International Journal of Aeronautical & Space Sciences (2018) 19:262–277
https://doi.org/10.1007/s42405-018-0021-7

IJASS International Journal of Aeronautical and Space Sciences

ORIGINAL PAPER

Provisional-Ideal-Point-Based Multi-objective Optimization Method for Drone Delivery Problem

Hiroki Omagari[1] · **Shin–Ichiro Higashino**[1]

Received: 17 July 2017 / Revised: 7 March 2018 / Accepted: 7 March 2018 / Published online: 4 April 2018

Abstract
In this paper, we proposed a new evolutionary multi-objective optimization method for solving drone delivery problems (DDP). It can be formulated as a constrained multi-objective optimization problem. In our previous research, we proposed the "aspiration-point-based method" to solve multi-objective optimization problems. However, this method needs to calculate the optimal values of each objective function value in advance. Moreover, it does not consider the constraint conditions except for the objective functions. Therefore, it cannot apply to DDP which has many constraint conditions. To solve these issues, we proposed "provisional-ideal-point-based method." The proposed method defines a "penalty value" to search for feasible solutions. It also defines a new reference solution named "provisional-ideal point" to search for the preferred solution for a decision maker. In this way, we can eliminate the preliminary calculations and its limited application scope. The results of the benchmark test problems show that the proposed method can generate the preferred solution efficiently. The usefulness of the proposed method is also demonstrated by applying it to DDP. As a result, the delivery path when combining one drone and one truck drastically reduces the traveling distance and the delivery time compared with the case of using only one truck.

Keywords Multi-objective optimization · Genetic algorithm · Drone delivery · Provisional-ideal point

1 Introduction

1.1 Drone Delivery Service

Attempts to utilize drones for delivery services have been rapidly expanding in recent years. In 2013, as soon as Amazon CEO announced that he had been aiming to realize drone delivery services [1], major venture companies had been competing for achieving this goal [2]. Using the drones for delivery services can reduce the delivery cost and the time. However, the payload of the drone and the flight duration are strictly limited. To solve these issues, some venture companies have been developing a new delivery service combining drones and trucks [3,4]. Figure 1 is the conceptual diagram of that [5].

In this service, after arriving close to a delivery destination, a driver launches a drone with a package and makes it fly toward there. In the meantime, the driver continues to travel his route. The drone returns to the truck after releasing the package, and finally, the driver collects the drone at the next destination.

It is expected that this concept can not only realize the cost reduction but also compensate for the limited performance of the drone. In addition, the complete automation of the delivery service may be possible in the future by combining the drone and an automatic driving vehicle [6]. The construction of such a delivery system will be extremely effective and useful for emergency transport of relief supplies in the event of a natural disaster.

1.2 Drone Delivery Problem

The drone delivery problem (DDP) can be formulated to a constrained multi-objective optimization problem as follows:

$$\begin{cases} \min\limits_{\boldsymbol{x}} \quad \boldsymbol{F}(\boldsymbol{x}) = \min\limits_{\boldsymbol{x}} \; [F_1(\boldsymbol{x}), F_2(\boldsymbol{x}), \ldots, F_i(\boldsymbol{x}), \ldots, F_M(\boldsymbol{x})]^T \\ \text{subject to } g_j(\boldsymbol{x}) \le 0, \quad j = 1, 2, \ldots, m, \end{cases} \tag{1}$$

An earlier version of this paper was presented at APISAT 2017, Seoul, Korea, October 2017.

✉ Hiroki Omagari
tail.sitter.presence@gmail.com

[1] Department of Aeronautics and Astronautics, Kyushu University, Fukuoka 819-0395, Japan

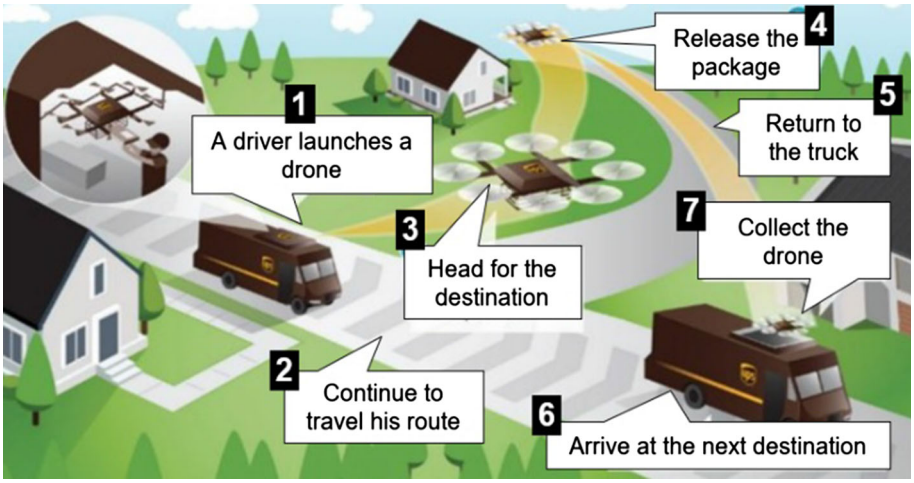


Fig. 1 Conceptual diagram of the drone delivery service

where $\boldsymbol{x}$ is a vector of design variables, F_i ($\boldsymbol{x}$) is ith objective function, M is the number of objective functions, g_j ($\boldsymbol{x}$) is jth constraint condition, and m is the number of constraint conditions. In DDP, the fuel cost of the delivery truck, the traveling time, the quality of the delivery services, etc. can be considered as F_i ($\boldsymbol{x}$). Drone's loadable weight or flight duration can be regarded as g_j ($\boldsymbol{x}$). $\boldsymbol{x}$ represents the delivery plan or route. DDP is typically formulated as some variant of TSP with various constraints [2]. Since TSP is already NP-hard, DDP is often formulated as an NP-hard problem, so it is expected that many computational resources are required to solve the problem.

The outline of this study is as follows. In Sect. 2, we introduce some studies related to DDP. In Sect. 3, we describe the detail of the proposed method and show the advantage of it. In Sect. 4, we show the usefulness of the proposed method by applying it to some benchmark problems. In Sect. 5, we evaluate the benefits of utilizing the drone in delivery service by applying the proposed method to DDP. In Sect. 6, we discuss the conclusion of this paper and future works.

2 Background

2.1 Related Works

Murray et al. first mentioned DDP as the flying sidekick traveling salesman problem (FSTSP) for the first time [2]. FSTSP is a new delivery concept that combining the drone and the truck for delivering packages to customers. They formulated FSTSP as a mixed integer programming problem and solved these problems by applying a solver called "Gurobi" or some simple heuristics. By comparing and analyzing the calculation results, they reported that it has a trade-off relationship between the drone speed and its endurance. In addition, they also referred to the form called "parallel drone scheduling TSP (PDSTSP)". This problem assumed a scenario that the drone delivers a package to one of the customers who lives near the depot, and the conventional truck is used for the others who live where the drone cannot use because of its battery restriction. The drone departing from the depot returned after delivering the package. This paper suggested that the benefit of utilizing the drone is greater in PDSTSP than in FSTSP in the problem set. Mathew et al. proposed a method for solving DDP by converting it into a generalized traveling salesman problem (GTSP) [7]. In this study, they showed that the truck moves to the vicinity of the destination and the drone delivers the package by flying remaining distance, thereby reducing not only the fuel consumption of the truck but also the time required for delivery. Dorling et al. showed that there was a trade-off relationship between the delivery cost and the time required for the delivery using simulated annealing (SA) heuristic [8].

What is common to those studies is that DDP has been solved as a single-objective optimization problem. However, as mentioned earlier, DDP should be considered as a constrained multi-objective optimization problem. Moreover, there is little time for planning the delivery schedules at the actual site. Unfortunately, there are few studies to solve these problems so far.

2.2 Genetic Algorithm

The genetic algorithm (GA) is one of the evolutionary computations (EC) advocated by John H. Holland, Michigan University in 1975 [9]. GA imitates the evolutionary process of living organisms, and it can calculate a suboptimal solution easily in a short time without being limited to a specific problem. It is known that GA is an effective method for solving NP-hard problems. Therefore, it can be said that it is suitable for solving DDP. A typical application example of GA is TSP. It is a problem of finding the shortest passing order of N cities without overlapping. When GA is applied to TSP, a suboptimal solution is obtained by considering the traveling path as

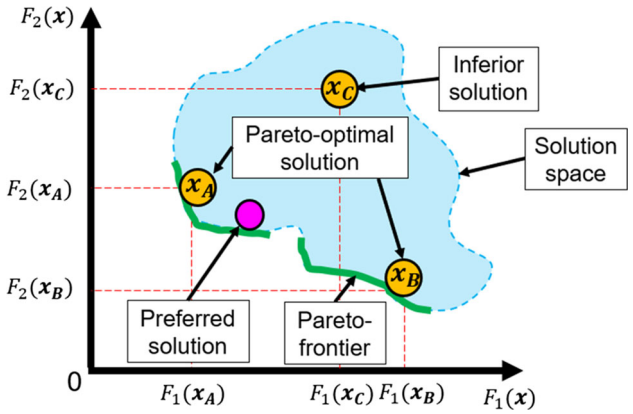


Fig. 2 Conceptual diagram of the multi-objective optimization

one individual and repeating three genetic operations called selection, crossover, and mutation. Meanwhile, in DDP, individuals need to include its task assignment, passage order of customers, and location of the take-off and landing waypoint (WP) of drone in the solution.

