Bulletin of the JSME Vol.15, No.6, 2021
Journal of Advanced Mechanical Design, Systems, and Manufacturing

Genetic algorithm for scheduling of parcel delivery by drones

Yohei HAZAMA*, Hitoshi IIMA*, Yoshiyuki KARUNO* and Kosuke MISHIMA*
*Kyoto Institute of Technology
Matsugasaki, Sakyo-ku, Kyoto 606-8585, Japan
E-mail: iima@kit.ac.jp

Received: 15 April 2021; Revised: 30 August 2021; Accepted: 30 September 2021

Abstract
In recent years, efficient logistics has become indispensable, and using unmanned aerial vehicles (UAVs) or drones is promising for considerably reducing the cost and time required for parcel delivery. This paper addresses a parcel delivery scheduling problem. In this problem, a truck loaded with drones and parcels leaves a distribution center and stops at some points on a fixed route. At each point, the drones take off and deliver parcels to customers. We define this problem as finding the assignment of customers to both the drones and their takeoff points. Then, we propose a genetic algorithm (GA) for finding a near-optimal solution in a short time. In the proposed GA, a solution is represented using sets of customers assigned to the takeoff points, and a heuristic rule determines the assignment to the drones. The crossover operation enables offspring to inherit the customer sets. Experimental results show that the proposed GA can successfully find an optimal or a near-optimal solution faster than an integer programming solver for almost all instances. In addition, it significantly outperforms other GAs using a different crossover.

Keywords : Engineering optimization, Scheduling, Delivery, Genetic algorithm, Grouping

1. Introduction

The development of electronic commerce increases the importance of efficient logistics. Unmanned aerial vehicles (UAVs) or drones are fast, light, and reasonable and do not require pilots and a crew. Therefore, they can significantly reduce the cost and time required for delivery. Many companies such as Amazon (Amazon.com, Inc., 2016), Google (Stewart, 2014), and DHL (DHL International GmbH, 2014) have announced that they will use drones for delivery. However, the drones have limitations on the weight and volume of cargo, and they can deliver only light and small cargoes like parcels. In addition, their battery capacity is limited, and their flight time is short, so it is difficult to deliver to customers far from a distribution center. It is accepted that in order to cover the shortcomings, the drones are used in combination with a truck. The truck is useful for mass transportation, and the drones are effective in mobility and economy (Chen et al., 2018). For example, Mathew et al. (2015) address the following delivery system. A truck carries parcels and drones over long-distance transportation, and the drones deliver the parcels from the truck to customers. As a result, the overall delivery time and cost can be reduced. For recent years, intensive studies have been conducted on scheduling of parcel delivery by drones. These studies have various problem settings, and they are broadly classified into two settings: only one drone delivers parcels (Murray and Chu, 2015; Othman et al., 2017; Agatz et al., 2018) and multiple drones do so (Kim and Moon, 2019; Karuno and Mishima, 2019).

First, in (Murray and Chu, 2015; Othman et al., 2017; Agatz et al., 2018), only one drone delivers parcels. Murray and Chu (2015) address two models based on a traveling salesman problem (TSP) (Laporte, 1992): the flying sidekick TSP (FSTSP) and the parallel drone scheduling TSP (PDSTSP). In FSTSP, a truck carrying a drone leaves a distribution center and delivers heavy or large cargoes that the drone cannot carry. Its route is determined by solving the TSP. The drone takes off from the truck and returns to it or the distribution center after delivering. In PDSTSP, a truck leaves a distribution center and delivers via a route determined by solving the TSP. A drone takes off from the distribution center and delivers to customers within its flight range. Othman et al. (2017) address a model in which a truck loaded with parcels and a drone follows a predetermined route. The drone delivers to customers in the last-stretch of the route. Agatz et al. (2018) address a model in which a truck and a drone share the same road network. Both the truck and the drone deliver via a route determined by solving the TSP. The drone takes off from the truck, delivers a parcel, and returns to the truck delivering to the next customer.

Second, in (Kim and Moon, 2019; Karuno and Mishima, 2019), multiple drones deliver parcels. Kim and Moon (2019) focus on a model using facilities that store drones and their batteries: drone stations (DS). A DS is located at the center of city, far from a distribution center. When a truck leaving the distribution center arrives at the DS, parcels are unloaded in the DS from it. The drones take off from the DS, deliver the parcels, and return to the DS. Karuno and Mishima (2019) address a scheduling model in which a truck loaded with parcels and drones stops at some points. At each point, the drones take off and deliver to customers. This scheduling problem is to assign customers to both the drones and their takeoff points.

Karuno and Mishima (2019) also propose formulating the parcel delivery scheduling problem as an integer programming problem. Their experimental results show that as the number of drones increases, the computation time takes longer. To such instances, it is appropriate to apply a genetic algorithm (GA) (Goldberg and Holland, 1988) which searches for a near-optimal solution in a short time. The GA is applied to actual problems (Zhuang et al., 2019; Ehyaei et al., 2020; Lee et al., 2018; Han et al., 2018) actively. When applied to a problem, the GA should comprise solution representation and crossover and mutation operations appropriate for the problem. The problem in (Karuno and Mishima, 2019) is an assignment problem, in other words, a problem of forming groups. A GA based on the characteristics of this type of problem was first proposed in (Falkenauer, 1994). Since then, many studies have proposed GAs for various grouping problems (Ramos-Figueroa et al., 2020, 2021; Mutingi and Mbohwa, 2014; Mutingi and Onwubolu, 2012; Jawahar and Subhaa, 2017; Cuadra et al., 2016; Ozcan et al., 2016; Moghaddam et al., 2015; Chen et al., 2019; Ülker et al., 2008) and are summarized in (Ramos-Figueroa et al., 2020, 2021).

In this paper, we propose a GA for scheduling of parcel delivery by drones. Our scheduling problem is the same as (Karuno and Mishima, 2019). The proposed method comprises effective solution representation and crossover and mutation operations for finding a near-optimal solution in a short time. Its performance is empirically evaluated by applying to many instances and comparing it with an integer programming solver in (Karuno and Mishima, 2019) and GAs for other grouping problems.

The rest of this paper is organized as follows. Section 2 defines our scheduling problem and presents related work on GAs for grouping problems. Section 3 proposes a GA for the defined problem. Section 4 shows experimental results and verifies the effectiveness of the proposed method. Finally, Section 5 concludes this paper.

2. Preliminaries

In this section, we first define our parcel delivery scheduling problem. Next, we describe studies related to GAs for grouping problems.

2.1 Problem formulation

A truck is loaded with n parcels and p drones and leaves a distribution center. As shown in Fig. 1, it stops at q points. At each point, the drones take off, deliver parcels to customers, and return to the truck. A drone can carry only one parcel. If p is less than the number of customers delivered from the takeoff point, some drones must deliver undelivered parcels after returning to the truck. When all drones finish delivering, the truck leaves the takeoff point. After leaving the final takeoff point, it moves to its destination.

