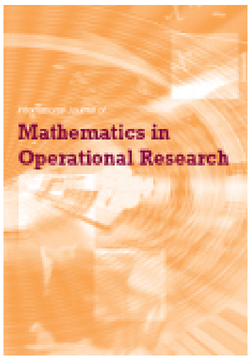


**International Journal of Mathematics in Operational Research**

ISSN online: 1757-5869 - ISSN print: 1757-5850
https://www.inderscience.com/ijmor

# Determining the optimal parcel delivery method using one drone and one truck

Kazuki Satoh, Hiroshi Morita

**DOI:** 10.1504/IJMOR.2022.10047093

**Article History:**

| | |
|---|---|
| Received: | 24 March 2022 |
| Accepted: | 26 March 2022 |
| Published online: | 05 April 2023 |

# Determining the optimal parcel delivery method using one drone and one truck

Kazuki Satoh and Hiroshi Morita*

Graduate School of Information Science and Technology,
Osaka University,
Suita, Osaka, Japan
Email: kazuki.sato@ist.osaka-u.ac.jp
Email: morita@ist.osaka-u.ac.jp
*Corresponding author

**Abstract:** Delivering parcels using a combination of drones and trucks represents a promising new package delivery method. In previous studies, several truck/drone delivery planning problems and their solutions have been proposed. However, little research has been done to determine which delivery method is most appropriate. The reason for this is that it is difficult to obtain an exact solution for such problems, and thus the accuracy of the solutions is an issue. In this study, we propose an accurate solution procedure for the flying sidekick travelling salesman problem (FSTSP) and the parallel drone scheduling travelling salesman problem (PDSTSP), and establish which delivery method is most suitable based on the solutions obtained.

**Keywords:** drone; vehicle routing problem; travelling salesman problem; parcel delivery; heuristics.

**Reference** to this paper should be made as follows: Satoh, K. and Morita, H. (2023) 'Determining the optimal parcel delivery method using one drone and one truck', *Int. J. Mathematics in Operational Research*, Vol. 24, No. 4, pp.451–474.

**Biographical notes:** Kazuki Satoh is a graduate student at the Department of Information and Physical Sciences, Graduate School of Information Science and Technology, Osaka University. He received his MS in Information Science and Technology from Osaka University in 2022. His research interests are combinatorial optimisation and operations research.

Hiroshi Morita is a Professor at the Department of Information and Physical Sciences, Graduate School of Information Science and Technology, Osaka University. He received his PhD in Engineering from Kyoto University in 1992. His research interests are related to system modelling and optimisation, operations research, data science and energy management. He has published research papers at *European Journal of Operational Research*, *Transportation Research*, *Applied Energy* and so on.

## 1 Introduction

In recent years, parcel deliveries have continued to increase due to the significant spread of Internet shopping in Japan (Ministry of Land, Infrastructure, Transport and Tourism, 2020). This has led to a number of problems, including a shortage of delivery personnel and the threat of an adverse environmental impact. Drone delivery is expected to be one of the solutions to these problems (Goodchild and Toy, 2018). In 2013, Amazon announced plans to use drones, and many companies are currently researching the possibility of developing a similar system.

The differences between drone delivery and truck delivery are summarised in Table 1. As indicated, drones move faster, as they are not limited by the road network or traffic congestion. Drones are also much lighter, which means they consume less energy when moving about. However, given their small size, it is difficult for drones to deliver more than one parcel at a time, and they must pick up another the parcel after each delivery. In addition, because they are powered by small batteries, delivery distances are limited. Thus, a proper combination of trucks and drones will likely be needed to provide efficient parcel delivery.

**Table 1** Drone delivery vs. truck delivery

| | *Speed* | *Weight* | *Capacity* | *Range* |
|---|---|---|---|---|
| Drone | High | Light | One | Short |
| Truck | Low | Heavy | Many | Long |

Murray and Chu (2015) first defined the flying sidekick travelling salesman problem (FSTSP) and parallel drone scheduling travelling salesman problem (PDSTSP) as delivery planning problems in which a combination of drones and trucks is used for parcel delivery. An example of each problem type is shown in Figure 1. In their paper, the authors compared the FSTSP and PDSTSP and sought to determine the superior delivery method. However, because of the difficulty of obtaining an exact solution, the authors used heuristic-based solutions in their comparisons, acknowledging that such comparisons were inexact because of the possibility of large errors due to the limited accuracy of their heuristic-based methods.

The objective of this study is to determine more rigorously the optimal delivery method in a delivery system that uses one drone and one truck. The main contributions of this paper are as follows:

1 a new FSTSP method is proposed and numerical experimental results are presented

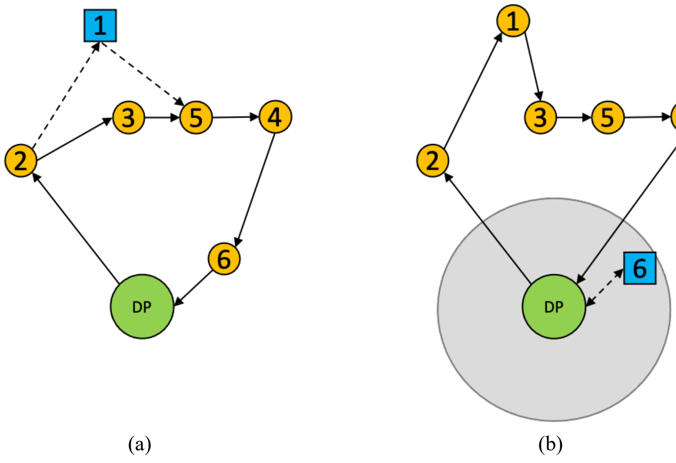
2 a new PDSTSP method is proposed and numerical experimental results are presented

3 the results of numerical experiments are used to determine which delivery method (FSTSP or PDSTSP) is superior for a given set of destinations.

The remainder of this paper is organised as follows: Section 2 discusses previous research on drone delivery systems. Section 3 defines the FSTSP and describes the proposed method. Section 4 defines the PDSTSP and describes the proposed method.

In Section 5, we present the results of numerical experiments comparing the proposed method with the conventional method for the FSTSP and the PDSTSP. Finally, Section 6 offers conclusions and suggests the direction of future work.

**Figure 1** Delivery methods in this study, (a) FSTSP (b) PDSTSP (see online version for colours)



## 2 Related literature

In their 2015 paper, Murray and Chu (2015) introduced the FSTSP, which involves a combination of one truck and one drone to deliver parcels. In the problem they defined, the drone can carry only one parcel at a time and can only deliver parcels to a certain defined subset of customers. The drone can start and end deliveries at a certain customer node, whether at the distribution centre or the truck, but the start and end points must not be the same. The problem is to determine the delivery route that minimises the delivery time, with the delivery time being the time it takes to deliver the parcels to all customers and return to the distribution centre. For this problem, Murray and Chu (2015) proposed a mixed integer liner programming (MILP) formulation and a heuristic.

Murray and Chu (2015) defined the PDSTSP as a separate problem from the FSTSP. In this problem, a drone delivers a parcel after departing from a distribution centre, then returns to retrieve another parcel from the distribution centre. Here, the drone delivers parcels near the distribution centre, while the truck delivers parcels far from the distribution centre. As with the FSTSP, Murray and Chu (2015) propose an MILP formulation and a heuristic. Kundu and Matis (2017) defined a model that takes into account wind effects and drone battery consumption as an advanced version of FSTSP, and new heuristics are proposed. Ponza (2016) extended the work of Murray and Chu (2015) in his master's thesis, offering a method based on the simulated annealing (SA) metaheuristic.