2.3 Multi-objective Optimization

To visualize concepts related to multi-objective optimization, we consider the following expression:

$$\min_{\boldsymbol{x}} \boldsymbol{F}(\boldsymbol{x}) = \min_{\boldsymbol{x}} \left[F_1(\boldsymbol{x}), F_2(\boldsymbol{x})\right]^T. \tag{2}$$

Equation (2) means searching for $\boldsymbol{x}$ to minimize F_1 and F_2 simultaneously. Figure 2 is the conceptual diagram of that problem. It assumed that there are three solutions $\boldsymbol{x_A}$, $\boldsymbol{x_B}$ and $\boldsymbol{x_C}$ in the solution space. The relationship between $\boldsymbol{x_A}$ and $\boldsymbol{x_B}$ is $F_1(\boldsymbol{x_A}) < F_1(\boldsymbol{x_B})$ and $F_2(\boldsymbol{x_A}) > F_2(\boldsymbol{x_B})$. In this case, we cannot distinguish the superiority or inferiority of the two solutions. A solution which has such the relationship with any other is called "Pareto-optimal solution". On the other hand, the relationship between $\boldsymbol{x_A}$ and $\boldsymbol{x_C}$ is $F_1(\boldsymbol{x_A}) < F_1(\boldsymbol{x_C})$ and $F_2(\boldsymbol{x_A}) < F_2(\boldsymbol{x_C})$. At this time, $\boldsymbol{x_C}$ is inferior to $\boldsymbol{x_A}$, so a solution like this is called "inferior solution". In this way, the generated solutions in the multi-objective problem can be divided into the "Pareto-optimal solution" or the "inferior solution". Thus, there is no solution which can optimize all objective functions simultaneously. The green lines are formed by a set of the Pareto-optimal solutions, which is called "Pareto-frontier". The Pareto-optimal solutions have a trade-off relationship with each other. A decision maker (DM) needs to choose only one solution from them. The selected solution is called "preferred solution", which, is chosen, depends on the preference information of the DM. One of the purposes in a multi-objective problem is to obtain the preferred solution.

Numerous prior studies on multi-objective optimization method have focused mainly on the generation of the Pareto-frontier [10]. Meanwhile, there are few papers mentioning the method for generating the preferred solution for the DM. For these reasons, we proposed a multi-objective optimization method which is called "Aspiration-point-based method" to solve this problem in the past research [11]. In this method, we define a virtual solution named "ideal point" which can optimize all the objective functions at the same time. The ideal point is obtained by optimizing each objective function independently. Then, we repeat generating and selecting a solution which is the closest to the ideal point within the pre-specified constraint conditions until the calculation condition is satisfied. We confirmed that the method could generate the preferred solution efficiently and with high probability compared with the weighted sum method [12].

However, this method does not consider the constraint conditions except for the objective functions, so the application scope is limited. Moreover, it needs to calculate the optimal value of each objective function in advance. For these reasons, it cannot be applied to solve DDP.

3 Proposed Method

We propose the "Provisional-Ideal-Point-Based Method" to solve those problems as mention above. The proposed method consists of two steps. The first one is to define "Penalty Value" to eliminate the limited application scope. The second one is to use "Provisional-Ideal Point" to omit the pre-calculation. The details of the proposed method are as follows.

Figure 3 is an example of the visualized constraint conditions. In this figure, if the value of $g_j(\boldsymbol{x})$ is in the blue area, it can be said that the constraint condition is satisfied. Conversely, if the value of $g_j(\boldsymbol{x})$ is in the red area, it can be said that the constraint condition is not satisfied. In other words, it becomes possible to eliminate the limited application scope by proposing a method that can generate a solution whose all $g_j(\boldsymbol{x})$ are included in the blue area.

To realize above, we define the penalty values $P(\boldsymbol{x})$ and $P_j(\boldsymbol{x})$ in the first step as follows:

$$P(\boldsymbol{x}) = \sum_{j=1}^{m} P_j(\boldsymbol{x}) \quad \text{s.t.} \begin{cases} P_j(\boldsymbol{x}) = g_j(\boldsymbol{x}), & g_j(\boldsymbol{x}) > 0 \\ P_j(\boldsymbol{x}) = 0, & g_j(\boldsymbol{x}) \le 0. \end{cases} \tag{3}$$

Then, we search for a feasible solution whose $P(\boldsymbol{x})$ equals to 0 using GA. At this time, no problem arises even if $P_j(\boldsymbol{x})$ having different units are handled as scalar values, because the purpose of this step is to search for the feasible solution whose $P(\boldsymbol{x})$ equals to 0. If the evolution of the solution stag-

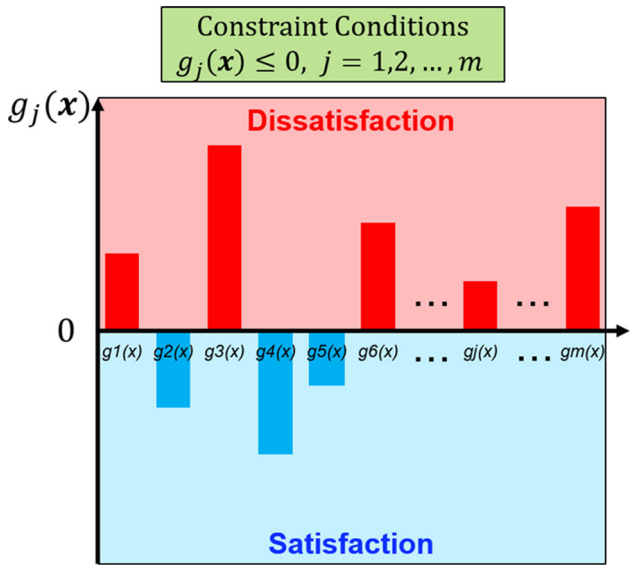


Fig. 3 Conceptual diagram of the constraint conditions

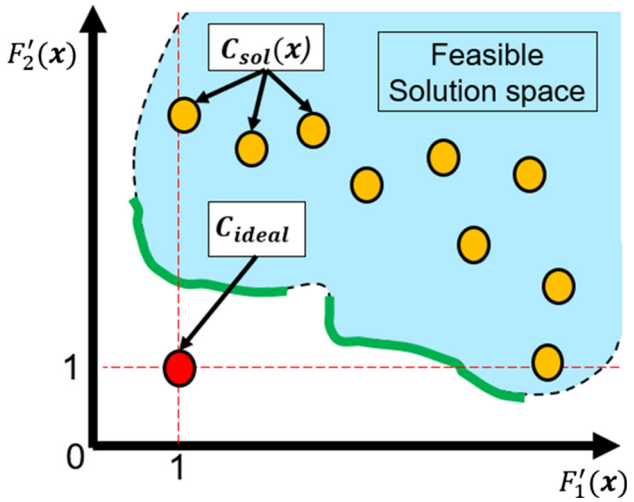


Fig. 4 Conceptual diagram of $\boldsymbol{C}_{\mathbf{sol}}(\boldsymbol{x})$ and $\boldsymbol{C}_{\mathbf{ideal}}$

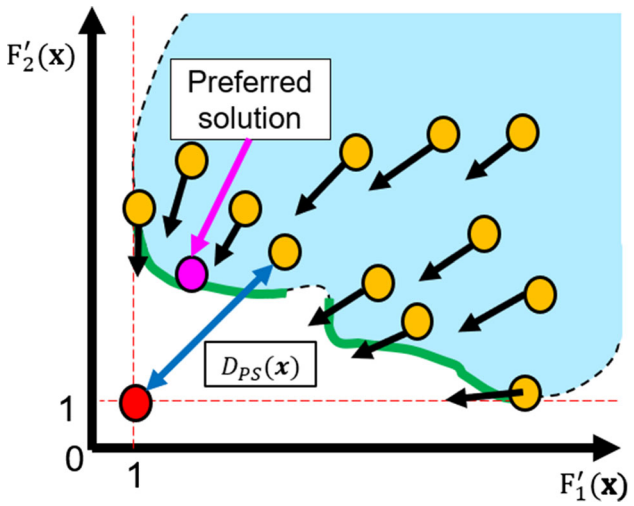


Fig. 5 Searching for the preferred solution

nates without generating a feasible solution, it requires to relax the constraint conditions or changes the problem setting. On the other hand, if one feasible solution is obtained, we move on to next step.

In the second step, we define solution point $\boldsymbol{C}_{\mathbf{sol}}(\boldsymbol{x})$ and provisional-ideal point $\mathbf{C}_{\mathbf{ideal}}$ as follows:

$$\begin{aligned}\boldsymbol{C}_{\mathbf{sol}}(\boldsymbol{x}) &= \left[\frac{F_1(\boldsymbol{x})}{F_{1\_\mathrm{pro}}}, \frac{F_2(\boldsymbol{x})}{F_{2\_\mathrm{pro}}}, \ldots, \frac{F_i(\boldsymbol{x})}{F_{i\_\mathrm{pro}}}, \ldots, \frac{F_M(\boldsymbol{x})}{F_{M\_\mathrm{pro}}}\right]^T \\ &= \left[F_1'(\boldsymbol{x}), F_2'(\boldsymbol{x}), \ldots, F_i'(\boldsymbol{x}), \ldots, F_M'(\boldsymbol{x})\right]^T \quad (4)\\ \boldsymbol{C}_{\mathbf{ideal}} &= \left[\frac{F_{1\_\mathrm{pro}}}{F_{1\_\mathrm{pro}}}, \frac{F_{2\_\mathrm{pro}}}{F_{2\_\mathrm{pro}}}, \ldots \frac{F_{i\_\mathrm{pro}}}{F_{i\_\mathrm{pro}}} \cdots \frac{F_{M\_\mathrm{pro}}}{F_{M\_\mathrm{pro}}}\right]^T \\ &= [1, 1, \ldots, 1, \ldots 1]^T, \quad (5)\end{aligned}$$

where F_{i_pro} is the provisional minimum value of $F_i(\boldsymbol{x})$ obtained by the solution search process so far. The value of F_{i_pro} is assigned as the same as $F_i(\boldsymbol{x})$ if it is greater than $F_i(\boldsymbol{x})$. $\boldsymbol{C}_{\mathbf{sol}}(\boldsymbol{x})$ are generated by normalizing the generated feasible solutions. On the other hand, $\boldsymbol{C}_{\mathbf{ideal}}$ is a virtual solution which optimizes each objective function $F_i(\boldsymbol{x})$ to F_{i_pro} simultaneously. The notation of the dash attached to $F_i(\boldsymbol{x})$ means that it is dimensionless. Figure 4 is a conceptual diagram in which these coordinate points are plotted in $F_1'(\boldsymbol{x}) - F_2'(\boldsymbol{x})$ non-dimensional coordinate system.

