Oper Res Int J (2020) 20:1913–1938
https://doi.org/10.1007/s12351-018-0402-5

ORIGINAL PAPER

Air refueling tanker allocation based on a multi-objective zero-one integer programming model

Farzaneh Ferdowsi[1] · Hamid Reza Maleki[1] · Sanaz Rivaz[2]

Received: 1 February 2017 / Revised: 18 April 2018 / Accepted: 10 May 2018 /
Published online: 19 May 2018

Abstract The need to refuel aircraft traveling long distances is important because fuel tank capacities limit the range of aircraft, and landing to refuel may not be practical or even possible. To overcome this difficulty, aerial refueling can be performed en route along the aircraft's travel path to extend its range. This paper considers the problem of identifying the locations along an aircraft flight path at which to conduct aerial refueling, given a limited number of refueling stations. Due to the inherent uncertainty of real-world cases, the cost of refueling is considered as an interval number, and the problem is mathematically presented as an interval multi-objective zero-one integer programming model. To solve the model, a new version of the modified label-correcting method and a genetic algorithm are proposed. Moreover, the applicability and efficiency of the proposed solution approaches are examined and compared using some randomly generated test problems.

Keywords Aerial refueling · Allocation problem · Interval multi-objective zero-one integer programming · Modified label-correcting method · Genetic algorithm

✉ Hamid Reza Maleki
maleki@sutech.ac.ir

Farzaneh Ferdowsi
f.ferdowsi@sutech.ac.ir

Sanaz Rivaz
srivaz@nit.ac.ir

[1] Faculty of Mathematics, Shiraz University of Technology, Shiraz 71557-13876, Iran

[2] Department of Mathematics, Faculty of Basic Sciences, Babol Noshirvani University of Technology, Shariati Ave, Babol 47148-71167, Iran

1 Introduction

Vehicles have different fuel tank capacities, and these capacities limit their range. Thus, any vehicle can travel a limited distance even with a full fuel tank. Aerial refueling is necessary for aircraft traveling beyond their fuel tank range when landing to refuel is undesirable, impractical, or impossible. To perform refueling operations, the following initial information may be needed in advance.

- The number of required refuelings along a certain route to reach the destination;
- The proper locations for a set of refueling operations;
- The amount of fuel that the vehicle consumes while traveling along a route.

Refueling problems have attracted attention from researchers in areas such as ground transportation and aviation organization (Berman and Krass 1998; Kuby et al. 2009; Jin et al. 2006; Kaplan and Rabadi 2012).

In the ground transportation area, Kuby and Lim (2005) presented a flow refueling location model (FRLM). The FRLM determines optimal locations for refueling stations considering the limitation of vehicle ranges. To solve the FRLM, Lim and Kuby (2010) adopted three heuristic algorithms. However, their algorithms took a long time to solve problems in large-scale networks. To overcome this shortcoming, Capar and Kuby (2011) presented a radically different mixed-integer programming formulation. The proposed model solved the FRLM to optimality as fast as or faster than the aforementioned heuristic algorithms. Wang and Lin (2009) proposed a refueling-station-location model based on a mixed-integer programming problem. The major advantage of this model is that it does not require additional preprocessing for the model solutions, in contrast to the FRLM. A hybrid model with two objective functions was also investigated by Wang and Wang (2010). They used a mixed-integer programming model to locate the refueling stations serving inter-city and intra-city travel. MirHassani and Ebrazi (2013) reformulated a previously introduced FRLM, and proposed a new mixed-integer linear programming problem. The computation time of the proposed model was also compared with that of Wang and Lin's (2009) model. Other studies concerning the FRLM include the capacitated flow refueling location model (Upchurch et al. 2009), comparison of p-median and the FRLM (Upchurch and Kuby 2010), drivers' deviations (Kim and Kuby 2012), heuristic approaches to solve the deviation flow refueling location (Kim and Kuby 2013), and dispersion of candidate sites on arcs (Kuby and Lim 2007).

As mentioned above, refueling problems have also been investigated in aviation organizations. Sundar and Rathinam (2014) presented a mixed-integer linear programming formulation for routing a small unmanned aerial vehicle by a series of targets and refueling depots. This formulation minimized the total required fuel such that all targets are visited. Moreover, Levy et al. (2014) considered a multiple depot, multiple unmanned vehicle routing problem such that all the specified targets should be visited and the total travel cost of the vehicles should be minimized.

Aerial refueling is one of the solutions used to refuel aircraft conducting operations over an extended duration. Transferring fuel from a tanker aircraft to a receiver aircraft in the air is called aerial refueling. Tanker aircraft can carry a significant amount of fuel to meet the requirements of receiver aircraft using special equipment and techniques during their flights. The first aerial refueling was performed in 1917 by the Russian-American Alexander de Seversky (Thomas et al. 2014). The key purpose of aerial refueling is to increase aircraft endurance. Due to its advantages, aerial refueling has been extensively investigated, developed, and employed in many aerial missions. Its first military application was in the Korean War (Thomas et al. 2014). A number of studies have been devoted to aerial refueling problems. In this context, Bush (2006) investigated the problem of routing an aircraft, locating a fixed number of aerial refueling points to be served by tanker aircraft based on predetermined locations, and minimizing the total fuel consumptions of both the receiver aircraft and the tanker aircraft. The problem of finding the least costly route for an aircraft between a fixed origin and destination locations while ensuring sufficient fuel is either on board or acquired en route via aerial refueling for the completion of the route was studied by Kannon et al. (2014). Kannon et al. also proved that this problem is NP-hard. Kannon et al. (2015) subsequently considered the aircraft routing problem with aerial refueling. To deal with this problem, they presented a multi-objective mixed-integer programming model.

Our main focus in this paper is the aerial refueling operation for an aircraft. An arbitrary route between a fixed origin and a specified destination from a network is considered. The aircraft should travel this route without running out of fuel. Obviously, if the amount of fuel that the aircraft needs to travel the route is greater than its range, one or more refueling operations are needed. Moreover, it is assumed that there are refueling tankers in some predetermined stations on the ground. We would like to determine the nodes at which aerial refueling should be performed and allocate the stations containing tanker aircraft to these nodes. As the allocation process and refueling operations involve some costs, it would be reasonable to seek appropriate solutions that minimize the total cost as well as the number of refueling operations. To deal with the problem mathematically, parameters such as the costs of serving aerial refueling nodes by tanker aircraft and fuel consumption of the receiver aircraft should be available. In some of the real-world situations, the values of parameters cannot be determined exactly. For instance, the amount of fuel that an aircraft needs to travel a route depends on weather conditions, altitude, gross weight of the aircraft, etc. Any change in the values of these parameters affects the amount of fuel consumed. Uncertainty is an inherent aspect of measurements, and it would be more realistic to consider uncertain inputs for the parameters of the aerial refueling problem. Uncertainty may be interpreted as randomness or fuzziness (Wu 2009). However, specifying distributions of random parameters and membership functions of fuzzy numbers is very questionable. Interval programming may provide an alternative choice for dealing with uncertain parameters (Oliveira and Antunes 2009; Urli and Nadeau 1992; Rivaz and Yaghoobi 2015; Hladik 2016). In interval programming, uncertain parameters are modeled by closed intervals, which may be easier to handle.

To deal with the newly proposed aerial refueling problem, a multi-objective zero-one integer programming model with interval objective function coefficients is proposed in this paper. The combination of multi-objective programming, interval programming, and refueling requirements of a receiver aircraft with limited range in this problem sets it apart from existing aerial refueling research. To handle the model, two new algorithms are developed based on the inherent multi-objective nature of the problem. Moreover, the performance of the algorithms is analyzed and compared through some examples.

The remainder of this paper is organized as follows. Some preliminaries are stated in Sect. 2. Section 3 describes the problem and proposes its mathematical programming model. In Sect. 4, two algorithms are presented to solve the proposed model. Numerical examples and computational results are provided in Sect. 5. Finally, Sect. 6 is devoted to the conclusion.

2 Preliminaries

In this section, some initial concepts and definitions that will be used later are reviewed.

A closed interval in $\mathbb{R}$ is defined as

$$\left[a^l, a^u\right] = \left\{t \in \mathbb{R} | a^l \leq t \leq a^u\right\},$$

where a^l and a^u denote the lower and upper bounds of the interval, respectively. An interval with equal lower and upper bounds identifies a real number, i.e., $a = [a, a], a \in \mathbb{R}$.

Let $A = [a^l, a^u]$ and $B = [b^l, b^u]$ be two closed intervals in $\mathbb{R}$. Then, by definition, their sum is

$$A + B = \{a + b | a \in A, b \in B\} = \left[a^l + b^l, a^u + b^u\right].$$

Different order relations have been proposed on closed intervals (Zapata et al. 2013; Wu 2009). The strong order, "$\leqslant_I$", will be applied in this paper. A is strongly less than or equal to B (written as $A \leqslant_I B$) if and only if $a^u \leq b^l$. In fact, $A \leqslant_I B$ indicates that any value from the interval $A = [a^l, a^u]$ is smaller than or equal to any value from the interval $B = [b^l, b^u]$. The strong order is the common and by far most prominent sense of interval order (Zapata et al. 2013). It is not difficult to say that "$\leqslant_I$" is a partial order on the set of all closed intervals in $\mathbb{R}$. The word "partial" in the name "partial order" is used as an indication that not every pair of intervals need be comparable. Intervals are frequently partially ordered and cannot be compared. When we are forced to compare many intervals, many incomparable intervals may exist (Okada and Gen 1993). To select the smaller interval of a set of incomparable intervals, some studies suggest converting them to crisp values (Sengupta et al. 2001). An important deficiency of this approach is that a considerable portion of the information of the data of the problem may be ignored during the conversion. Therefore, to overcome this deficiency in comparisons, we retain all incomparable intervals for further consideration. For more details on interval analysis, the studies of Okada and Gen (1993) and Moore et al. (2009) are suggested.