This scheduling problem is to determine the drone delivering a parcel to each customer and the point at which the drone takes off. Its objective is to minimize the completion time of all parcel delivery. The route of the truck is predetermined, so the travel time does not change. Therefore, minimizing the completion time is equal to minimizing the sum of the maximal time from all the drones taking off at each point to returning there.

Karuno and Mishima (2019) formulate this scheduling problem as an integer programming problem. This formulation is described in Appendix A. In this paper, we formulate the scheduling problem in a different way for applying our GA. Let X_{jk} be a set of customers delivered by the drone $j\ (= 1,2,\dots,p)$ taking off at the point $k\ (= 1,2,\dots,q)$. Let w_{ik} be the round-trip flight time between the takeoff point k and the customer $i\ (= 1,2,\dots,n)$. The scheduling problem is formulated as follows:

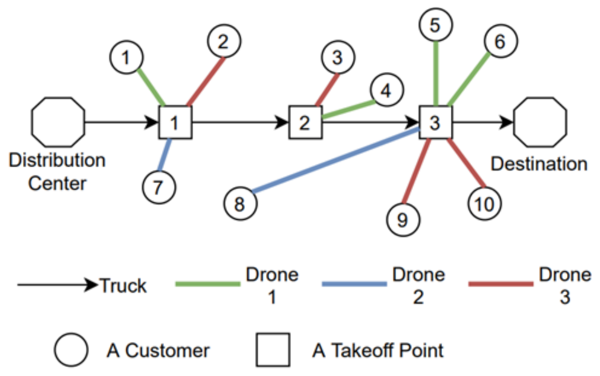

Fig. 1 Illustrative delivery route ($n = 10, p = 3, q = 3$).

$$\min\ f = \sum_{k=1}^{q} \max_{j} \sum_{i \in X_{jk}} w_{ik}, \tag{1}$$

$$\text{s.t.}\ \ X_{j_1 k_1} \cap X_{j_2 k_2} = \emptyset\ (\forall j_1, k_1, j_2, k_2\ ; j_1 \neq j_2 \text{ or } k_1 \neq k_2), \tag{2}$$

$$X_{11} \cup X_{12} \cup \ldots \cup X_{pq} = \{1,2, \ldots, n\}, \tag{3}$$

where f is the objective function. Equations (2) and (3) mean that each customer must be assigned to one of the takeoff points and one of the drones. The decision variables are X_{jk}. The solution shown in Fig. 1 is $X_{11} = \{1\}, X_{12} = \{4\}, X_{13} = \{5,6\}, X_{21} = \{7\}, X_{22} = \{\}, X_{23} = \{8\}, X_{31} = \{2\}, X_{32} = \{3\}$, and $X_{33} = \{9,10\}$, and its objective function value $f = \max\{w_{11}, w_{71}, w_{21}\} + \max\{w_{42}, w_{32}\} + \max\{w_{53} + w_{63}, w_{83}, w_{93} + w_{10,3}\} = w_{21} + w_{42} + w_{83}$.

2.2 Grouping genetic algorithm

Grouping problems are problems that group or assign items, considering their objective and constraints (Ramos-Figueroa et al., 2021). More formally, they are to partition a set V of n items into D mutually disjoint subsets $G_i\ (i = 1,2, \ldots, D)$ in such a way that their objective function is maximized or minimized. The main constraints on groups G_i are as follows:

$$G_i \cap G_j = \emptyset, i \neq j, \tag{4}$$

$$V = \bigcup_{i=1}^{D} G_i. \tag{5}$$

The scheduling problem defined in the previous subsection is a kind of grouping problems. Equations (2) and (3) are equivalent to Eqs. (4) and (5), respectively.

The grouping problems include classic problems such as bin packing and scheduling and real-world problems such as logistics and planning. They are difficult to solve, and many studies have adopted GAs for finding a near-optimal solution. A GA for a grouping problem (Grouping GA: GGA) was first proposed in (Falkenauer, 1994) and showed promising results for the bin packing problem. If a solution representation used for many other problems is adopted for the grouping problems, a solution is represented by arranging group numbers i to which items belong. For example, a solution 21231 represents $G_1 = \{2,5\}$, $G_2 = \{1,3\}$, and $G_3 = \{4\}$. However, items' combination in each group G_i is destroyed easily by a crossover operation, which deteriorates the performance of the GA. To avoid this destruction, the GGA uses a solution represented by arranging groups. It outperforms other metaheuristics such as tabu search and simulated annealing. Therefore, many GGAs have been proposed for various grouping problems (Ramos-Figueroa et al., 2020, 2021; Mutingi and Mbohwa, 2014; Mutingi and Onwubolu, 2012; Jawahar and Subhaa, 2017; Cuadra et al., 2016; Ozcan et al., 2016; Moghaddam et al., 2015; Tucker et al., 2005; Rossi et al., 2010; Chen et al., 2019; Ülker et al., 2008).

To find a near-optimal solution efficiently, it is necessary to use crossover and mutation operations specific to GGA's solution representation. Such crossovers include One-point crossover (1PX) (Mutingi and Mbohwa, 2014; Mutingi and Onwubolu, 2012), Multi-point crossover (MPX) (Jawahar and Subhaa, 2017), and Uniform crossover (UX)

(Cuadra et al., 2016), and they are often used for not only the grouping problems but also many others. Mutations include Elimination (Ramos-Figueroa et al., 2020; Ozcan et al., 2016; Moghaddam et al., 2015), Merge and split (Tucker et al., 2005), Swap (Mutingi and Mbohwa, 2014; Mutingi and Onwubolu, 2012), Insertion (Mutingi and Mbohwa, 2014; Cuadra et al., 2016; Chen et al., 2019), and Item elimination (Rossi et al., 2010).

Crossovers generate two new solutions (offspring) from two solutions (parents), and the offspring inherit parents' characteristics. In 1PX and MPX, they inherit some consecutive groups. In UX, they inherit scattered groups. Unlike many other problems, the three crossovers frequently generate infeasible solutions, so it is necessary to apply an additional procedure to make them feasible. Mutations alter one solution. Elimination and Merge and split mutate groups selected randomly. Swap, Insertion, and Item elimination mutate some items in some groups.

3. Proposed method

This section proposes a GA for finding a near-optimal solution of the scheduling problem formulated in Subsection 2.1. First, we describe its basic idea and then its flow. Next, we describe the solution representation and the crossover and mutation operations in our GA.

3.1 Basic idea

Our scheduling problem is to assign all customers to both takeoff points and drones. There are two strategies for applying a GA to this problem. One of them is to determine the two kinds of assignment by the GA. The other is to determine only one of the two kinds of assignment by the GA and determine the other by some heuristic rule. In the former strategy, the solution space searched by the GA is large, and it takes a long time to obtain a near-optimal solution. In the latter strategy, if the rule is effective, the search space can be reduced and includes near-optimal solutions. Therefore, it is promising to obtain a near-optimal solution in a short time.