Agatz et al. (2018) defined the travelling salesman problem with drone (TSP-D), a modified version of the FSTSP. In this version of the problem, the drone uses the same network as the truck, and the drone can depart from and retrieve parcels at the same vertex. Agatz et al. (2018) solved this problem with an integer programming (IP) formulation and a heuristic. Bouman et al. (2018) extended this work and developed a new exact solution method using dynamic programming (DP) for large cases. Yurek and Ozmutlu (2018) proposed a decomposition-based iterative optimisation algorithm for heuristic approach for medium-sized instances. Ha et al. (2018) introduced a new variant of the TSP-D, called the min-cost TSP-D, in which the objective is to minimise operational costs. Ha et al. solved this problem with an MILP formulation and two heuristics: a greedy randomised adaptive search procedure (GRASP) and TSP-LS. Furthermore, Ha et al. (2020) proposed a new hybrid genetic algorithm (HGA) with adaptive diversity control to solve the TSP-D under both min-cost and min-time objectives.

In Wang et al. (2017) and Poikonen et al. (2017), the authors considered the vehicle routing problem with drones (VRP-D) and presented worst-case results to evaluate the benefit of using additional drones. VRP-D is a generalisation of TSP-D, using multiple drones and multiple trucks to deliver parcels. However, a restriction that the drones must take off from and land on the same truck is imposed.

Chung et al. (2020) provide a detailed review of optimisation models and methods for drone-truck combined operations. Table 2 provides an overview of the various works described above, along with the problem addressed in the present paper. In each case, it shows the model, number of trucks, number of drones, and the proposed solution method.

**Table 2** Related works and characteristics

| *Reference* | *Problem* | *Trucks* | *Drones* | *Proposed solution* |
|---|---|---|---|---|
| Murray and Chu (2015) | FSTSP | 1 | 1 | MILP, heuristic |
| | PDSTSP | 1 | 1 | MILP, heuristic |
| Kundu and Matis (2017) | FSTSP | 1 | 1 | Heuristic |
| Ponza (2016) | FSTSP | 1 | 1 | SA |
| Agatz et al. (2018) | TSP-D | 1 | 1 | IP, DP, heuristic |
| Yurek and Ozmutlu (2018) | TSP-D | 1 | 1 | Heuristic |
| Ha et al. (2018) | TSP-D | 1 | 1 | MILP, GRASP |
| Ha et al. (2020) | TSP-D | 1 | 1 | HGA |
| Carlsson and Song (2018) | HRP | 1 | m | Heuristic |
| Wang et al. (2017) | VRPD | n | m | Theoretical insights |
| Poikonen et al. (2017) | VRPD | n | m | Theoretical insights |
| This work | FSTSP | 1 | 1 | Heuristic |
| | PDSTSP | 1 | 1 | Heuristic |

## 3 Flying sidekick travelling salesman problem

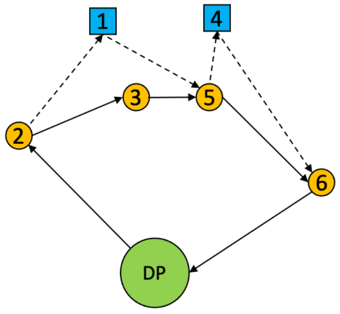
As previously noted, the FSTSP was first defined by Murray and Chu (2015). The model seeks to minimise the delivery time when a combination of one truck and one drone is used for delivery, and is classified as an NP-hard problem.

In this section, we describe the FSTSP used in this study and propose a new algorithm for solving it. The algorithm, which we call the TSP+DRP algorithm, divides the FSTSP into a truck routing problem (TSP) and a drone routing problem (DRP).

## *3.1 Problem setting*

Figure 2 shows the FSTSP problem setting. The numbered vertices represent delivery destinations. The distribution centre is labelled DP. The round vertices are the delivery destinations of the truck; the square vertices are the delivery destinations of the drone. The solid arrow indicates the delivery route of the truck. The dotted arrow is the delivery route of the drone.

**Figure 2** An example of a FSTSP solution (see online version for colours)



Here, a drone and the parcels to be delivered are loaded onto a truck and depart from the distribution centre. When the drone and the truck arrive at a certain vertex, the drone departs from the truck. The drone delivers the parcel to the specified vertex and moves to the collection vertex with the truck. Meanwhile, the truck delivers a parcel to another vertex and moves to the collection vertex of the drone. The truck then collects the drone at the collection vertex. For example, in Figure 2, the truck moves in the order DP, 2, 3, 5, 6, DP. The drone then departs when it arrives at vertex 2. The drone delivers the parcel to vertex 1 and moves to vertex 5, the collection point. Meanwhile, the truck delivers the package to vertex 3 and moves to vertex 5, the drone's collection point, to collect the drone.

In this way, we define as the delivery time the time it takes for the truck and the drone to depart from the distribution centre, deliver the parcels to all the destinations, and return to the distribution centre. The objective is to find the delivery route that minimises this delivery time.

The key assumptions for this problem are summarised below. The last item was not included in Murray and Chu (2015); however, we added it to make the conditions of the problem more realistic.

- A truck and a drone will always visit every vertex only once.
- A drone can only carry one parcel in one flight. A truck can deliver any number of parcels during its flight.

- It is not possible to launch or recover the drone while the truck is moving. All this work is done at a vertex.
- It is not possible to recover the drone at the same vertex from which it departed.
- The drone must be recovered while it is in flight.

## 3.2 Overview of the proposed method

In this section, we introduce the TSP+DRP algorithm, which is a combination of the travelling salesman problem (TSP) and the drone routing problem (DRP). As mentioned earlier, with this method, the FSTSP is divided into a truck delivery routing problem and a drone delivery routing problem, and the best solution is obtained.

The TSP+DRP algorithm involves four major steps: TruckCustomers is the set of vertices to be delivered to by truck; DroneCustomers is the set of vertices to be delivered to by drone; TruckRoute is the delivery route of the truck; and $t$ is the delivery time.

*TSP+DRP algorithm*

Step 1 Assign all deliveries to TruckCustomers and solve TSP for all vertices to get TruckRoute and $t$.

Step 2 Assign one vertex of TruckCustomers to DroneCustomers and update the TruckRoute.

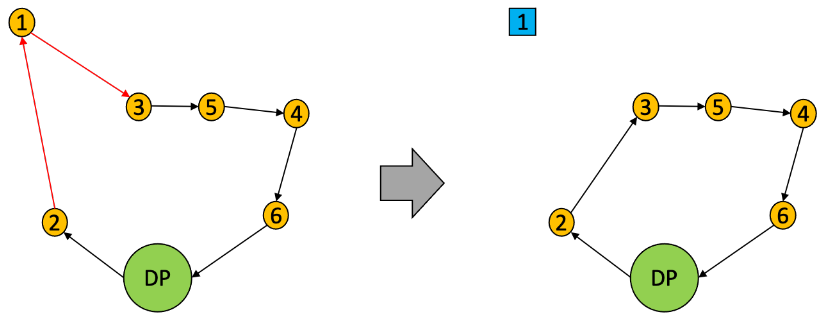
Step 3 Solve the DRP; if the delivery time is shorter than $t$, update $t$.

Step 4 If $t$ is not updated in step 3, exit. If $t$ is updated, return to step 2.

## 3.3 Change assignment

In this section, we describe step 2 of the TSP+DRP algorithm in detail.

**Figure 3** Example of a change in assignment (see online version for colours)



The method for changing assignments is based on the greedy idea of using drones to deliver to destinations that take too long to be delivered to by truck. This method takes advantage of the fact that the drone can deliver a parcel without being affected by the traffic network. In other words, the vertex that spends the most travel time in the TruckRoute is assigned to DroneCustomers. Figure 3 shows an example. Here, since it takes the longest time to travel to vertex 1 by truck, we change this vertex to the delivery destination of the drone.

To implement a change of assignment, we use the pseudo-code given in Algorithm1. It calculates the savings of the truck's travel time when vertex $j$ is excluded from the truck's destinations, and adds the vertex with the largest savings (AddNode) to the set of drone destinations (DroneCusutomers). C' represents the set of vertices that can be delivered to by the drone, and $\tau$ is the matrix of the time taken by the truck to move between each vertex. That is, $\tau_{ij}$ is the travel time for the truck to move from vertex $i$ to vertex $j$.