In the following, a traditional multi-objective integer programming problem is defined as

$$\begin{aligned} \min\ :\ & \mathbf{Cx} = (\mathbf{c}_1\mathbf{x}, \dots, \mathbf{c}_p\mathbf{x}) \\ s.t. \quad & \mathbf{x} \in X, \end{aligned} \tag{1}$$

where

- $X \subseteq \mathbb{Z}^n$ is a finite feasible set (here $\mathbb{Z}$ denotes the set of all integers).
- $\mathbf{c}_i\mathbf{x} = \sum_{j=1}^{n} c_{ij}x_j$ for $i = 1, \dots, p$ and $c_{ij}, i = 1, \dots, p, j = 1, \dots, n$ are real numbers.

More explicitly, $X = \{\mathbf{x} \in \mathbb{Z}^n | A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$, $A \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$. Many multi-objective integer programs will actually have $X \subseteq \{0, 1\}^n$.

In multi-objective optimization problems, it is rarely possible to find a solution that optimizes all objective functions simultaneously. Hence, the concept of an efficient solution is used instead of the optimal solution. In the efficient solution set, no improvement in any objective function is possible without sacrificing at least one of the other objective functions. To present the definition of efficiency, an order relation in $\mathbb{R}^p$ is needed. Let $\mathbf{a} = (a_1, \dots, a_p)^t$ and $\mathbf{b} = (b_1, \dots, b_p)^t$ be two vectors in $\mathbb{R}^p$. Then $\mathbf{a} \preccurlyeq \mathbf{b}$ if $a_i \leq b_i$ for $i = 1, \dots, p$ and there is also at least one $1 \leq q \leq p$ with $a_q < b_q$.

Definition 1 A feasible solution $\mathbf{x}^0$ of Problem (1) is efficient if there is no $\mathbf{x} \in X$ such that $C\mathbf{x} \preccurlyeq C\mathbf{x}^0$.

The above definition can be found in many references, such as Steuer (1986) and Ehrgott (2005).

A multi-objective integer programming problem with interval objective function coefficients is as follows:

$$\begin{aligned} \min\ :\ \quad & Z(\mathbf{x}) = \mathbf{Cx} \\ s.t. \quad & \mathbf{x} \in X, \\ & \mathbf{C} \in \Psi, \end{aligned} \tag{2}$$

where Ψ is a set of $p \times n$ matrices with each row in the form of $\mathbf{c}_i$, with elements $c_{ij} = [c_{ij}^l, c_{ij}^u]$ and $i = 1, \dots, p$, $j = 1, \dots, n$.

For every $\mathbf{x} \in X$, $Z(\mathbf{x}) = ([z_1^l(\mathbf{x}), z_1^u(\mathbf{x})], \dots, [z_p^l(\mathbf{x}), z_p^u(\mathbf{x})]) = ([\sum_{j=1}^{n} c_{1j}^l x_j, \sum_{j=1}^{n} c_{1j}^u x_j], \dots, [\sum_{j=1}^{n} c_{pj}^l x_j, \sum_{j=1}^{n} c_{pj}^u x_j])$, is an interval-valued vector. Therefore, Problem (2) could also be written as

$$\begin{aligned} \min\ :\ \quad & z_i(\mathbf{x}) = \sum_{j=1}^{n} \left[c_{ij}^l, c_{ij}^u\right] x_j, \quad i = 1, \dots, p, \\ s.t.\ & \mathbf{x} \in X. \end{aligned} \tag{3}$$

It is clear that if coefficients $[c_{ij}^l, c_{ij}^u]$, $i = 1, \dots, p$, $j = 1, \dots, n$ are degenerated to real numbers, then Problem (2) is reduced to Problem (1). For simplicity, (2) is called an interval multi-objective integer (IMOI) programming problem.

By $[a^l, a^u] <_I [b^l, b^u]$, we mean $a^u < b^l$, and hence the following order on interval-valued vectors is stated. Let $\mathcal{A}$ and $\mathcal{B}$ be two interval-valued vectors as

$$\mathcal{A} = \left([a_1^l, a_1^u], [a_2^l, a_2^u], \dots, \left[a_p^l, a_p^u\right] \right)$$

and

$$\mathcal{B} = \left([b_1^l, b_1^u], [b_2^l, b_2^u], \dots, \left[b_p^l, b_p^u\right] \right).$$

By $\mathcal{A} \preccurlyeq_{IV} \mathcal{B}$, we mean $[a_i^l, a_i^u] \leqslant_I [b_i^l, b_i^u]$, $i = 1, \dots, p$, and there exists at least one $1 \le j \le p$ such that $[a_j^l, a_j^u] <_I [b_j^l, b_j^u]$. Now, considering the above-mentioned order on interval vectors, the concepts of efficiency, nondominance, and dominance are introduced for the IMOI Problem (2).

Definition 2 A feasible solution $\mathbf{x}^0$ of Problem (2) is IV-efficient if there exists no $\mathbf{x} \in X$ such that $C\mathbf{x} \preccurlyeq_{IV} C\mathbf{x}^0$. If $\mathbf{x}^0$ is IV-efficient, $C\mathbf{x}^0$ is called a nondominated interval-valued vector. If $\mathbf{x}^1, \mathbf{x}^2 \in X$ and $C\mathbf{x}^1 \preccurlyeq_{IV} C\mathbf{x}^2$, $C\mathbf{x}^1$ dominates $C\mathbf{x}^2$.

3 Problem statement

Let $\mathfrak{p}$ be the shortest path from $s^{\mathfrak{p}}$ to $t^{\mathfrak{p}}$ with the set of nodes $N^{\mathfrak{p}}$ (which also contains $s^{\mathfrak{p}}$ and $t^{\mathfrak{p}}$) and the set of arcs $E^{\mathfrak{p}}$. Now, consider an aircraft that is traveling from the origin, $s^{\mathfrak{p}}$, to the destination, $t^{\mathfrak{p}}$, along the path $\mathfrak{p}$. It is assumed that the aircraft has a full tank of fuel at the origin and may not refuel at the destination node (Kannon et al. 2015). Due to the limited range of the aircraft, it may run out of fuel before reaching the destination; therefore, aerial refueling is necessary. It is assumed that the nodes with no possibility of refueling are removed from the path at the beginning. Hence, there is the possibility of refueling on all nodes of the path. To perform aerial refuelings, all valid combinations of refueling nodes are found (Kuby and Lim 2005). Each valid combination is a subset of $N^{\mathfrak{p}}$, which contains the possible aerial refueling nodes.

Undoubtedly, the range of the aircraft, R, and the amount of fuel that is consumed by the aircraft to travel a subpath between any two nodes i and j on path $\mathfrak{p}$, $f_{\mathfrak{p}}(i,j)$, play essential roles in determining the valid combinations. The amount of fuel that the aircraft consumes to travel a subpath may change depending on different factors such as weather, airspeed, altitude and gross weight (Kannon et al. 2015). Hence, it would be more meaningful to consider the amount of consumable fuel needed to travel a subpath as an uncertain value. In this sense, it can be interpreted as an interval number, i.e. $f_{\mathfrak{p}}(i,j) = [f^l_{\mathfrak{p}_{ij}}, f^u_{\mathfrak{p}_{ij}}]$. Here, R could also be considered as an interval number with equal lower and upper bounds.

In what follows, to find valid combinations of aerial refueling nodes on path $\mathfrak{p}$, a network denoted by $\mathcal{G}^{\mathfrak{p}} = \mathcal{G}^{\mathfrak{p}}(\mathcal{N}^{\mathfrak{p}}, \mathcal{E}^{\mathfrak{p}})$ is constructed (MirHassani and Ebrazi 2013). In this network, $\mathcal{N}^{\mathfrak{p}}$ and $\mathcal{E}^{\mathfrak{p}}$ define the sets of nodes and arcs on path $\mathfrak{p}$. Before starting to construct $\mathcal{G}^{\mathfrak{p}}$, the notation $ord_{\mathfrak{p}}(i)$ is defined as an ordering index of node i in the path sequence $\mathfrak{p}$. For instance, the ordering index of node C on path $\mathfrak{p} : s^{\mathfrak{p}} - A - B - C - D - t^{\mathfrak{p}}$ is $ord_{\mathfrak{p}}(C) = 4$. To construct $\mathcal{G}^{\mathfrak{p}}$, each node $i \in N^{\mathfrak{p}}$ should be connected to any other node $j \in N^{\mathfrak{p}}$ if the ordering index of i is less than the ordering index of j in the path sequence $\mathfrak{p}$ ($ord_{\mathfrak{p}}(i) < ord_{\mathfrak{p}}(j)$) and the aircraft is able to start from i with a full tank of fuel and can reach j without running out of fuel. In other words,

$$(ord_{\mathfrak{p}}(i) < ord_{\mathfrak{p}}(j))\&(f_{\mathfrak{p}}(i,j) \leqslant_I R) \Longrightarrow (i,j) \in \mathcal{E}^{\mathfrak{p}}, \quad \forall i,j \in N^{\mathfrak{p}}.$$

As stated before, " $\leqslant_I$ " denotes the strong order between interval numbers. Since $f_{\mathfrak{p}}(i,j) \leqslant_I R$ means that any value from the interval $f_{\mathfrak{p}}(i,j)$ is smaller than or equal to any value from the interval R. Thus, the aircraft is guaranteed to never run out of fuel on the subpath between nodes i and j on path $\mathfrak{p}$. If in this process we encounter a node that could not be connected to any other nodes with a larger ordering index, then path $\mathfrak{p}$ is infeasible, i.e., this path cannot be traveled along due to the fuel capacity restriction. In $\mathcal{G}^{\mathfrak{p}}$, each arc corresponds to the consecutive feasible aerial refueling, and each directed path from $s^{\mathfrak{p}}$ to $t^{\mathfrak{p}}$ shows a feasible combination of aerial refuelings that can refuel path $\mathfrak{p}$. To further clarify, a simple example is given here.

