 1$ and $\alpha = 3$. Note that HGA has only "Avg.Cost" but no "Std", since only the average cost was reported in [24]. For each instance, we conducted Wilcoxon rank sum test between each compared algorithm and MATSP-D. The compared result is marked with $(+)/(-)$, if it performs significantly better/worse than MATSP-D, and $(=)$ if there is no statistical difference.

The results show similar patterns as on the first dataset. When $\alpha = 1$, TSP-EP performs the best, followed by MATSP-D with slightly worse performance (only 1.00% gap). TSP-EP is significantly better than MATSP-D on 16 out of the 24 instances. However, MATSP-D performs significantly better than all the other compared algorithms on all the instances. When $\alpha = 3$, MATSP-D performs much better than TSP-EP on all the 24 instances, and their gap is -7.24%. We omit other algorithms in Table VI, as they are even much worse than TSP-EP. In addition, MATSP-D is superior to TSP-EP in terms of runtime when $N > 100$.

Since HGA is the most similar algorithm with MATSP-D, comparison between MATSP-D and HGA might be of particular interest. The tables show that MATSP-D performs much better than HGA, although its runtime is much longer. Further analysis on convergence curve in the supplementary file shows that given the same runtime as in [24], MATSP-D can still converge to much better solutions than HGA.

TABLE VI
THE AVERAGE PERFORMANCE AND RUNTIME OF THE COMPARED ALGORITHMS ON THE **SECOND** LARGE DATASET [13] WITH $\boldsymbol{\alpha = 3}$.

	TSP-EP [6]		MATSP-D		
	Cost	Time(s)	Avg.Cost	Std	Time(s)
berlin52	130.4(-)	40	118.5	1.8	103
eil51	7.6(-)	41	6.9	0.1	83
eil76	9.2(-)	303	8.2	0.2	432
kroA100	393.7(-)	1652	373.5	4.6	2206
kroC100	410.5(-)	2262	375.8	6.1	2036
kroD100	390.8(-)	1874	379	4.5	2196
kroE100	435.6(-)	2112	385.6	12.3	1915
rat99	24(-)	1768	21.1	0.6	1534
rd100	153.3(-)	1180	131.3	2	1933
st70	12.6(-)	198	11.4	0.1	359
bier127	1930.7(-)	6284	1876.2	53	3582
ch130	116.8(-)	10145	107.3	2.7	5782
d198	370.3(-)	73467	362.8	5.2	26778
kroA150	496.4(-)	24723	475.2	8.6	11290
kroA200	569.2(-)	109208	535	11	38516
kroB150	508.4(-)	15566	460.3	13.9	11454
kroB200	570(-)	90712	523.3	15.8	37729
lin105	282.6(-)	2046	275	4.5	2001
pr107	882.5(-)	223	867	8.6	3049
pr124	1223.6(-)	2778	1160.6	22.1	3298
pr136	1730.7(-)	7168	1566.4	36.9	6224
pr144	1472.5(-)	7293	1388.6	53.8	7594
pr152	1677.2(-)	17071	1592.2	19	10658
rat195	44.8(-)	88068	40.7	0.7	27152
W-D-L	0-0-24	-	-	-	-
gap	-7.24%	-	-	-	-

TABLE VII
RESULTS OVER 30 RUNS OF MATSP-D AND MA-HGVNS ON THE UNIFORM INSTANCES WITH $\alpha = 2, N = 50$.

	MATSP-D			MA-HGVNS		
U-n50	Avg.Cost	Std	Time(s)	Avg.Cost	Std	Time(s)
71	388.3	2.6	137	405.4(-)	4.9	658
72	422.9	2.4	90	442(-)	8.4	660
73	398	3.2	91	408.2(-)	5.3	585
74	411.9	1.6	95	423.4(-)	6.6	640
75	415.9	3.5	88	430.7(-)	6.5	579
76	374.4	2.1	89	386.8(-)	5.8	570
77	416.7	4.3	94	433.6(-)	7	581
78	422.2	1.6	90	438.8(-)	4.6	579
79	384.4	3.6	85	395.6(-)	5.2	656
80	368.3	2.9	102	378.3(-)	6.8	616
W-D-L	-	-	-	0-0-10	-	-

B. Further Analysis

1) Effect of New Local Search Operators: An important contribution of MATSP-D is the newly proposed local search operators. To verify the efficacy of the local search operators, we replace the VNS process in MATSP-D with the HGVNS process [13] (namely MA-HGVNS), and compare between them on 10 instances with uniform distribution and $\alpha = 2$. Table VII shows the comparative results. From the table, we can see that MATSP-D can obtain significantly better solutions than MA-HGVNS on most instances with a much shorter time. Similar patterns can be observed on other instances. This verifies the effectiveness of the newly proposed local search operators.

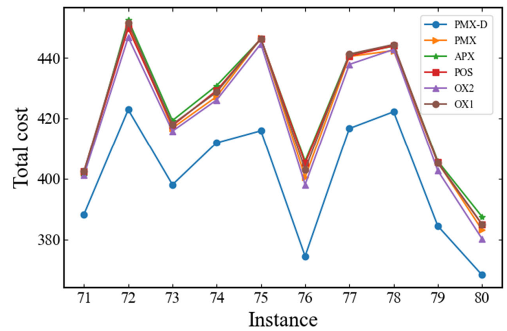
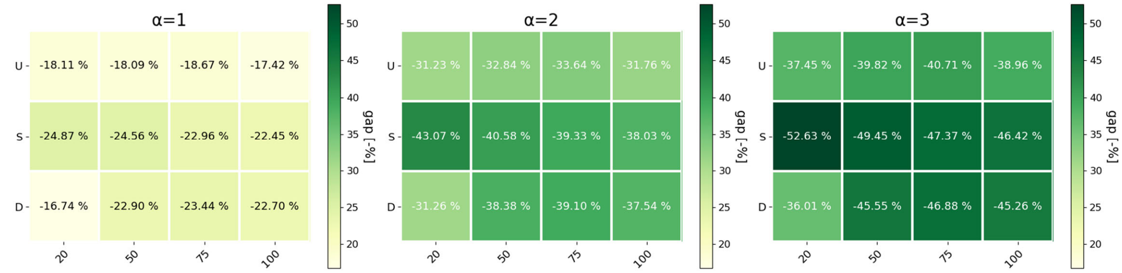


Fig. 6. Results over 30 runs of PMX-D and other common crossover operators on the uniform instances with $\alpha = 2, N = 50$.

2) Effect of the PMX-D Crossover Operator: The PMX-D operator extends the PMX operator to adapt to the specific representation of MATSP-D. It first applies PMX to the node sequence, and then develops a new repair process for the state vector. To verify the effectiveness of the PMX-D operator, we replace it with six other common crossover operators [65], i.e., PMX, APX, POS, OX2, OX1, and CX. To prevent infeasible offspring, we set the state of all the nodes to T. We compare the MATSP-D counterparts with different crossover operators on 10 instances with uniform distribution and $\alpha = 2$, and the results are shown in Fig. 6. We can see that the results of PMX-D are significantly better than that of the other crossover operators on all the instances. Similar trends can be observed on other instances. This verifies the effectiveness of the proposed PMX-D operator. Furthermore, the advantage over the original PMX operator verifies the efficacy of the newly proposed state vector repair process in PMX-D.

3) Practical Implication on Customer Distribution: We further analyze the practical implication of TSP-D, i.e., when it is good to use drones together with trucks. Here, we investigate the effect of customer distribution, by comparing MATSP-D with Gurobi that considers no drone (i.e., solving TSP) on the first large dataset.

Fig. 7 shows the heatmaps of the *gap* between with drone (MATSP-D) and without drone (TSP by Gurobi), which is calculated as $\frac{\mathtt{cost}_{\text{MATSP-D}} - \mathtt{cost}_{\text{TSP}}}{\mathtt{cost}_{\text{TSP}}}$. The horizontal and vertical axes indicate the number of customers and their distributions, respectively. We can see that all the gaps are negative, i.e., the drone-truck collaborative system always outperforms the truck-only systems. Besides, the gap increases when the drone speed (α) increases, which is consistent with intuition that a faster drone speed requires more collaboration between the truck and drone. Furthermore, it is shown that the gap is the largest on the single-center instances, followed by the double-center instances. As shown in [66], the urban population distribution tends to be single-center or double-center. The most advantage of MATSP-D over TSP on the single-center and double-center instances suggests that the collaborative drone-truck delivery system will play a vital role in urban logistics transportation. To fully utilize the advantage of the truck-drone delivery



(a) $\alpha = 1$ (b) $\alpha = 2$ (c) $\alpha = 3$

Fig. 7. Heatmaps of gap between TSP-D and TSP routes (a) $\alpha = 1$ (b) $\alpha = 2$ (c) $\alpha = 3$.

systems, we recommend deploying them separately for each region/sub-center of the city.

V. CONCLUSIONS

This paper proposes a novel memetic algorithm to solve the challenging TSP-D more effectively, especially for the cases when the drone speed is faster than the truck. To represent the coupling relationship between the truck route and drone route accurately, we develop a new two-level solution representation for the node sequence and state vector. Unlike traditional TSP-D algorithms, the proposed algorithm, namely MATSP-D, optimizes the truck route and drone route simultaneously by the extended PMX-D crossover operator and six new local search operators.