**Algorithm 1** Change assignment

```
Input: DroneCustomers, TruckRoute, $C'$, $\tau$
Output: DroneCustomers, TruckRoute
 1: MaxSavings ← 0
 2: for (TruckRoute with consecutive $i, j, k$) do
 3:     if $j \in C'$ then
 4:         savings ← $\tau_{ij} + \tau_{jk} - \tau_{ik}$
 5:         if savings > MaxSavings then
 6:             MaxSavings ← savings
 7:             AddNode ← $j$
 8:         end if
 9:     end if
10: end for
11: Remove AddNode from TruckRoute.
12: Add an AddNode to DroneCustomers.
13: return DroneCustomers, TruckRoute
```

### *3.4 Drone routing problem*

In the DRP, we seek to determine from which vertex the drone should be launched and retrieved, given that the route of the truck and the delivery destination of the drone have been determined. Let the number of all vertices be $N$ and the number of vertices to be delivered to by the drone be $n$. If we solve the DRP by searching all vertices in an exhaustive search, the number of searches is

$$_{N-1}\mathrm{C}_{2n} \times n! \tag{1}$$

Therefore, we propose two methods for solving the DRP: one is to formulate the DRP and solve it using an integer programming solver; the other is to solve it using dynamic programming. Both methods are capable of calculating the exact solution to the DRP.

#### *3.4.1 Notation and mathematical formulation*

We next define the sets, parameters, and decision variables necessary for the formulation of the DRP. Let $c$ be the number of destinations to be delivered to; let 0 be the vertex when departing from the distribution centre and $c + 1$ be the vertex when returning to the distribution centre.

*Set*

- $C = \{1, 2, ..., c\}$: set of all customers.
- $N = \{0, 1, ..., c + 1\}$: set of all vertices in the network.
- $N_0 = \{0, 1, ..., c\}$: set of vertices from which a vehicle may depart.
- $N_1 = \{1, 2, ..., c + 1\}$: set of vertices which a vehicle may visit.
- $L$: set of all customers to be delivered to by the drone.

*Parameter*

- $\tau_{ij}, (i \in N_0, j \in N_1)$: time required for the truck to travel from vertex $i$ to vertex $j$.
- $\tau'_{ij}, (i \in N_0, j \in N_1)$: time required for the drone to travel from vertex $i$ to vertex $j$.
- $s_L$: time required by the truck driver to prepare the drone for launch.
- $s_R$: time required by the truck driver to recover the drone upon rendezvous.
- $x_{ij} \in \{0, 1\}, (i \in N_0, j \in N_1)$: 1, if the truck travels from vertex $i$ to vertex $j$.
- $p_{ij} \in \{0, 1\}, (i \in N_0, j \in N_1)$: 1, if customer $i$ is visited at some time before customer $j$ in the truck's path.
- $e$: maximum flight time for the drone.
- $M$: a value sufficiently greater than the delivery time.

*Decision variable*

- $t_i, (i \in N)$: time at which the truck arrives at vertex $i$.
- $y_{ilk} \in \{0, 1\}, (i \in N_0, l \in L, k \in N_1)$: 1, if the drone is launched from vertex $i$, travels to vertex $j$, and returns to the truck or the ending depot.

Based on this notation, we formulate the DRP as follows:

$$\text{Minimise } t_{n+1} \tag{2}$$

Subject to

$$\sum_{i \in N_0} \sum_{\substack{k \in N_1 \\ k \neq i}} y_{ilk} = 1, \qquad \forall l \in L, \tag{3}$$

$$\sum_{l \in L} \sum_{\substack{k \in N_1 \\ k \neq i}} y_{ilk} \leq 1, \qquad \forall i \in N_0, \tag{4}$$

$$\sum_{l \in L} \sum_{\substack{i \in N_0 \\ i \neq k}} y_{ilk} \leq 1, \qquad \forall k \in N_1, \tag{5}$$

$$y_{ilk} \leq p_{ik}, \qquad \forall l \in L, i \in N_0, k \in N_1, i \neq k, \tag{6}$$

$$p_{kl} \geq 1 - M\left(3 - \sum_{j \in L} y_{ijk} - \sum_{m \in L} \sum_{\substack{n \in N_1 \\ n \neq i,k,l}} y_{lmn} - p_{il}\right), \quad \forall i \in N_0, k \in N_1, l \in C, i \neq k, l \neq i, k, \tag{7}$$

$$t_k \geq t_h + \tau_{hk} + M(1 - x_{hk}) + s_l \sum_{l \in L} \sum_{\substack{m \in N_1 \\ m \neq h}} y_{hlm} + s_r \sum_{l \in L} \sum_{\substack{i \in N_0 \\ i \neq k}} y_{ilk}, \quad \forall h \in N_0, k \in N_1, h \neq k, \tag{8}$$

$$t_l \geq t_i + \tau'_{il} + s_l - M(1 - y_{ilk}), \qquad \forall l \in L, i \in N_0, k \in N_1, i \neq k, \tag{9}$$

$$t_k \geq t_l + \tau'_{lk} + s_r - M(1 - y_{ilk}), \qquad \forall l \in L, i \in N_0, k \in N_1, i \neq k, \tag{10}$$

$$t_k - t_i \leq e + M(1 - y_{ilk}), \qquad \forall l \in L, i \in N_0, k \in N_1, i \neq k, \tag{11}$$

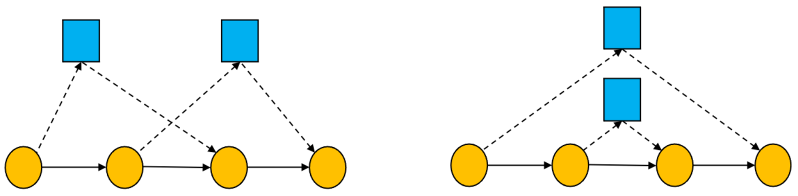
$$t_0 = 0, \tag{12}$$

$$t_i \geq 0, \qquad \forall i \in N_1. \tag{13}$$

Equation (2) serves as the objective function. The goal is to deliver the packages to all the destinations and minimise the time until the truck and drone return to the distribution centre.

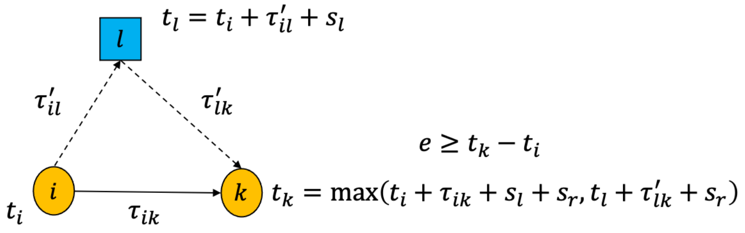
Equations (3) through (7) are constraints on $y$. Constraint (3) requires that all destinations assigned to the drone are delivered to by the drone. Constraint (4) ensures that the drone can depart only once from a vertex; constraint (5) ensures that the drone can only be retrieved once at a vertex. Constraint (6) imposes the condition that the drone cannot fly in a direction opposite to that of the truck. Constraint (7) prevents the drone from moving to another delivery destination during delivery, as shown in Figure 4.

**Figure 4** Example violation of the constraint (7) (see online version for colours)



Equations (8) through (13) are constraints on time $t$. An overview of these constraints is shown in Figure 5. Constraints (8) through (10) determine the arrival times of the drone and truck at each vertex. Constraint (8) represents the time at which the truck arrives at each vertex, and adds $s_l$, the drone's preparation time for departure if the drone is to depart from that vertex, and adds $s_r$, the drone's recovery time if the drone is to be recovered at that vertex. Constraints (9) and (10) represent the time at which the drone arrives at each vertex, where constraint (9) represents the time at which the drone arrives at the vertex to be delivered to and constraint (10) represents the time at the vertex at which the drone is collected by the truck. Constraint (11) is a constraint on the maximum flight time of the drone. Finally, constraint (12) represents the time at which the drone departs the distribution centre, and constraint (13) requires the time to be non-negative.