Example 1 Consider a specified path $\mathfrak{p}$ with six nodes, $s^{\mathfrak{p}} - A - B - C - D - t^{\mathfrak{p}}$, and an aircraft with $R = [90{,}000, 90{,}000] = 90{,}000$ (measured in pounds) that is going to fly from origin $s^{\mathfrak{p}}$ to destination $t^{\mathfrak{p}}$. Moreover, suppose $f_{\mathfrak{p}}(s^{\mathfrak{p}}, A) = [20{,}000, 25{,}000]$, $f_{\mathfrak{p}}(A, B) = [38{,}000, 42{,}000]$, $f_{\mathfrak{p}}(B, C) = [69{,}000, 73{,}000]$, $f_{\mathfrak{p}}(C, D) = [30{,}000, 32{,}000]$ and $f_{\mathfrak{p}}(D, t^{\mathfrak{p}}) = [8000, 11{,}000]$ (measured in pounds). To find all valid combinations of aerial refueling nodes, the mentioned rules are applied. Obviously, $ord_{\mathfrak{p}}(s^{\mathfrak{p}}) < ord_{\mathfrak{p}}(A)$ and $f_{\mathfrak{p}}(s^{\mathfrak{p}}, A) \leqslant_I R$; accordingly, $\mathcal{G}^{\mathfrak{p}}$ includes arc $(s^{\mathfrak{p}}, A)$. Furthermore, $ord_{\mathfrak{p}}(s^{\mathfrak{p}}) < ord_{\mathfrak{p}}(B)$ and $f_{\mathfrak{p}}(s^{\mathfrak{p}}, B) = f_{\mathfrak{p}}(s^{\mathfrak{p}}, A) + f_{\mathfrak{p}}(A, B) \leqslant_I R$ imply that $\mathcal{G}^{\mathfrak{p}}$ contains arc $(s^{\mathfrak{p}}, B)$. By performing similar comparisons for all nodes, $\mathcal{G}^{\mathfrak{p}}$ is constructed (Fig. 1). According to Fig. 1, the four valid combinations of aerial refueling nodes are $\{A, B, C, D\}$, $\{B, C, D\}$, $\{A, B, C\}$, and $\{B, C\}$.

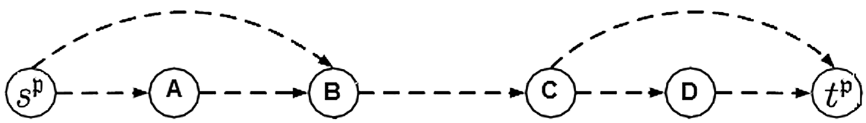

Fig. 1 Network $\mathcal{G}^{\mathfrak{p}}$ for Example 1

There are several criteria for selecting one or more appropriate combinations for aerial refuelings among all valid combinations. In an aerial refueling operation, a tanker aircraft must fly from its specified station on the ground to an aerial refueling node in the air. Obviously, each refueling operation includes some costs, which are affected by different factors, such as the distance that should be covered by the tanker aircraft and the fuel consumed by the tanker aircraft to reach the aerial refueling node. Thus, it is reasonable to search for refueling operations with minimum cost. Furthermore, to avoid wasting time and cost, identifying the minimum number of necessary aerial refuelings to complete the aircraft mission is another criterion for selecting one or more valid combinations of aerial refueling nodes. Therefore, minimizing the total cost of refueling and minimizing the number of refuelings are the two criteria considered in this paper. In addition, it is assumed that there sufficient fuel and tanker aircraft available at each station. Hence, there is no limitation on the frequency of use of the stations.

To find appropriate aerial refueling combinations among all valid combinations on the specified path $\mathfrak{p}$, according to the mentioned criteria, a bi-objective programming model is presented. The following set, parameter, and decision variables are used in the proposed model.

1 *Set*

K: The set of possible stations.

1 *Parameter*

$[c_{kj}^l, c_{kj}^u]$: The interval cost of an aerial refueling operation at node j, which is served by a tanker aircraft from the k-th station.

1 *Decision variables*

$y_{kj}^{\mathfrak{p}}$: 1 if the aerial refueling at node j on path $\mathfrak{p}$ is served by a tanker aircraft from the k-th station; otherwise 0.

$x_{ij}^{\mathfrak{p}}$: 1 if arc $(i,j) \in \mathcal{E}^{\mathfrak{p}}$ is traveled along by the aircraft; otherwise 0.

The model is formulated as follows:

$$\min: \quad Z_1 = \sum_{k \in K} \sum_{\substack{j \in \mathcal{N}^{\mathfrak{p}} \\ j \neq s^{\mathfrak{p}}, t^{\mathfrak{p}}}} \left[c_{kj}^l, c_{kj}^u\right] y_{kj}^{\mathfrak{p}}, \tag{4}$$

$$\min: \quad Z_2 = \sum_{k \in K} \sum_{\substack{j \in \mathcal{N}^{\mathfrak{p}} \\ j \neq s^{\mathfrak{p}}, t^{\mathfrak{p}}}} y_{kj}^{\mathfrak{p}}, \tag{5}$$

$$s.t. \quad \sum_{\{j|(i,j)\in\mathcal{E}^{\mathfrak{p}}\}} x_{ij}^{\mathfrak{p}} - \sum_{\{j|(j,i)\in\mathcal{E}^{\mathfrak{p}}\}} x_{ji}^{\mathfrak{p}} = \begin{cases} 1 \text{ if } i = s^{\mathfrak{p}}, \\ 0 \text{ if } i \neq s^{\mathfrak{p}}, t^{\mathfrak{p}}, \forall i \in \mathcal{N}^{\mathfrak{p}}, \\ -1 \text{ if } i = t^{\mathfrak{p}}, \end{cases} \tag{6}$$

$$\sum_{k\in K} y_{kj}^{\mathfrak{p}} \geq \sum_{\{i|(i,j)\in\mathcal{E}^{\mathfrak{p}}\}} x_{ij}^{\mathfrak{p}}, \qquad \forall j \in \mathcal{N}^{\mathfrak{p}}, j \neq s^{\mathfrak{p}}, t^{\mathfrak{p}}, \tag{7}$$

$$\sum_{k\in K} y_{kj}^{\mathfrak{p}} \leq 1, \qquad \forall j \in \mathcal{N}^{\mathfrak{p}}, j \neq s^{\mathfrak{p}}, t^{\mathfrak{p}}, \tag{8}$$

$$x_{ij}^{\mathfrak{p}} \in \{0, 1\}, \qquad \forall (i,j) \in \mathcal{E}^{\mathfrak{p}}, \tag{9}$$

$$y_{kj}^{\mathfrak{p}} \in \{0, 1\}, \qquad \forall j \in \mathcal{N}^{\mathfrak{p}}, j \neq s^{\mathfrak{p}}, t^{\mathfrak{p}}, \forall k \in K. \tag{10}$$

The objective function (4) minimizes the sum of all aerial refueling operation costs. Furthermore, in the objective function (5), the number of aerial refuelings is minimized. Constraints (6) demonstrate that the aircraft must start flying from node $s^{\mathfrak{p}}$ and end its flight at node $t^{\mathfrak{p}}$. Moreover, they show that when the aircraft enters an intermediate node, it must also exit that node. Constraints (7) ensure that when the aircraft enters a node, it is refueled by a tanker aircraft. To prevent allocating more than one tanker aircraft to an aerial refueling node, Constraints (8) are applied. Constraints (9) and (10) are required as all variables are binary.

In most real-world problems, aircraft refuelings on different paths should be investigated. To extend the proposed problem, it is sufficient to first consider all paths as a network, $G(N, E)$, where N and E contain the nodes and arcs of all paths, respectively. Then, the network, $\mathcal{G}(\mathcal{N}, \mathcal{E})$, corresponding to $G(N, E)$ can be easily obtained simply by using the previously mentioned rules on each path. Finally, the proposed model is applied to each path in $\mathcal{G}(\mathcal{N}, \mathcal{E})$. Any $O - D$ pairs could be considered as a path. In fact, each node can play the role of an origin or destination.

4 Solution methodology

Before starting to solve the proposed model, its computational complexity is investigated. Here, we attempt to show the computational complexity of the proposed formulation by empirical evidence. For this purpose, the proposed model was solved exactly by considering each objective function separately in non-interval form. Some network instances were generated, and the single objective models were solved by the CPLEX solver of GAMS software. The computational times of the optimal solution obtained for each objective function are presented in Table 1. The network with 50 nodes needs more than 70,000 s of running

Table 1 The results of computational complexity experiments

Network no.	Nodes	Time (s)	
		The first objective function	The second objective function
		Z_1	Z_2
1	5	2.474	2.246
2	10	13.765	13.325
3	15	136.976	136.722
4	20	1073.845	1070.648
5	25	3038.706	3030.366
6	30	7683.889	7675.596
7	50	> 70,000	> 70,000

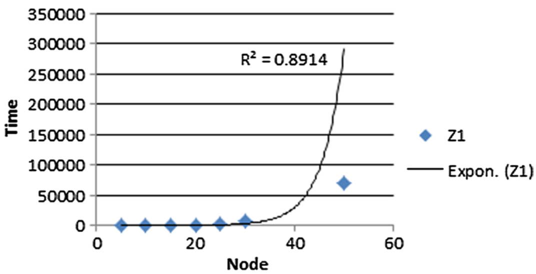

Fig. 2 The computational times of the optimal solution for the first objective function and an exponential function fit to them

time to be solved optimally. The phenomenon of combinatorial explosion can be observed here. Exponential functions for the first and second objectives are fit to the CPU times of GAMS with $R^2 = 0.8914$ and $R^2 = 0.8892$, respectively (Figs. 2, 3).