Then, the multi-objective optimization can be realized by searching for solution points which minimize a distance between $\boldsymbol{C}_{\mathbf{ideal}}$ and $\boldsymbol{C}_{\mathbf{sol}}(\boldsymbol{x})$ as follows:

$$\min_{\boldsymbol{x}} D_{\mathrm{PS}}(\boldsymbol{x}) = \min_{\boldsymbol{x}} \left|\boldsymbol{C}_{\mathbf{ideal}} - \boldsymbol{C}_{\mathbf{sol}}(\boldsymbol{x})\right|. \quad (6)$$

In this way, the generated feasible solutions $\boldsymbol{C}_{\mathbf{sol}}(\boldsymbol{x})$ close to $\boldsymbol{C}_{\mathbf{ideal}}$ are inherited to the next generation preferentially. Of course, there would be some cases that the population includes some infeasible solutions whose penalty value is not 0, but all of them will be given lower priority than $\boldsymbol{C}_{\mathbf{sol}}(\boldsymbol{x})$. Therefore, it is possible which not only prevents the feasible solution from being lost in the second step, but also realizes multi-objective optimization simultaneously. In addition, it can be said that the second step can omit the pre-calculation which is required in the previous method [11].

Figure 5 shows a conceptual diagram of generating the preferred solution for the DM by minimizing $D_{\mathrm{PS}}(\boldsymbol{x})$. This method is based on the solution search technique called "Proximate Optimality Principle (POP)" [13].

Table 1 Calculation conditions

Items	Values
Number of individuals	100
Elite population	5
Crossover probability	80%
Mutation probability	1%
Max number of generation	3000

Table 2 Simulation result

Problem	Optimal value	Proposed method	Calculation time (s)
GO1	− 15.0000	− 14.7338	821.2191
GO2	0.8036	0.7103	1309.1
GO3	1.0000	0.9978	614.9905
GO4	− 30,665	− 30,636	45.0231
GO5	5126.5	5151.2	252.5760
GO6	− 6961.8	− 6925.7	26.0341
GO7	24.3062	26.4786	58.9143
GO8	0.0958	0.0958	0.8372
GO9	680.6300	681,2954	40.6823
GO10	7049.3	7262.0	502.1607

4 Benchmark Test

To verify the usefulness of the proposed method, we used two kinds of benchmark test problems called "GO problems" [14] and "DTLZ problem [15]". The GO problems can be used to confirm the capability of generating feasible solutions. Meanwhile, the DTLZ problems can be used to confirm the capability of the multi-objective optimization. Therefore, we applied the proposed method to both test problems, respectively, and verified the two performances of the proposed method. Note that there is no need to normalize the objective functions given in the two benchmark test problems, because these are originally dimensionless.

In this paper, we used the "simple genetic algorithm (SGA)" [16] to apply the proposed method. The numerical simulation was done using MATLAB ver. 2016b. The solutions were computed on a laptop computer with Intel Core i7-4790 3.60GHz CPU with 8GBytes of RAM.

4.1 Global Optimization Problem

The global optimization (GO) problems [14] are constrained single-objective optimization problems. The optimal values and that of the design variables are known in advance. We used ten kinds of the test problems among them. Table 1 shows the calculation conditions.

We also assumed that each solution search process was terminated when it reached the maximum number of generations or when the deviation from the optimal solution became 0.001% or less. Table 2 shows the comparison between the results obtained using the proposed method and the optimal solution.

The results show that the proposed method can generate the suboptimal solutions in all benchmark problems within at least 20 min. Figure 6a–j shows the transition of each evaluation function value and the penalty value, respectively. Magenta lines represent the cases where the penalty values are larger than 0. Blue lines represent the cases where the values are 0. The red lines are already known as optimal values for each benchmark problem.

As can be seen from these figures, the proposed method first searches for feasible solutions that satisfy all of the constraint conditions. Then, it searches for the optimal value while satisfying the constraint conditions. In Fig. 6b, h, since the feasible solutions were generated at the first generation, the penalty values were 0 from the first generation and kept constant. These results show that the proposed method can search for the optimal solution while satisfying all constraint conditions.

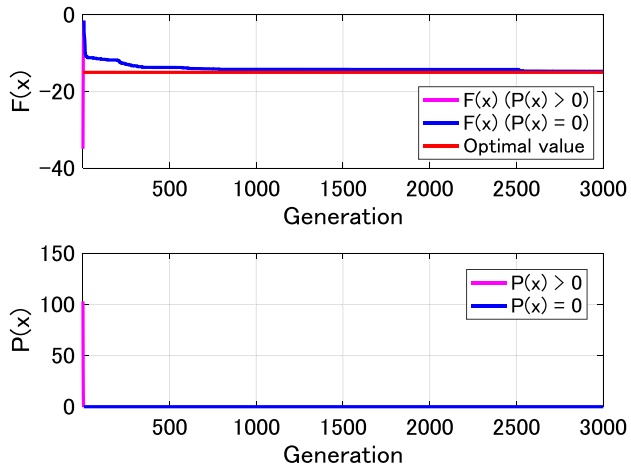
4.2 DTLZ Problems

We also used some scalable benchmark test problems known as "DTLZ problems"[15]. The DTLZ problems have frequently been used to evaluate the performance of the multi-objective optimization methods [17]. The test problems can freely change the number of objective functions and the number of design variables. In addition, the shape of the Pareto-frontier is known in advance, so that we can calculate the Euclidean distance from the solution point to the Pareto-frontier. We used the three benchmark test problems called "DTLZ1," "DTLZ3," and "DTLZ7". The Pareto-frontier shape of DTLZ1 is linear, DTLZ3 is a fan shape, and DTLZ7 is a discontinuous shape, as shown in Figs. 7, 8, and 9.

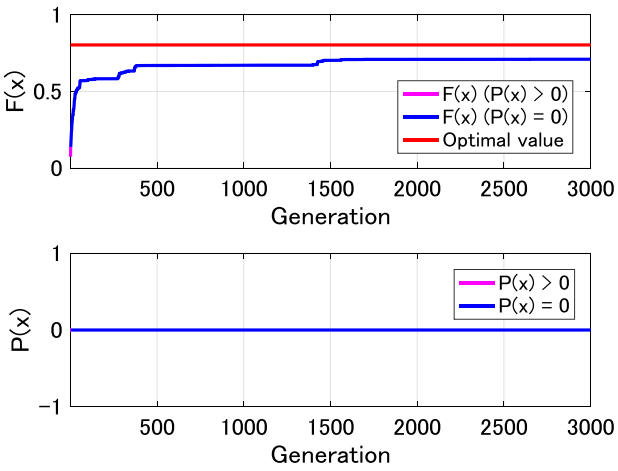
Since any DTLZ problems originally are not given the constraint condition, so we arbitrary gave those as shown in Table 3, those conditions are drawn with blue translucent regions as shown in Figs. 7, 8, and 9. In other words, it can be said that these regions represent the feasible region of the test problems.

The other calculation conditions were given as below.

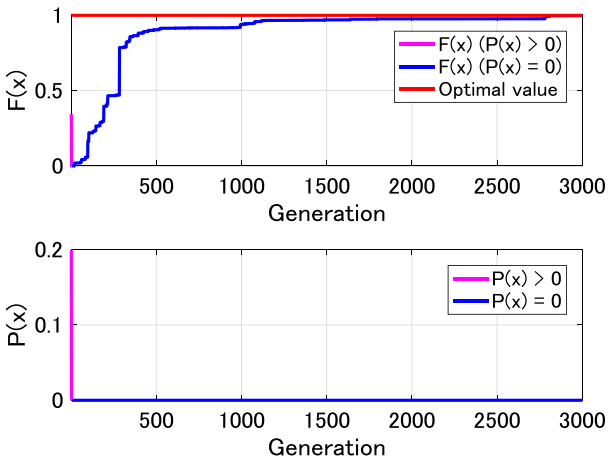
The objective functions of DTLZ problems are originally dimensionless, so there was also no need for normalization process when we applied the proposed method to DTLZ problems. In addition, since the minimum value of each objective function is 0, we set the coordinates of the provisional-ideal point exceptionally to $\boldsymbol{C}_{\mathbf{ideal}} = \left[F_{1\_\mathrm{pro}}, F_{2\_\mathrm{pro}}\right]^T$ to avoid the definition of that becoming