**Figure 5** Example of the constraint (8) to (13) (see online version for colours)


$t_l = t_i + \tau'_{il} + s_l$
$l$
$\tau'_{il}$
$\tau'_{lk}$
$e \geq t_k - t_i$
$i$
$k$
$t_k = \max(t_i + \tau_{ik} + s_l + s_r, t_l + \tau'_{lk} + s_r)$
$t_i$
$\tau_{ik}$

### *3.4.2 Introduction of an adjacency list*

In order to reduce the number of unnecessary constraints in the formulation of the DRP, we introduce what we call the adjacency list. We will first describe the essential property of the DRP; we then reduce the number of constraints and the computational cost by constructing an adjacency list using this property.

Theorem 3.1 always holds for the vertices of the drone departure and recovery points determined as the optimal solution of the DRP. In other words, by narrowing the list of candidates in advance and retaining them in an adjacency list rather than searching all the drone departure and recovery points, we can reduce the number of constraints and thus reduce the computational cost.

*Theorem 3.1:* Let $i$ be the delivery vertex of the drone and $j$ be the vertex closest to $i$ among the delivery vertices of the truck. Also, let the maximum flight time of the drone be $e$. In this case, the following relation holds between the departure vertex $l$ and the recovery vertex $c$ of the drone:

$$e - \tau'_{ij} \geq \tau'_{il} \tag{14}$$

$$e - \tau'_{ij} \geq \tau'_{ic} \tag{15}$$

*Proof:* The time it takes for the drone to depart the truck, deliver the parcel to the determined vertex, and be recovered must always be within the maximum flight time of the drone, thus satisfying equation (16).

$$e \geq \tau'_{il} + \tau'_{ic} \tag{16}$$

Also, vertex $j$ is the closest vertex from vertex $i$, so equation (17) and equation (18) are satisfied.

$$\tau'_{ij} \leq \tau'_{il} \tag{17}$$

$$\tau'_{ij} \leq \tau'_{ic} \tag{18}$$

From the above, equations (14) and (15) hold. □

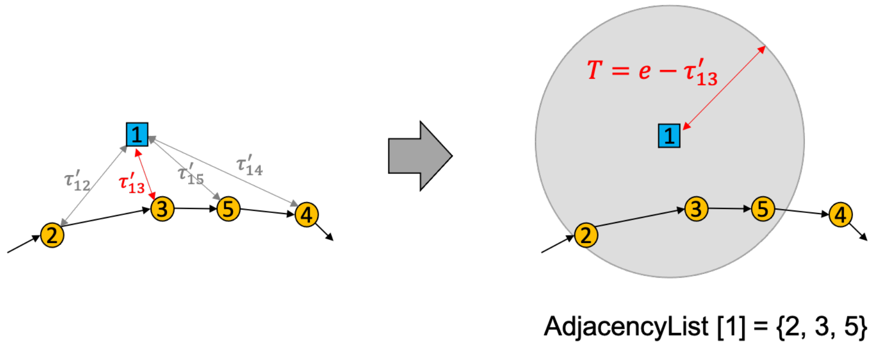
The method for creating the adjacency list is described in Algorithm 2; let $l$ be the delivery vertex of the drone. Search for the vertex closest to the vertex $l$ among the delivery vertices of the truck. Let this vertex be $j$. Let $T$ be the value obtained by taking the difference between the maximum flight time $e$ and the travel time $\tau'_{lj}$ from vertex $l$ to vertex $j$ by the drone. Add the vertices that can be moved within $T$ from the delivery vertex $l$ of the drone to the adjacency list. This is done for all the drone delivery vertices.

**Algorithm 2** Create an adjacency list

```
Input: DroneCustomers, TruckCustomers, $\tau'$, $e$
Output: AdjacencyList
 1: Create AdjacencyList, an empty dictionary.
 2: for $l \in$ DroneCustomers do
 3:     $m \leftarrow \infty$
 4:     for $j \in$TruckCustomers do
 5:         if $\tau'_{lj} < m$ then
 6:             $m \leftarrow \tau'_{lj}$
 7:         end if
 8:     end for
 9:     $T = e - m$
10:     for $j \in$TruckCustomers do
11:         if $\tau'_{lj} \leq T$ then
12:             Add $j$ to AdjacencyList[$l$].
13:         end if
14:     end for
15: end for
16: return AdjacencyList
```

A concrete example is shown in Figure 6. Vertex 1 is the vertex to be delivered to by the drone. The nearest vertex among the delivery vertices of the truck is 3. Therefore, we calculate $T = e - \tau'_{13}$. We then add $\{2, 3, 5\}$, the vertices that can be moved within $T$ from vertex 1, to AdjacencyList[1].

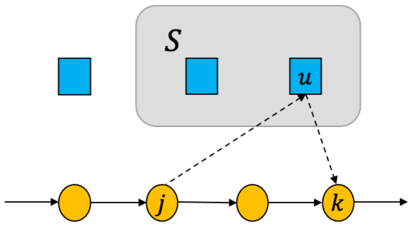
**Figure 6** Example of creating an adjacency list (see online version for colours)



## 3.4.3 *Solution by dynamic programming*

In order to efficiently search for the departure and recovery vertices of the drone, we use a dynamic programming method. This dynamic programming method is an application of the dynamic programming approach to the TSP (Held and Karp, 1962; Bellman, 1962). In this section, we introduce the transition equation and computational complexity of the dynamic programming method.

**Figure 7** Explanation of variables in dynamic programming (see online version for colours)



We first describe the variables: let $S$ be a subset of the set of delivery destinations for the drone, $u$ be an element in $S$, and $k$ be a vertex at the end of the truck's delivery path. Let $j$ denote some vertex up to $k$. An explanation of the above variables is summarised in Figure 7. Here, $dp(S, k)$ represents the minimum delivery time to vertex $k$ when subset $S$ is delivered to $k$, the last point of the truck. The transition equations for $dp(S, k)$ can be expressed in equations (19) and (20). By using these transition equations to find $dp(L,\mathrm{DP})$, we obtain the optimal value.

$$dp(S, k) = \min(dp(S, k-1) + \tau_{k-1,k}, X) \tag{19}$$

$$X = \min_{\substack{u \in S \\ j=1,\ldots,k-1}} (dp(S - \{u\}, j) + \max(dp(\emptyset, k) - dp(\emptyset, j), \tau'_{ju} + \tau'_{uk}) \tag{20}$$

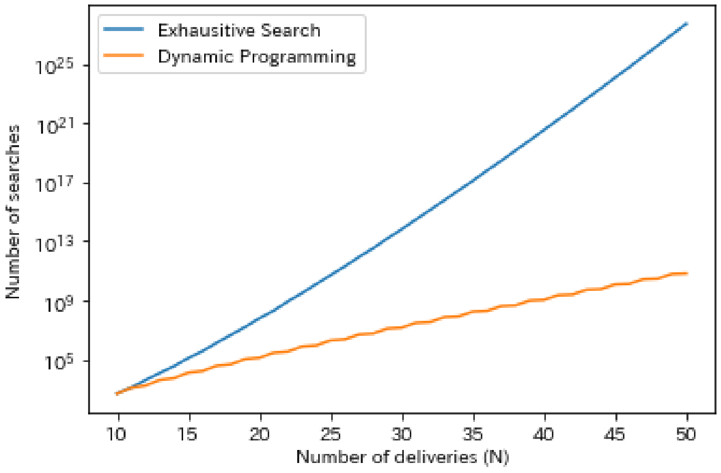
Equations (19) and (20) are explained below. The first part represents the minimum delivery time when vertex $k$ is not the collection point of the drone; the second part represents the minimum delivery time when vertex $k$ is the collection point of the drone. Here, $dp(S, k-1) + \tau_{k-1,k}$, represents the minimum delivery time $dp(S, k-1)$ when all $S$ deliveries have been completed and packages have been collected through $k-1$,.] plus the travel time $\tau_{k-1,k}$ of the truck from vertex $k-1$ to $k$. In the second part, we calculate the delivery time when an element $u$ in $S$ is the starting point $j$ and the collection point $k$; then, by replacing $j$ and $u$, we calculate the minimum delivery time when the collection point of the drone always includes $k$. The $dp(\emptyset, k)$ represents the delivery time of the truck only.