The results on both the synthetic and realistic datasets show that for small instances, MATSP-D can almost always obtain the optimal solution. For large instances, MATSP-D can significantly outperform the state-of-the-art algorithms when $\alpha \geq 2$ (the drone is at least as twice fast as the truck) in most cases. Besides, we demonstrate the effectiveness of the newly developed crossover and local search operators. We further investigate the applicable scenarios for the collaborative drone-truck delivery system. According to a case study, the collaborative drone-truck delivery system is more suitable to be deployed in urban scenarios.

A limitation of MATSP-D is its high computational cost. To address this issue, there can be several possible future directions. First, we will improve the efficiency of the local search by designing more intelligent schemes to select the solutions with more potential. Second, we will improve the effectiveness of the method for large-scale instances, e.g., through the divide-and-conquer strategies and cooperative co-evolution. Last but not least, we will extend our method to more complex and practical problem models such as multiple delivery trucks and drones.

REFERENCES

[1] Protecting Ghana's Election: Instant Agility With Zipline's Autonomous Delivery Network. (2021). [Online]. Available: https://www.flyzipline.com/

[2] I. A. Hamilton. A tiny Irish drone startup is delivering emergency cake, pizza, and medical supplies to Moneygall, the ancestral home of Barack Obama. (2020). [Online]. Available: https://www.businessinsider.com/drone-delivery-startup-manna-adapted-to-coronavirus-pandemic-2020-5?r=US&IR=T

[3] S. Jingli. SF Express completes first drone delivery to hospital in coronavirus-hit Wuhan. (2020). [Online]. Available: https://kr-asia.com/sf-express-completes-first-drone-delivery-to-hospital-in-coronavirus-hit-wuhan

[4] C. C. Murray and A. G. Chu, "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery," *Transp. Res. Pt. C-Emerg. Technol.*, vol. 54, pp. 86–109, 2015.

[5] M. Wohlsen. The Next Big Thing You Missed: Amazon's Delivery Drones Could Work—They Just Need Trucks. (2014). [Online]. Available: https://www.wired.com/2014/06/the-next-big-thing-you-missed-delivery-drones-launched-from-trucks-are-the-future-of-shipping/

[6] N. Agatz, P. Bouman, and M. Schmidt, "Optimization approaches for the traveling salesman problem with drone," *Transp. Sci.*, vol. 52, no. 4, pp. 965–981, 2018.

[7] M. S. bin Othman, A. Shurbevski, Y. Karuno, and H. Nagamochi, "Routing of carrier-vehicle systems with dedicated last-stretch delivery vehicle and fixed carrier route," *J. Inf. Process.*, vol. 25, pp. 655–666, 2017.

[8] E. E. Yurek and H. C. Ozmutlu, "A decomposition-based iterative optimization algorithm for traveling salesman problem with drone," *Transp. Res. Pt. C-Emerg. Technol.*, vol. 91, pp. 249–262, 2018.

[9] P. Bouman, N. Agatz, and M. Schmidt, "Dynamic programming approaches for the traveling salesman problem with drone," *Networks*, vol. 72, no. 4, pp. 528–542, 2018.

[10] R. Roberti and M. Ruthmair, "Exact methods for the traveling salesman problem with drone," *Transp. Sci.*, vol. 55, no. 2, pp. 315–335, 2021.

[11] M. E. Bruni, S. Khodaparasti, and M. Moshref-Javadi, "A logic-based benders decomposition method for the multi-trip traveling repairman problem with drones," *Comput. Oper. Res.*, vol. 145, SEP 2022.

[12] A. Ponza, "Optimization of drone-assisted parcel delivery," Master's thesis, Dept. Eng., Universita Degli Studi DI Padova, Padua, Italy, 2016.

[13] J. C. de Freitas and P. H. V. Penna, "A variable neighborhood search for flying sidekick traveling salesman problem," *Int. Trans. Oper. Res.*, vol. 27, no. 1, pp. 267–290, 2020.

[14] S. T. W. Mara, A. P. Rifai, and B. M. Sopha, "An adaptive large neighborhood search heuristic for the flying sidekick traveling salesman problem with multiple drops," *Expert Syst. Appl.*, vol. 205, NOV 1 2022.

[15] A. Heimfarth, M. Ostermeier, and A. Huebner, "A mixed truck and robot delivery approach for the daily supply of customers," *Eur. J. Oper. R.*, vol. 303, no. 1, pp. 401–421, NOV 16 2022.

[16] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts : Towards memetic algorithms," Caltech Concurrent Comput. Program, California Inst. Technol., Pasadena, CA, USA, Tech. Rep. 826, 1989.

[17] Y. Chen and J.-K. Hao, "Memetic search for the generalized quadratic multiple knapsack problem," *IEEE Trans. Evol. Comput.*, vol. 20, no. 6, pp. 908–923, 2016.

[18] H. Handa, D. Lin, L. Chapman, and X. Yao, "Robust solution of salting route optimisation using evolutionary algorithms," in *2006 IEEE Congr. Evol. Comput.*, 2006, pp. 3098–3105.

[19] J. Wang, W. Ren, Z. Zhang, H. Huang, and Y. Zhou, "A hybrid multiobjective memetic algorithm for multiobjective periodic vehicle routing problem with time windows," *IEEE Trans. Syst. Man Cybern. -Syst.*, vol. 50, no. 11, pp. 4732–4745, 2020.

[20] K. Tang, Y. Mei, and X. Yao, "Memetic algorithm with extended neighborhood search for capacitated arc routing problems," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1151–1166, 2009.

[21] S.-Y. Wang and L. Wang, "An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem," *IEEE Trans. Syst. Man Cybern. -Syst.*, vol. 46, no. 1, pp. 139–149, 2016.

[22] Z. Li, L. Tang, and J. Liu, "A memetic algorithm based on probability learning for solving the multidimensional knapsack problem," *IEEE Trans. Cybern.*, vol. 52, no. 4, pp. 2284–2299, 2022.

[23] Q. M. Ha, Y. Deville, Q. D. Pham, and M. H. Hà, "On the min-cost traveling salesman problem with drone," *Transp. Res. Pt. C-Emerg. Technol.*, vol. 86, pp. 597–621, 2018.

[24] Q. M. Ha, Y. Deville, Q. D. Pham, and M. H. Hà, "A hybrid genetic algorithm for the traveling salesman problem with drone," *J. Heuristics*, vol. 26, no. 2, pp. 219–247, 2020.

[25] M. D. Phung, C. H. Quach, T. H. Dinh, and Q. Ha, "Enhanced discrete particle swarm optimization path planning for uav vision-based surface inspection," *Autom. Constr.*, vol. 81, pp. 25–33, 2017.

[26] D. Ma, Y. Li, X. Hu, H. Zhang, and X. Xie, "An optimal three-dimensional drone layout method for maximum signal coverage and minimum interference in complex pipeline networks," *IEEE Trans. Cybern.*, pp. 1–11, 2021.

[27] F. Mufalli, R. Batta, and R. Nagi, "Simultaneous sensor selection and routing of unmanned aerial vehicles for complex mission plans," *Comput. Oper. Res.*, vol. 39, no. 11, pp. 2787–2799, 2012.

[28] B. D. Song, K. Park, and J. Kim, "Persistent uav delivery logistics: Milp formulation and efficient heuristic," *Comput. Ind. Eng.*, vol. 120, pp. 418–428, 2018.

[29] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Trans. Syst. Man Cybern. -Syst.*, vol. 47, no. 1, pp. 70–85, 2017.

[30] S. H. Chung, B. Sah, and J. Lee, "Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions," *Comput. Oper. Res.*, vol. 123, p. 105004, 2020.

[31] S. Poikonen and B. Golden, "The mothership and drone routing problem," *INFORMS J. Comput.*, vol. 32, no. 2, pp. 249–262, 2020.

[32] H. Y. Jeong, B. D. Song, and S. Lee, "Truck-drone hybrid delivery routing: Payload-energy dependency and no-fly zones," *Int. J. Prod. Econ.*, vol. 214, pp. 220–233, 2019.

[33] S. M. Ferrandez, T. Harbison, T. Weber, R. Sturges, and R. Rich, "Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm," *J. Ind. Eng. Manag.*, vol. 9, no. 2, pp. 374–388, 2016.

[34] Z. Luo, Z. Liu, and J. Shi, "A two-echelon cooperated routing problem for a ground vehicle and its carried unmanned aerial vehicle," *Sensors*, vol. 17, no. 5, p. 1144, 2017.

[35] P. L. Gonzalez-R, D. Canca, J. L. Andrade-Pineda, M. Calle, and J. M. Leon-Blanco, "Truck-drone team logistics: A heuristic approach to multi-drop route planning," *Transp. Res. Pt. C-Emerg. Technol.*, vol. 114, pp. 657–680, 2020.

[36] Y. Liu, Z. Liu, J. Shi, G. Wu, and W. Pedrycz, "Two-echelon routing problem for parcel delivery by cooperated truck and drone," *IEEE Trans. Syst. Man Cybern. -Syst.*, vol. 51, no. 12, pp. 7450–7465, 2021.