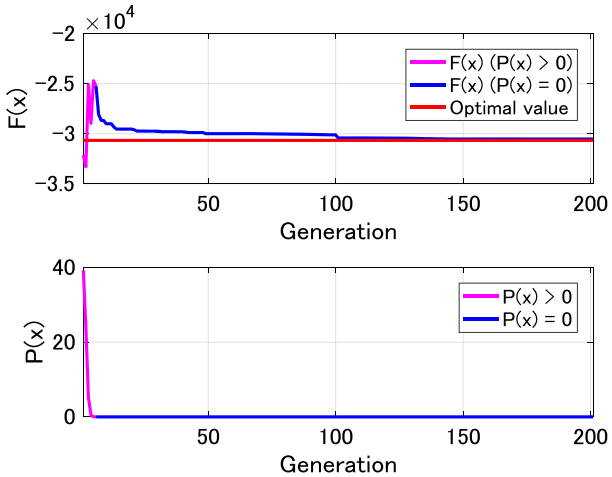
(a) GO 1



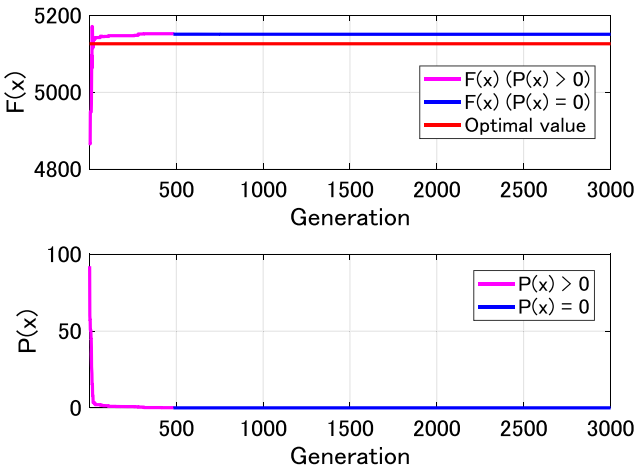
(b) GO 2



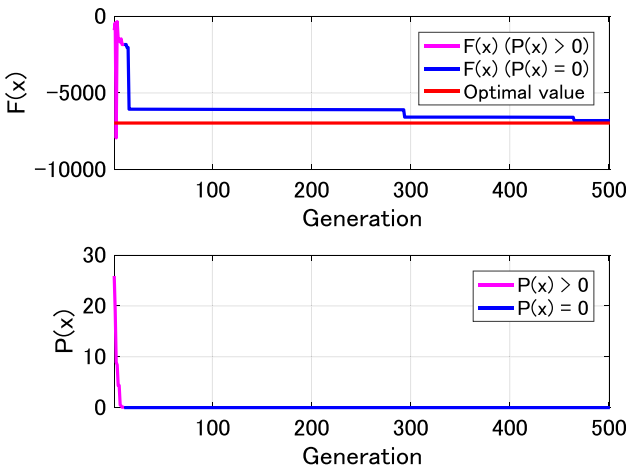
(a) GO 3



(b) GO4

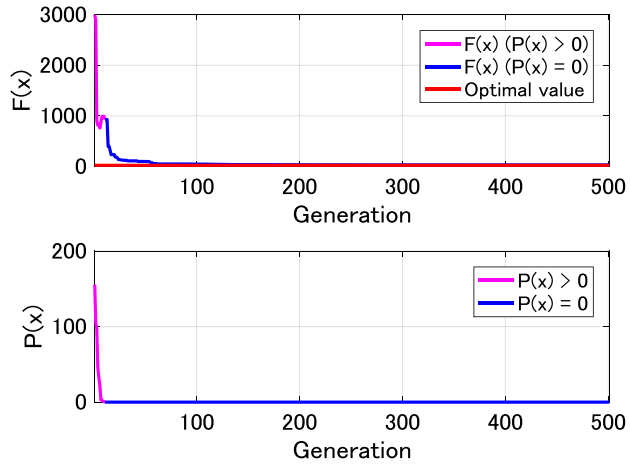


(a) GO 5

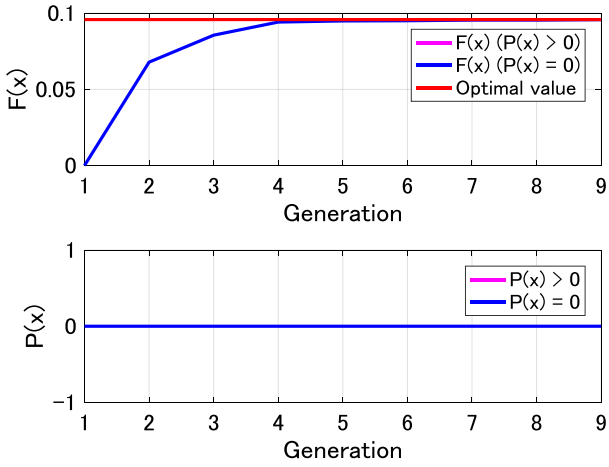


(b) GO 6

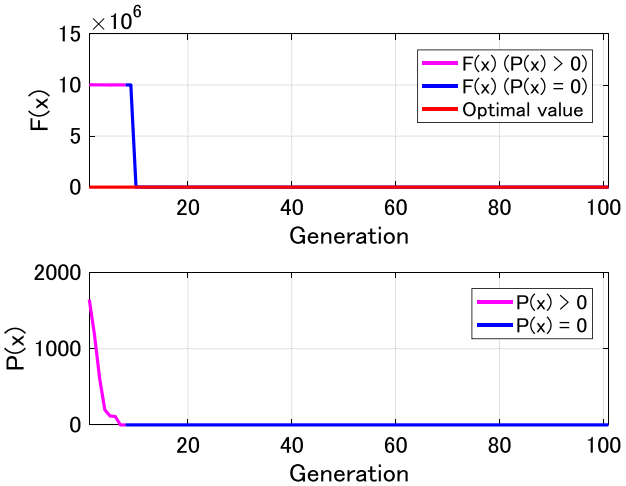
Fig. 6 Simulation results of GO problems



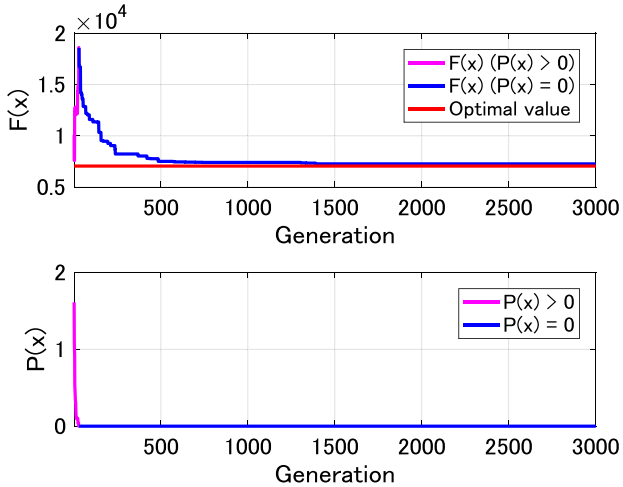
(a) GO7



(b) GO8

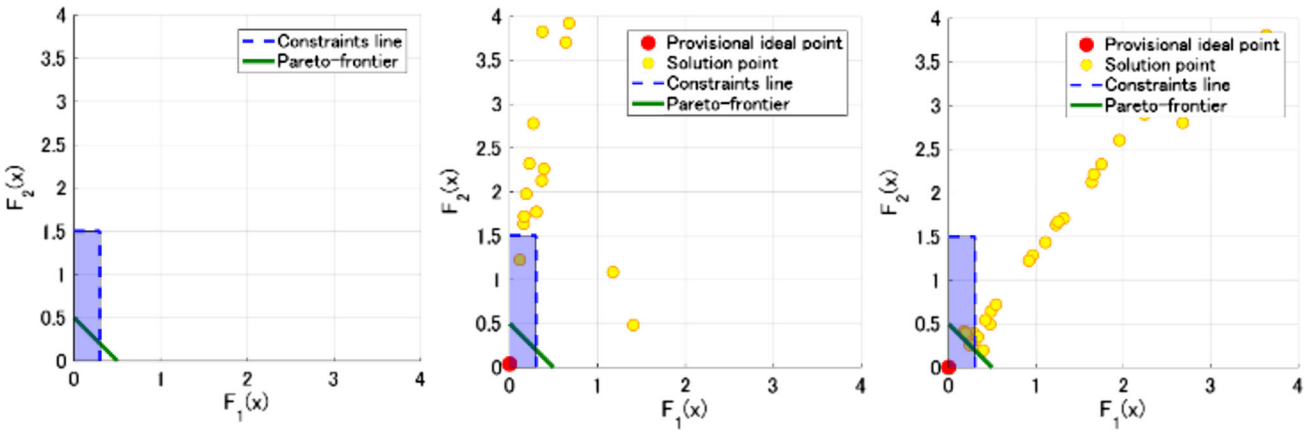


(a) GO9



(b) GO10

Fig. 6 continued



(a) State 1 (b) State 2 (c) State 3

Fig. 7 DTLZ1

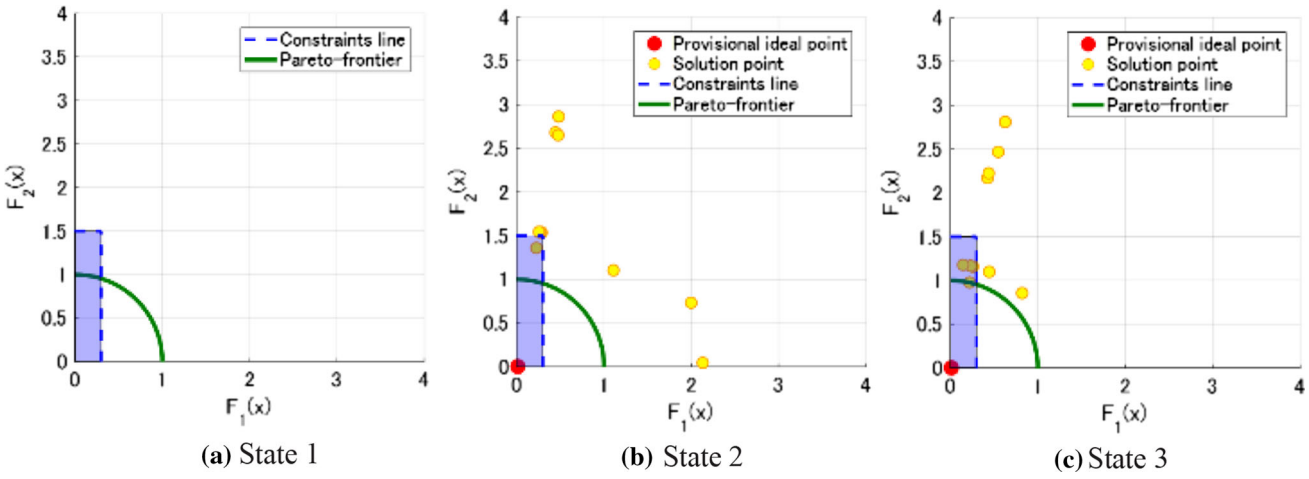


Fig. 8 DTLZ3

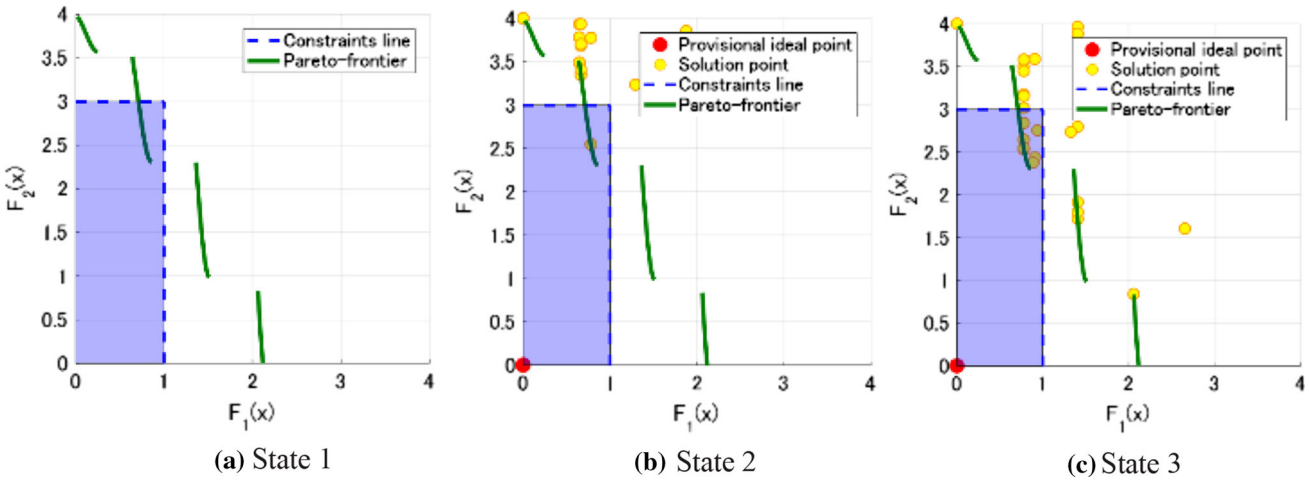


Fig. 9 DTLZ7

Table 3 Constraint conditions

Types of DTLZ	Given constraint conditions
DTLZ1	$g_1(\boldsymbol{x}) = F_1(\boldsymbol{x}) - 1.5 < 0$
DTLZ3	$g_2(\boldsymbol{x}) = F_2(\boldsymbol{x}) - 0.3 < 0$
DTLZ7	$g_1(\boldsymbol{x}) = F_1(\boldsymbol{x}) - 3 < 0$
	$g_2(\boldsymbol{x}) = F_2(\boldsymbol{x}) - 4 < 0$

indeterminate form. We also assumed that each solution search process was terminated when it reached the maximum number of generations or when the preferred solution was generated on the Pareto-frontier.