The assignment of customers to drones at each takeoff point is equal to the parallel identical-machine scheduling problem where a machine, a job, and its processing time, respectively, correspond to a drone, a customer, and a round-trip time w_{ik}. For this scheduling problem, several approximation methods are known. Among them, the rule called the Longest Processing Time (LPT) is a simple and good approximation method (Graham, 1966). In LPT, the job with the longest processing time is assigned to the machine with the shortest total processing time.

Considering the above discussion, in our method, the GA determines the assignment to takeoff points, and LPT determines the assignment to drones.

3.2 Flow

The flow of our proposed method is based on general GAs (Goldberg and Holland, 1988), which repeat to update a population of multiple solutions called individuals. It is as follows.

N: Population size
G: Number of generations
S: Tournament size
p_c: Probability of crossover
p_m: Probability of mutation

Step 1 Generate N individuals $s_1, s_2, \ldots, s_N$ randomly. Let pop be their set ($pop = \{s_1, s_2, \ldots, s_N\}$).

Step 2 Calculate the objective function value of each individual of pop. In this calculation, LPT assigns customers to drones. Let $nextpop$ be the set of new individuals and set $nextpop \leftarrow \emptyset$ initially.

Step 3 Pick up S individuals from pop randomly and select one of them by applying tournament selection based on their objective function values. Similarly, select another individual. The two selected individuals are denoted as P_1 and P_2.

Step 4 Generate a uniformly random number r in [0,1). If $r \leq p_c$, go to Step 5. If $p_c < r \leq p_c + p_m$, go to Step 6. If $r > p_c + p_m$, go to Step 7.

Step 5 Generate two individuals O_1 and O_2 by applying a crossover to P_1 and P_2. Set $nextpop \leftarrow nextpop \cup \{O_1, O_2\}$. Go to Step 8.

Step 6 Generate two individuals O_1 and O_2, respectively, by applying a mutation to P_1 and P_2. Set $nextpop \leftarrow nextpop \cup \{O_1, O_2\}$. Go to Step 8.

$$\{1,6,9\},\{\},\{2,4,5,8\},\{3,7\},\{10\}$$

Fig. 2 An example of solution representation.

$$
\begin{aligned}
Y_k = \{1,2,3,4,5\} &\xrightarrow[\text{Step 1}]{} \{4,3,2,5,1\} \\
&\downarrow \text{Step 2} \\
X_{1k} = \{\}, X_{2k} &= \{\}, X_{3k} = \{\} \\
&\downarrow \text{Step 3,4} \\
X_{1k} = \{4\}, X_{2k} &= \{\}, X_{3k} = \{\} \\
&\downarrow \text{Step 3,4} \\
X_{1k} = \{4\}, X_{2k} &= \{3\}, X_{3k} = \{\} \\
&\downarrow \text{Step 3,4} \\
X_{1k} = \{4\}, X_{2k} &= \{3\}, X_{3k} = \{2\} \\
&\downarrow \text{Step 3,4} \\
X_{1k} = \{4\}, X_{2k} &= \{3\}, X_{3k} = \{2,5\} \\
&\downarrow \text{Step 3,4} \\
X_{1k} = \{4\}, X_{2k} &= \{1,3\}, X_{3k} = \{2,5\}
\end{aligned}
$$

Fig. 3 An example of determining X_{jk} from Y_k.

Step 7 Set $nextpop \leftarrow nextpop \cup \{P_1, P_2\}$.

Step 8 If the number of elements of $nextpop$ is less than N, return to Step 3.

Step 9 Set $pop \leftarrow nextpop$.

Step 10 Repeat Steps 2-9 G times. Determine the final solution with the smallest objective function value among all individuals generated so far. In calculating an objective function value, LPT assigns customers to drones.

3.3 Solution representation

A solution is represented by arranging the set of customers assigned to each takeoff point, in other words, by $Y_1, Y_2, \ldots, Y_q$ where $Y_k = \bigcup_j X_{jk}$. Figure 2 shows an example of the solution representation for $n = 10$ and $q = 5$. In this example, customers $i = 1,6,9$, $i = 2,4,5,8$, $i = 3,7$, and $i = 10$ are, respectively, assigned to takeoff points $k = 1$, $k = 3$, $k = 4$, and $k = 5$. No customers are assigned to takeoff point $k = 2$.

The method of assigning customers to drones at a takeoff point k, that is, determining $X_{1k}, X_{2k}, \ldots, X_{pk}$ from Y_k by LPT is as follows.

Step 1 Sort the customers of Y_k in descending order of round-trip flight time w_{ik}. The sorted customers are denoted as $\left\{y_k^1, y_k^2, \ldots, y_k^{n(Y_k)}\right\}$. $n(Y_k)$ is the number of elements of Y_k.

Step 2 Set initially $X_{jk} = \{\}$ and $g_{jk} = 0\ (\forall j)$ where g_{jk} is the total flight time of drone j. Set initially $t = 1$ where t is an index to the sorted customers.

Step 3 Assign the customer y_k^t to the drone j^* with the shortest total flight time, that is, set $j^* = \arg\min_j g_{jk}$ and $X_{j^*k} \leftarrow X_{j^*k} \cup \{y_k^t\}$. Set $g_{j^*k} \leftarrow g_{j^*k} + w_{y_k^t k}$.

Step 4 If $t = n(Y_k)$, terminate this algorithm. Otherwise, set $t \leftarrow t + 1$ and return to Step 3.

Figure 3 shows an example of determining X_{jk} from Y_k. In this example, let $p = 3$, $Y_k = \{1,2,3,4,5\}$, $w_{1k} = 2$, $w_{2k} = 6$, $w_{3k} = 7$, $w_{4k} = 9$, and $w_{5k} = 4$. First, the customers sorted in descending order of w_{ik} are $\{4,3,2,5,1\}$. Next, set $g_{1k} = g_{2k} = g_{3k} = 0$, $X_{1k} = X_{2k} = X_{3k} = \{\}$, and $t = 1$. The first customer $i = 4$ is assigned to $j^* = 1$, and then set $X_{1k} = \{4\}$ and $g_{1k} = 9$. Since $t \neq n(Y_k)$, set $t = 2$. The second customer $i = 3$ is assigned to $j^* = 2$, and then set $X_{2k} = \{3\}$ and $g_{2k} = 7$. Since $t \neq n(Y_k)$, set $t = 3$. Similarly, X_{1k}, X_{2k}, and X_{3k} are updated. When $X_{1k} = \{4\}$, $X_{2k} = \{1,3\}$, $X_{3k} = \{2,5\}$, and $t = 5$, this algorithm is terminated.