[37] G. Zhang, N. Zhu, S. Ma, and J. Xia, "Humanitarian relief network assessment using collaborative truck-and-drone system," *Transp. Res. E-log.*, vol. 152, p. 102417, 2021.

[38] M. Marinelli, L. Caggiani, M. Ottomanelli, and M. Dell'Orco, "En route truck–drone parcel delivery for optimal vehicle routing strategies," *IET Intell. Transp. Syst.*, vol. 12, no. 4, pp. 253–261, 2018.

[39] N. Boysen, D. Briskorn, S. Fedtke, and S. Schwerdfeger, "Drone delivery from trucks: Drone scheduling for given truck routes," *Networks*, vol. 72, no. 4, pp. 506–527, 2018.

[40] I. Dayarian, M. Savelsbergh, and J.-P. Clarke, "Same-day delivery with drone resupply," *Transp. Sci.*, vol. 54, no. 1, pp. 229–249, 2020.

[41] J. C. Pina-Pardo, D. F. Silva, and A. E. Smith, "The traveling salesman problem with release dates and drone resupply," *Comput. Oper. Res.*, vol. 129, p. 105170, 2021.

[42] S. Kim and I. Moon, "Traveling salesman problem with a drone station," *IEEE Trans. Syst. Man Cybern. -Syst.*, vol. 49, no. 1, pp. 42–52, 2019.

[43] R. G. Ribeiro, L. P. Cota, T. A. M. Euzébio, J. A. Ramírez, and F. G. Guimarães, "Unmanned-aerial-vehicle routing problem with mobile charging stations for assisting search and rescue missions in postdisaster scenarios," *IEEE Trans. Syst. Man Cybern. -Syst.*, pp. 1–15, 2021.

[44] Q. Luo, G. Wu, B. Ji, L. Wang, and P. N. Suganthan, "Hybrid multi-objective optimization approach with pareto local search for collaborative truck-drone routing problems considering flexible time windows," *IET Intell. Transp. Syst.*, 2021.

[45] S. Zhang, S. Liu, W. Xu, and W. Wang, "A novel multi-objective optimization model for the vehicle routing problem with drone delivery and dynamic flight endurance," *Comput. Ind. Eng.*, p. 108679, 2022.

[46] Z. Ghelichi, M. Gentili, and P. B. Mirchandani, "Logistics for a fleet of drones for medical item delivery: A case study for louisville, ky," *Comput. Oper. Res.*, vol. 135, p. 105443, 2021.

[47] M. Boccia, A. Masone, A. Sforza, and C. Sterle, "A column-and-row generation approach for the flying sidekick travelling salesman problem," *Transp. Res. Pt. C-Emerg. Technol.*, vol. 124, p. 102913, 2021.

[48] J. F. Campbell, D. Sweeney, and J. Zhang, "Strategic design for delivery with trucks and drones," Supply Chain Analytics Report, Univ. Missouri–St. Louis, St. Louis, MO, USA, Tech. Rep. SCMA-2017-0201, 2017.

[49] J. G. Carlsson and S. Song, "Coordinated logistics with a truck and a drone," *Manage. Sci.*, vol. 64, no. 9, pp. 4052–4069, 2018.

[50] Y.-H. Jia, W.-N. Chen, T. Gu, H. Zhang, H. Yuan, Y. Lin, W.-J. Yu, and J. Zhang, "A dynamic logistic dispatching system with set-based particle swarm optimization," *IEEE Trans. Syst. Man Cybern. -Syst.*, vol. 48, no. 9, pp. 1607–1621, 2018.

[51] Y. Chen, M. Chen, Z. Chen, L. Cheng, Y. Yang, and H. Li, "Delivery path planning of heterogeneous robot system under road network constraints," *Comput. Electr. Eng.*, vol. 92, p. 107197, 2021.

[52] C. Chen, E. Demir, Y. Huang, and R. Qiu, "The adoption of self-driving delivery robots in last mile logistics," *Transp. Res. E-log.*, vol. 146, p. 102214, 2021.

[53] G. Wu, N. Mao, Q. Luo, B. Xu, J. Shi, and P. N. Suganthan, "Collaborative truck-drone routing for contactless parcel delivery during the epidemic," *IET Intell. Transp. Syst.*, pp. 1–15, 2022.

[54] G. Reinelt, "Tsplib—a traveling salesman problem library," *ORSA J. Computing*, vol. 3, no. 4, pp. 376–384, 1991.

[55] J. C. de Freitas and P. H. V. Penna, "A randomized variable neighborhood descent heuristic to solve the flying sidekick traveling salesman problem," *Electron. Notes Discret. Math.*, vol. 66, pp. 95–102, 2018.

[56] J. C. Freitas, P. H. V. Penna, and T. A. Toffolo, "Exact and heuristic approaches to truck-drone delivery problems," *EURO Journal on Transportation and Logistics*, p. 100094, 2022.

[57] C. Chen, E. Demir, and Y. Huang, "An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots," *Eur. J. Oper. Res.*, vol. 294, no. 3, pp. 1164–1180, 2021.

[58] G. Wu, N. Mao, Q. Luo, B. Xu, J. Shi, and P. N. Suganthan, "Collaborative truck-drone routing for contactless parcel delivery during the epidemic," *IET Intell. Transp. Syst.*, 2022.

[59] D. Lei, Z. Cui, and M. Li, "A dynamical artificial bee colony for vehicle routing problem with drones," *Eng. Appl. Artif. Intel.*, vol. 107, p. 104510, 2022.

[60] B. L. Miller, D. E. Goldberg *et al.*, "Genetic algorithms, tournament selection, and the effects of noise," *Complex Syst.*, vol. 9, no. 3, pp. 193–212, 1995.

[61] K. Wu, J. Liu, and S. Wang, "Reconstructing networks from profit sequences in evolutionary games via a multiobjective optimization approach with lasso initialization," *Sci. Rep.*, vol. 6, no. 1, pp. 1–11, 2016.

[62] D. E. Goldberg, R. Lingle *et al.*, "Alleles, loci, and the traveling salesman problem," in *Proc. 1st Int. Conf. Genet. Algorithms*, vol. 154, 1985, pp. 154–159.

[63] N. Mladenović and P. Hansen, "Variable neighborhood search," *Comput. Oper. Res.*, vol. 24, no. 11, pp. 1097–1100, 1997.

[64] J. Berger, M. Barkaoui, and O. Bräysy, "A route-directed hybrid genetic approach for the vehicle routing problem with time windows," *Proc.INFOR*, vol. 41, no. 2, pp. 179–194, 2003.

[65] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: A review of representations and operators," *Artificial intelligence review*, vol. 13, no. 2, pp. 129–170, 1999.

[66] D. P. McMillen and S. C. Smith, "The number of subcenters in large urban areas," *J. Urban Econ.*, vol. 53, no. 3, pp. 321–338, 2003.

Supplementary Material of "A Collaborative Drone-Truck Delivery System with Memetic Computing Optimization"

Ruonan Zhai, *Student Member, IEEE,* Yi Mei, *Senior Member, IEEE,* Tong Guo, *Student Member, IEEE,* and Wenbo Du, *Member, IEEE*

S.I. Detailed Results on the Small Dataset

Table SI shows the summarized results of CPLEX and MATSP-D on the small dataset [1]. Each group contains 10 instances with N uniformly distributed customers. We conducted the Wilcoxon rank sum test to compare the results of MATSP-D with the optimum. The optimal result is marked with $(+)$, if MATSP-D performs significantly worse than the optimum, and $(=)$ if there is no statistical difference. For each group, the summary results are listed in the "W-D" row. "W" indicates the number of $(+)$, and "D" the number of $(=)$.

From the table, we can see that MATSP-D consistently reached the optimum on 22 out of the 30 instances. The average gap from the optimum over all the instances is as small as 0.1%.

An interesting observation is that the standard deviation of MATSP-D is always zero, even when the solution is not optimal. In other words, MATSP-D converges to the same sub-optimal solution in all the runs. We further looked into the solutions of the MATSP-D for these instances and compared with the optimal solutions, and found that the optimal solutions require the drone to depart and return to the same truck node. The representation of MATSP-D (each operation must have different start and end nodes) does not allow such a situation. However, MATSP-D still obtains the optimal solutions with the assumption that the drone must depart and return to different truck nodes.

TABLE SI
The mean (std) of MATSP-D and runtime over 30 independent runs on the small instances.