Figures 7a, 8a, 9a represent the states before starting the solution search (State 1); and Figs. 7b, 8b, 9b show just after generating the feasible solution (State 2). Figures 7c, 8c, 9c are just after generating the preferred solution on the Pareto-frontier (State 3). The red marker indicates the position of the provisional-ideal point. The yellow marker represents the position of the solution point.

As can be seen in these figures, it was confirmed that the provisional-ideal point had been generated at the origin when the feasible solutions were found. It was also shown that the proposed method was able to generate the solution points on the Pareto-frontier within the constraint conditions in all cases. The calculation times until the search processes were terminated were 56.32 s on DTLZ1, 50.93 s on DTLZ3, and 4.21 s on DTLZ7, respectively. Therefore, it can be said that the proposed method can obtain the preferred solution without any pre-calculation. It can also be said that the computational load of the proposed method is relatively small, because the calculation times required for searching these solutions were less than 1 min in any cases.

4.3 Advantages of the Proposed Method

The advantages of the proposed method are summarized as follows:

Table 4 Calculation conditions

Items	Values
Number of individuals	200
Elite population	30
Crossover probability	100 (%)
Mutation probability	1 (%)
Max number of generation	300

- Even if many constraint conditions are given, the feasible solutions can be generated easily and efficiently by searching for a solution whose penalty value equals to 0.
- There is no need to pre-calculate the optimal values of each objective function unlike the previous method [11].
- It is possible to efficiently generate the preferred solutions for the DM by searching for solutions which are close to the provisional-ideal point preferentially.

5 Drone Delivery Problem

In this section, we describe how to apply the proposed method to DDP which assumed FSTSP. We also show the results of the numerical simulation. The computing environment is the same as Sect. 4. The calculation conditions are shown in Table 4.

5.1 Problem Setting

As mentioned above, we assume a drone delivery service, as shown in Fig. 1. The target area considered in this paper is a part of Japan. The map was excerpted from the Google map. Figure 10 shows the map of the delivery area. We set passable WPs to the roads, delivery destinations, corners, intersections, etc., as shown in this figure.

5.1.1 Parameter Definition

The set of passable WPs is represented by $WP_{\text{passible}} = [w_1, w_2, \ldots, w_N]$. Assume that the number of delivery destinations is n and these sets are expressed by $C = [C_1, C_2, \ldots, C_n]$, $C \subset WP_{\text{passible}}$. In addition, Let $C' = [C'_1, C'_2, \ldots, C'_{n'}]$, $C' \subset C$ be the delivery destinations that need to carry a parcel exceeding the loadable weight of the drone, $C'' = [C''_1, C''_2, \ldots, C''_{n''}]$, $C'' \subset C$ be the places where the time zone of home delivery is preset by customer.

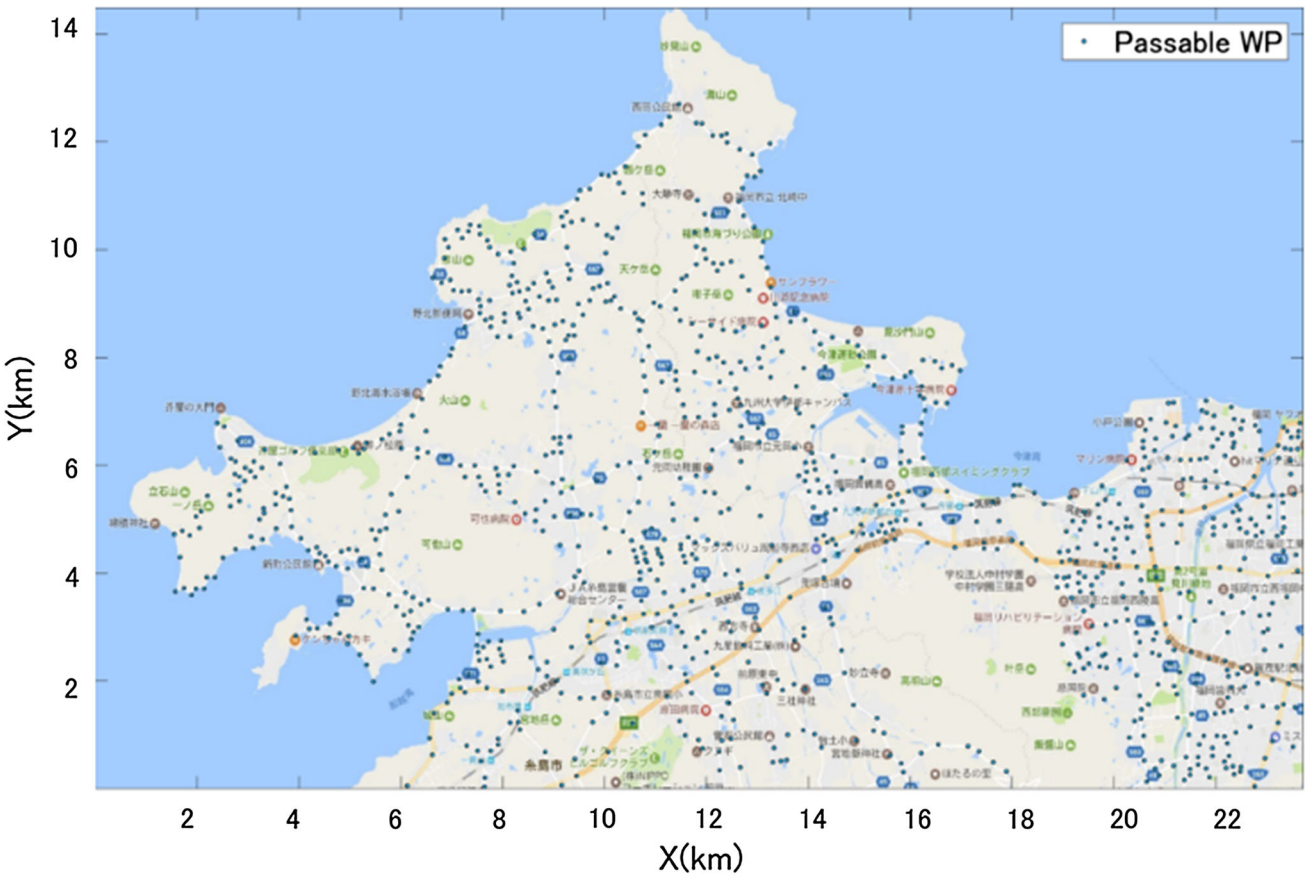


Fig. 10 Map of the delivery area

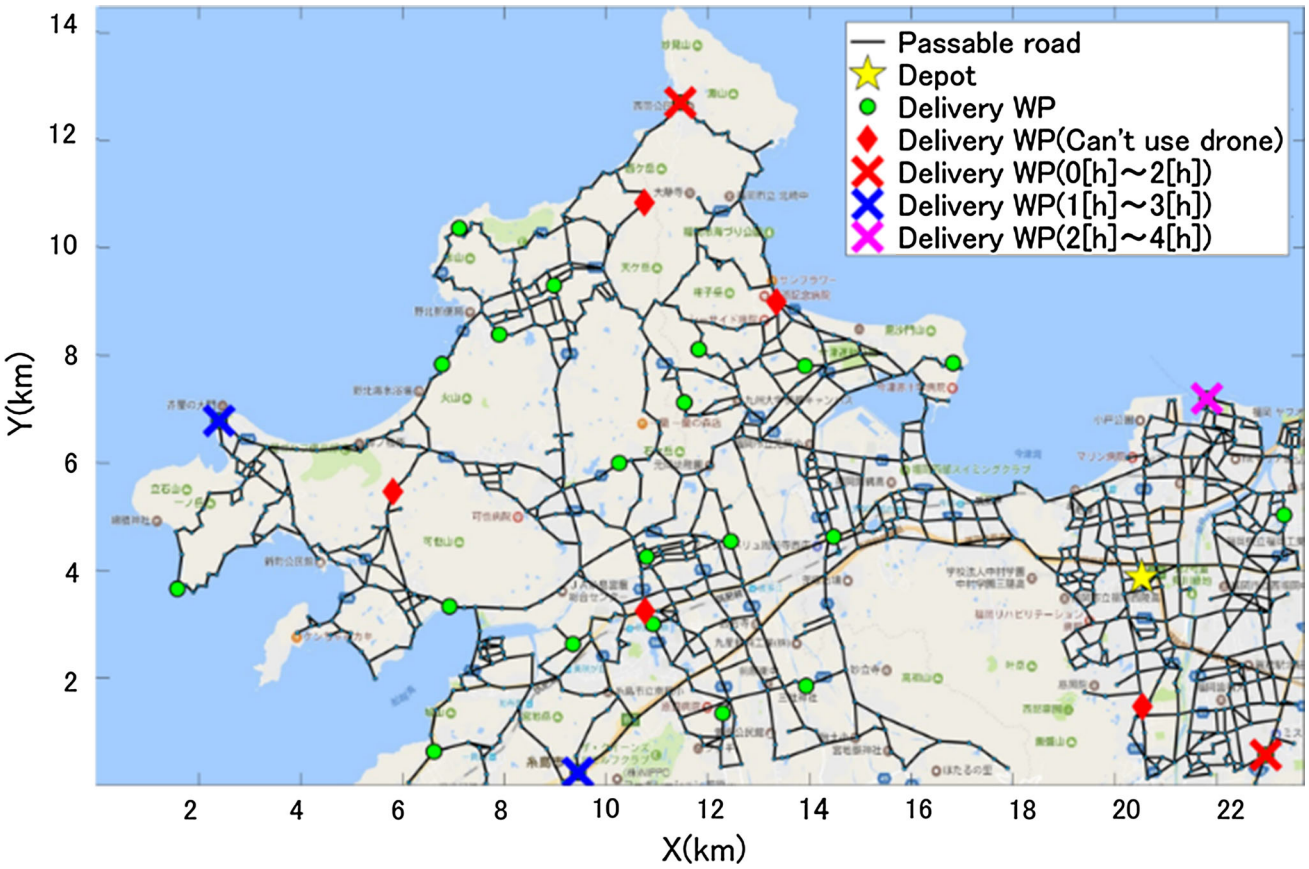


Fig. 11 Location of the delivery WPs

The location of the delivery base the so-called "depot" is arbitrarily selected from WP_{passible}, and it is represented by L_{depo}, $L_{\text{depo}} \in WP_{\text{passible}}$.