The computational complexity for our dynamic programming solution is shown in equation (21), where $n$ is the number of drone deliveries and $N$ is the total number vertices, including the delivery centre.

$$O(2^n \times n \times (N-n)^2) \tag{21}$$

Figure 8 shows a comparison of the number of searches when using the dynamic programming method and when using an exhaustive search. The horizontal axis represents the number of deliveries, and the vertical axis represents the number of searches. The blue line indicates the exhaustive search results; the orange line indicates the results for our dynamic programming approach. The number of searches used by the exhaustive search method is calculated according to equation (1). Figure 8 shows that as the number of deliveries increases, the difference in the number of searches for the two methods becomes larger. Thus, the computational cost is greatly reduced by using the dynamic programming method.

**Figure 8** Comparison of the number of searches using exhaustive search and dynamic programming (see online version for colours)



## 4 Parallel drone scheduling travelling salesman problem

In the PDSTSP model defined by Murray and Chu (2015), the objective is to minimise delivery time by using a drone to deliver to destinations near the distribution centre and a truck to deliver to other destinations. In other words, the PDSTP is an assignment problem that divides vertices into those to be delivered to by drone and those to be delivered to by truck.

In this section, we describe the PDSTSP problem in this study and propose a new algorithm for solving the problem.

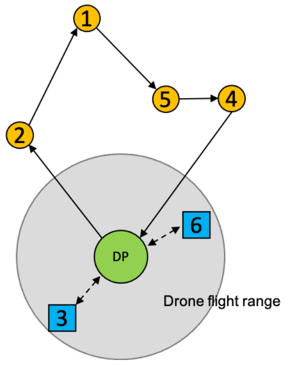
### *4.1 Problem setting*

The PDTSP setting is shown in Figure 2. The numbered vertices represent delivery destinations. The distribution centre is labelled DP. The round vertices are the delivery destinations of the truck; the square vertices are the delivery destinations of the drone. The solid arrow indicates the delivery route of the truck; the dotted arrow is the delivery route of the drone. The grey circle shows the drone's range of movement.

The PDTSP model differs from the FSTSP model in that drone departures and collections are always at the distribution centre. That is, the drone departs from the distribution centre, delivers a parcel to a certain vertex, then returns to the distribution centre. This process is used repeatedly to deliver the parcels. Meanwhile, the truck continues to deliver parcels to its designated destinations as before. The time taken by the truck and the drone to deliver the parcels to the specified vertices and return to the distribution centre is defined as the delivery time, and the assignment that minimises this delivery time is determined.

In Figure 9, the truck moves in the order of DP, 2, 1, 5, 4, DP. In this case, the drone moves along the path DP, 3, DP, 6, DP. The delivery time for the truck and that for the drone are calculated, and the larger of the two is taken as the overall delivery time.

**Figure 9** An example of a PDSTSP solution (see online version for colours)

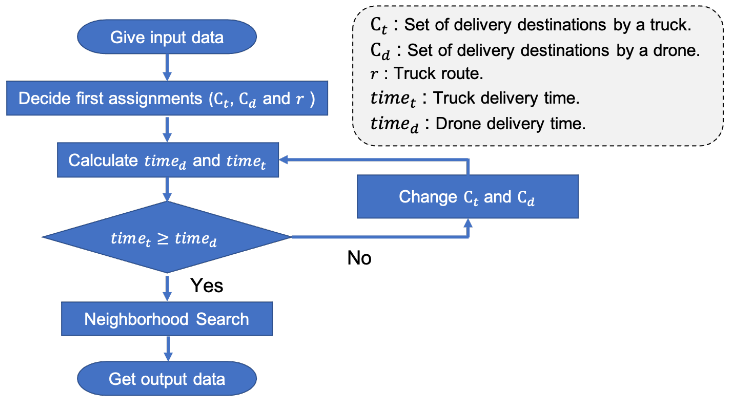


In Murray and Chu (2015), the problem was defined with multiple drones. However, since the purpose of this study is to compare with the FSTSP, we added the assumption that there is only one drone in order to match the conditions of the FSTSP.

## *4.2 Overview of the proposed method*

The basic idea of the proposed method is to consider all the vertices that can be delivered to by the drone, and then change the drone delivery destinations to truck delivery destinations until the conditions are satisfied, and thus produce the optimal solution.

**Figure 10** Flow of the proposed method for the PDSTSP (see online version for colours)

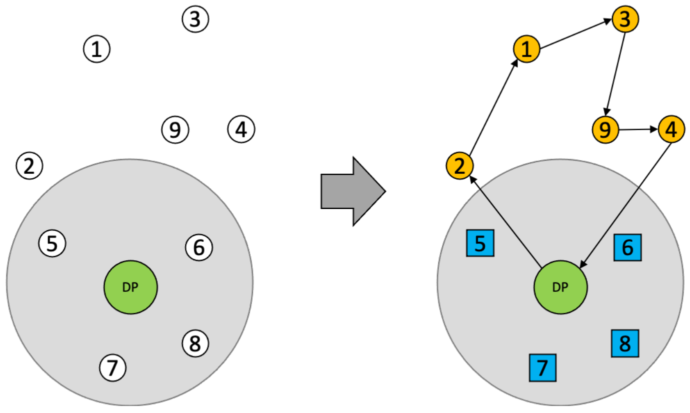

Give input data
Decide first assignments ($C_t$, $C_d$ and $r$ )
Calculate $time_d$ and $time_t$
$time_t \geq time_d$
No
Change $C_t$ and $C_d$
Yes
Neighborhood Search
Get output data
$C_t$ : Set of delivery destinations by a truck.
$C_d$ : Set of delivery destinations by a drone.
$r$ : Truck route.
$time_t$ : Truck delivery time.
$time_d$ : Drone delivery time.

The flow of the proposed method is shown in Figure 10. Let $C_t$ be the set of vertices to be delivered to by the truck and $C_d$ be the set of vertices to be delivered to by the drone. Further, let $r$ be the delivery route of the truck and $time_t$ be the delivery time of the truck; $time_d$ is the delivery time of the drone. The method proceeds as follows: first, the input is provided. The initial solution $(C_t, C_d, r)$ is then determined from the input information. Next, from this initial solution, the delivery times $time_t, time_d$ are obtained. If $time_t$ is greater than or equal to $time_d$, we proceed to the next step; otherwise, the assignment is changed and the delivery time is recalculated. Finally, if the previous conditions are satisfied, a neighbourhood search is performed and the solution is output.

## 4.3 Generating the initial solution

Figure 11 can be used to explain how the initial solution is generated. The basic idea is to first deliver to all the vertices that can be delivered to by drone, with the rest being delivered to by truck. That is, if the drone departs from the distribution centre, delivers the parcel to a defined vertex, and returns to the distribution centre within the maximum flight time, it is assigned to $C_d$, the set of drone delivery destinations. Then, the vertices that do not satisfy this condition are assigned to $C_t$, the set of truck delivery destinations. In Figure 11, the grey area is the area within which the drone can travel, and vertices 5, 6, 7, and 8 that fall within this area are assigned to the drone's delivery destination set $C_d$. The remaining vertices 1, 2, 3, 4, and 9 are assigned to the set of truck delivery destinations $C_t$. Then, by solving TSP for $C_t$ and the delivery centre, we obtain the delivery route $r$ for the truck.