Instance	CPLEX	Time(s)	MATSP-D	Std	Time(s)
U-n10-51	250.7(=)	45	250.7	0.0	4
U-n10-52	189.5(+)	48	191.3	0.0	5
U-n10-53	192.2(=)	43	192.2	0.0	4
U-n10-54	224.9(=)	41	224.9	0.0	5
U-n10-55	253.1(=)	39	253.1	0.0	5
U-n10-56	226.9(=)	37	226.9	0.0	5
U-n10-57	197.2(+)	59	197.5	0.0	4
U-n10-58	213.3(=)	38	213.3	0.0	7
U-n10-59	204(=)	52	204	0.0	5
U-n10-60	225.9(+)	44	226	0.0	4
W-D	3-7	-	-	-	-
Avg.	217.8	45	218.0	0.0	5
Instance	**CPLEX**	**Time(s)**	**MATSP-D**	**Std**	**Time(s)**
U-1-n11	221.2(+)	433	223.4	0.0	10
U-2-n11	205.8(=)	542	205.8	0.0	5
U-3-n11	193(=)	655	193	0.0	5
U-4-n11	241.3(=)	395	241.3	0.0	5
U-5-n11	248.1(=)	386	248.1	0.0	5
U-6-n11	217.7(=)	523	217.7	0.0	6
U-7-n11	237.3(=)	727	237.3	0.0	5
U-8-n11	214.8(=)	418	214.8	0.0	5
U-9-n11	256.3(+)	682	256.8	0.0	5
U-10-n11	227.9(=)	635	227.9	0.0	5
W-D	2-8	-	-	-	-
Avg.	226.3	540	226.6	0.0	5
Instance	**CPLEX**	**Time(s)**	**MATSP-D**	**Std**	**Time(s)**
U-1-n12	239.7(=)	5264	239.7	0.0	6
U-2-n12	221.3(=)	5824	221.3	0.0	8
U-3-n12	247.1(+)	6255	247.5	0.0	7
U-4-n12	230(=)	8469	230	0.0	7
U-5-n12	243.3(=)	7645	243.3	0.0	7
U-6-n12	222(=)	5249	222	0.0	6
U-7-n12	225.8(=)	5702	225.8	0.0	7
U-8-n12	227.9(+)	6759	229.3	0.0	8
U-9-n12	243.9(=)	5313	243.9	0.0	6
U-10-n12	196.1(+)	6698	198.8	0.0	7
W-D	3-7	-	-	-	-
Avg.	229.7	6318	230.2	0.0	7

S.II. Detailed Results on The First Large DataSet

Table SII-SX shows the detailed results of the four compared algorithms on the nine groups of the first large instances. Each group is distinct in three distributions (uniform, single-center and double-center), and three ratios between truck and drone speeds ($\alpha = c^d/c = 1, 2, 3$). Each row shows the cost and runtime of the compared deterministic algorithms. As MATSP-D was run 30 times independently for each instance, we first calculate the mean and standard deviation over the 30 runs for each instance, and then present the average of the mean cost and standard deviation over the 10 instances in the table. We conducted Wilcoxon rank sum test between MATSP-D and the compared algorithms. The compared result is marked with $(+)$, if it performs significantly better than MATSP-D, $(=)$ if there is no statistical difference, and $(-)$ if it performs significantly worse than MATSP-D. For each scale, the summary results are listed in the "W-D-L" row. "W" indicates the number of $(+)$, "D" the number of $(=)$, and "L" the number of $(-)$.

TABLE SII
THE AVERAGE PERFORMANCE AND RUNTIME OF THE COMPARED ALGORITHMS ON THE UNIFORM DATASET WITH $\alpha = 1$

Instance		TSP-EP [1]		TSP-GP [1]		TSP-MC [2]		HGVNS [3]		MATSP-D		
N	id	Cost	Time(s)	Cost	Time(s)	Cost	Time(s)	Avg.Cost	Time(s)	Avg.Cost	Std	Time(s)
50	71	479.8(+)	13	530.1(-)	1	529.9(-)	2	506.2(-)	2	481.2	1.8	86
	72	510.6(+)	18	600(-)	1	568.5(-)	1	541.7(-)	2	513.6	5.2	137
	73	488.3(+)	15	563.8(-)	1	568.6(-)	1	518.6(-)	1	491.3	2.7	148
	74	484.5(-)	10	560.2(-)	1	541(-)	1	526.5(-)	2	483.9	1.6	142
	75	523.4(-)	8	578.9(-)	1	571.7(-)	1	555.2(-)	1	516.1	5	146
	76	470.5(+)	14	534.6(-)	1	552.4(-)	1	505.3(-)	2	472.4	3.6	156
	77	513.7(=)	14	583.6(-)	1	602.1(-)	1	545.7(-)	1	513.1	4.4	138
	78	506.8(-)	21	554(-)	1	569.4(-)	1	550.5(-)	1	492.9	2.8	166
	79	466.7(-)	15	522.2(-)	1	533.7(-)	2	533.7(-)	2	464.5	3.5	131
	80	458(-)	7	501.1(-)	1	503.2(-)	2	501.4(-)	2	452.7	3.3	73
W-D-L		4-1-5	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
75	81	575.5(=)	176	644.1(-)	6	670.7(-)	6	621.6(-)	7	575.9	5.6	334
	82	527.8(+)	262	600.9(-)	3	612.9(-)	5	558.6(-)	6	530	2.6	340
	83	544.8(=)	201	609(-)	5	602(-)	7	588.3(-)	8	544.9	4.9	358
	84	566.9(+)	324	651.1(-)	4	646.6(-)	6	636.8(-)	7	574.4	5.5	353
	85	599.5(=)	234	689.5(-)	2	696.8(-)	5	671(-)	6	598.4	7.1	347
	86	579(=)	216	695.2(-)	1	645.3(-)	6	645.3(-)	6	579.8	4.3	342
	87	579.2(+)	198	674.4(-)	2	646.8(-)	7	631(-)	9	587	5.1	366
	88	601.3(-)	234	655.9(-)	5	678.9(-)	6	634.7(-)	8	580.9	5.7	360
	89	526.7(-)	185	591.6(-)	5	599.7(-)	7	560.4(-)	9	518.8	7.3	348
	90	565.9(-)	160	653.3(-)	4	660.6(-)	6	651.8(-)	7	563.2	4.4	369
W-D-L		3-4-3	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
100	91	676(+)	595	749.3(-)	11	769.7(-)	18	726.3(-)	23	678.7	5.9	1109
	92	615.8(=)	774	683.8(-)	12	700.9(-)	21	676.4(-)	23	615.8	6.5	1080
	93	610.2(+)	898	701.6(-)	11	707.7(-)	22	678.7(-)	25	613.2	5.1	1025
	94	634.2(+)	1239	734.3(-)	9	719.5(-)	24	673.8(-)	27	639.7	6.5	1056
	95	651.3(+)	394	728.3(-)	10	763.1(-)	23	700.5(-)	27	657.6	6.2	1091
	96	649.9(=)	863	725.3(-)	15	758(-)	23	730.1(-)	27	654.1	9.1	1087
	97	674.1(=)	1263	763.9(-)	8	766.1(-)	22	730(-)	25	673.9	6.8	1131
	98	620.7(+)	557	738(-)	12	720.7(-)	26	671.3(-)	29	628.3	7.2	1047
	99	652.3(=)	1217	731.1(-)	13	742.3(-)	27	723.2(-)	29	653.8	7.9	1113
	100	669.7(-)	653	746.9(-)	16	749.8(-)	26	749.8(-)	27	651.7	13	1041
W-D-L		5-4-1	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-

It is shown that MATSP-D significantly outperforms TSP-GP, TSP-MC and HGVNS for all 270 instances. Compared with TSP-EP, the performance is comparable when $\alpha = 1$ (32 wins and 32 losses). When $\alpha = 2$, MATSP-D significantly outperforms TSP-EP on 60 instances, while losing on only 16 instances. When $\alpha = 3$, MATSP-D shows significantly better performance than TSP-EP on 81 out of the 90 instances.

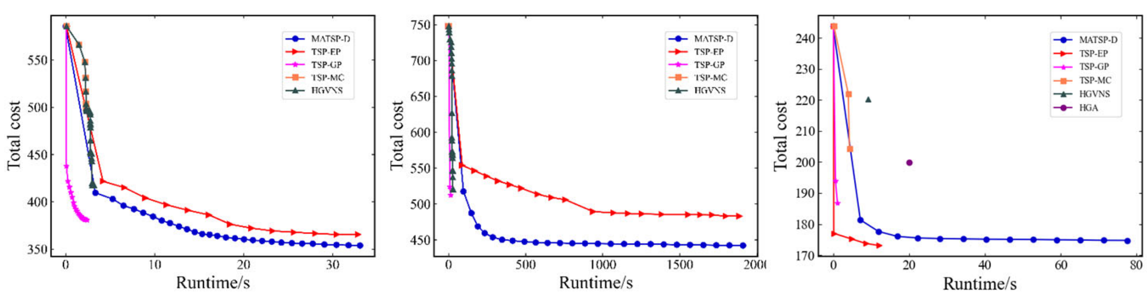
S.III. CONVERGENCE CURVES

Fig. S1 shows the convergence curves of the compared algorithms on three representative instances: (a) U_3-75-n50 (b) U_3-100-n100 and (c) berlin52 with $\alpha = 1$. From Fig. 1(a), we can see that when $N = 50$, MATSP-D converged slightly more slowly than TSP-GP, but similar to other compared algorithms. It can reach better solutions than all the other algorithms. Note that TSP-GP, TSP-MC and HGVNS had much shorter convergence curves than TSP-EP and MATSP-D, since they are designed as fast heuristics and have much shorter computational complexity than TSP-EP and MATSP-D. They are deterministic algorithms, and thus we cannot extend their runtime to further improve their performance. In fact, the original literature [1] already showed that TSP-EP can achieve better final performance than TSP-GP, although with higher computational complexity. From Fig. 1(b), we see similar patterns. MATSP-D had slightly slower convergence than TSP-GP, TSP-MC and HGVNS, but the same as TSP-EP at the beginning. After around 100 seconds, TSP-EP started to stagnate and its curve became flatter. However, MATSP-D can still improve substantially, and reached much better final solutions than TSP-EP.