The number of drones and trucks used for the delivery service is represented by N_{drone} and N_{truck}, respectively. Flight velocity of the drone is notated by V_{drone}, Loadable weight of the drone is W_{drone}, the duration is t_{duration}, the launching time is t_{launch}, the landing time is t_{landing}, and the time required for changing the drone's battery and loading parcel is t_{load}. The traveling velocity is notated by V_{truck}. The time at which the driver parks the truck to deliver the parcel at the delivery destination is represented by t_{driver}.

We also define C_{WP}, S_{path} and P_{cost} to shorten the time required to generate a delivery path in DDP:

$$C_{\text{WP}} = \begin{bmatrix} 0 & c_{12} & \cdots & c_{1N} \\ c_{21} & 0 & & c_{2N} \\ \vdots & & \ddots & \vdots \\ c_{N1} & c_{N2} & \cdots & 0 \end{bmatrix},$$

$$\begin{pmatrix} \text{WP } \alpha \text{ and WP } \beta \text{ are adjacent} & \Rightarrow c_{\alpha\beta} = c_{\beta\alpha} = 1 \\ \text{Otherwise} & \Rightarrow c_{\alpha\beta} = c_{\beta\alpha} = 0 \\ \alpha, \beta = 1, 2, \ldots, N, \alpha \neq \beta & \end{pmatrix} \tag{7}$$

$$S_{\text{path}} = \begin{bmatrix} [\,] & \text{Path}_{12} & \cdots & \text{Path}_{1N} \\ \text{Path}_{21} & [\,] & & \text{Path}_{2N} \\ \vdots & & \ddots & \vdots \\ \text{Path}_{N1} & \text{Path}_{N2} & \cdots & [\,] \end{bmatrix}, \tag{8}$$

$$P_{\text{cost}} = \begin{bmatrix} 0 & \text{Cost}_{12} & \cdots & \text{Cost}_{1N} \\ \text{Cost}_{21} & 0 & & \text{Cost}_{2N} \\ \vdots & & \ddots & \vdots \\ \text{Cost}_{N1} & \text{Cost}_{N2} & \cdots & 0 \end{bmatrix}, \tag{9}$$

where we call C_{WP} "adjacency matrix", S_{path} "shortest path matrix", and P_{cost} "distance cost matrix". C_{WP} assumes that $c_{\alpha\beta} = 1$ if WP α and WP β are adjacent to each other, and $c_{\alpha\beta} = 0$ otherwise. The shortest path $\text{Path}_{\alpha\beta}$ between WP α and WP β is calculated by applying the "A* algorithm" [18] to C_{WP}, and all $\text{Path}_{\alpha\beta}$ are stored in S_{path}. The distance cost $\text{Cost}_{\alpha\beta}$ between WP α and WP β is calculated using $\text{Path}_{\alpha\beta}$ and all $\text{Cost}_{\alpha\beta}$ are stored in P_{cost}. In this way, we can speed up the evaluation of generated solution using P_{cost} when applying GA to DDP. In addition, S_{path} can be used when drawing the finally obtained delivery path on the map. The above method is based on the concept of the "A*-EC algorithm" [19].

Figure 11 represents some passable roads with the black lines by connecting these passable WPs. In addition, the yellow star represents the location of the depot. These green circles are the delivery WPs. These red diamond markers represent the delivery WPs where exceeds the loadable weight of the drone. The cross markers mean the delivery WPs where customers have specified the time zone of the delivery service.

5.1.2 Assumption

The following conditions are assumed:

(1) Only one package can be loaded on the drone.
(2) After completing the delivery of the package, the drone heads to a pre-specified rendezvous WP.
(3) The driver will not move to the next delivery WP until he collects the drone at the rendezvous WP.
(4) A WP to launch the drone can be selected from any passable WPs, but a WP to collect the drone are limited to the delivery WPs.
(5) The driver must have the drone when the truck returns to the depot.
(6) The number of times which the driver can launch or collect a drone at the same WP is limited to once, respectively.
(7) Parcels exceeding the loadable weight of the drone must be transported by the truck.
(8) The battery of the drone must be exchange each time when the driver collects the drone at the rendezvous WP.
(9) The drone rises in the vertical direction when it is launched and moves in the horizontal flight. Then, it descends vertically toward the rendezvous WP.
(10) The drone can fly autonomously.
(11) Related laws of drones, obstacle avoidance, weather conditions, and influences on road conditions such as traffic congestion or accidents are not considered.
(12) All parameter values (i.e., V_{drone}, V_{truck}, W_{drone}, t_{duration}, t_{launch}, t_{landing}, t_{load}, and t_{driver}) are constant values.

5.1.3 Individual Expression

The solution structure of DDP in this paper consists of three kinds of chromosomes called "D_{WP}", "T_{WP}" and "R_N". D_{WP} represents the delivery WPs assigned to the drone. T_{WP} represents the passage order of the delivery WP assigned to the truck. R_N has the same individual length as that of T_{WP}, and it is expressed by the number of 0 or 1. The place where the element of R_N becoming 1 means one of the rendezvous WPs. The drone heads to the rendezvous WP from the corresponding delivery WP after releasing the package. On the other hand, the WP to launch the drone can be automatically determined by selecting the place where the time lag when the truck and the drones merge in the rendezvous WP becomes the smallest. The notations D_{WP}, T_{WP} and R_N are shown below:

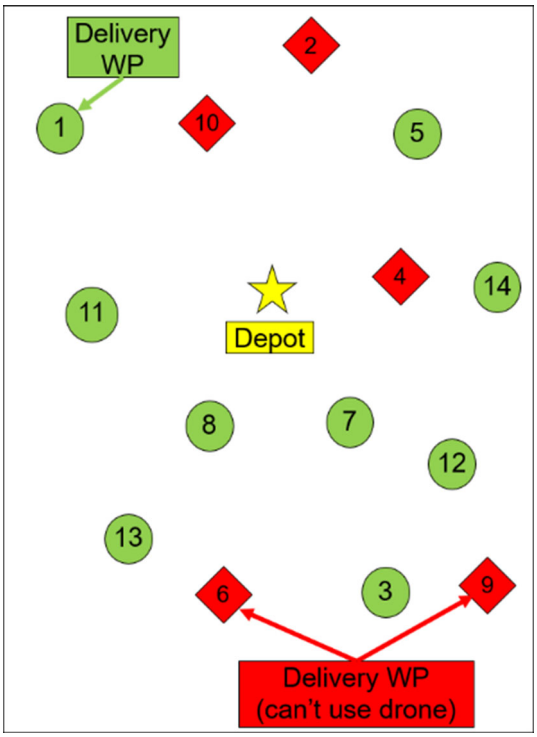


Fig. 12 Sample problem of DDP

$$D_{\text{WP}} = \left[D_{\text{WP}_1}, D_{\text{WP}_2}, \ldots, D_{\text{WP}_i}, \ldots, D_{\text{WP}_{D_{\text{all}}}}\right], \tag{10}$$

$$D_{\text{WP}_i} = \left[D_{\text{launch}_i}, C_{D_i}, D_{\text{land}_i}\right], \quad \forall D_{\text{launch}_i} \in WP_{\text{passible}}, \forall C_{D_i}, \forall D_{\text{land}_i} \in C, \tag{11}$$

$$T_{\text{WP}} = \left[L_{\text{depo}}, C_{T_{\text{WP}1}}, C_{T_{\text{WP}2}}, \ldots, C_{T_{\text{WP}j}}, \ldots, C_{T_{\text{WP}T_{\text{all}}}}, L_{\text{depo}}\right], \quad \forall C_{T_{\text{WP}j}} \in C, \tag{12}$$

$$R_N = \left[r_1, r_2, \ldots, r_j, \ldots, r_{T_{\text{all}}}\right], \tag{13}$$

$$r_j = \begin{cases} 1 & \text{the path gose from } C_{D_i} \text{ to } D_{\text{land}_i} \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

where D_{all} is total number of WP where delivered by using the drone, D_{launch_i} is ith launching WP of the drone, C_{D_i} is ith place where delivered by the drone, D_{land_i} is ith landing WP of the drone, $C_{T_{\text{WP}j}}$ is jth place where delivered by the truck, and T_{all} is total number of WP where delivered by using the truck.