Considering the computational results obtained from the aforementioned empirical evidence as well as the interval multi-objective nature of the proposed problem, for the case of large size instances, the presented model faces serious computational difficulties that must be solved. These difficulties motivated us to develop some algorithms to solve the problem. In this section, we attempt to present effective algorithms for solving the proposed model. An attempt has been made to ensure that the algorithms preserve the interval multi-objective nature of the problem and have acceptable CPU times and reasonable computational efforts. In the rest of this section, two new algorithms based on the modified label-correcting algorithm (Ahuja et al. 1993) and genetic algorithm (GA) are proposed, and their structures are discussed in detail.

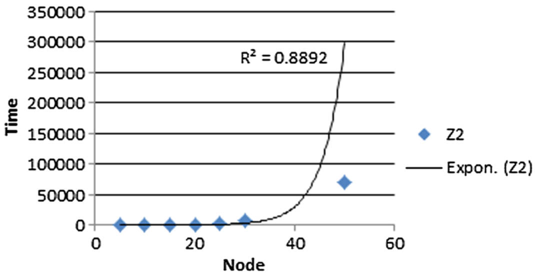

Fig. 3 The computational times of the optimal solution for the second objective function and an exponential function fit to them

4.1 Labeling algorithm

The node labeling algorithm as an applicable method was used by Kannon et al. (2014, 2015) to solve the aircraft routing problem with refueling. A new version of the modified label-correcting method, called the labeling algorithm (LA), is presented in this paper. According to the proposed problem, additional entries are used in the node labels. Instead of using traditional methods for converting two objectives into a single objective function, the values of the two objectives are considered as the elements of a vector. This new algorithm is detailed below.

Generally, in LA, the label corresponding to each node of $\mathcal{G}^{\mathfrak{p}}$ is

$$L^{\mathfrak{p}}(j) = \left(\left(\left[w_j^l, w_j^u\right], n_j\right), sta(pred(j)), pred(j)\right).$$

The first component of $L^{\mathfrak{p}}(j)$, i.e., $([w_j^l, w_j^u], n_j)$, is a vector including two elements. Moreover, the first element of this vector represents the total cost of the aerial refueling operations from node $s^{\mathfrak{p}}$ to j, which is given by an interval number, where w_j^l and w_j^u are lower and upper bounds, respectively. The second element in this vector shows the total number of aerial refueling operations from the origin $s^{\mathfrak{p}}$ to node j. The first component of the node label shows that the inherent multi-objective nature of the problem is preserved, which may be considered an advantage of LA. In this manner, both objective functions are investigated simultaneously. Moreover, $pred(j)$ in the second and third components of $L^{\mathfrak{p}}(j)$ denotes the node immediately prior to node j on path $\mathfrak{p}$. The station from which a tanker aircraft is allocated to $pred(j)$ is given by $sta(pred(j))$.

At the beginning of the algorithm, each node of path $\mathfrak{p}$ is labeled as follows:

$$L^{\mathfrak{p}}(s^{\mathfrak{p}}) = (([0,0],0),-,-),$$
$$L^{\mathfrak{p}}(j) = (([\infty,\infty],\infty),-,-) \quad \forall j \in \mathcal{N}^{\mathfrak{p}}, \quad j \neq s^{\mathfrak{p}}.$$

A set $\mathcal{D}^{\mathfrak{p}}(j)$ of labels is associated with each node $j \in \mathcal{N}^{\mathfrak{p}}$. Initially, this set contains the abovementioned labels. The set $\mathcal{D}^{\mathfrak{p}}(j)$ is updated during the algorithm in such a way that no first component of each element (each label) in $\mathcal{D}^{\mathfrak{p}}(j)$ is dominated by another first component of the labels in $\mathcal{D}^{\mathfrak{p}}(j)$. We try to improve LA according to the inherent interval multi-objective nature of the proposed problem and to make it more applicable to real-world situations. To do so, the order " $\preccurlyeq_{IV}$ " is used in the process of determining the dominant objectives, as explained in detail during the algorithm procedure. Another set that is required is SE. This set initially contains $s^{\mathfrak{p}}$ (i.e., $SE = \{s^{\mathfrak{p}}\}$) and is updated during the algorithm. At first, the only member of SE is removed, and all arcs $(s^{\mathfrak{p}}, j)$ emanating from node $s^{\mathfrak{p}}$ in $\mathcal{G}^{\mathfrak{p}}$ are found. Next, to create a new label for each node j on path $\mathfrak{p}$, where $(s^{\mathfrak{p}}, j) \in \mathcal{E}^{\mathfrak{p}}$, a label-scanning process should be started. With regard to the label of $s^{\mathfrak{p}}$ and the fact that no refueling occurs at node $s^{\mathfrak{p}}$, the new label of j will be $L^{\mathfrak{p}}(j) = (([0,0],0),-,s^{\mathfrak{p}})$. Then, $L^{\mathfrak{p}}(j)$ is added to the set $\mathcal{D}^{\mathfrak{p}}(j)$, and the first components of the two labels in $\mathcal{D}^{\mathfrak{p}}(j)$ are compared with each other according to the order " $\preccurlyeq_{IV}$ ". Obviously, $([0,0],0) \preccurlyeq_{IV} ([\infty,\infty],\infty)$, which indicates that the vector ([0, 0] , 0) dominates the vector $([\infty,\infty],\infty)$ due to the minimization inherent in the objectives in the proposed model. Hence, the label of j with the dominated first component is removed from $\mathcal{D}^{\mathfrak{p}}(j)$. Accordingly, the set $\mathcal{D}^{\mathfrak{p}}(j)$ is updated. Since $j \notin SE$, it is added to the set SE.

In the preceding discussion, the algorithm was described for the case in which the set SE contains only node $s^{\mathfrak{p}}$, at which no refueling is performed. In the following descriptions, the algorithm is explained for the case in which SE contains nodes at which refueling may be performed.

In the rest of the algorithm, a member of SE, suppose i, is selected and removed from SE. Then, all arcs $(i,j) \in \mathcal{E}^{\mathfrak{p}}$ on path $\mathfrak{p}$ are determined, and the label-scanning process is performed. Obviously, $\mathcal{D}^{\mathfrak{p}}(i)$ may contain some labels whose first components are incomparable, considering the order " $\preccurlyeq_{IV}$ ". To find possible new labels for each node j, first the following vector is computed regarding the first component of each member of $\mathcal{D}^{\mathfrak{p}}(i)$:

$$\left(\left[w_i^l, w_i^u\right], n_i\right) + \left(\left[c_{ki}^l, c_{ki}^u\right], 1\right) = \left(\left[w_i^l + c_{ki}^l, w_i^u + c_{ki}^u\right], n_i + 1\right) \quad \forall k \in K.$$

Then, the vectors obtained for all members of $\mathcal{D}^{\mathfrak{p}}(i)$ are compared with respect to the order " $\preccurlyeq_{IV}$ ", and the nondominated ones are kept. The labels for these nondominated vectors are created and added to the set $\mathcal{D}^{\mathfrak{p}}(j)$. Finally, the first components of all labels in the set $\mathcal{D}^{\mathfrak{p}}(j)$ are compared with each other, and all dominated ones are removed from $\mathcal{D}^{\mathfrak{p}}(j)$. Now, the set $\mathcal{D}^{\mathfrak{p}}(j)$ is updated. Node j should be added to the set SE if the set does not include it. Once again a member of SE is selected, and the same process is executed while $SE \neq \phi$. The route traveled by the aircraft and the allocations of tanker aircraft from stations to aerial refueling nodes, which may not be unique, can be determined via post-processing by backtracking through the predecessor labels. The proposed algorithm is shown by the pseudo-code in Fig. 4.

```
Initialization. Set
L^p(s^p) := (([0,0],0),−,−);
L^p(j) := (([∞,∞],∞),−,−), for j ∈ N^p − {s^p};
D^p(j) := {L^p(j)};
SE := {s^p};
while SE ≠ φ
    remove a node i from SE;
    for each arc (i,j) ∈ E^p emanating from node i
        if i = s^p replace L^p(j) by (([0,0],0),−,s^p) and update the set D^p(j)
        else (i ≠ s^p) for each L^p(i) = (([w_i^l, w_i^u], n_i), sta(pred(i)), pred(i)) ∈ D^p(i)
        compute {([w_i^l, w_i^u], n_i) + ([c_ki^l, c_ki^u], 1)}, ∀k ∈ K, compare the obtained vectors
        for all members of D^p(i) and keep nondominated ones. Then, create labels
        for nondominated vectors. Finally, add the obtained labels to the set D^p(j)
        and remove labels with dominated first components.
            if j ∉ SE
            add j to SE;
            end;
          end;
        end;
    end;
end;
```

Fig. 4 The pseudo-code of the labeling algorithm

4.2 Genetic algorithm

Recently, genetic algorithms (GA) have been applied to many problems (Roy and Mula 2016; Spiliopoulou et al. 2017; Marinakis et al. 2009). GA is a stochastic search technique based on the mechanism of natural selection and natural genetics (Holland 1992). GA begins with a randomly selected population of chromosomes represented by strings. The chromosomes in the population represent potential solutions to the problem of interest. Each chromosome in the population is evaluated using some measure of its fitness. By using some neighborhood operators, e.g., crossover and mutation, GA can find new solutions (named offspring) with characteristics that may not already exist in the population and that differ from those of some of the selected solutions of the population (named parents). Afterward, the offspring are compared with the population (according to their fitness values) and substituted for the worst chromosomes in the population; thus, the next generation is created. This procedure continues until a termination condition is satisfied. In the following paragraphs, the implementation details of GA according to our problem are explained.