In Fig. 1(c), the final performance and runtime of HGVNS and HGA were directly obtained from their literature, which are plotted as two points in the figure. Since the compared algorithms were implemented on different computers, normalization has been carried out to make fair comparisons on runtime. That is, all the runtimes presented in this experiment were obtained by dividing the runtimes in the original publications by some factors. To be specific, HGA was implemented on Intel Core i7-6700 (3.4GHz); therefore the runtimes presented in [4] were divided by 2.4/3.4. We can observe consistent patterns with that on the other two instances. Furthermore, we can directly obtain the results of HGA on this instance from [4]. Since HGA is the only population-based algorithm among the five compared algorithms, comparison between MATSP-D and HGA might be of particular interest. The figure shows that if given the

TABLE SIII
THE AVERAGE PERFORMANCE AND RUNTIME OF THE COMPARED ALGORITHMS ON THE UNIFORM DATASET WITH $\alpha = 2$

Instance		TSP-EP [1]		TSP-GP [1]		TSP-MC [2]		HGVNS [3]		MATSP-D		
N	id	Cost	Time(s)	Cost	Time(s)	Cost	Time(s)	Avg.Cost	Time(s)	Avg.Cost	Std	Time(s)
50	71	404.9(-)	25	422.1(-)	1	508.2(-)	2	455.8(-)	3	388.3	2.6	137
	72	429.1(-)	60	459.3(-)	1	558(-)	2	497.3(-)	2	422.9	2.4	90
	73	399.2(-)	20	418.4(-)	2	550.8(-)	2	446.6(-)	2	398	3.2	91
	74	417.7(-)	25	424.6(-)	1	500.5(-)	2	425.3(-)	2	411.9	1.6	95
	75	445.4(-)	25	464.7(-)	1	555.4(-)	2	480.8(-)	2	415.9	3.5	88
	76	370.7(+)	33	397.7(-)	2	509.2(-)	2	444.8(-)	3	374.4	2.1	89
	77	433.6(-)	22	446(-)	1	584.3(-)	1	465.4(-)	2	416.7	4.3	94
	78	436.9(-)	32	448.4(-)	1	562.9(-)	2	459(-)	2	422.2	1.6	90
	79	381.7(+)	75	409.6(-)	1	524(-)	2	524(-)	2	384.4	3.6	85
	80	379.5(-)	23	391.6(-)	1	471.7(-)	3	391.3(-)	3	368.3	2.9	102
W-D-L		2-0-8	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
75	81	481.7(-)	427	504(-)	8	649.8(-)	9	557(-)	10	471.5	6.7	493
	82	454.1(-)	239	475.8(-)	4	601.5(-)	8	502(-)	10	440.2	4.3	483
	83	455.1(-)	393	466.4(-)	4	570.2(-)	11	504.7(-)	13	446.5	6.7	448
	84	470.8(=)	533	500(-)	5	622.6(-)	9	622.6(-)	9	472.8	5.4	422
	85	508.9(-)	229	534(-)	6	678.1(-)	7	572.1(-)	9	486.2	6.5	555
	86	488.9(-)	415	512(-)	6	616.6(-)	9	604.4(-)	10	469.1	7	461
	87	473.2(=)	523	491.9(-)	6	628.1(-)	11	573.5(-)	13	471.7	5.4	451
	88	508.9(-)	340	529.4(-)	6	651.5(-)	8	567(-)	10	462.7	4	436
	89	446.9(-)	535	458.5(-)	5	576.1(-)	9	467.7(-)	11	428.6	7.2	431
	90	462.9(=)	335	482.5(-)	8	643.4(-)	8	523.7(-)	10	463.6	7.5	466
W-D-L		0-3-7	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
100	91	565.7(-)	3186	585.1(-)	18	761(-)	24	634.2(-)	30	559.7	7.5	1546
	92	517.7(-)	2914	533.2(-)	19	679.6(-)	28	594.6(-)	30	499.2	7.6	1394
	93	526.2(-)	2393	546.4(-)	15	683.3(-)	25	563.4(-)	32	501.7	8	1458
	94	541.8(-)	2295	558.2(-)	21	711.3(-)	26	609.9(-)	29	534.2	9.4	1435
	95	549.3(-)	1451	560.9(-)	20	758.3(-)	24	595.7(-)	27	538.7	6.4	1575
	96	555(-)	1729	577.2(-)	18	739.1(-)	26	636.9(-)	32	545	8.2	1469
	97	582.8(-)	2505	607.1(-)	21	742(-)	26	662(-)	34	563.2	8.9	1508
	98	520.1(=)	1085	529.4(-)	20	681.1(-)	29	583.7(-)	35	522.2	5.6	1567
	99	556(-)	1438	577.4(-)	16	719.5(-)	34	643.8(-)	41	544.3	5.4	1537
	100	571(-)	1728	596.6(-)	20	730.1(-)	29	656.8(-)	32	535.7	8.3	1498
W-D-L		0-1-9	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-



(a) (b) (c)

Fig. S1. The convergence curves of the compared algorithms on (a) U_3-75-n50 (b) U_3-100-n100 and (c) berlin52 ($\alpha = 1$).

same runtime as in [4], MATSP-D can converge to much better solutions than HGA.

REFERENCES

[1] N. Agatz, P. Bouman, and M. Schmidt, "Optimization approaches for the traveling salesman problem with drone," *Transp. Sci.*, vol. 52, no. 4, pp. 965–981, 2018.

[2] C. C. Murray and A. G. Chu, "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery," *Transp. Res. Pt. C-Emerg. Technol.*, vol. 54, pp. 86–109, 2015.

[3] J. C. de Freitas and P. H. V. Penna, "A variable neighborhood search for flying sidekick traveling salesman problem," *Int. Trans. Oper. Res.*, vol. 27, no. 1, pp. 267–290, 2020.

[4] Q. M. Ha, Y. Deville, Q. D. Pham, and M. H. Hà, "A hybrid genetic algorithm for the traveling salesman problem with drone," *J. Heuristics*, vol. 26, no. 2, pp. 219–247, 2020.

TABLE SIV
THE AVERAGE PERFORMANCE AND RUNTIME OF THE COMPARED ALGORITHMS ON THE UNIFORM DATASET WITH $\alpha = 3$

Instance		TSP-EP [1]		TSP-GP [1]		TSP-MC [2]		HGVNS [3]		MATSP-D		
N	id	Cost	Time(s)	Cost	Time(s)	Cost	Time(s)	Avg.Cost	Time(s)	Avg.Cost	Std	Time(s)
50	71	365.4(-)	33	380.8(-)	3	496.1(-)	2	452.9(-)	3	349.9	3.5	161
	72	392.4(-)	48	427.7(-)	1	551.5(-)	2	453(-)	2	384.4	2.8	114
	73	383(-)	31	394.8(-)	2	536.3(-)	2	394.3(-)	2	364.6	2.5	125
	74	390.1(-)	27	396.1(-)	2	490.2(-)	2	439(-)	2	372.7	3.7	119
	75	415.1(-)	32	414.4(-)	3	538.3(-)	2	482.5(-)	2	364.3	2.6	115
	76	347.9(-)	30	363.6(-)	2	481.4(-)	2	427.5(-)	3	333.7	3	119
	77	386.6(-)	38	396.3(-)	2	572.9(-)	2	440.6(-)	2	377.6	2.7	109
	78	397.4(-)	30	424.9(-)	2	560.5(-)	1	451.5(-)	2	369.7	3.8	105
	79	365.1(-)	40	366.7(-)	2	518.5(-)	2	433.1(-)	2	334.2	2.9	131
	80	359.4(-)	45	371.7(-)	1	460.2(-)	2	378.5(-)	3	336.1	3	108
W-D-L		0-0-10	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
75	81	457.3(-)	710	468.5(-)	9	637.9(-)	8	515.4(-)	9	425.3	7.2	597
	82	433.6(-)	376	434.8(-)	9	594(-)	7	456.8(-)	9	403.9	4.9	542
	83	407.8(-)	367	426.9(-)	8	576.5(-)	10	470(-)	12	393.7	5	668
	84	438.1(-)	434	441(-)	11	616.7(-)	8	488.8(-)	10	414.1	5.9	629
	85	467.4(-)	472	474.9(-)	12	670.5(-)	7	546.1(-)	8	445.6	4.9	705
	86	450.4(-)	539	452.9(-)	10	605(-)	9	590.3(-)	11	408.7	6.1	625
	87	438.5(-)	548	454.3(-)	11	595.8(-)	11	518.3(-)	11	415.1	4.2	573
	88	485.5(-)	486	504(-)	10	639.9(-)	8	537(-)	9	410.5	4.6	608
	89	420.1(-)	344	429(-)	10	569.3(-)	9	477.1(-)	11	390.5	6.1	625
	90	426.7(-)	530	431.8(-)	11	617.3(-)	8	616.9(-)	9	414	7.8	670
W-D-L		0-0-10	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
100	91	531.2(-)	2226	538.2(-)	31	759.4(-)	22	584.9(-)	26	501.2	6.6	2206
	92	482.4(-)	2488	511.7(-)	24	678.8(-)	26	550.8(-)	31	442.2	7.6	1909
	93	484.6(-)	2772	499.5(-)	33	676.4(-)	24	523.5(-)	30	446	7.2	1981
	94	506.4(-)	2477	528.2(-)	32	705.4(-)	25	594(-)	29	476.1	10.7	2014
	95	502.3(-)	2814	528.9(-)	26	757.2(-)	23	557(-)	32	484.8	5.8	2258
	96	523(-)	2918	528.5(-)	29	724.3(-)	25	585.7(-)	29	501.5	3.4	1944
	97	547.9(-)	2966	563(-)	33	739.9(-)	24	582.3(-)	34	493.4	15	1978
	98	479.6(-)	1828	499.2(-)	29	691.1(-)	26	547.7(-)	30	461.9	9.6	2200
	99	524.3(-)	2040	531.9(-)	31	707.9(-)	31	578.9(-)	40	498.5	7.4	1920
	100	539.3(-)	2122	547.8(-)	35	715.4(-)	28	676.8(-)	31	474.5	15.2	2245
W-D-L		0-0-10	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-