3.4 Crossover and mutation

Since a solution is represented using the combination Y_k of customers assigned to each takeoff point k, we propose a crossover such that offspring inherits this combination. The crossover generates offspring O_1 and O_2 from parents P_1 and P_2. Let the solutions of O_1 and O_2 be $Y_1^{O1}, \ldots, Y_q^{O1}$ and $Y_1^{O2}, \ldots, Y_q^{O2}$, respectively, and those of P_1 and P_2 be $Y_1^{P1}, \ldots, Y_q^{P1}$ and $Y_1^{P2}, \ldots, Y_q^{P2}$, respectively. First, a takeoff point k' is selected randomly. O_1 inherits $Y_{k'}^{P1}$, and O_2 inherits $Y_{k'}^{P2}$. These combinations are inherited from the parents with the same number. Next, O_1 inherits $Y_1^{P2}, \ldots, Y_{k'-1}^{P2}, Y_{k'+1}^{P2}, \ldots, Y_q^{P2}$, and O_2 inherits $Y_1^{P1}, \ldots, Y_{k'-1}^{P1}, Y_{k'+1}^{P1}, \ldots, Y_q^{P1}$. These combinations are inherited from the parents with the different number.

$$\begin{array}{lllllll}
P_1 : & \{1,6,9\}, & \{\}, & \{2,4,5,8\}, & \{3,7\}, & \{10\} \\
P_2 : & \{1,8,9,10\}, & \{3,4\}, & \{5\}, & \{6,7\}, & \{2\} \\
 & & & \downarrow & & \\
O_1 : & \{1,9,10\}, & \{3\}, & \{2,4,5,8\}, & \{6,7\}, & \{\} \\
O_2 : & \{1,4,6,9\}, & \{2,8\}, & \{5\}, & \{3,7\}, & \{10\}
\end{array}$$

Fig. 4 An example of crossover.

$$\begin{array}{l}
P : \{1,6,9\},\{\},\{2,4,5,8\},\{3,7\},\{10\} \\
\downarrow \\
O : \{1,6,9\},\{10\},\{2,4,5,8\},\{3,7\},\{\}
\end{array}$$

Fig. 5 An example of mutation.

If customers in $Y_{k'}^{P2}$ are not in $Y_{k'}^{P1}$, they do not appear in O_1, and the constraint (3) is not satisfied. Thus, each of them is assigned to one customer set selected randomly from $Y_1^{O1}, \dots, Y_{k'-1}^{O1}, Y_{k'+1}^{O1}, \dots, Y_q^{O1}$. If customers in $Y_{k'}^{P1}$ are not in $Y_{k'}^{P2}$, each of them appears twice in O_1, and the constraint (2) is not satisfied. In this case, $Y_{k'}^{O1}$ is prioritized, and customers inherited from Y_k^{P2} $(k \in \{1, \dots, k'-1, k'+1, \dots, q\})$ are deleted. These operations for satisfying the constraints (2) and (3) are applied to also generating O_2.

The flow of the crossover is as follows.

```
Step 1  Select a takeoff point $k'$ randomly.
Step 2  Set $Y_{k'}^{O1} \leftarrow Y_{k'}^{P1}$ and $Y_{k'}^{O2} \leftarrow Y_{k'}^{P2}$.
Step 3  Set $Y_k^{O1} \leftarrow Y_k^{P2}$ and $Y_k^{O2} \leftarrow Y_k^{P1}$ $(k = 1, \dots, k'-1, k'+1, \dots, q)$.
Step 4  Set a customer number $i \leftarrow 1$.
Step 5  If $i \in Y_{k'}^{P2}$ and $i \notin Y_{k'}^{P1}$, go to Step 6. Otherwise, go to Step 8.
Step 6  Select a takeoff point number $k$ randomly from $1, \dots, k'-1, k'+1, \dots, q$, and set $Y_k^{O1} \leftarrow Y_k^{O1} \cup \{i\}$.
Step 7  Find the takeoff point $k$ such that $i \in Y_k^{P1}$ $(k \in \{1, \dots, k'-1, k'+1, \dots, q\})$, and set $Y_k^{O2} \leftarrow Y_k^{O2} \setminus \{i\}$.
Step 8  If $i \in Y_{k'}^{P1}$ and $i \notin Y_{k'}^{P2}$, go to Step 9. Otherwise, go to Step 11.
Step 9  Select a takeoff point number $k$ randomly from $1, \dots, k'-1, k'+1, \dots, q$, and set $Y_k^{O2} \leftarrow Y_k^{O2} \cup \{i\}$.
Step 10 Find the takeoff point $k$ such that $i \in Y_k^{P2}$ $(k \in \{1, \dots, k'-1, k'+1, \dots, q\})$, and set $Y_k^{O1} \leftarrow Y_k^{O1} \setminus \{i\}$.
Step 11 If $i < n$, set $i \leftarrow i + 1$, and return to Step 5. Otherwise, terminate this algorithm.
```

Figure 4 shows an example of the crossover for $n = 10$ and $q = 5$. In this example, a takeoff point $k' = 3$ is selected. In Step 1, $Y_3^{O1} = \{2,4,5,8\}$, and $Y_3^{O2} = \{5\}$. In Step 2, $Y_1^{O1} = \{1,8,9,10\}, Y_2^{O1} = \{3,4\}, Y_4^{O1} = \{6,7\}, Y_5^{O1} = \{2\}, Y_1^{O2} = \{1,6,9\}, Y_2^{O2} = \{\}, Y_4^{O2} = \{3,7\}$, and $Y_5^{O2} = \{10\}$. For $i = 1$, all of Y_k^{O1} and Y_k^{O2} are unchanged in Steps 5-10. For $i = 2$, $Y_2^{O2} = \{2\}$ in Step 9, and $Y_5^{O1} = \{\}$ in Step 10. For $i > 2$, $Y_1^{O1}, Y_2^{O1}, Y_1^{O2}$, and Y_2^{O2} are altered in the same way.

Our mutation selects a customer i and a takeoff point k randomly and assigns i to k. Figure 5 shows an example of the mutation for $n = 10$ and $q = 5$. In this example, $i = 10$ and $k = 2$ are selected at random.

4. Numerical experiments

This section shows experimental results of applying the method proposed in Section 3 to many instances and its performance compared with other methods. We describe our experimental method, show the experimental results, and discuss them.

4.1 Method

We use 20 instances in (Mishima and Karuno, 2019). In these instances, let $n = 50$ and $q = 10$. The coordinates (x_i, y_i) of each customer i and those (X_k, Y_k) of each takeoff point k are on a plane whose size is 50×50. The coordinates of the customers are different depending on instances. For example, in instance 1, they are (0,31), (1,2), (4,44), (5,1), (6,48), (10,10), (10,21), (11,30), (12,0), (12,27), (14,33), (15,16), (15,47), (20,20), (20,26), (20,37), (21,4), (22,40), (23,1), (24,23), (24,38), (26,21), (28,31), (28,40), (29,29), (30,28), (30,32), (31,9), (31,24), (31,49), (32,16), (35,4), (35,9), (35,46), (36,37), (38,18), (38,36), (38,44), (39,31), (39,37), (39,43), (39,50), (42,26), (44,39), (46,12), (46,48), (47,41), (49,20), (49,29), and (50,20). The coordinates of the takeoff points are the same for all instances and are (0,3), (2,44), (9,43), (10,32), (19,14), (23,13), (30,43), (32,11), (40,10), and (45,17). The round-trip flight time w_{ik} between i and k is given by

$$w_{ik} = \mathrm{ceil}\left(\sqrt{(x_i - X_k)^2 + (y_i - Y_k)^2}\right), \tag{6}$$

where ceil means the ceiling function. The number p of drones is set to 4, 5, 6, and 7 for each of the 20 instances. The total of instances is 80.