GA requires a genetic representation of the solution domain. An appropriate chromosome representation is important for the successful application of GA. Therefore, according to the problem in this study, we consider each chromosome in a fixed-length multiple-string form (matrix form). Each chromosome represents a feasible solution in $\mathcal{G}^{\mathfrak{p}}$. The first string represents the set of nodes $\mathcal{N}^{\mathfrak{p}}$. The locus of the chromosome in the first string shows a node on $\mathfrak{p}$. Each gene takes the value 1 if the receiver aircraft enters the corresponding node on path $\mathfrak{p}$ and 0 otherwise . The genes of the first locus and the last locus are equal to one and always reserved for the source node, $s^{\mathfrak{p}}$, and the destination node, $t^{\mathfrak{p}}$, respectively. The length of the chromosome is equal to the total number of nodes on path $\mathfrak{p}$. Each of the remaining strings of the chromosome shows one refueling station. In fact, the number of strings in the chromosome is equal to the size of set K plus one. Each gene in the string of the k-th station takes the value 1 if the aerial refueling nodes in the first string are served by a tanker aircraft from the k-th station and 0 otherwise. An example of the chromosome encoding for aerial refueling nodes $\{B, C\}$ on path $\mathfrak{p}$, $s^{\mathfrak{p}} - A - B - C - D - t^{\mathfrak{p}}$, in $\mathcal{G}^{\mathfrak{p}}$, assuming two stations (node B is served by a tanker aircraft from the second station and node C is served by the first station) is shown in Fig. 5.

Once the chromosome representation is defined, GA proceeds to generate an initial population of solutions (pop_0). The initial population is generated randomly. The fitness function of each chromosome in the population is then calculated. In fact, the fitness function measures the quality of the chromosomes. In this study, the fitness function is considered as a vector-valued function including two elements. The first element is an interval that represents the total cost of aerial refuelings, and the second element is the total number of refuelings of the chromosome. Then, the fitness function is used as a criterion for selecting chromosomes for the crossover operation. In this regard, the fitness functions of chromosomes are compared using the order " $\preccurlyeq_{IV}$ ", and two nondominated chromosomes are selected as parents for the crossover operation.

Now, the selected parents are ready for crossover to create new offspring. The chromosomes selected for crossover (parents) should have at least one gene (node) in common other than the source and destination nodes. If they do not have any nodes

Locus	$s^{\mathfrak{p}}$	A	B	C	D	$t^{\mathfrak{p}}$
	1	0	1	1	0	1
First station	0	0	0	1	0	0
Second station	0	0	1	0	0	0

Fig. 5 An example of a chromosome representation

in common, another pair of parents should be selected from the population. The node in common is a crossover site. If the parents have more than one node in common, one node is selected randomly as a cross site. To create offspring, the fitness function of the parents should be compared according to the order " $\preccurlyeq_{IV}$ " in two steps. First, the fitness functions corresponding to the source node up to the crossover site of the parents are compared, and the nondominated part is preserved for the first part of the offspring. Then, the fitness functions corresponding to the crossover site up to the destination node of the parents are compared and, the nondominated part is preserved for the second part of the offspring. The offspring are thus created. If in each step of the comparison procedure the fitness functions are incomparable according to the order " $\preccurlyeq_{IV}$ ", then both parts of the parents are preserved. As an example, the crossover operator of the parents in Fig. 6 can be referred to, where the interval costs of aerial refueling operations at nodes B, C and D, which are served by stations 1 and 2, are $c_{1B} = [3, 4], c_{2B} = [5, 6], c_{1C} = [2, 4]$, and $c_{2D} = [3, 2]$.

Once the offspring are created, the mutation operator is applied to them. The purpose of mutation is to preserve diversity. Mutation alters one gene of the offspring with a probability equal to the mutation rate p_m. In this paper, bit-reverse type mutation is adopted for one of the strings, except the first string of the offspring. In the end, each offspring is compared with all chromosomes of the population, and the worst chromosome is replaced by the offspring. If the offspring are incomparable with the

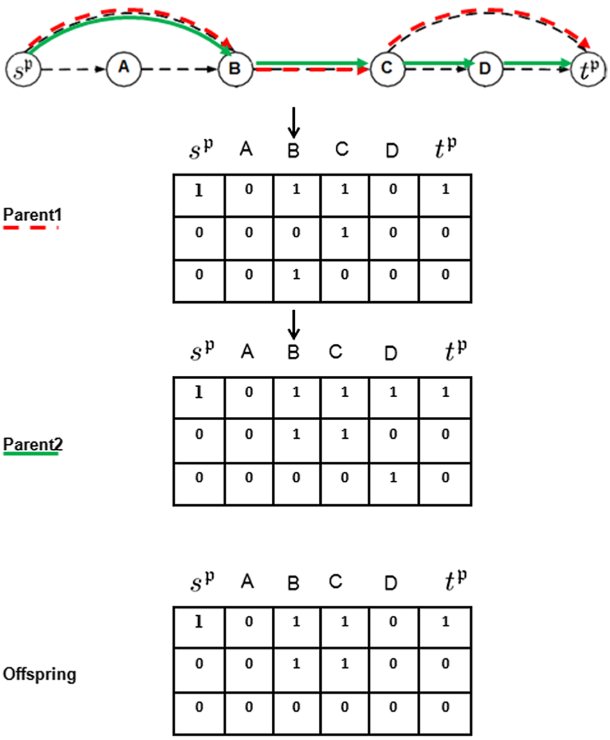

Fig. 6 An example of a crossover operation

chromosomes, they are added to the population, and the next generation is created. This procedure continues until a termination condition is satisfied, which can be stated in terms of either CPU time or the number of iterations.

5 Numerical experiments

To tackle the problem introduced in Sect. 3, two algorithms were proposed in Sect. 4, and the performance of the algorithms is studied in this section. The required experiments are performed in the following subsections. In Sect. 5.1, an example network with 11 nodes, which is sufficiently small to report the details of the obtained solutions, is presented. In Sect. 5.2, we test and compare the solution methods in 21 instances of networks in different cases, which will allow us to evaluate how well the algorithms perform in large, real-world instances. The algorithms were coded in MATLAB 8.1.0.604 (R2013a), and all experiments were performed on a laptop equipped with a Core i5 Processor (2.40 GHz) and 4.00 GB RAM.

5.1 Network with 11 nodes

Consider an example network with 11 nodes and some arcs (Fig. 7). It is assumed that two predetermined stations containing tanker aircraft exist on the ground. The interval number, which is written on each arc, denotes the interval value for fuel consumption for traveling that arc (measured in pounds). Furthermore, interval costs, c_{kj}, $k = 1, 2, j = 1, 2, \ldots, 11$, for allocating stations to the nodes are specified as follows:

$$c_{11} = [3, 4] \quad c_{12} = [12, 14] \quad c_{13} = [21, 22] \quad c_{14} = [4, 5] \quad c_{15} = [2, 4]$$
$$c_{16} = [1, 2] \quad c_{17} = [7, 8] \quad c_{18} = [11, 13] \quad c_{19} = [3, 4] \quad c_{1,10} = [2, 3]$$
$$c_{1,11} = [9, 10] \quad c_{21} = [7, 8] \quad c_{22} = [20, 21] \quad c_{23} = [19, 21] \quad c_{24} = [7, 9]$$
$$c_{25} = [3, 4] \quad c_{26} = [3, 4] \quad c_{27} = [10, 11] \quad c_{28} = [5, 7] \quad c_{29} = [5, 6]$$
$$c_{2,10} = [4, 4] \quad c_{2,11} = [11, 13].$$

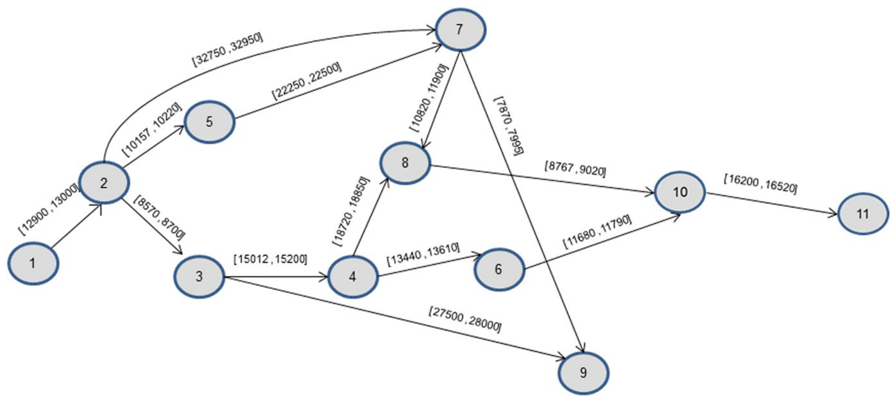

Fig. 7 The 11-node network

Moreover, two ranges are considered for receiver aircraft. One is $R = 26{,}000$ pounds, which was also used in the study by Kannon et al. (2015). They utilized $R = 26{,}000$ based on the fuel capacity of a specified aircraft. Another range considered in this subsection is $R = 22{,}000$ pounds. The shortest path between each $O - D$ pair is found. The results for the 11-node network obtained by using the labeling algorithm for the two ranges, $R = 26{,}000$ and $R = 22{,}000$ (measured in pounds), are presented in Table 2. Using LA, IV-efficient solutions and their corresponding objective function values for some specific $O - D$ pairs are presented in this table. Here, the symbol $i[k]$ denotes that the aircraft is refueled at node i by a tanker aircraft from the k-th station. $i[-]$ indicates that the aircraft is not refueled at node i. As an example, in Table 2, $2[-] \longrightarrow 4[1] \longrightarrow 6[-]$ is an IV-efficient solution between nodes 2 and 6 when $R = 26{,}000$. This solution implies that, for traveling through nodes 2 and 6, refueling at node 4 is necessary. In addition a tanker aircraft from the first station is allocated to node 4. Table 2 indicates that for $R = 22{,}000$, three IV-efficient solutions are obtained along the route between nodes 2 and 6. All of these solutions specify the same path, $2 \longrightarrow 3 \longrightarrow 4 \longrightarrow 6$; however, the allocations of the stations to nodes 3 and 4 along this path, at which refueling should be performed, are not the same. The IV-efficient solutions for the $O - D$ pair $1 - 11$ considering $R = 26{,}000$ show that it is possible to have IV-efficient solutions with different numbers of refueling operations. In this sense, the solution $1[-] \longrightarrow 2[1] \longrightarrow 4[1] \longrightarrow 10[1] \longrightarrow 11[-]$ indicates that three refueling operations are necessary. However, the solution $1[-] \longrightarrow 2[1] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$ shows that four refueling operations are needed. In total, LA obtained 51 and 71 IV-efficient solutions for all $O - D$ pairs in the 11-node network for ranges 26,000 and 22,000, respectively.