TABLE SV
THE AVERAGE PERFORMANCE AND RUNTIME OF THE COMPARED ALGORITHMS ON THE SINGLE-CENTER DATASET WITH $\alpha = 1$

Instance		TSP-EP [1]		TSP-GP [1]		TSP-MC [2]		HGVNS [3]		MATSP-D		
N	id	Cost	Time(s)	Cost	Time(s)	Cost	Time(s)	Avg.Cost	Time(s)	Avg.Cost	Std	Time(s)
50	71	564.1(-)	16	664.8(-)	1	613.4(-)	2	610(-)	2	550.6	2.4	85
	72	623.2(=)	12	731.9(-)	1	738.1(-)	2	685(-)	2	622.9	4.2	160
	73	470.5(-)	20	534.7(-)	2	581.3(-)	2	520.1(-)	2	468.9	4.5	164
	74	711.9(-)	26	842.2(-)	1	805.7(-)	2	791.9(-)	2	695.8	3.1	156
	75	722.7(-)	23	878.5(-)	1	814.7(-)	2	772.1(-)	2	705.9	8.1	178
	76	696.1(-)	8	837.8(-)	1	745.5(-)	2	745.5(-)	2	687.9	7.7	160
	77	567.8(=)	11	681.8(-)	1	603(-)	2	602.7(-)	2	569.3	4.2	193
	78	719.9(=)	5	789.3(-)	0	743.6(-)	2	743.6(-)	2	719.1	4.2	138
	79	587.3(-)	16	672.7(-)	1	648.2(-)	2	614.3(-)	2	577.1	5.7	91
	80	744.9(-)	7	900.4(-)	1	822.4(-)	2	803.7(-)	3	721.7	6	85
W-D-L		0-3-7	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
75	81	952.2(+)	369	1128.7(-)	5	1161.6(-)	7	1005(-)	7	959	13.9	394
	82	762.3(=)	149	922.8(-)	4	817.2(-)	11	811.3(-)	12	761.7	8.4	353
	83	788.9(+)	274	904.1(-)	4	897.2(-)	7	854(-)	8	796.9	6.3	365
	84	884.8(+)	113	989(-)	4	1014.8(-)	8	947.4(-)	9	893.8	10.1	367
	85	743.3(+)	175	956.2(-)	4	809.8(-)	8	795.3(-)	8	759.2	6.9	399
	86	941.9(=)	77	1100.7(-)	5	1121.6(-)	7	1040.3(-)	8	939.9	9.3	401
	87	903.2(-)	197	1025.2(-)	5	1030.9(-)	7	966.6(-)	9	897.9	9.4	394
	88	864.2(+)	179	1089.5(-)	4	927.7(-)	10	920.7(-)	11	881.4	11.9	416
	89	815.4(-)	89	915.8(-)	5	881.5(-)	9	859.9(-)	10	808.4	12.7	376
	90	937.2(-)	169	1099.9(-)	5	1138.3(-)	7	1076.5(-)	8	929.5	14.6	372
W-D-L		5-2-3	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
100	91	987.5(+)	521	1209.1(-)	12	1144(-)	26	1105.6(-)	29	996	15.1	1141
	92	1112.4(=)	865	1302.4(-)	10	1291.7(-)	23	1197.8(-)	25	1111.9	15.2	1114
	93	1097.2(=)	660	1282.7(-)	5	1186.9(-)	26	1162(-)	29	1096.6	12.4	1129
	94	1234.6(-)	740	1430.6(-)	8	1388(-)	23	1387.6(-)	24	1208.5	15.1	1136
	95	958.9(=)	906	1133.5(-)	13	1100.5(-)	27	1045.3(-)	28	964.6	12.7	1112
	96	1076.3(-)	992	1264.8(-)	13	1171.9(-)	30	1090.1(-)	32	1056.3	13.1	1119
	97	1069.4(+)	982	1295.3(-)	11	1188(-)	27	1159.6(-)	29	1086.7	8.9	1187
	98	1030.7(=)	1198	1208(-)	13	1165.1(-)	34	1108.2(-)	35	1034.5	13.2	1205
	99	846.8(=)	847	952.7(-)	17	1021.2(-)	22	947.5(-)	24	847.9	8.8	1123
	100	962.6(-)	750	1119.5(-)	14	1079.6(-)	25	1079.6(-)	27	954.8	10.8	1090
W-D-L		2-5-3	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-

TABLE SVI
THE AVERAGE PERFORMANCE AND RUNTIME OF THE COMPARED ALGORITHMS ON THE SINGLE-CENTER DATASET WITH $\alpha = 2$

Instance		TSP-EP [1]		TSP-GP [1]		TSP-MC [2]		HGVNS [3]		MATSP-D		
N	id	Cost	Time(s)	Cost	Time(s)	Cost	Time(s)	Avg.Cost	Time(s)	Avg.Cost	Std	Time(s)
50	71	457.7(-)	21	480.3(-)	2	602.3(-)	2	505.7(-)	2	437.3	6.4	98
	72	484.3(+)	17	533.8(-)	1	730.4(-)	2	566.2(-)	2	489.4	6.2	200
	73	375.1(-)	16	410(-)	1	564.8(-)	2	457.3(-)	2	368	3.4	178
	74	556.2(-)	31	597.2(-)	1	729.4(-)	2	693.5(-)	3	545.1	9.7	187
	75	587.1(-)	24	648.1(-)	1	798(-)	2	702.5(-)	2	572.9	7.5	181
	76	569.6(-)	22	634.2(-)	1	734.9(-)	2	664.3(-)	2	552.7	6.6	198
	77	439.9(-)	16	489.1(-)	2	572.4(-)	2	543.2(-)	2	436.6	4.9	186
	78	553.5(+)	53	628.6(-)	1	691(-)	2	671.7(-)	3	559.8	5.7	83
	79	463.4(-)	70	483.2(-)	2	639.3(-)	2	513.8(-)	2	450	6.8	93
	80	576.8(-)	24	643.1(-)	2	717.4(-)	2	672.9(-)	3	565.4	10.1	91
W-D-L		2-0-8	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
75	81	734.1(-)	504	803.4(-)	8	1017.4(-)	9	840(-)	11	724.3	15.6	447
	82	596.9(-)	298	650.4(-)	7	769.5(-)	11	736.9(-)	12	588.1	11.9	460
	83	640.7(-)	309	664.8(-)	9	879.5(-)	8	735.1(-)	9	626	7.8	522
	84	738.5(-)	260	785(-)	9	955.5(-)	9	811.6(-)	10	710.6	9.2	506
	85	589.5(+)	282	636.4(-)	7	739.5(-)	9	697.2(-)	10	601.1	8.6	453
	86	743.9(=)	291	804.2(-)	8	1104.5(-)	7	828.5(-)	8	746.9	16.3	476
	87	711.9(=)	453	781.4(-)	9	1024(-)	7	868.8(-)	8	709.5	12.7	485
	88	697.5(+)	232	770.2(-)	8	860.3(-)	15	847.7(-)	16	707.7	12.4	443
	89	665.8(-)	353	709.7(-)	6	841.6(-)	9	773.3(-)	11	657.1	8.4	505
	90	743(-)	414	807.3(-)	10	1088.2(-)	7	855.2(-)	10	722.3	10.9	508
W-D-L		2-2-6	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
100	91	820(-)	1250	869(-)	17	1099.2(-)	28	916.4(-)	32	801	14.7	1449
	92	877.3(+)	1481	944.1(-)	24	1243(-)	26	1059.2(-)	29	888.1	19.8	1477
	93	932.2(-)	1904	969.1(-)	22	1135.7(-)	27	1018.7(-)	30	883.1	17.1	1743
	94	987.4(-)	1679	1026.3(-)	23	1311.5(-)	25	1108.2(-)	29	976.1	16.4	1416
	95	735.7(+)	2276	792.6(-)	23	1072.5(-)	29	879.3(-)	33	760.3	14.1	1545
	96	813.1(+)	1984	912.6(-)	18	1093.4(-)	33	962(-)	37	842.6	19.8	1480
	97	843.7(+)	1829	943.5(-)	20	1129.8(-)	29	976.7(-)	33	853.9	16.2	1677
	98	824.7(-)	1483	916.7(-)	21	1026.1(-)	34	913.2(-)	38	806.2	12.3	1405
	99	699.3(-)	2055	737.2(-)	25	990.2(-)	25	775.9(-)	31	683.4	10.4	1643
	100	805.5(-)	2273	835.8(-)	20	1042.8(-)	25	916.1(-)	29	782.1	14.3	1668
W-D-L		4-0-6	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-