Figure 12 shows a sample problem of DDP which has 14 delivery WPs, and Fig. 13 describes an example of "D_{WP}", "T_{WP}", and "R_N", respectively. Figure 14 is a part of the solution structure in Fig. 13, and it shows the diagram of the above concept. Figure 15 represents the entire delivery path

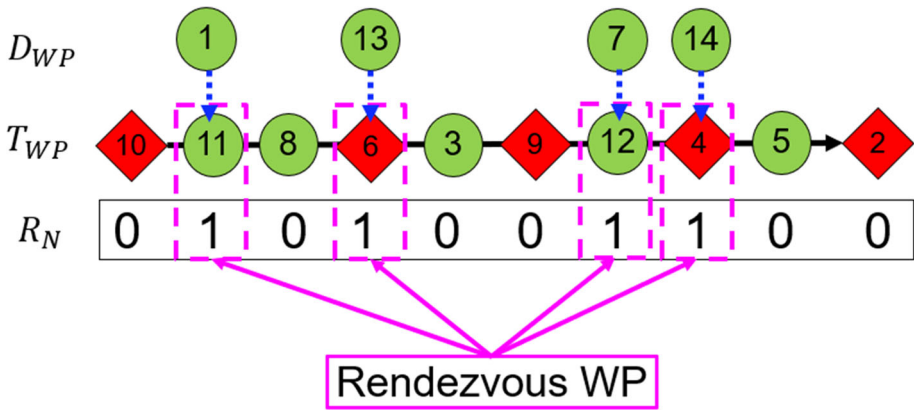


Fig. 13 Solution structure of DDP

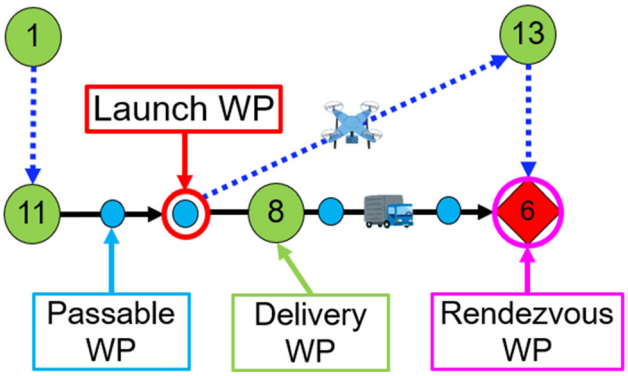


Fig. 14 Part of the solution structure in Fig. 13

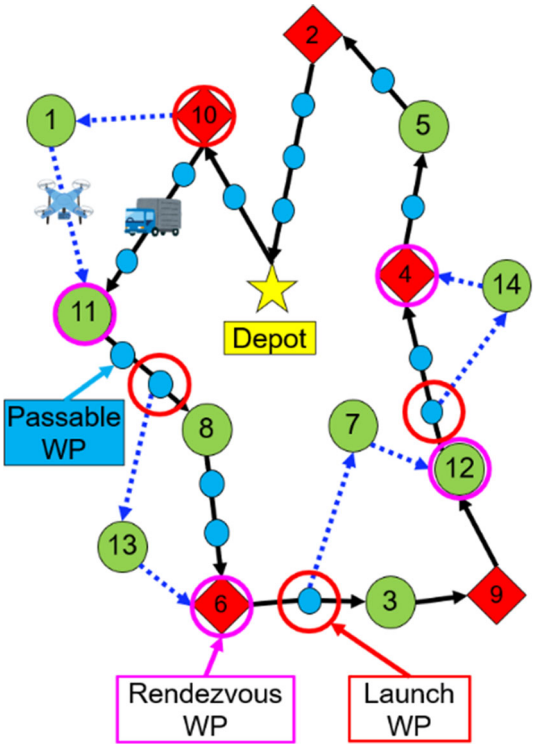


Fig. 15 Entire delivery path of FSTSP

which is expressed by the solution structure described above. Therefore, the delivery plan can be completely represented by these chromosomes.

5.1.4 Formulation

As mentioned above, DDP has many constraint conditions to be considered in the real delivery scenarios. In this study, we consider the traveling distance and the delivery time as the objective functions. Meanwhile, the flight duration, the loadable weight of drones, the designated delivery time zone, and allowable time of stopping the truck at the same place can be considered as the constraint conditions. Based on the above, we formulated DDP as a constrained multi-objective optimization problem as follows:

$$\min_{\boldsymbol{x}} \boldsymbol{F}(\boldsymbol{x}) = \min_{\boldsymbol{x}} [F_1(\boldsymbol{x}), F_2(\boldsymbol{x})]^T \tag{15}$$

$$F_1(\boldsymbol{x}) = \mathrm{Cost}_{L_{\mathrm{depo}} T_{\mathrm{WP}_1}} + \sum_{j=1}^{T_{\mathrm{all}}-1} \mathrm{Cost}_{T_{\mathrm{WP}_j} T_{\mathrm{WP}_{j+1}}} + \mathrm{Cost}_{T_{\mathrm{WP}_{T_{\mathrm{all}}}} L_{\mathrm{depo}}} \tag{16}$$

$$F_2(\boldsymbol{x}) = F_1(\boldsymbol{x})/V_{\mathrm{truck}} + \sum_{i=1}^{D_{\mathrm{all}}} \times |(\| D_{\mathrm{launch}_i} - C_{D_i} \| + \| C_{D_i} - D_{\mathrm{land}_i} \|)/V_{\mathrm{drone}} - \mathrm{Cost}_{D_{\mathrm{launch}_i} D_{\mathrm{land}_i}}/V_{\mathrm{truck}}| \tag{17}$$

$$\text{subject to } g_1(\boldsymbol{x}) = F_1(\boldsymbol{x}) - 100{,}000 \le 0, \tag{18}$$

$$g_2(\boldsymbol{x}) = F_2(\boldsymbol{x}) - 14{,}400 \le 0, \tag{19}$$

$$g_3(\boldsymbol{x}) = \left|\left(\| D_{\mathrm{launch}_i} - C_{D_i} \| + \| C_{D_i} - D_{\mathrm{land}_i}\right)/V_{\mathrm{drone}} - \mathrm{Cost}_{D_{\mathrm{launch}_i} \| D_{\mathrm{land}_i}}/V_{\mathrm{truck}}\right| - 600 \le 0, \tag{20}$$

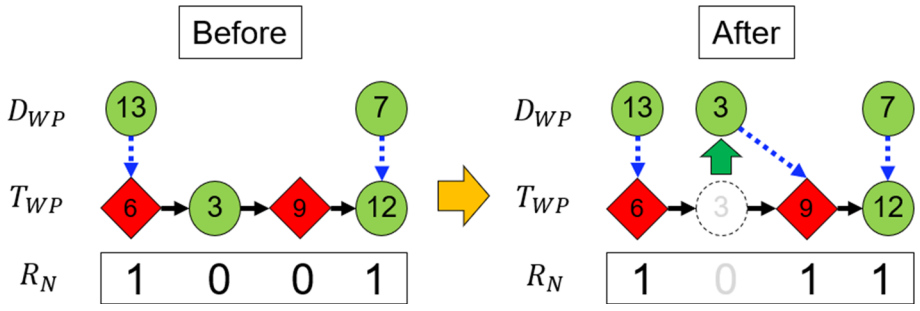


Fig. 16 Genetic operation for multiple chromosomes

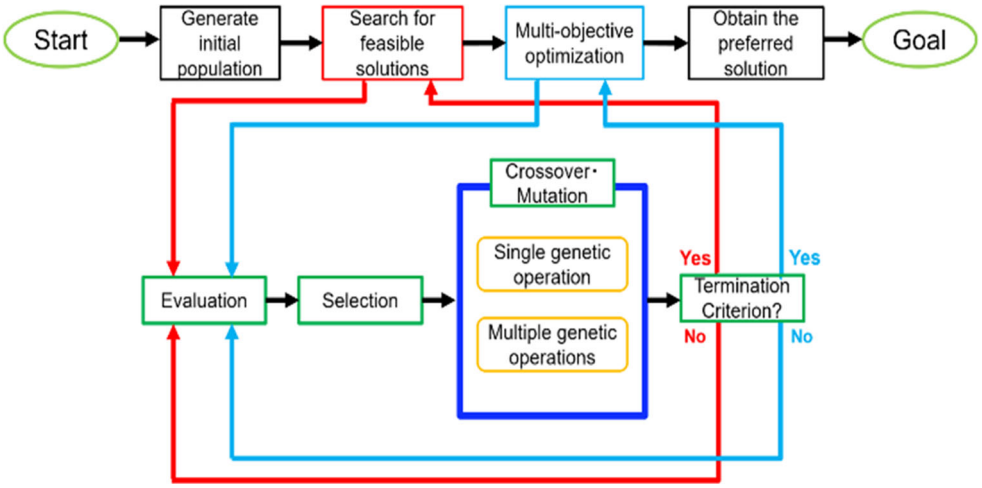


Fig. 17 Flow chart of the proposed method

$$g_4(\boldsymbol{x}) = W_{\text{drone}} - 5 \le 0, \tag{21}$$
$$g_5(\boldsymbol{x}_1) = F_2(\boldsymbol{x}_1) - 7200 \le 0, \tag{22}$$
$$g_6(\boldsymbol{x}_2) = 3600 - F_2(\boldsymbol{x}_2) \le 0, \tag{23}$$
$$g_7(\boldsymbol{x}_2) = F_2(\boldsymbol{x}_2) - 10{,}800 \le 0, \tag{24}$$
$$g_8(\boldsymbol{x}_3) = 7200 - F_2(\boldsymbol{x}_3) \le 0, \tag{25}$$
$$g_9(\boldsymbol{x}_3) = F_2(\boldsymbol{x}_3) - 14{,}400 \le 0, \tag{26}$$

where the one with the suffix attached to $\boldsymbol{x}$ represents the route from the depot to the place where delivery time is specified. $\boldsymbol{x}_1$ is within 2 h from the start of home delivery, $\boldsymbol{x}_2$ is 3 h from 1 h later, and $\boldsymbol{x}_3$ needs to arrive at the delivery destination within 4 h as from 2 h later.

The values of various parameters were set as follows.

$$N_{\text{drone}} = 1, \tag{27}$$
$$N_{\text{truck}} = 1, \tag{28}$$
$$V_{\text{drone}} = 13\ (\text{m/s}), \tag{29}$$
$$V_{\text{truck}} = 15\ (\text{m/s}), \tag{30}$$
$$t_{\text{duration}} = 15\ (\text{min}), \tag{31}$$
$$t_{\text{launch}} = 15\ (\text{s}), \tag{32}$$
$$t_{\text{landing}} = 15\ (\text{s}), \tag{33}$$
$$t_{\text{load}} = 30\ (\text{s}), \tag{34}$$

Table 5 Calculation results

Items	Truck only	FSTSP
Traveling distance	114.14 (km)	67.74 (km)
Delivery time	4 (h) 14 (min)	3 (h) 1 (min)
Calculation time	29.26 (s)	73.23 (s)

$$t_{\text{driver}} = 120\ (\text{s}). \tag{35}$$

5.2 Genetic Operation

Since the individual representing the delivery plan is composed of three kinds of chromosomes and these are related to each other, the conventional method which adds genetic operation to one chromosome is insufficient. To compensate for this, we proposed some methods for applying genetic operations to multiple chromosomes simultaneously as shown below.