We apply the proposed GA 20 times to each instance and compare it with an integer programming solver (IP solver) CPLEX (IBM) in (Mishima and Karuno, 2019) and GGAs in (Ramos-Figueroa et al., 2020, 2021; Mutingi and Mbohwa, 2014; Mutingi and Onwubolu, 2012; Jawahar and Subhaa, 2017; Cuadra et al., 2016; Ozcan et al., 2016; Moghaddam et al., 2015; Tucker et al., 2005; Rossi et al., 2010; Chen et al., 2019; Ülker et al., 2008). The results of the IP solver are cited from (Mishima and Karuno, 2019). The GGAs are described in Appendix B.

The parameters of GAs are as follows:

- Population size: $N = 30000$,
- Number of generations: $G = 80$,
- Tournament size: $S = 5$,
- Probability of crossover: $p_c = 0.6$,
- Probability of mutation: $p_m = 0.01$.

We use Python as a programming language. The experiments are conducted on a computer with Intel Core i7-7500U CPU (2.70GHz). The IP solver in (Mishima and Karuno, 2019) was run on a computer with Intel Core i7-8550U CPU (1.80GHz) and 20GB memory.

4.2 Results

Table 1 shows the average $\bar{f}_{GA}$ of objective function values by applying the proposed GA 20 times and the objective function value f_{IP} by the IP solver for each instance. Table 2 shows the average $\bar{T}_{GA}$ of computation time by applying the GA and the computation time T_{IP} by the IP solver. The IP solver is not applied to instances for $p = 7$, and its computation is limited to about 3600 seconds (Mishima and Karuno, 2019). Table 3 shows for $p = 6$, average objective function values $\bar{f}_{1PX}$, $\bar{f}_{MPX}$, $\bar{f}_{UX}$, $\bar{f}_{E}$, $\bar{f}_{MS}$, $\bar{f}_{S}$, $\bar{f}_{I}$, and $\bar{f}_{IE}$, respectively, by applying eight kinds of GAs described in Appendix B: GAs using 1PX, MPX, UX, Elimination, Merge and split, Swap, Insertion, and Item elimination. Figure 6 shows the convergence curve of the proposed GA for instance 1.

Table 1 shows the proposed GA can successfully find solutions which are as good as the IP solver capable of searching for an optimal solution. The difference between the two is 1.21, 1.38, and 1.44 for $p = 4$, 5, and 6, respectively. Table 2 shows the computation time of the GA for $p = 6$ is shorter than that of the IP solver in all instances and about 1/76 on average. It for $p = 5$ and 4 is shorter in 19 and 17 instances and about 1/48 and 1/13 on average, respectively. Table 3 shows our crossover is more effective than the other crossovers: 1PX, MPX, and UX. Our mutation is slightly better than the other mutations. Figure 6 shows the proposed GA finds the solution by iteratively generating individuals with better objective function values from initial individuals with bad ones.

4.3 Discussion

For the proposed GA, Table 1 shows the objective function value decreases by about ten whenever the number p of drones increases by one, and Fig. 6 shows the shapes of the four graphs are similar. These two facts imply that the GA does not deteriorate for a larger number of the drones and that the solutions obtained for $p = 7$ are good. For the IP solver, Table 2 shows as the number of drones increases, its computation time is longer than 3600 seconds in more instances. For $p = 7$, such results are expected for almost all instances, which makes the difference in computation time between the two methods even larger. The IP solver for $p = 7$ may not find an optimal solution within 3600 seconds. Therefore, the proposed GA is more effective as the number of drones increases. In contrast, for $p = 4$, there are some instances where the GA takes more time. However, fluctuations in the computation time are minor, and the average in 20 instances is smaller. These are advantages of the GA. Table 3 shows the difference of objective function values between our mutation and each of the other mutations is small. However, our mutation is the simplest since it only assigns one customer to another takeoff point.

As mentioned in Subsection 3.4, our crossover assigns a disappeared customer to a random takeoff point to satisfy the constraint (3). It also makes offspring not inherit one of the duplicate customers assigned to two takeoff points to satisfy the constraint (2). These operations to satisfy (2) and (3) are named op1 and op2, respectively. They are used in

Table 1 Comparison of objective function values by the proposed method and the IP solver.

	$\bar{f}_{GA}$				f_{IP}		
Instance	$p = 4$	5	6	7	$p = 4$	5	6
1	131.0	112.0	97.3	87.8	129	109	96
2	112.0	94.2	85.6	77.7	111	93	84
3	112.0	98.2	86.0	78.8	111	99	85
4	115.1	99.1	88.2	80.0	115	98	88
5	118.0	100.1	88.7	81.7	116	100	88
6	113.7	97.3	87.5	79.3	111	96	86
7	116.0	96.8	85.1	77.8	115	94	83
8	125.0	108.1	95.9	85.0	123	106	93
9	119.2	100.0	88.0	80.0	117	99	88
10	128.0	107.0	95.3	85.8	126	106	94
11	112.1	93.1	82.5	73.2	111	91	82
12	124.0	104.5	92.1	82.8	123	104	92
13	131.0	111.2	97.4	87.7	131	109	96
14	114.0	95.1	85.0	78.0	112	94	84
15	130.0	110.0	99.1	90.1	129	109	95
16	130.0	108.2	96.0	85.5	129	107	93
17	130.0	112.1	99.9	91.9	130	109	96
18	116.0	99.0	84.0	75.4	116	97	84
19	131.3	109.6	94.7	86.7	130	108	93
20	128.0	110.3	96.8	87.8	127	110	96
Average	121.81	103.28	91.24	82.64	120.6	101.9	89.8

Table 2 Comparison of computation time [s] by the proposed method and the IP solver.