TABLE SVII
THE AVERAGE PERFORMANCE AND RUNTIME OF THE COMPARED ALGORITHMS ON THE SINGLE-CENTER DATASET WITH $\alpha = 3$

Instance		TSP-EP [1]		TSP-GP [1]		TSP-MC [2]		HGVNS [3]		MATSP-D		
N	id	Cost	Time(s)	Cost	Time(s)	Cost	Time(s)	Avg.Cost	Time(s)	Avg.Cost	Std	Time(s)
50	71	410.5(-)	74	436(-)	2	588.6(-)	2	470.1(-)	2	386.2	4.3	145
	72	433.8(=)	52	465.4(-)	2	724.2(-)	2	525.8(-)	2	433.7	4.9	109
	73	322(-)	41	339.9(-)	2	549.2(-)	2	361.7(-)	2	309	3.4	109
	74	484(-)	90	514(-)	2	715.7(-)	2	589.8(-)	3	467.2	7.8	120
	75	499.3(-)	41	539.5(-)	2	745.3(-)	2	610.9(-)	2	481.8	7.8	99
	76	531.2(-)	34	550.9(-)	1	720.7(-)	1	546.6(-)	2	486.8	7.6	107
	77	384.6(-)	25	401.1(-)	2	552.8(-)	2	436.7(-)	2	368.1	4.3	109
	78	467.6(-)	29	523.4(-)	2	646.8(-)	2	594.8(-)	2	453.3	6.5	98
	79	379.3(-)	41	427.6(-)	2	639.3(-)	2	432.4(-)	2	374.1	5.2	94
	80	479(-)	47	543.2(-)	2	681.8(-)	3	639.6(-)	3	473.8	9	101
W-D-L		0-1-9	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
75	81	650.9(-)	362	715(-)	8	964.1(-)	9	664.7(-)	11	621.7	13.4	540
	82	505.9(-)	633	550.7(-)	10	757(-)	10	584(-)	15	498.9	9.7	507
	83	564.3(-)	554	572.8(-)	11	855.8(-)	7	675.2(-)	9	556.5	9.6	576
	84	666.1(-)	489	676.1(-)	10	935(-)	9	769.4(-)	10	622.7	10.1	553
	85	535.8(-)	503	548.7(-)	10	723.5(-)	9	622.7(-)	10	522.4	9.9	505
	86	685.5(-)	731	724.7(-)	9	1075.1(-)	7	777.2(-)	8	657	13.3	535
	87	590.2(=)	707	648.7(-)	10	1009.4(-)	7	697.4(-)	8	592.6	10.2	544
	88	659(-)	470	685.8(-)	11	818.9(-)	16	818.9(-)	16	617.9	13.7	504
	89	562.5(=)	804	576.5(-)	13	843.5(-)	9	667.5(-)	11	563.6	10.2	547
	90	665.7(-)	612	712(-)	11	1075.2(-)	7	773.2(-)	9	640.3	7.8	544
W-D-L		0-2-8	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
100	91	712.5(-)	2100	752.9(-)	24	1081.8(-)	28	798(-)	35	698.6	13.2	1696
	92	789.6(-)	3497	806.5(-)	33	1206.2(-)	25	906.8(-)	34	756.6	15.5	1675
	93	875.3(-)	3488	881.1(-)	27	1112.5(-)	27	918.6(-)	33	779.3	16	1964
	94	837.1(-)	2901	916(-)	33	1253.1(-)	25	1051.4(-)	27	823.6	16.2	1586
	95	632.5(+)	3614	684.3(-)	25	1040.7(-)	28	791(-)	33	647.1	18.7	1740
	96	757.9(-)	2258	779.3(-)	25	1079.4(-)	31	1079.4(-)	33	722.9	16.1	1711
	97	741.7(+)	2176	781.1(-)	35	996.4(-)	31	901.8(-)	34	756.8	14.2	1870
	98	685(=)	2085	731.9(-)	35	948.9(-)	37	766.6(-)	42	685.1	18.3	1672
	99	611.8(-)	2142	643.3(-)	32	947(-)	24	764.1(-)	28	594.9	12.8	1735
	100	721(-)	1635	760.6(-)	26	1014.7(-)	27	864.3(-)	30	691.1	13.6	2084
W-D-L		2-1-7	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-

TABLE SVIII
THE AVERAGE PERFORMANCE AND RUNTIME OF THE COMPARED ALGORITHMS ON THE DOUBLE-CENTER DATASET WITH $\alpha = 1$

Instance		TSP-EP [1]		TSP-GP [1]		TSP-MC [2]		HGVNS [3]		MATSP-D		
N	id	Cost	Time(s)	Cost	Time(s)	Cost	Time(s)	Avg.Cost	Time(s)	Avg.Cost	Std	Time(s)
50	71	1060.7(=)	8	1260.3(-)	2	1178(-)	2	1110.7(-)	2	1057.4	8.1	120
	72	989.2(-)	8	1134.2(-)	1	1170.7(-)	2	1082(-)	2	985.6	8	86
	73	900.6(-)	10	1101.2(-)	1	1004(-)	2	1004(-)	2	893.5	7.7	82
	74	903.7(+)	13	1012.3(-)	1	1080.5(-)	1	950.1(-)	1	910.5	7.4	81
	75	1017.1(=)	10	1170.1(-)	1	1185(-)	2	1055.8(-)	2	1016.1	5	85
	76	1044.1(-)	8	1280.9(-)	2	1171.9(-)	2	1142(-)	3	1029.3	8	85
	77	961.2(+)	10	1098.4(-)	1	1048.4(-)	2	1048.4(-)	2	966.2	5.4	87
	78	949.6(+)	22	1173.1(-)	1	1167.3(-)	1	1020.7(-)	2	961.7	8.3	86
	79	955.7(=)	18	1091.3(-)	1	1080.4(-)	1	1080.2(-)	2	957.9	4.8	85
	80	1070.5(-)	22	1202(-)	1	1196.7(-)	2	1127.2(-)	2	1067.9	7.6	79
W-D-L		3-3-4	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
75	81	1091.6(+)	129	1247.8(-)	3	1292(-)	8	1188(-)	9	1107.4	8.5	406
	82	1170.2(-)	150	1380.8(-)	6	1363.1(-)	6	1303.8(-)	8	1161.3	13.2	424
	83	1128.7(-)	98	1312(-)	5	1231.6(-)	9	1179.9(-)	10	1121.8	8.1	421
	84	1452.3(-)	235	1680.7(-)	4	1646.5(-)	8	1630.1(-)	8	1416.1	18.6	368
	85	1356.3(=)	179	1742.8(-)	4	1575.4(-)	7	1490.6(-)	9	1357.8	12.7	413
	86	1130.4(+)	98	1359.3(-)	3	1262.5(-)	7	1218.8(-)	8	1139	4.7	367
	87	1196.9(=)	161	1392.1(-)	4	1446(-)	6	1307.7(-)	8	1198.1	10.6	367
	88	1353.9(+)	312	1697.6(-)	4	1553.3(-)	9	1521.1(-)	10	1372.7	9.7	433
	89	1159.7(+)	220	1399.4(-)	6	1335.6(-)	7	1234.4(-)	8	1167.2	12.6	391
	90	1100.8(=)	158	1262.7(-)	7	1205.3(-)	10	1184.6(-)	12	1101.9	7	405
W-D-L		4-3-3	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
100	91	1318.6(+)	667	1609.6(-)	12	1572.4(-)	22	1527.5(-)	25	1332.9	13.5	1099
	92	1312.2(-)	1156	1485.3(-)	17	1483.2(-)	22	1468.2(-)	24	1301.6	11.3	1234
	93	1216.5(+)	1440	1407.9(-)	18	1407.5(-)	23	1304.8(-)	25	1233.2	9.1	1203
	94	1332.5(=)	2064	1552.5(-)	20	1517.8(-)	28	1439.2(-)	29	1328.2	17.1	1158
	95	1395.5(+)	1359	1717.4(-)	16	1601.2(-)	24	1543.5(-)	26	1431.6	12.3	1178
	96	1486.5(-)	1191	1684(-)	12	1635.8(-)	25	1622.4(-)	30	1459.5	14.9	1174
	97	1487.7(+)	723	1779.6(-)	15	1690.3(-)	29	1638.4(-)	31	1506.5	16.7	1138
	98	1397.4(-)	863	1565.7(-)	17	1518.1(-)	27	1467.2(-)	29	1385.3	18.6	1220
	99	1250.6(+)	1095	1557.1(-)	15	1422.9(-)	27	1387.6(-)	30	1263.2	11.4	1176
	100	1407.1(+)	1444	1740.7(-)	18	1595.7(-)	30	1564.8(-)	32	1421.3	11.1	1230
W-D-L		6-1-3	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-