The results for the 11-node network, obtained using GA with population sizes of 5, 10, and 15 for 100 generations with the same ranges as LA are presented in Tables 3, 4, and 5. The IV-efficient solutions and the corresponding objective functions are obtained for the same $O - D$ pairs that were selected for LA.

The results in Table 3 are obtained for $pop_0 = 5$. As shown in the table, for $R = 26{,}000$, five IV-efficient solutions are found for traveling from node 1 to node 11. All of these solutions are dominated by the solutions obtained by LA (Table 2) in the same condition. Similar results are obtained for $O = 3$ and $D = 11$. For $O = 2$ and $D = 11$, considering $R = 26{,}000$, GA yields two solutions. One, ([6, 8], 2), is the same as that obtained by LA. However, the other, ([7, 10], 3), is new and is incomparable with LA's solutions. In fact, due to the characteristics of LA and the defined order " $\preccurlyeq_{IV}$ " for comparison, LA misses this solution. The final $O - D$ in $R = 26{,}000$ with $O = 2$ and $D = 6$ has the same solution as LA's solution in Table 2. Consider the solutions in the case of $R = 22{,}000$ in Table 3. For $O = 1$ and $D = 11$, GA can find only three of the nine solutions found by LA in Table 2. For the path from $O = 3$ to $D = 11$, the vector ([11, 13], 3) in Table 3 is dominated by ([7, 10], 3) in Table 2. Furthermore, GA can find an extra solution, ([10, 14], 3), that is incomparable with all solutions obtained by LA. For $O = 2$ and $D = 11$, GA finds two more solutions than LA. The solutions in Tables 2 and 3 are the same for the path from $O = 2$ to $D = 6$.

Table 2 Results of 11-node network for LA

Range	$O-D$	IV-efficient solution	Obj. functions
26,000	$O=1, D=11$	$1[-] \longrightarrow 2[1] \longrightarrow 4[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([18, 22],3)
		$1[-] \longrightarrow 2[1] \longrightarrow 4[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([20, 23]),3)
		$1[-] \longrightarrow 2[1] \longrightarrow 4[2] \longrightarrow 10[1] \longrightarrow 11[-]$	([21, 26],3)
		$1[-] \longrightarrow 2[1] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([19, 24],4)
		$1[-] \longrightarrow 2[1] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([21, 25],4)
	$O=3, D=11$	$3[-] \longrightarrow 4[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([6, 8],2)
		$3[-] \longrightarrow 4[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([8, 9],2)
	$O=2, D=11$	$2[-] \longrightarrow 4[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([6, 8],2)
		$2[-] \longrightarrow 4[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([8, 9],2)
	$O=2, D=6$	$2[-] \longrightarrow 4[1] \longrightarrow 6[-]$	([4, 5],1)
22,000	$O=1, D=11$	$1[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([28, 32],4)
		$1[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([30, 33],4)
		$1[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 6[2] \longrightarrow 10[1] \longrightarrow 11[-]$	([30, 34],4)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([26, 31],4)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([28, 32],4)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[2] \longrightarrow 10[1] \longrightarrow 11[-]$	([28, 33],4)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[2] \longrightarrow 10[2] \longrightarrow 11[-]$	([30, 34],4)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[2] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([29, 35],4)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[2] \longrightarrow 6[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([31, 36],4)
	$O=3, D=11$	$3[-] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([9, 11],3)
		$3[-] \longrightarrow 4[1] \longrightarrow 6[2] \longrightarrow 10[1] \longrightarrow 11[-]$	([9, 12],3)
		$3[-] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([7, 10],3)
	$O=2, D=11$	$2[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([28, 32],4)
		$2[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([30, 33],4)
		$2[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 6[2] \longrightarrow 10[1] \longrightarrow 11[-]$	([30, 34],4)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([26, 31],4)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([28, 32],4)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[2] \longrightarrow 10[1] \longrightarrow 11[-]$	([28, 33],4)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[2] \longrightarrow 10[2] \longrightarrow 11[-]$	([30, 34],4)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[2] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([29, 35],4)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[2] \longrightarrow 6[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([31, 36],4)
	$O=2, D=6$	$2[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 6[-]$	([25, 27],2)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[-]$	([23, 26],2)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[2] \longrightarrow 6[-]$	([26, 30],2)

To obtain the results in Tables 4 and 5, GA is run for $pop_0 = 10$ and $pop_0 = 15$, respectively. With small population sizes such as 5 and 10, some of the solutions generated by GA are dominated by the solutions obtained by LA. In the results obtained by GA when $pop_0 = 15$, all of LA's solutions plus some extra solutions are obtained for some selected $O-D$ pairs. In this case, as observed in Table 3, the extra solutions are incomparable with all of LA's solutions. In

Table 3 Results of 11-node network for GA with $pop_0 = 5$

Range	$O-D$	IV-efficient solution	Obj. functions
26,000	$O=1, D=11$	1[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([26, 31],4)
		1[−] ⟶ 3[1] ⟶ 4[1] ⟶ 10[2] ⟶ 11[−]	([29, 31],3)
		1[−] ⟶ 3[2] ⟶ 4[2] ⟶ 10[1] ⟶ 11[−]	([28, 33],3)
		1[−] ⟶ 3[2] ⟶ 4[1] ⟶ 10[2] ⟶ 11[−]	([27, 30],3)
		1[−] ⟶ 3[2] ⟶ 4[2] ⟶ 10[2] ⟶ 11[−]	([30, 34],3)
	$O=3, D=11$	3[−] ⟶ 4[2] ⟶ 10[2] ⟶ 11[−]	([11, 13],2)
		3[−] ⟶ 4[2] ⟶ 10[1] ⟶ 11[−]	([9, 12],2)
	$O=2, D=11$	2[−] ⟶ 4[1] ⟶ 10[1] ⟶ 11[−]	([6, 8],2)
		2[−] ⟶ 4[1] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([7, 10],3)
	$O=2, D=6$	2[−] ⟶ 4[1] ⟶ 6[−]	([4, 5],1)
22,000	$O=1, D=11$	1[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([26, 31],4)
		1[−] ⟶ 3[1] ⟶ 4[1] ⟶ 6[1] ⟶ 10[2] ⟶ 11[−]	([30, 33],4)
		1[−] ⟶ 3[1] ⟶ 4[1] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([28, 32],4)
	$O=3, D=11$	3[−] ⟶ 4[2] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([10, 14],3)
		3[−] ⟶ 4[1] ⟶ 6[2] ⟶ 10[2] ⟶ 11[−]	([11, 13],3)
		3[−] ⟶ 4[1] ⟶ 6[1] ⟶ 10[2] ⟶ 11[−]	([9, 11],3)
		3[−] ⟶ 4[1] ⟶ 6[2] ⟶ 10[1] ⟶ 11[−]	([9, 12],3)
	$O=2, D=11$	2[−] ⟶ 3[1] ⟶ 4[1] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([28, 32],4)
		2[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[1] ⟶ 10[2] ⟶ 11[−]	([28, 32],4)
		2[−] ⟶ 3[1] ⟶ 4[1] ⟶ 6[1] ⟶ 10[2] ⟶ 11[−]	([30, 33],4)
		2[−] ⟶ 3[1] ⟶ 4[1] ⟶ 6[2] ⟶ 10[1] ⟶ 11[−]	([30, 34],4)
		2[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[2] ⟶ 10[2] ⟶ 11[−]	([30, 34],4)
		2[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([26, 31],4)
		2[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[2] ⟶ 10[1] ⟶ 11[−]	([28, 33],4)
		2[−] ⟶ 3[2] ⟶ 4[2] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([29, 35],4)
		2[−] ⟶ 3[1] ⟶ 4[2] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([31, 36],4)
		2[−] ⟶ 3[2] ⟶ 4[2] ⟶ 6[2] ⟶ 10[1] ⟶ 11[−]	([31, 37],4)
		2[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[1] ⟶ 10[2] ⟶ 11[−]	([31, 36],4)
	$O=2, D=6$	2[−] ⟶ 3[1] ⟶ 4[1] ⟶ 6[−]	([25, 27],2)
		2[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[−]	([23, 26],2)
		2[−] ⟶ 3[2] ⟶ 4[2] ⟶ 6[−]	([26, 30],2)

fact, GA's results depend on the size of the initial population. Increasing the initial population size improves the solutions obtained by GA and helps GA obtain extra solutions that are incomparable with those of LA. In the 11-node network, GA with $pop_0 = 15$ obtains 53 and 77 IV-efficient solutions for all $O-D$ pairs for $R = 26{,}000$ and $R = 22{,}000$, respectively. Obviously, the number of GA solutions is larger than the number of LA solutions. However, in this case, the running time also increases. In fact, GA even requires a long time to obtain the same solutions obtained by LA. Computational times are not reported in these tables

Table 4 Results of 11-node network for GA with $pop_0 = 10$

Range	$O-D$	IV-efficient solution	Obj. functions
26,000	$O=1, D=11$	$1[-] \longrightarrow 2[2] \longrightarrow 4[2] \longrightarrow 10[1] \longrightarrow 11[-]$	([29, 33]),3)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([26, 31],4)
		$1[-] \longrightarrow 2[2] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([27, 31]),4)
		$1[-] \longrightarrow 2[2] \longrightarrow 4[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([26, 29]),3)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([25, 29]),3)
		$1[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([27, 30]),3)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[2] \longrightarrow 10[1] \longrightarrow 11[-]$	([28, 33]),3)
		$1[-] \longrightarrow 2[2] \longrightarrow 4[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([28, 30]),3)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([27, 30]),3)
		$1[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([29, 31]),3)
		$1[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([28, 32]),4)
	$O=3, D=11$	$3[-] \longrightarrow 4[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([6, 8],2)
		$3[-] \longrightarrow 4[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([8, 9],2)
		$2[-] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([7, 10],3)
	$O=2, D=11$	$2[-] \longrightarrow 4[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([6, 8],2)
		$2[-] \longrightarrow 4[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([8, 9],2)
		$2[-] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([7, 10],3)
	$O=2, D=6$	$2[-] \longrightarrow 4[1] \longrightarrow 6[-]$	([4, 5],1)

because the network is very small, but these times are discussed in detail in the next subsection.