	$\bar{T}_{GA}$				T_{IP}		
Instance	$p = 4$	5	6	7	$p = 4$	5	6
1	39.3	39.5	39.4	40.5	81.5	2872.3	3610.9
2	39.2	39.3	39.5	40.3	186.3	474.1	2476.6
3	39.1	39.8	39.3	40.5	5.3	3613.6	3612.3
4	39.3	39.4	40.1	40.2	3606.1	3605.4	3614.6
5	39.1	39.5	39.4	40.2	32.3	3607.1	1099.6
6	39.9	39.2	39.4	40.9	40.4	697.5	2138.1
7	40.1	39.2	39.6	40.0	691.7	184.8	351.8
8	39.4	39.6	40.0	40.3	166.2	3611.6	3609.9
9	40.8	38.9	38.9	39.9	25.5	474.9	3604.6
10	39.4	39.3	39.3	40.1	43.7	430.7	3605.3
11	39.6	39.8	39.9	39.7	56.2	9.2	3607.0
12	39.8	40.3	39.9	39.9	284.4	2863.8	3610.5
13	39.0	40.3	39.3	40.2	138.2	286.5	3610.6
14	39.7	40.2	39.6	40.4	49.5	104.8	3606.2
15	39.0	39.5	39.8	39.6	606.2	3605.8	3607.6
16	39.3	39.4	39.1	39.7	246.3	498.0	3606.3
17	38.9	39.6	40.1	40.9	90.4	3605.8	3612.3
18	39.6	39.5	39.3	41.4	65.5	614.7	261.5
19	39.4	39.5	40.2	40.5	3608.2	3603.5	3605.7
20	39.2	40.1	40.2	40.7	602.8	3613.5	3610.3
Average	39.45	39.60	39.62	40.27	531.33	1918.87	3028.08

Table 3 Comparison of objective function values by the proposed method and other GGAs ($p = 6$).

Instance	$\bar{f}_{GA}$	$\bar{f}_{1PX}$	$\bar{f}_{MPX}$	$\bar{f}_{UX}$	$\bar{f}_{E}$	$\bar{f}_{MS}$	$\bar{f}_{S}$	$\bar{f}_{I}$	$\bar{f}_{IE}$
1	97.3	139.1	140.2	119.9	97.3	97.4	97.2	97.1	97.1
2	85.6	127.1	128.2	113.5	85.6	85.5	85.6	85.9	85.6
3	86.0	124.1	126.9	106.2	86.1	86.0	86.3	86.1	86.2
4	88.2	120.9	126.8	113.1	88.2	88.5	88.3	88.3	88.3
5	88.7	134.5	132.7	119.4	88.9	88.7	88.7	88.8	89.1
6	87.5	128.5	130.0	104.7	87.8	87.6	87.7	87.3	87.2
7	85.1	124.3	127.3	110.5	85.3	85.1	83.7	85.3	85.3
8	95.9	140.3	136.8	118.4	96.3	95.8	96.1	96.2	96.0
9	88.0	135.9	139.5	119.7	88.0	88.0	88.0	88.0	88.0
10	95.3	134.6	135.1	125.4	95.1	95.2	95.2	95.0	95.2
11	82.5	119.5	120.5	103.3	82.8	82.8	82.5	82.6	82.6
12	92.1	128.2	130.3	111.1	92.4	92.0	92.0	92.0	92.1
13	97.4	131.1	134.2	116.0	97.5	97.5	97.3	97.4	97.4
14	85.0	121.6	124.9	111.6	86.1	85.7	86.0	85.1	85.3
15	99.1	143.9	146.3	119.4	100.7	100.1	99.9	100.1	100.1
16	96.0	139.7	143.5	117.4	96.1	96.0	96.0	96.0	96.0
17	99.9	139.6	141.2	121.8	100.3	99.5	100.6	100.2	99.8
18	84.0	119.8	119.9	110.1	84.2	84.2	84.0	84.0	84.0
19	94.7	133.9	140.3	117.6	94.0	94.3	94.3	94.5	94.6
20	96.8	128.7	133.6	121.0	97.3	96.8	96.8	96.8	96.9
Average	91.24	130.77	132.91	115.01	91.48	91.32	91.28	91.32	91.32

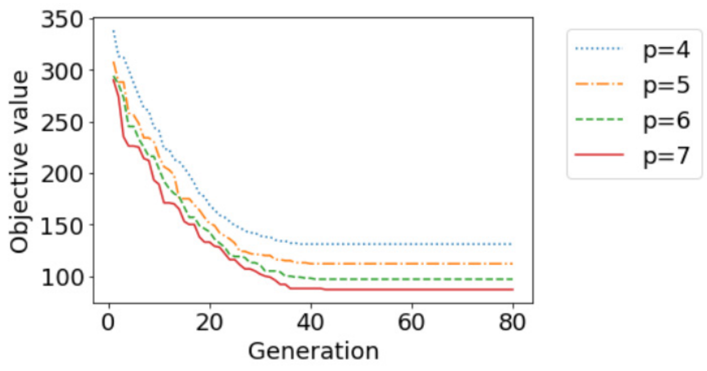

Fig. 6 Convergence curve of the proposed GA for instance 1.

Table 4 Numbers of applying two operations op1 and op2 in crossovers.

Operation	Proposed Crossover	1PX	MPX	UX
op1	9.00	18.34	15.71	22.50
op2	9.00	18.34	15.71	22.50

also the other crossovers: 1PX, MPX, and UX. They break the combination of customers assigned to each takeoff point, and therefore they should be applied as few times as possible. Thus, we compare the number of times op1 and op2 are applied. We generate offspring O_1 and O_2 from parents P_1 and P_2 generated randomly 1000000 times, and we count the numbers of applying op1 and op2. Table 4 shows the results. The numbers of these operations are essentially equal, and this fact is seen in this table. The numbers in our crossover are less than those of the other crossovers. These small numbers are one of the reasons for the good performance of ours.

5. Conclusion

In this paper, we have defined a scheduling problem of parcel delivery by drones as assigning all customers to both takeoff points and the drones. We have also proposed a GA for finding a near-optimal solution in a short time. Experimental results show the proposed GA can successfully find it faster than the IP solver for almost all instances. Compared with other GGAs, our crossover is the best, and our mutation is effective. This best performance results from a small number of applying operations which break customers' combination.

This paper has addressed a model in which a drone carries only one parcel. With the development of drone technology, the drone will carry multiple parcels. Peng et al. (2019) address a problem with such drones. Scheduling with the drones is an extended topic of our problem.

This work was partly supported by JSPS KAKENHI Grant Number JP20K11988.