TABLE SIX
THE AVERAGE PERFORMANCE AND RUNTIME OF THE COMPARED ALGORITHMS ON THE DOUBLE-CENTER DATASET WITH $\alpha = 2$

Instance		TSP-EP [1]		TSP-GP [1]		TSP-MC [2]		HGVNS [3]		MATSP-D		
N	id	Cost	Time(s)	Cost	Time(s)	Cost	Time(s)	Avg.Cost	Time(s)	Avg.Cost	Std	Time(s)
50	71	871.8(-)	22	946.6(-)	2	1106.9(-)	2	960(-)	2	836.4	20.1	133
	72	823.1(-)	20	908.5(-)	2	1105.8(-)	2	1105.8(-)	2	806.8	9.3	82
	73	756.9(-)	18	769.7(-)	2	953.4(-)	2	879(-)	2	725.7	7.8	83
	74	705.3(+)	55	776.5(-)	2	1073.2(-)	1	788(-)	2	710.1	7.1	88
	75	847.2(-)	20	914.3(-)	1	1169.6(-)	1	959.7(-)	2	836.5	10.3	85
	76	856(-)	22	942.1(-)	1	1102.1(-)	2	902.5(-)	3	810.1	7.3	84
	77	736.6(=)	28	785.4(-)	2	1028(-)	2	961.1(-)	3	738.1	8.6	96
	78	762.5(-)	22	816.8(-)	1	1096.9(-)	2	1095.2(-)	2	747	6.9	88
	79	814.8(-)	20	851.4(-)	1	1076.3(-)	1	1076.3(-)	1	787	11.2	88
	80	871.9(=)	36	950(-)	2	1113.1(-)	2	991.8(-)	3	871.3	4.9	89
W-D-L		1-2-7	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
75	81	868.1(+)	452	969.5(-)	10	1279.9(-)	7	957.4(-)	10	873	10.2	434
	82	919.2(-)	420	1021.8(-)	6	1297.5(-)	7	1074.8(-)	9	906.9	11.5	420
	83	894.7(-)	470	992.9(-)	9	1100.2(-)	14	1077.1(-)	15	881.3	8.3	435
	84	1150.1(=)	575	1287.4(-)	12	1619.1(-)	8	1594.3(-)	9	1149	14.9	418
	85	1067.8(=)	458	1201.2(-)	8	1436.1(-)	7	1241.2(-)	10	1073.4	18	423
	86	928.8(+)	319	974.7(-)	8	1219(-)	8	1098.1(-)	12	932.5	9.3	448
	87	944.5(+)	252	1009.2(-)	9	1412.8(-)	7	1052.4(-)	11	953	10.3	394
	88	1171(-)	291	1241.4(-)	9	1477.2(-)	11	1271.4(-)	12	1096.9	21.4	444
	89	998.3(-)	358	1081.8(-)	9	1330(-)	7	1111.4(-)	10	940.6	13.8	437
	90	861.2(-)	492	973.3(-)	9	1019.4(-)	14	991.6(-)	15	853.7	8.7	409
W-D-L		3-2-5	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
100	91	1063.4(=)	1751	1142.2(-)	23	1515.9(-)	23	1326.9(-)	27	1058.6	13.1	1330
	92	1013.3(=)	1761	1140.7(-)	26	1460.9(-)	23	1169.6(-)	30	1014.3	15.1	1260
	93	1041(-)	1935	1087.5(-)	22	1374.5(-)	23	1161.6(-)	27	1014.5	12.7	1414
	94	1092.2(=)	2581	1205.6(-)	25	1470.1(-)	29	1295.5(-)	32	1095	19	1361
	95	1131(-)	2372	1230.3(-)	18	1557.9(-)	25	1357.8(-)	31	1115.3	13	1338
	96	1210.9(-)	2003	1280.7(-)	30	1551.5(-)	27	1406.3(-)	31	1188.9	16.5	1367
	97	1247.5(-)	1869	1392.3(-)	24	1550.2(-)	32	1344.6(-)	36	1240.6	12.8	1354
	98	1154.1(=)	3009	1195.7(-)	28	1461.8(-)	31	1292.2(-)	37	1148.7	15.7	1563
	99	1008.6(+)	1612	1098.1(-)	22	1333.7(-)	27	1206.3(-)	32	1020.7	10.1	1396
	100	1113.7(+)	1380	1248.4(-)	29	1494.1(-)	32	1325.6(-)	38	1144.3	11.7	1305
W-D-L		2-4-4	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-

TABLE SX
THE AVERAGE PERFORMANCE AND RUNTIME OF THE COMPARED ALGORITHMS ON THE DOUBLE-CENTER DATASET WITH $\alpha = 3$

Instance		TSP-EP [1]		TSP-GP [1]		TSP-MC [2]		HGVNS [3]		MATSP-D		
N	id	Cost	Time(s)	Cost	Time(s)	Cost	Time(s)	Avg.Cost	Time(s)	Avg.Cost	Std	Time(s)
50	71	787.8(-)	44	822(-)	3	1062.4(-)	2	878.2(-)	3	730.1	7.5	146
	72	750(-)	42	802.5(-)	2	1099.8(-)	1	1099.8(-)	2	691	8.5	100
	73	671.6(-)	59	725.4(-)	2	967.8(-)	2	909.9(-)	2	654.5	6.7	96
	74	658.2(-)	25	662.8(-)	2	1060.4(-)	1	720.5(-)	2	624.5	4.6	95
	75	795.7(-)	36	790.2(-)	4	1146.4(-)	1	875.6(-)	2	735.5	7.9	104
	76	797.1(-)	24	791.4(-)	3	1074.9(-)	2	942.7(-)	3	722.8	7	92
	77	700.4(-)	23	704.1(-)	2	995.3(-)	2	833.8(-)	3	656.7	6.9	97
	78	693.8(-)	35	694.6(-)	2	1096.9(-)	2	1095.2(-)	2	653	4.4	94
	79	737.5(-)	33	759.3(-)	2	1059.5(-)	2	1041.8(-)	2	691.4	7	95
	80	831.6(-)	47	854.8(-)	3	1056.7(-)	2	895.1(-)	3	794.1	9.9	104
W-D-L		0-0-10	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
75	81	784.2(-)	684	816(-)	12	1243.2(-)	8	878.1(-)	10	768.3	8.4	515
	82	816.9(-)	714	850.9(-)	14	1281.3(-)	7	1036.6(-)	9	802.3	10.9	594
	83	789.3(=)	617	835.1(-)	12	983.3(-)	15	899.2(-)	19	787.4	5.3	535
	84	1021.2(-)	974	1076.5(-)	15	1618.6(-)	8	1285.4(-)	11	980.3	12.2	483
	85	970.5(-)	502	1014.5(-)	9	1402.1(-)	8	1155.5(-)	11	932.6	15.5	487
	86	858.6(-)	523	926.8(-)	7	1196.7(-)	8	1196.7(-)	9	816.5	10.2	475
	87	843.2(-)	389	908.4(-)	11	1399.3(-)	6	965(-)	10	837.9	10.4	470
	88	957.4(-)	739	1077.3(-)	10	1374.1(-)	15	1116.8(-)	16	910.2	17.3	468
	89	869.7(-)	502	940.3(-)	10	1286.3(-)	8	945.5(-)	11	826.4	10.1	478
	90	802.7(-)	364	838.7(-)	10	951.7(-)	17	867.6(-)	22	763.9	8.9	520
W-D-L		0-1-9	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-
100	91	922(=)	2440	971.4(-)	38	1503.1(-)	23	1206.9(-)	30	922.8	14.9	1578
	92	896.5(-)	3223	985.6(-)	29	1359.7(-)	24	1008.5(-)	30	863.1	14.7	1404
	93	911.8(-)	4255	1005.6(-)	29	1376.6(-)	24	1079.6(-)	28	902.7	11.5	1582
	94	951.1(=)	4048	1017.2(-)	43	1366.2(-)	34	1151.2(-)	40	960.3	23.9	1529
	95	982.6(-)	2606	1048.5(-)	29	1511.2(-)	26	1178(-)	30	974.8	14.7	1658
	96	1104.9(-)	3356	1116.1(-)	39	1511.6(-)	29	1342.5(-)	34	1037.2	17.4	1570
	97	1167.2(-)	1444	1198.4(-)	39	1480.2(-)	35	1188(-)	43	1088.5	19	1539
	98	1086.8(-)	2009	1126.6(-)	29	1452.6(-)	32	1286.2(-)	35	1026.3	18	1732
	99	924.1(-)	1495	961.6(-)	27	1309.7(-)	27	1179.4(-)	30	906.5	10.6	1667
	100	1021.5(-)	1984	1070(-)	37	1422.5(-)	30	1249(-)	34	993.1	9.8	1531
W-D-L		0-2-8	-	0-0-10	-	0-0-10	-	0-0-10	-	-	-	-