(1) Select one WP from D_{WP} and T_{WP}, respectively, and exchange.

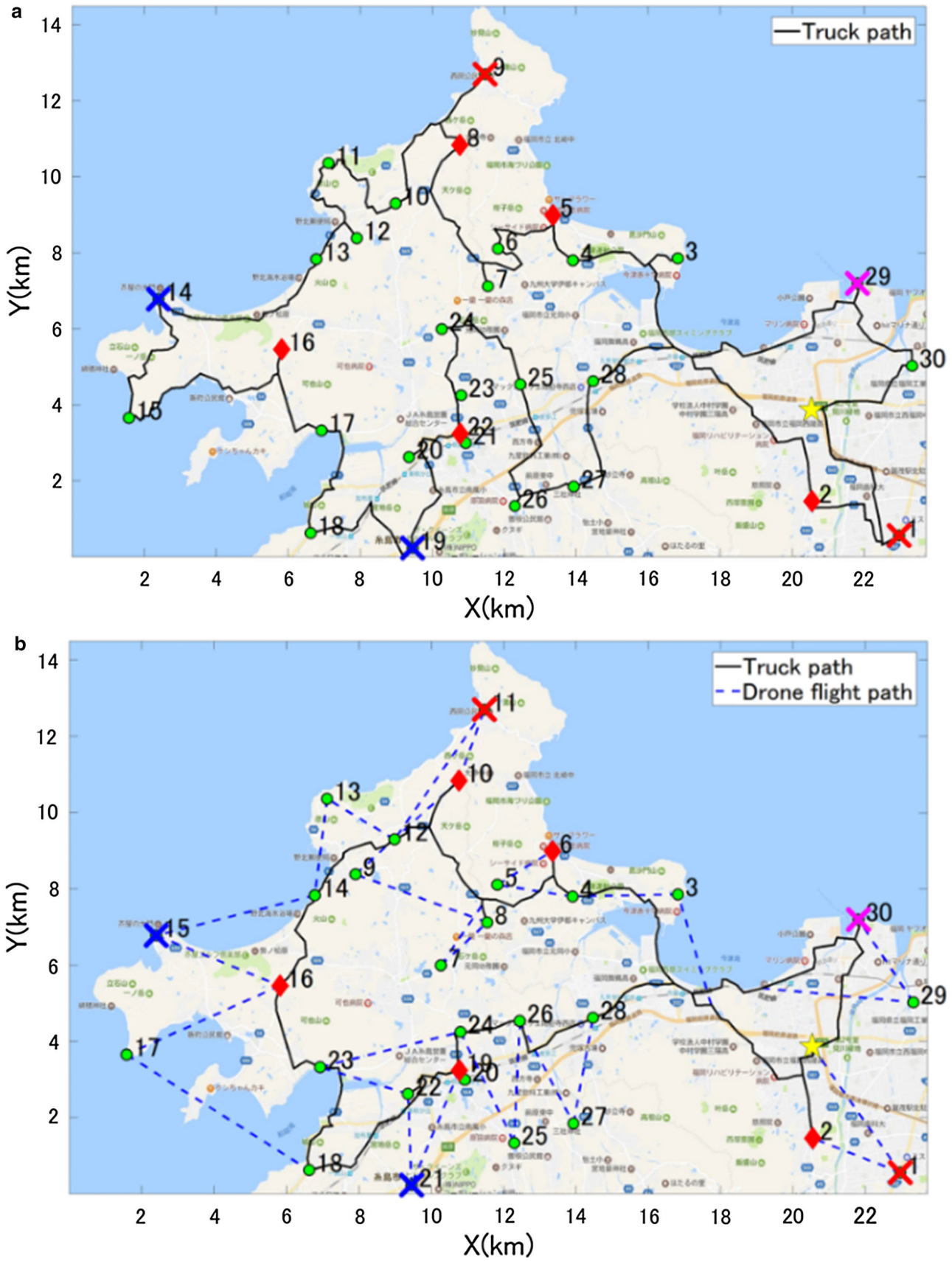


Fig. 18 **a** Delivery path using only one truck. **b** Delivery path of FSTSP

(2) Select one WP from D_{WP} and move it to an arbitrary position of T_{WP}.
(3) Select one WP from T_{WP} and move it to an arbitrary position of D_{WP}.

Note that the operations of (2) and (3) need to change the individual structure of R_N. For example, the operation of (3) needs to reduce the individual length of R_N when moving one of the WP of T_{WP} to D_{WP}, as shown in Fig. 16.

5.3 Flow Chart of the Proposed Method

Figure 17 shows the flow chart to generate the delivery plan by applying the proposed method to FSTSP. As shown in the figure, there are two methods of crossover and mutation. The first one is to add genetic operation to one chromosome like the conventional method. This is expressed as "Single genetic operation". The second one is to add the operation to multiple chromosomes at the same time. This is expressed as "Multiple genetic operations". In the proposed method, solutions are searched by adding one of the two operations or both to individuals simultaneously in a random manner.

5.4 Simulation Results

Table 5 shows the calculation results obtained using the proposed method. Figure 18a shows the delivery path when using only one truck. Figure 18b shows the delivery path when combining one drone and one truck. The black lines are the truck paths, and the blue dash lines are the flight paths of the drone. Each number attached to the delivery WPs represents the passage order of the delivery service.

From these simulation results, we found that the delivery path combining one drone and one truck can reduce the traveling distance by 40.7% and the delivery time by 28.7% compared with the case where using one truck while satisfying all constraint conditions. In addition, both calculation times were about 1 min. Therefore, it can be said that the purpose of this research had achieved.

6 Conclusion

We proposed the provisional-ideal-point-based method for solving the drone delivery problem. This method defines the penalty value by summing all evaluation values that do not satisfy given constraint conditions. The feasible solutions can be generated by searching for a solution whose penalty value is equal to 0. After that, it also defines the provisional-ideal point and the solution point using the provisional minimum value of each objective function. The former is a virtual solution which can optimize all objective functions at the same time, and the later represents one of the normalized feasible solutions. The preferred solution for a decision maker can be generated by minimizing the distance between the two points.

We showed the usefulness of the proposed method using two kinds of test problems. As a result, the proposed method was able to generate the feasible solutions and the preferred solution efficiently without any preliminary calculations. We also applied this method to DDP which assumes FSTSP, and evaluated its validity and the potential benefits of the drone delivery service. The simulation results showed that the drone has the potential of drastically reducing the delivery costs or times while maintaining its quality.

References

1. Holloway CM, Knight JC, McDermid JA (2014) Neither Pollyanna nor Chicken Little: thoughts on the ethics of automation. In: 2014 IEEE international symposium on ethics in science, technology and engineering, pp 1–7
2. Murray CC, Chu AG (2015) The flying sidekick traveling salesman problem: optimization of drone-assisted package delivery. Transp Res Part C Emerg Technol 54:86–109
3. Ramadan ZB, Farah MF, Mrad M (2017) An adapted TPB approach to consumers' acceptance of service-delivery drones. Technol Anal Strateg Manag 29(7):817–828
4. Ferrandez SM, Harbison T, Weber T, Sturges R, Rich R (2016) Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. J Ind Eng Manag 9(2):374–388
5. The Verge (2017) UPS has a delivery truck that can launch a drone. https://www.theverge.com/2017/2/21/14691062/ups-drone-delivery-truck-test-completed-video. Accessed 7 Jul 2017
6. Parker GG, Van Alstyne MW, Choudary SP (2016) Platform revolution: how net worked markets are transforming the economy and how to make them work for you. WW Norton & Company, New York
7. Mathew N, Smith SL, Waslander SL (2015) Planning paths for package delivery in drone multirobot teams. IEEE Trans Autom Sci Eng 12(4):1298–1308
8. Dorling K, Heinrichs J, Messier GG, Magierowski S (2017) Vehicle routing problems for drone delivery. IEEE Trans Syst Man Cybern Syst 47(1):70–85
9. Goldberg DE, Holland JH (1988) Genetic algorithms and machine learning. Mach Learn 3(2):95–99
10. Shin H, Todoroki A, Hirano Y (2013) Elite-initial population for efficient topology optimization using multi-objective genetic algorithms. Int J Aeronaut Space Sci 14(4):324–333
11. Omagari H, Higashino S (2017) Aspiration-point-based multi-objective path planning method for an unmanned aerial vehicle. Trans Jpn Soc Aeronaut Space Sci 15:a99–a108 (APISAT-2016 special issue of Aerospace Technology Japan)
12. Kim IY, De Weck OL (2005) Adaptive weighted sum method for bi-objective optimization, Pareto front generation. Struct Multidiscip Optim 29:149–158
13. Yaguchi K, Tamura K, Yasuda K, Ishigame A (2011) Basic study of proximate optimality principle based combinatorial optimization method. In: IEEE international conference on systems, man, and cybernetics (SMC), pp 1753–1758
14. Runarsson TP, Yao X (2000) Stochastic ranking for constrained evolutionary optimization. IEEE Trans Evol Comput 4(3):284–294

15. Deb K, Thiele L, Laumanns M, Zitzler E (2005) Scalable test problems for evolutionary multi-objective optimization. Springer, London, pp 105–145
16. Vose MD (1999) "The simple genetic algorithm" foundations and theory, vol 12. MIT Press, Cambridge
17. Ishibuchi H, Hitotsuyanagi Y, Tsukamoto N, Nojima, Y. (2010) Many-objective test problems to visually examine the behavior of multi-objective evolution in a decision space. In: International conference on parallel problem solving from nature. Springer, Berlin, Heidelberg, pp 91–100
18. Hart PE, Nilsson NJ, Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. IEEE Trans Syst Sci Cybern SSC 4(2):100–107
19. Mitsutake K, Higashino S (2008) An A*-EC hybrid path planning method for waypoints traveling problem considering terrain. In: AIAA guidance, navigation and control conference and exhibit