Appendix A

This appendix describes the formulation of the problem of Subsection 2.1 in (Karuno and Mishima, 2019). Let x_{ijk} be a decision variable whose value is 1 if a customer i is delivered from a takeoff point k by a drone j. Otherwise, x_{ijk} is 0. The problem is formulated as follows:

$$\min \sum_{k=1}^{q} f_k, \tag{7}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} w_{ik} x_{ijk} \le f_k, \forall j \in \{1,2,\dots,p\};\ \forall k \in \{1,2,\dots,q\}, \tag{8}$$

$$\sum_{j=1}^{p} \sum_{k=1}^{q} x_{ijk} = 1,\ \forall i \in \{1,2,\dots,n\}, \tag{9}$$

$$x_{ijk} \in \{0,1\},\ \forall i \in \{1,2,\dots,n\};\ \forall j \in \{1,2,\dots,p\};\ \forall k \in \{1,2,\dots,q\}, \tag{10}$$

$$f_k \ge 0,\ \forall k \in \{1,2,\dots,q\}. \tag{11}$$

$$
\begin{array}{lllllll}
P_1: & \{1,6,9\}, & \{\}, & \{2,4,5,8\}, & \{3,7\}, & \{10\} & \\
P_2: & \{1,8,9,10\}, & \{3,4\}, & \{5\}, & \{6,7\}, & \{2\} & \\
 & & & \downarrow & & & \\
O_1: & \{1,6,9\}, & \{\}, & \{2,4,5,8\}, & \{\cancel{6},7\}, & \{\cancel{2}\} & \leftarrow 3,10 \\
O_2: & \{1,8,9,10\}, & \{3,4\}, & \{5\}, & \{\cancel{3},7\}, & \{\cancel{10}\} & \leftarrow 2,6
\end{array}
$$

Fig. 7 Example of 1PX ($k = 3$).

$$
\begin{array}{lllllll}
P_1: & \{1,6,9\}, & \{\}, & \{2,4,5,8\}, & \{3,7\}, & \{10\} & \\
P_2: & \{1,8,9,10\}, & \{3,4\}, & \{5\}, & \{6,7\}, & \{2\} & \\
 & & & \downarrow & & & \\
O_1: & \{1,6,9\}, & \{\cancel{3},4\}, & \{5\}, & \{3,7\}, & \{2\} & \leftarrow 8 \\
O_2: & \{1,8,9,10\}, & \{\}, & \{2,4,5,\cancel{8}\}, & \{6,7\}, & \{10\} & \leftarrow 3
\end{array}
$$

Fig. 8 Example of MPX ($M = 3, k_1 = 1, k_2 = 3, k_3 = 4$).

$$
\begin{array}{lllllll}
P_1: & \{1,6,9\}, & \{\}, & \{2,4,5,8\}, & \{3,7\}, & \{10\} & \\
P_2: & \{1,8,9,10\}, & \{3,4\}, & \{5\}, & \{6,7\}, & \{2\} & \\
 & & & \downarrow & & & \\
O_1: & \{1,6,9\}, & \{3,\cancel{4}\}, & \{2,4,5,8\}, & \{\cancel{6},7\}, & \{10\} & \\
O_2: & \{1,8,9,10\}, & \{\}, & \{5\}, & \{3,7\}, & \{2\} & \leftarrow 4,6
\end{array}
$$

Fig. 9 Example of UX.

$$
\begin{array}{c}
P: \{1,6,9\},\{\},\{2,4,5,8\},\{3,7\},\{10\} \\
\downarrow \\
O: \{\},\{1\},\{2,4,5,8\},\{3,6,7\},\{9,10\}
\end{array}
$$

Fig. 10 Example of Elimination ($k' = 1$).

$$
\begin{array}{c}
P: \{1,6,9\},\{\},\{2,4,5,8\},\{3,7\},\{10\} \\
\downarrow \text{Merge} \\
\{1,6,9\},\{\},\{2,3,4,5,7,8\},\{\},\{10\} \\
\downarrow \text{Split} \\
O: \{1,6,9\},\{\},\{2,5,8\},\{3,4,7\},\{10\}
\end{array}
$$

Fig. 11 Example of Merge and split ($k_1 = 3, k_2 = 4$).

$$
\begin{array}{c}
P: \{1,6,9\},\{\},\{2,4,5,8\},\{3,7\},\{10\} \\
\downarrow \\
O: \{1,3,7\},\{\},\{2,4,5,8\},\{6,9\},\{10\}
\end{array}
$$

Fig. 12 Example of Swap ($k_1 = 1, k_2 = 4$).

$$
\begin{array}{c}
P: \{1,6,9\},\{\},\{2,4,5,8\},\{3,7\},\{10\} \\
\downarrow \\
O: \{1,6,9\},\{4\},\{2,8\},\{3,5,7\},\{10\}
\end{array}
$$

Fig. 13 Example of Insertion ($k' = 3$).

$$
\begin{array}{c}
P: \{1,6,9\},\{\},\{2,4,5,8\},\{3,7\},\{10\} \\
\downarrow \\
O: \{5,6\},\{1,9\},\{8\},\{3,4,7\},\{2,10\}
\end{array}
$$

Fig. 14 Example of Item elimination.

Equations (7) and (8) mean the objective function is the sum of maximum delivery time f_k of all takeoff points. Equations (9) and (10) mean each customer is delivered by a drone from a takeoff point.

Appendix B

This appendix describes GGAs used in the experiments of Section 4. Crossovers of the GGAs are 1PX, MPX, and UX and generate offspring O_1 and O_2 from parents P_1 and P_2. Mutations are Elimination, Merge and split, Swap, Insertion, and Item elimination and generate offspring O from a parent P.

In 1PX, one takeoff point k is randomly selected. O_1 (O_2) inherits $Y_{k'}$ ($k' \le k$) from P_2 (P_1) and inherits $Y_{k'}$ ($k' > k$) from P_1 (P_2). However, O_1 and O_2 do not usually satisfy the constraints (2) and (3). Thus, two operations op1 and op2, described in Subsection 4.3, are applied to O_1 and O_2. In op2, offspring do not inherit customers of $Y_{k'}$ ($k' > k$). In MPX, M (≥ 3) takeoff points $k_1, k_2, \ldots, k_M$ are randomly selected, and each offspring inherits Y_k alternately from P_1 and P_2. For example, O_1 inherits $Y_1, \ldots, Y_{k_1}$ from P_1, and $Y_{k_1+1}, \ldots, Y_{k_2}$ from P_2, and so on. Then, op1 and op2 are applied. In UX, O_1 inherits each Y_k from either P_1 or P_2 with the same probability. O_2 inherits from P_1 (P_2) Y_k which O_1 inherits from P_2 (P_1). Then, op1 and op2 are applied.

Figures 7-9 show examples of these crossovers. In these figures, a diagonal line means a customer who is not inherited in op2. An arrow means op1. In Fig. 9, O_1 inherits Y_1, Y_3, and Y_5 from P_1 and inherits Y_2 and Y_4 from P_2. O_2 inherits Y_1, Y_3, and Y_5 from P_2 and inherits Y_2 and Y_4 from P_1.

Elimination mutation removes all customers assigned to a randomly selected takeoff point k' and randomly assigns each of them to Y_k ($k \ne k'$). Merge and split mutation randomly selects two takeoff points k_1 and k_2, merges all customers assigned to them into one set, and then splits the set into Y_{k_1} and Y_{k_2}. Swap mutation selects two takeoff points k_1 and k_2 and swaps some customers assigned to them. Insertion mutation removes some customers assigned to a randomly selected takeoff point k' and randomly assigns each of them to Y_k ($k \ne k'$). Item elimination mutation removes each customer with a certain probability and randomly assigns the removed customers. Figures 10-14 show examples of these mutations. In Fig. 14, customers 1, 2, 4, 5, and 9 are removed and randomly assigned.