**Figure 11** Example of how to generate an initial solution (see online version for colours)

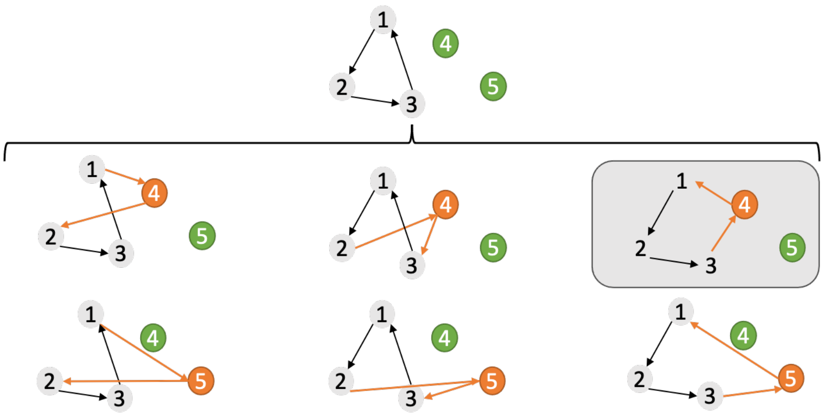


## 4.4 Change assignment

We applied the cheapest insertion algorithm (Rosenkrantz et al., 1977; Goetschalckx, 2011) to assign one element in the set of drone delivery destinations to the set of truck delivery destinations. This has been proven to produce a 2-approximate solution (Rosenkrantz et al., 1977). The algorithm calculates the additional cost of adding the remaining vertices from the partial circuit and adds the vertex with the lowest additional cost to the partial circuit. The computational complexity of solving the TSP with $N$ vertices using this method is $O(N^2 \log_2 N)$ (Rosenkrantz et al., 1977).

The reason for using the cheapest insertion algorithm to change the PDSTSP assignment in this study is that we believe that it is more efficient to deliver by truck for destinations close to the route of the truck. In other words, the drone delivery destinations closest to the existing truck route are added to the truck route. By doing so, we expect that the truck delivery time will not increase much and the drone delivery time will decrease significantly.

**Figure 12** Example of cheapest insertion algorithm (see online version for colours)



**Algorithm 3** Cheapest insertion algorithm

```
Input: $time_t$, $time_d$, DroneCustomers, TruckCustomers, TruckRoute
Output: $time_t$, $time_d$, DroneCustomers, TruckCustomers, TruckRoute
 1: MinTime $\leftarrow \infty$
 2: for $i \in$DroneCustomers do
 3:     for $j \in$ TruckRoute do
 4:         SaveTime $\leftarrow$ Delivery time when truck delivery route excludes vertex $j$.
 5:         if SaveTime $<$ MinTime then
 6:             SaveMinTime $\leftarrow$MinTime
 7:             $v \leftarrow i$
 8:             $o \leftarrow j$
 9:         end if
10:     end for
11: end for
12: Remove $v$ from DroneCustomers.
13: Add $v$ to TruckCustomers.
14: Add $v$ to the $o$-th TruckRoute.
15: $time_d \leftarrow$ DroneTime $-(\tau'_{0v} + \tau'_{v0})$
16: $time_t \leftarrow$ MinTime
17: return $time_t$, $time_d$, DroneCustomers, TruckCustomers, TruckRoute
```

Figure 12 illustrates this in detail. The current truck path is 1, 2, 3, 1, and the candidate vertices to add are 4 and 5. We calculate the cost of adding 4 and 5 between the truck paths 1, 2, 3, and 1, respectively. The route with the lowest additional cost, 1, 2, 3, 4, 1, is the next truck delivery route.

The pseudo-code of the program is shown in Algorithm 3. The computation time of the cheapest insertion algorithm is $O(NM^2)$ when the total number of deliveries is $N$ and the number of vertices that can be delivered by the drone (i.e., the number of elements in the initial solution $C_d$) is $M$. It is possible to reduce the computational complexity to $O(M^2 \log N)$ if the distance between vertices is managed by a priority queue.

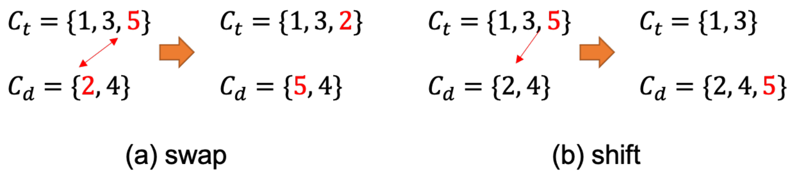
## *4.5 Neighbourhood search*

In this study, we used a nearest neighbour search to improve the accuracy of the solution. We attempted to improve the accuracy of the solution by using both a swap neighbourhood and a shift neighbourhood approach.

Swap neighbourhood is a method for updating a temporary solution by swapping elements of one set with elements of another set. Here, one element is selected from each set of truck delivery destinations $C_t$ and drone delivery destinations $C_d$; the delivery time is calculated when the elements are exchanged and updated if the solution is improved. A concrete example is shown in Figure 13(a).

Shift neighbourhood is a method for updating a temporary solution by performing an operation that changes an element of one set to an element of another set. In this study, an element is selected from $C_t$, the set of truck delivery destinations, and the delivery time is calculated when the vertex is assigned to $C_d$, the drone delivery destination, and updated if the solution is improved. A concrete example is shown in Figure 13(b).

**Figure 13** Examples of swap and shift neighbourhood (see online version for colours)



## *4.6 Computational complexity*

Determining the computational complexity of the proposed method is fairly straightforward. Let $N$ be the total number of deliveries and $M$ be the number of deliveries that can be made by the drone. Since the TSP must be solved to generate the initial solution, the solution method of the TSP determines the computational complexity. If an exhaustive search is used, $O(n!)$ of computation is required. Here, the cheapest insertion algorithm is used to change assignments, and its computational complexity is $O(NM)$. Since the maximum number of assignment changes is $M$, the total computational complexity of the delivery time calculation and assignment changes is $O(NM^2)$. Neighbourhood search also has a computational complexity of $O(NM^2)$

since the cheapest insertion algorithm is applied. From the above, if the computational complexity of TSP is $O(\text{TSP})$, the overall computational complexity of the proposed method is $O(\text{TSP} + NM^2)$.

## 5 Computational experiments

In this section, we describe the numerical experiments used to evaluate the FSTSP, the PDSTSP, and the proposed delivery method.

### *5.1 Experimental conditions*

In the datasets created for the numerical experiments, the number of deliveries was set from 10 to 40, and the deliveries were restricted to a 40 km $\times$ 40 km area. The coordinates of the distribution centre and the delivery destinations were randomly determine. Since the path of the drone is through the air, the distance travelled by the drone was calculated as a Euclidean distance. In contrast, the distance travelled by the truck was calculated as a Manhattan distance. This means that the drone's travel distance would be less than or equal to the truck's travel distance. Since a drone is generally faster than a truck, the speeds of the drone and truck were set to 50 km/h and 40 km/h, respectively. The preparation time for starting the drone and the recovery time for retrieving the drone were both set to 1 minute, and the maximum flight time of the drone was set to 30 minutes.

All experiments were conducted on a computer with a 3.7 GHz 6-core Intel Core i5 processor and 16 GB memory. The integer programming solver was IBM ILOG CPLEX Optimization Studio 12.9.0 (CPLEX, online) and Gurobi Optimizer 9.5.0 (GUROBI, online). The programs were written in Python 3.7.9.

To indicate how close a solution was to the optimal solution, we used equation (22) to calculate a measure that we called GAP, based on the results of the numerical experiments.

$$\text{GAP}(\%) = \frac{\text{Difference from optimal solution}}{\text{Optimal solution}} \times 100 \tag{22}$$

The closer the GAP value is to 0, the closer the solution is to the optimal solution.

The reduction ratio of the TSP, which serves as an indicator of how much the delivery time improves from the truck-only delivery time, was calculated according to equation (23):

$$\text{Reduction ratio of TSP}(\%) = \frac{\text{Difference from TSP solution}}{\text{TSP solution}} \times 100 \tag{23}$$

Here, larger values correspond to shorter delivery times.

Determining the truck path (TSP) required to generate the initial solutions for the FSTSP and PDSTSP was accomplished by solving the formulation with an integer programming solver. The time limit was set to 600 seconds.

## *5.2 FSTSP numerical experiments*

In the numerical experiments involving the SFTSP, the number of deliveries was set to either 10, 15, 20, 25, 30, 35, or 40. Thirty sets of test data were generated for each of these seven cases.