5.2 Randomly generated networks

To generate an example network, n nodes were randomly generated in the square $[0, 30{,}000] \times [0, 30{,}000]$. Then, according to the Euclidean distance between these nodes, each node was connected to at most m (an arbitrary natural number) nearest adjacent nodes. It was assumed that the direction of the arc between nodes i and j was from i to j if $i < j$. The amount of fuel consumed for each arc (i, j) was determined considering the Euclidean distance between nodes i and j. Moreover, we generated integer values for the lower and upper bounds of the fuel consumption interval for each arc (i, j) by multiplying the specified Euclidean distance by the coefficients 0.95 and 1.05, respectively, and subsequently rounding (the values of the coefficients chosen to construct the interval were taken from the study by Kannon et al. (2015)). Then, the shortest path between each $O-D$ pair was found. The network example was constructed using all of these shortest paths.

To investigate the efficiency of the proposed formulation and the algorithms in real-world conditions, seven random networks were generated by the above method. It was assumed that there were 10 predetermined stations on the ground, and the receiver's range was set to $R = 26{,}000$.

Table 7 compares the solution times and the number of solutions of both LA and GA on 21 instances of networks in different cases. In this table, "No. of solutions"

Table 4 (continued)

Range	$O-D$	IV-efficient solution	Obj. functions
22,000	$O=1, D=11$	1[−] ⟶ 3[1] ⟶ 4[1] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([28, 32],4)
		1[−] ⟶ 3[2] ⟶ 4[2] ⟶ 6[2] ⟶ 10[1] ⟶ 11[−]	([31, 37],4)
		1[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[1] ⟶ 10[2] ⟶ 11[−]	([28, 32],4)
		1[−] ⟶ 3[2] ⟶ 4[2] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([29, 35],4)
		1[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([26, 31],4)
		1[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[2] ⟶ 10[1] ⟶ 11[−]	([28, 33],4)
		1[−] ⟶ 3[1] ⟶ 4[1] ⟶ 6[2] ⟶ 10[1] ⟶ 11[−]	([30, 34],4)
		1[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[2] ⟶ 10[2] ⟶ 11[−]	([30, 34],4)
		1[−] ⟶ 3[1] ⟶ 4[2] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([31, 36],4)
	$O=3, D=11$	3[−] ⟶ 4[1] ⟶ 6[1] ⟶ 10[2] ⟶ 11[−]	([9, 11],3)
		3[−] ⟶ 4[1] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([7, 10],3)
		3[−] ⟶ 4[1] ⟶ 6[2] ⟶ 10[1] ⟶ 11[−]	([9, 12],3)
		3[−] ⟶ 4[2] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([10, 14],3)
	$O=2, D=11$	2[−] ⟶ 3[1] ⟶ 4[1] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([28, 32],4)
		2[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[1] ⟶ 10[2] ⟶ 11[−]	([28, 32],4)
		2[−] ⟶ 3[1] ⟶ 4[1] ⟶ 6[1] ⟶ 10[2] ⟶ 11[−]	([30, 33],4)
		2[−] ⟶ 3[1] ⟶ 4[1] ⟶ 6[2] ⟶ 10[1] ⟶ 11[−]	([30, 34],4)
		2[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[2] ⟶ 10[2] ⟶ 11[−]	([30, 34],4)
		2[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([26, 31],4)
		2[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[2] ⟶ 10[1] ⟶ 11[−]	([28, 33],4)
		2[−] ⟶ 3[2] ⟶ 4[2] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([29, 35],4)
		2[−] ⟶ 3[1] ⟶ 4[2] ⟶ 6[1] ⟶ 10[1] ⟶ 11[−]	([31, 36],4)
		2[−] ⟶ 3[2] ⟶ 4[2] ⟶ 6[2] ⟶ 10[1] ⟶ 11[−]	([31, 37],4)
		2[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[1] ⟶ 10[2] ⟶ 11[−]	([31, 36],4)
	$O=2, D=6$	2[−] ⟶ 3[1] ⟶ 4[1] ⟶ 6[−]	([25, 27],2)
		2[−] ⟶ 3[2] ⟶ 4[1] ⟶ 6[−]	([23, 26],2)
		2[−] ⟶ 3[2] ⟶ 4[2] ⟶ 6[−]	([26, 30],2)

is the number of IV-efficient solutions obtained by the algorithms, and "Time" is the CPU time required to run the algorithms to obtain IV-efficient solutions for all $O-D$ pairs. We used the results generated by LA as a guideline to determine whether good solutions were reached by GA. As mentioned previously, on the basis of the initial population, GA may find solutions that are dominated by the solutions of LA that are not in common with those of GA. In Table 7, "No. of solutions in common" shows the number of solutions that LA and GA have in common. To investigate the effect of the initial population size on the obtained solutions and the running time, GA was run with three initial population sizes, $pop_0 = 50, 100$, and 200, for 100 generations. We tested mutation rates of 0.0, 0.1, 0.2, 0.7, 0.8, and 0.9 for only two random networks with 50 and 100 nodes and $pop_0 = 50$ to determine how this rate affects the performance of GA (Table 6). In some cases, as the mutation rate incrased, the number of solutions obtained by GA and the number

Table 5 Results of 11-node network for GA with $pop_0 = 15$

Range	$O-D$	IV-efficient solution	Obj. functions
26,000	$O=1, D=11$	$1[-] \longrightarrow 2[1] \longrightarrow 4[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([18, 22],3)
		$1[-] \longrightarrow 2[1] \longrightarrow 4[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([20, 23]),3)
		$1[-] \longrightarrow 2[1] \longrightarrow 4[2] \longrightarrow 10[1] \longrightarrow 11[-]$	([21, 26],3)
		$1[-] \longrightarrow 2[1] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([19, 24],4)
		$1[-] \longrightarrow 2[1] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([21, 25],4)
	$O=3, D=11$	$3[-] \longrightarrow 4[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([6, 8],2)
		$3[-] \longrightarrow 4[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([8, 9],2)
		$3[-] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([7, 10],3)
	$O=2, D=11$	$2[-] \longrightarrow 4[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([6, 8],2)
		$2[-] \longrightarrow 4[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([8, 9],2)
		$2[-] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([7, 10],3)
	$O=2, D=6$	$2[-] \longrightarrow 4[1] \longrightarrow 6[-]$	([4, 5],1)
22,000	$O=1, D=11$	$1[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([28, 32],4)
		$1[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([30, 33],4)
		$1[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 6[2] \longrightarrow 10[1] \longrightarrow 11[-]$	([30, 34],4)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([26, 31],4)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([28, 32],4)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[2] \longrightarrow 10[1] \longrightarrow 11[-]$	([28, 33],4)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[2] \longrightarrow 10[2] \longrightarrow 11[-]$	([30, 34],4)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[2] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([29, 35],4)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[2] \longrightarrow 6[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([31, 36],4)
		$1[-] \longrightarrow 3[1] \longrightarrow 4[2] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([31, 36],4)
		$1[-] \longrightarrow 3[2] \longrightarrow 4[2] \longrightarrow 6[2] \longrightarrow 10[1] \longrightarrow 11[-]$	([31, 37],4)
	$O=3, D=11$	$3[-] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([9, 11],3)
		$3[-] \longrightarrow 4[1] \longrightarrow 6[2] \longrightarrow 10[1] \longrightarrow 11[-]$	([9, 12],3)
		$3[-] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([7, 10],3)
		$3[-] \longrightarrow 4[2] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([10, 14],3)
	$O=2, D=11$	$2[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([28, 32],4)
		$2[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([30, 33],4)
		$2[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 6[2] \longrightarrow 10[1] \longrightarrow 11[-]$	([30, 34],4)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([26, 31],4)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([28, 32],4)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[2] \longrightarrow 10[1] \longrightarrow 11[-]$	([28, 33],4)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[2] \longrightarrow 10[2] \longrightarrow 11[-]$	([30, 34],4)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[2] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([29, 35],4)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[2] \longrightarrow 6[1] \longrightarrow 10[2] \longrightarrow 11[-]$	([31, 36],4)
		$2[-] \longrightarrow 3[1] \longrightarrow 4[2] \longrightarrow 6[1] \longrightarrow 10[1] \longrightarrow 11[-]$	([31, 36],4)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[2] \longrightarrow 6[2] \longrightarrow 10[1] \longrightarrow 11[-]$	([31, 37],4)
	$O=2, D=6$	$2[-] \longrightarrow 3[1] \longrightarrow 4[1] \longrightarrow 6[-]$	([25, 27],2)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[1] \longrightarrow 6[-]$	([23, 26],2)
		$2[-] \longrightarrow 3[2] \longrightarrow 4[2] \longrightarrow 6[-]$	([26, 30],2)

of solutions in common increased, and the running time decreased. Therefore, we assumed $p_m = 0.8$ in Table 7.