References

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

Amazon.com, Inc., Amazon Prime Air (online), available from <http://www.amazon.com/primeair>, (accessed on 6 April, 2021).

Chen, C. -H., Shen, W. -Y., Wu, M. -E., and Hong, T. -P., A divide-and-conquer-based approach for diverse grouping stock portfolio optimization using Island-based genetic algorithms, Proceedings of IEEE Congress on Evolutionary Computation (2019), pp.1471–1477.

Chen, M., Tian, Y., Fortino, G., Zhang, J., and Humar, I., Cognitive Internet of vehicles, Computer Communications, Vol.120 (2018), pp.58–70.

Cuadra, L., Aybar-Ruíz, A., Del Arco, M. A., Navío-Marco, J., Portilla-Figueras, J. A., and Salcedo-Sanz, S., A Lamarckian hybrid grouping genetic algorithm with repair heuristics for resource assignment in WCDMA networks, Applied Soft Computing, Vol.43 (2016), pp.619–632.

DHL International GmbH, DHL parcelcopter launches initial operations for research purposes (online), available from http://www.dhl.com/en/press/releases/releases_2014/group/dhl_parcelcopter_launches_initial_operations_for_research_purposes.html, (accessed on 6 April, 2021).

Ehyaei, M. A., Ahmadi, A., Rosen, M. A., and Davarpanah, A., Thermodynamic optimization of a geothermal power plant with a genetic algorithm in two stages, Processes, Vol.8, No.10 (2020), 1277.

Falkenauer, E., A new representation and operators for genetic algorithms applied to grouping problems, Evolutionary Computation, Vol.2, No.2 (1994), pp.123–144.

Goldberg, D. E. and Holland, J. H., Genetic algorithms and machine learning, Machine Learning, Vol.3 (1988), pp. 95–99.

Graham, R. L., Bounds for certain multiprocessing anomalies, The Bell System Technical Journal, Vol.45, No.9 (1966), pp.1563–1581.

Han, B., Lianghai, J., and Schotten, H. D., Slice as an evolutionary service: Genetic optimization for inter-slice resource management in 5G networks, IEEE Access, Vol.6 (2018), pp.33137–33147.

IBM, IBM CPLEX Optimizer (online), available from <https://www.ibm.com/analytics/cplex-optimizer>, (accessed on 6 April, 2021).

Jawahar, N., and Subhaa, R., An adjustable grouping genetic algorithm for the design of cellular manufacturing system integrating structural and operational parameters, Journal of Manufacturing Systems, Vol.44, No.1 (2017), pp.115–142.

Karuno, Y. and Mishima, K., Makespan minimization scheduling for a truck-drone parcel delivery system: An integer programming formulation, Proceedings of International Symposium on Scheduling (2019), pp.103–107.

Kim, S. and Moon, I., Traveling salesman problem with a drone station, IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol.49, No.1 (2019), pp.42–52.

Laporte, G., The traveling salesman problem: An overview of exact and approximate algorithms, European Journal of Operational Research, Vol.59, No.2 (1992), pp.231–247.

Lee, S. -M., Kim, J. W., and Myung, H., Split-and-merge-based genetic algorithm (SM-GA) for LEGO brick sculpture optimization, IEEE Access, Vol.6 (2018), pp.40429–40438.

Mathew, N., Smith, S. L., and Waslander, S. L., Planning paths for package delivery in heterogeneous multirobot teams, IEEE Transactions on Automation Science and Engineering, Vol.12, No.4 (2015), pp.1298–1308.

Mishima, K. and Karuno, Y., A parcel delivery scheduling problem with multiple unmanned aerial vehicles and a single truck, Proceedings of the Scheduling Symposium (2019), pp.19–24.

Moghaddam, F. F., Moghaddam, R. F., and Cheriet, M., Carbon-aware distributed cloud: Multi-level grouping genetic algorithm, Cluster Computing, Vol.18 (2015), pp.477–491.

Murray, C. C. and Chu, A. G., The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery, Transportation Research Part C: Emerging Technologies, Vol.54 (2015), pp.86–109.

Mutingi, M. and Mbohwa, C., A fuzzy grouping genetic algorithm for care task assignment, Proceedings of the World Congress on Engineering and Computer Science, Vol.1 (2014), pp.454–459.

Mutingi, M. and Onwubolu, G. C., Integrated cellular manufacturing system design and layout using group genetic algorithms, Manufacturing System (2012), pp.205–222 (online), available from <https://www.intechopen.com/chapters/36411>, (accessed on 6 April, 2021).

Othman, M. S., Shurbevski, A., Karuno, Y., and Nagamochi, H., Routing of carrier-vehicle systems with dedicated last-stretch delivery vehicle and fixed carrier route, Journal of Information Processing, Vol.25 (2017), pp.655–666.

Ozcan, S. O., Dokeroglu, T., Cosar, A., and Yazici, A., A novel grouping genetic algorithm for the one-dimensional bin packing problem on GPU, Proceedings of International Symposium on Computer and Information Sciences (2016), pp.52–60.

Peng, K., Du, J., Lu, F., Sun, Q., Dong, Y., Zhou, P., and Hu, M., A hybrid genetic algorithm on routing and scheduling for vehicle-assisted multi-drone parcel delivery, IEEE Access, Vol.7 (2019), pp.49191–49200.

Ramos-Figueroa, O., Quiroz-Castellanos, M., Mezura-Montes, E., and Kharel, R., Variation operators for grouping genetic algorithms: A review, Swarm and Evolutionary Computation, Vol.60 (2021), 100796.

Ramos-Figueroa, O., Quiroz-Castellanos, M., Mezura-Montes, E., and Schütze, O., Metaheuristics to solve grouping problems: A review and a case study, Swarm and Evolutionary Computation, Vol.53 (2020), 100643.

Rossi, A., Singh, A., and Sevaux, M., A metaheuristic for the fixed job scheduling problem under spread time constraints, Computers & Operations Research, Vol.37, No.6 (2010), pp.1045–1054.

Stewart, J., Google tests drone deliveries in Project Wing trials. BBC (online), available from <http://www.bbc.com/news/technology-28964260>, (accessed on 6 April, 2021).

Tucker, A., Crampton, J., and Swift, S., RGFGA: An efficient representation and crossover for grouping genetic algorithms, Evolutionary Computation, Vol.13, No.4 (2005), pp.477–499.

Ülker, Ö., Korkmaz, E. E., and Özcan, E., A grouping genetic algorithm using linear linkage encoding for bin packing, Proceedings of International Conference on Parallel Problem Solving from Nature (2008), pp.1140–1149.

Zhuang, J., Wang, Y., Zhang, S., Wan, P., and Sun, C., A multi-antenna spectrum sensing scheme based on main information extraction and genetic algorithm clustering, IEEE Access, Vol.7 (2019), pp.119620–119630.