Table 3 shows the solution performance and computation time for Murray and Chu (2015), for the proposed method, and for the integer programming solver for the case of 10 deliveries. Although the integer programming solver was able to produce the optimal solution, with a GAP value of 0, it required a substantial amount of computation time, having an average computation time of 9,401.13 seconds. In comparison, Murray's method yielded a solution in an average computation time of only 0.28 seconds; however, the average GAP was a rather poor 9.00%. On the other hand, the proposed TSP+DRP algorithm required less computation time and had a smaller average GAP (3.72%) than the conventional method, resulting in a better solution in a shorter time.

**Table 3** Comparison of optimal solution and runtime for 10 deliveries

| | *GAP (%)* | | | *Runtime (s)* | | |
|---|---|---|---|---|---|---|
| | *Avg* | *Min* | *Max* | *Avg* | *Min* | *Max* |
| Murray and Chu | 9.00 | 0.00 | 29.44 | 0.28 | 0.25 | 0.52 |
| TSP+DRP | 3.72 | 0.00 | 17.02 | 0.39 | 0.16 | 0.74 |
| Solver | 0.00 | 0.00 | 0.00 | 9,401.13 | 3,798.49 | 65,290.25 |

Figure 14 shows the reduction ratio of the TSP for each number of deliveries. Blue indicates the method used by Murray and Chu; orange indicates the proposed method. The results show that the proposed method has a larger reduction ratio for any number of deliveries from 10 to 40, and performs better than the Murray and Chu method.

**Figure 14** Comparison of solution accuracy between conventional and proposed methods for the number of deliveries (see online version for colours)

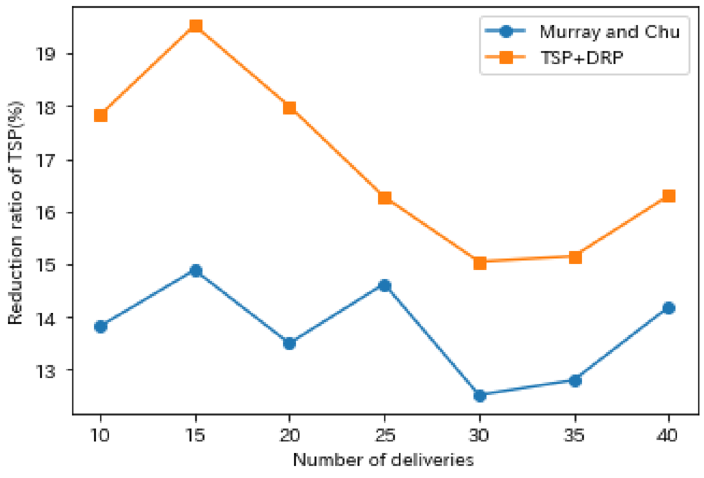


**Figure 15** Comparison of runtime between conventional and proposed methods for various numbers of deliveries (see online version for colours)

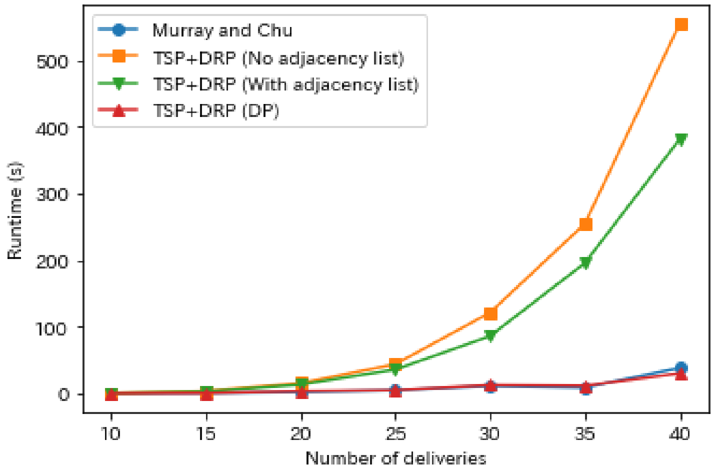


Figure 15 shows the computation time for the various numbers of deliveries. Here, blue indicates the Murray and Chu method, orange indicates the DRP solved with an integer programming solver, green represents the DRP solved with an integer programming solver when there is an adjacency list, and red represents the DRP solved with the proposed method using dynamic programming. The results show that solving the DRP with an integer programming solver requires more computation time as the number of deliveries increases. On the other hand, when an adjacency list is added, the computation time is reduced compared to that without an adjacency list, and when dynamic programming is used for solving the DRP, the computation time is significantly reduced compared to other proposed methods. There was also little difference when compared to the method of Murray and Chu. However, the computational complexity increases enormously when the number of deliveries is further increased from equation (21); thus, more computation time is expected to be required for cases larger than those considered in this study.

## *5.3 PDSTSP numerical experiments*

In this experiment, 30 test datasets were created for the cases of 10, 20, and 30 deliveries, respectively.

Table 4 compares the GAP and computation time for each number of deliveries using the Murray and Chu method, the proposed method, and the integer programming solver. The values in the table represent averages for the 30 test datasets. The Murray and Chu method involves formulating the TSP and solving it with an integer programming solver, while the nearest neighbour (NN) method solves the TSP with a nearest neighbour approach, one of the greedy solution methods (Rosenkrantz et al., 1977).

**Table 4** Comparison of GAP and computation time by number of deliveries

| *Number of deliveries* | *10 customers* | | *20 customers* | | *30 customers* | |
|---|---|---|---|---|---|---|
| | *GAP (%)* | *Runtime (s)* | *GAP (%)* | *Runtime (s)* | *GAP (%)* | *Runtime (s)* |
| Murray and Chu (formulation) | 6.66 | 0.39 | 0.97 | 154.57 | 4.16 | 13,275.52 |
| Murray and Chu (NN) | 8.37 | 0.01 | 5.01 | 6.76 | 8.50 | 147.61 |
| Proposed method | 1.81 | 0.01 | 2.28 | 10.14 | 6.54 | 154.01 |
| Solver | 0.00 | 0.14 | 0.00 | 128.01 | 0.00 | 1502.23 |

The integer programming solver is able to produce the optimal solution, but it requires a substantial computation time when the number of cases is 30. The conventional formulation of the method has a smaller GAP than the other methods. In other words, the accuracy of the solution is good. However, the average computation time for 30 cases was 13,275.52 seconds, which is more than the time for the integer programming solver. This is due to the fact that the TSP is solved multiple times. The conventional nearest neighbour method requires less computation time, but has a larger GAP value than the other methods. On the other hand, the average computation time of the proposed method is smaller and the GAP is better than that of the nearest neighbour method.

### 5.4 Comparison of delivery methods

Figure 16 shows the relationship between the delivery range and the reduction ratio of the TSP when the number of deliveries is set to 30. The blue line represents the FSTSP and the orange line represents the PDSTSP. The PDSTSP has a larger reduction rate when the delivery range is up to 25 km. In other words, the delivery time is smaller. However, the PDSTSP shows a sharp decrease in the rate of decrease from the point where the delivery range is 30 km, and the delivery time is larger than that of the FSTSP. The larger the delivery range, the greater the distance between delivery points, which is thought to be due to the decrease in the number of vertices that can be delivered to by the drone. Similarly, for the PDSTSP, the decrease in the number of delivery destinations near the distribution centre is thought to have resulted in a smaller decrease rate relative to the TSP solution.

From the results of Figure 16, it appears that the number of destinations near the distribution centre has a significant impact on the delivery time in the PDSTSP. Accordingly, we tested how the delivery time varied with the percentage of destinations near the delivery centre when the delivery range is set to 30 km. The results are shown in Figure 17. The blue line represents the FSTSP; the orange line represents the PDSTSP. As shown, the FSTSP is not significantly affected by the number of destinations near the distribution centre, with a decrease rate ranging from 12.5 to 17.5%. On the other hand, for the PDSTSP, the percentage of destinations near the distribution centre increased monotonically up to approximately 0.4, and converged after 0.4, resulting in a greater rate of decrease than for the FSTSP. From the above, in terms of the number of delivery destinations near the distribution centre, the FSTSP is superior where the percentage of delivery destinations is smaller than 0.4, and PDSTSP is superior, on average, after 0.4.