The results in Table 7 show that GA can find a smaller or equal number of solutions compared with LA. Moreover, GA requires too much time to obtain these solutions. Despite the long time, GA is still unable to find all of LA's solutions in cases in which "No. of solutions in common" is smaller than "No. of solutions" in LA. In such cases (where GA is unable to find all of LA's solutions), the solutions obtained by GA that are not in common with those found by LA may be dominated by those solutions of LA that are not in common with the solutions of GA. For example, in the network with 50 nodes and $pop_0 = 50$, LA takes only 183.25 seconds to yield 3118 IV-efficient solutions. GA reports 1952 solutions in a time equivalent to almost seven times longer than the running time for LA. Moreover, GA finds only 1562 of LA's solutions. Thus, 390 of the solutions ($1952 - 1562 = 390$) obtained by GA may be dominated by the 1556 solutions ($3118 - 1562 = 1556$) obtained by LA that are not in common with those of GA.

The results show that an increase in the initial population size leads not only to an increase in the number of GA solutions but also to an increase in the running time. In large-scale networks, such as networks with 200, 300, 400, and 500 nodes, as the population increases, the running time increases exponentially. As an example, in the network with 200 nodes and $pop_0 = 200$, GA requires more than 12 hours of running time. Therefore, we terminate GA within 7200 s (i.e., 2 hrs). In contrast to the long running time required by GA, LA can obtain IV-efficient solutions in large-scale networks in a very short running time (less than 3 minutes). Generally, we can conclude that the performance of LA is better than that of GA.

6 Conclusion

In this paper, an aerial refueling problem was investigated. Due to the limited range of receiver aircraft, aerial refueling is necessary. In this context, valid combinations of aerial refueling nodes were determined. To select suitable valid

Table 6 Results of using different mutation rates for GA

Mutation rate	Nodes = 50			Nodes = 100		
	No. of GA solutions	No. of solutions in common	Time (s)	No. of GA solutions	No. of solutions in common	Time (s)
0.0	1660	1296	1424.68	1494	1463	2644.10
0.1	1947	1555	1604.81	1532	1522	3779.55
0.2	1932	1546	1444.49	1533	1522	3777.75
0.7	1933	1548	1235.44	1532	1519	2787.05
0.8	1952	1562	1248.77	1528	1514	2662.90
0.9	1933	1533	1192.62	1531	1518	2634.51

Table 7 Comparison of LA and GA with different population sizes

Network no.	Nodes	LA		GA			No. of solutions in common
		No. of solutions	Time (s)	pop_0	No. of solutions	Time (s)	
1	20	111	0.26	50	111	73.61	111
				100	111	41.04	111
				200	111	18.63	111
2	50	3118	183.25	50	1952	1248.77	1562
				100	2149	1037.90	1670
				200	2237	1446.91	1725
3	100	1577	4.94	50	1528	2662.90	1514
				100	1567	2481.83	1566
				200	1575	3534.46	1575
4	200	3655	12.87	50	3466	7167.16	3435
				100	3628	6876.95	3615
				200	2579	7200	2579
5	300	7086	40.32	50	4322	7200	3752
				100	4651	7200	4052
				200	3998	7200	3340
6	400	8258	37.48	50	4115	7200	3843
				100	2606	7200	2413
				200	1215	7200	1207
7	500	15,816	91.04	50	2578	7200	2469
				100	2248	7200	2168
				200	992	7200	962

combinations, two criteria, minimization of the total cost of tanker aircraft allocations to aerial refueling nodes and minimization of the number of aerial refueling operations, were considered. Furthermore, due to the inherent uncertainty of real-world applications, the main problem of this paper was modeled as an interval multi-objective zero-one integer programming problem. Two new algorithms, namely, customized variants of labeling and genetic algorithms, respectively, were design to solve this model. Comparisons of the performance of the algorithms showed that the labeling algorithm obtains solutions in a short running time. However, as population size increases, the genetic algorithm tends to generate more nondominated solutions than the labeling algorithm at the expense of a much longer running time. As a result, it can be concluded that the performance of LA is better than that of GA.

Because a large set of IV-efficient solutions may be possible, compromise solutions may be desirable. Filtering methods would be suitable for this problem and could be considered as a topic for further research.

Acknowledgements The authors thank the reviewers' valuable comments and suggestions that helped them improve and clarify the paper. They would also like to express their gratitude to Dr. S. Niroomand and Dr. M. Hladik for editing this paper.

References

Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: theory, algorithms and applications. Prentice Hall, Upper Saddle River

Berman O, Krass D (1998) Flow intercepting spatial interaction model: a new approach to optimal location of competitive facilities. Locat Sci 6(1–4):41–65

Bush BA (2006) Analysis of fuel consumption for an aircraft deployment with multiple aerial refuelings. Thesis, North Carolina State University

Capar I, Kuby M (2011) An efficient formulation of the flow refueling location model for alternative-fuel stations. IIE Trans 44(8):622–636

Ehrgott M (2005) Multicriteria optimization, 2nd edn. Springer, Berlin

Hladik M (2016) Robust optimal solutions in interval linear programming with for all-exists quantifiers. Eur J Oper Res 254(3):705–714

Holland JH (1992) Adaptation in natural and artificial systems. University of Michigan Press, MIT Press, Cambridge

Jin Z, Shima T, Schumacher CJ (2006) Optimal scheduling for refueling multiple autonomous aerial vehicles. IEEE Trans Robot 22(4):682–693

Kannon TE, Nurre SG, Lunday BJ, Hill RR (2014) The aircraft routing with air refueling problem: exact and greedy approaches. In: Proceedings of the industrial and systems engineering research conference, Montreal, Canada, pp 817–824

Kannon TE, Nurre SG, Lunday BJ, Hill RR (2015) The aircraft routing problem with refueling. Optim Lett 9(8):1609–1624

Kaplan S, Rabadi G (2012) Exact and heuristic algorithms for the aerial refueling parallel machine scheduling problem with due date-to-deadline window and ready times. Comput Ind Eng 62(1):276–285

Kim JG, Kuby M (2012) The deviation-flow refueling location model for optimizing a network of refueling stations. Int J Hydrogen Energy 37(6):5406–5420

Kim J, Kuby M (2013) A network transformation heuristic approach for the deviation flow refueling location model. Comput Oper Res 40(4):1122–1131

Kuby M, Lim S (2005) The flow-refueling location problem for alternative-fuel vehicles. Socio Econ Plan Sci 39(2):125–145

Kuby M, Lim S (2007) Location of alternative-fuel stations using the flow-refueling location model and dispersion of candidate sites on arcs. Netw Spat Econ 7(2):129–152

Kuby M, Lines L, Schultz R, Xie Z, Kim JG, Lim S (2009) Optimization of hydrogen stations in Florida using the flow refueling location model. Int J Hydrogen Energy 34(15):6045–6064

Levy D, Sundar K, Rathinam S (2014) Heuristics for routing heterogeneous unmanned vehicles with fuel constraints. Math Probl Eng 2014:1–12

Lim S, Kuby M (2010) Heuristic algorithms for siting alternative fuel stations using the flow refueling location model. Eur J Oper Res 204(1):51–61

Marinakis Y, Marinaki M, Matsatsinis N, Zopounidis C (2009) Evolution of the population of a genetic algorithm using particle swarm optimization: application to clustering analysis. Oper Res Int J 9(1):105–120

MirHassani SA, Ebrazi R (2013) A flexible reformulation of the refueling station location problem. Transp Sci 47(4):617–628

Moore RM, Kearfott RB, Cloud MJ (2009) Introduction to interval analysis. SIAM Press, Philadelphia

Okada Sh, Gen M (1993) Order relation between intervals and its application to shortest path problem. Comput Ind Eng 25:147–150

Oliveira C, Antunes CH (2009) An interactive method of tackling uncertainty in interval multiple objective linear programming. J Math Sci 161(6):854–866

Rivaz S, Yaghoobi MA (2015) Weighted sum of maximum regrets in an interval MOLP problem. Int Trans Oper Res. https://doi.org/10.1111/itor.12216

Roy SK, Mula P (2016) Solving matrix game with rough payoffs using genetic algorithm. Oper Res Int J 16(1):117–130

Sengupta A, Kumar Pal T, Chakraborty D (2001) Interpretation of inequality constraints involving interval coefficients and a solution to interval linear programming. Fuzzy Sets Syst 119(1):129–138

Spiliopoulou A, Papamichail I, Papageorgiou M, Tyrinopoulos Y, Chrysoulakis J (2017) Macroscopic traffic flow model calibration using different optimization algorithms. Oper Res Int J 17(1):145–164

Steuer RE (1986) Multiple criteria optimization: theory, computation and application. Wiley, New York

Sundar K, Rathinam S (2014) Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots. IEEE Trans Autom Sci Eng 11(1):287–294

Thomas P, Bhandari U, Bullock S, Richardson T, Bois J (2014) Advances in air to air refueling. Prog Aerosp Sci 71:14–35

Upchurch C, Kuby M, Lim S (2009) A model for location of capacitated alternative-fuel stations. Geogr Anal 41(1):85–106

Upchurch C, Kuby M (2010) Comparing the p-median and flow refueling models for locating alternative-fuel stations. J Trans Geogr 18(6):750–758

Urli B, Nadeau R (1992) An interactive method to multi-objective linear programming problems with interval coefficients. INFOR 30(2):127–137

Wang Y-W, Lin C-C (2009) Locating road-vehicle refueling stations. Transp Res Part E Logist Transp Rev 45(5):821–829

Wang Y-W, Wang C-R (2010) Locating passenger vehicle refueling stations. Transp Res Part E Logist Transp Rev 46(5):791–801

Wu HC (2009) The Karush–Kuhn–Tucker optimality conditions in multi-objective programming problems with interval-valued objective functions. Eur J Oper Res 196(1):49–60

Zapata F, Kreinovich V, Joslyn C, Hogan E (2013) Orders on intervals over partially ordered sets: extending Allen's algebra and interval graph results. Soft Comput 17(8):1379–1391