**Figure 16** Relationship between delivery range and percentage decrease in delivery time (see online version for colours)

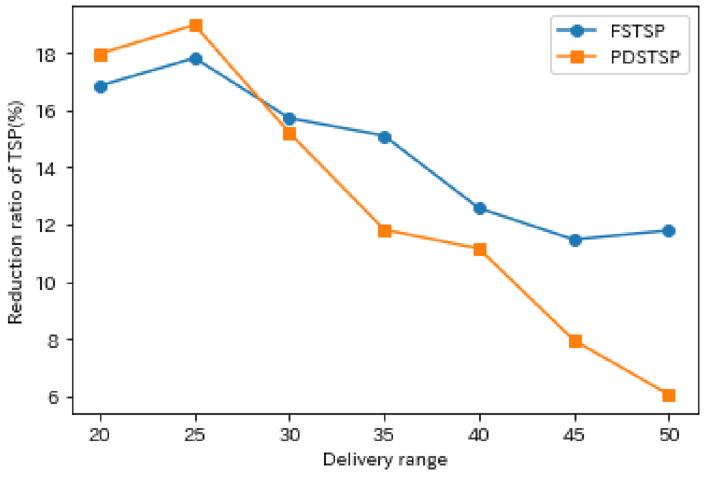
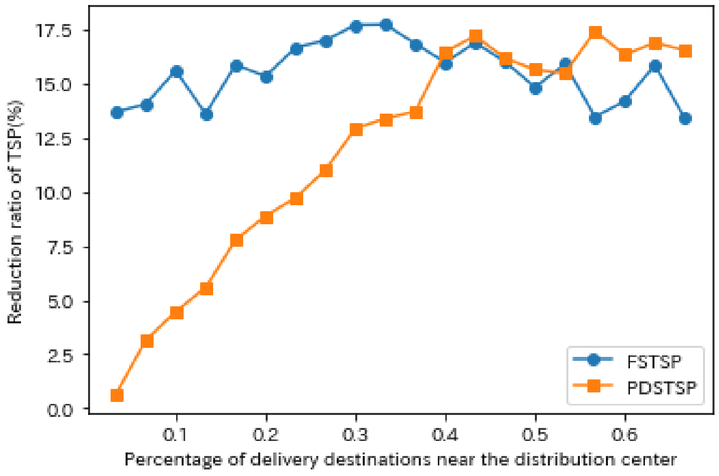


**Figure 17** Percentage of delivery destinations near the distribution centre vs. percentage decrease in delivery time (see online version for colours)



## 6 Conclusions

This study examined the FSTSP and PDSTSP, two delivery planning problems involving a drone and a truck, and sought to determine which delivery method was capable of delivering parcels more efficiently. The same problems have been investigated in previous studies, but the potentially large accuracy errors associated with the proposed solution methods have been acknowledged as a serious limitation. Since obtaining an exact solution has a high computational cost, a solution method that yields a good solution in a reasonable amount of time is needed.

We proposed a new heuristic for solving the FSTSP and PDSTSP and conducted a series of numerical experiments involving one drone and one truck and compared our results with previous studies. We found that our proposed algorithm is computationally inexpensive and achieves better accuracy than the conventional methods.

A comparison in terms of the ratio (percentage) of delivery destinations near the distribution centre confirmed that the FSTSP had shorter delivery times when the ratio was less than 0.4 (i.e., less than 40%), but that the PDSTSP had shorter delivery times when the ratio was greater than 0.4. However, it is possible that other factors, such as the location of the delivery centre, the speed of the drone, and the maximum flight distance, may also affect the results. To investigate these aspects, additional experiments are needed.

## Acknowledgements

This work was supported by JSPS Grants-in-Aid for Scientific Research, Grant Number 20K04962.

## References

Agatz, N., Bouman, P. and Schmidt, M. (2018) 'Optimization approaches for the traveling salesman problem with drone', *Transportation Science*, Vol. 52, No. 4, pp.965–981.

Bellman, R. (1962) 'Dynamic programming treatment of the travelling salesman problem', *Journal of the ACM*, Vol. 9, No. 1, pp.61–63.

Bouman, P., Agatz, N. and Schmidt, M. (2018) 'Dynamic programming approaches for the traveling salesman problem with drone', *Networks*, Vol. 72, No. 4, pp.528–542.

Carlsson, J.G. and Song, S. (2018) 'Coordinated logistics with a truck and a drone', *Management Science*, Vol. 64, No. 9, pp.4052–4069.

CPLEX [online] https://www.ibm.com/jp-ja/analytics/cplex-optimizer (accessed 13 March 2022).

Chung, S.H., Sah, B. and Lee, J. (2020) 'Optimization for drone and drone-truck combined operations: a review of the state of the art and future directions', *Computers & Operations Research*, Vol. 123, p.105004 [online] https://doi.org/10.1016/j.cor.2020.105004.

Goetschalckx, M. (2011) *Supply Chain Engineering*, Vol. 161, Springer Science & Business Media, Boston.

Goodchild, A. and Toy, J. (2018) 'Delivery by drone: an evaluation of unmanned aerial vehicle technology in reducing $CO_2$ emissions in the delivery service industry', *Transportation Research Part D: Transport and Environment*, Vol. 61, pp.58–67 [online] https://doi.org/10.1016/j.trd.2017.02.017.

GUROBI [online] https://www.gurobi.com (accessed 13 March 2022).

Ha, Q.M., Deville, Y., Pham, Q.D., and Hà, M.H. (2018) 'On the min-cost traveling salesman problem with drone', *Transportation Research Part C: Emerging Technologies*, Vol. 86, pp.597–621 [online] https://doi.org/10.1016/j.trc.2017.11.015.

Ha, Q.M., Deville, Y., Pham, Q.D. and Hà, M.H. (2020) 'A hybrid genetic algorithm for the traveling salesman problem with drone', *Journal of Heuristics*, Vol. 26, No. 2, pp.219–247.

Held, M. and Karp, R.M. (1962) 'A dynamic programming approach to sequencing problems', *Journal of the Society for Industrial and Applied mathematics*, Vol. 10, No. 1, pp.196–210.

Kundu, A. and Matis, T.I. (2017) ‘A delivery time reduction heuristic using drones under windy conditions’, *Proceedings of 67th Annual Conference and Expo of the Institute of Industrial Engineering 2017*, Institute of Industrial Engineers, Pittsburgh, USA, pp.1864–1869.

Ministry of Land, Infrastructure, Transport and Tourism (2020) *Survey and Tabulation Method of the Number of Parcel Deliveries* [online] https://www.mlit.go.jp/report/press/content/001418260.pdf (accessed 25 February 2022).

Murray, C. C., and Chu, A. G. (2015) ’The flying sidekick traveling salesman problem: optimization of drone-assisted parcel delivery’, *Transportation Research Part C: Emerging Technologies*, Vol. 54, pp.86–109 [online] https://doi.org/10.1016/j.trc.2015.03.005.

Poikonen, S., Wang, X. and Golden, B. (2017) ‘The vehicle routing problem with drones: extended models and connections’, *Networks*, Vol. 70, No. 1, pp.34–43.

Ponza, A. (2016) *Optimization of Drone-assisted Parcel Delivery*, Master’s thesis, University of Padova, Italy.

Rosenkrantz, D.J., Stearns, R.E. and Lewis II, P.M. (1977) ‘An analysis of several heuristics for the traveling salesman problem’, *SIAM Journal on Computing*, Vol. 6, No. 3, pp.563–581.

Wang, X., Poikonen, S. and Golden, B. (2017) ‘The vehicle routing problem with drones: several worst-case results’, *Optimization Letters*, Vol. 11, No. 4, pp.679–697.

Yurek, E.E. and Ozmutlu H.C. (2018) ‘A decomposition-based iterative optimization algorithm for traveling salesman problem with drone’, *Transportation Research Part C: Emerging Technologies*, Vol. 91, pp.249–262 [online] https://doi.org/10.1016/j.trc.2018.04.009.