J Optim Theory Appl (2011) 150:379–394
DOI 10.1007/s10957-011-9838-y

An Interactive Algorithm for Multi-objective Route Planning

Diclehan Tezcaner · Murat Köksalan

Published online: 19 April 2011

Abstract We address the route selection problem for Unmanned Air Vehicles (UAV) under multiple objectives. We consider a general case for this problem, where the UAV has to visit several targets and return to the base. We model this problem as a combination of two combinatorial problems. First, the path to be followed between each pair of targets should be determined. We model this as a multi-objective shortest path problem. Additionally, we need to determine the order of the targets to be visited. We model this as a multi-objective traveling salesperson problem (MOTSP). The overall problem is a combination of these two problems, which we define as a generalized MOTSP. We develop an exact interactive approach to identify the best paths and the best tour of a decision maker under a linear utility function.

Keywords Multi-objective decision making · Combinatorial optimization · Interactive method · Multi-objective shortest path · Multi-objective traveling salesperson problem · Unmanned Air Vehicle

Mathematics Subject Classification (2000) 90C29 · 90C27

1 Introduction

Unmanned Air Vehicles (UAVs) are unpiloted aircrafts that are used for both military and civilian purposes. In the military context, finding the "best" route that the UAV should follow through a defended area is critically important. This "best" route can be determined based on several criteria such as total distance traveled, fuel consumption, total flight time, detection threat avoidance, and navigation performance. Some of

Communicated by Harold P. Benson.

D. Tezcaner · M. Köksalan (✉)
Department of Industrial Engineering, Middle East Technical University, Ankara 06531, Turkey
e-mail: koksalan@ie.metu.edu.tr

these criteria are related and they are typically represented by two main criteria: total distance traveled and radar detection threat [1].

The route planning problem for UAVs could take one of the following two forms:

- Route planning with a single target, where the aim is to find the "best" path(s) between the starting point and the target.
- Route planning with multiple targets, where the aim is to find the "best" tour(s) that starts from the starting point, visits all targets and returns to the starting point.

This problem has been studied by several researchers using heuristic approaches. Gudaitis [1] assumes that the aircraft starts from an initial point and visits a single target in a three dimensional terrain, while minimizing the total travel distance and the radar detection threat. He combines these two objectives linearly and minimizes the resulting single objective problem using the A* algorithm that has been developed to solve the shortest path problems efficiently. Olsan [2] considers the same problem but uses a genetic algorithm to find a solution. Yavuz [3], on the other hand, assumes that the aircraft has to visit multiple targets during a mission. He again considers the same two objectives and tries to minimize a linear combination of them. He first tries to determine the best order to visit the targets using an algorithm called Particle Swarm Optimization and Ant System. He then tries to find the paths to be used between consecutive targets using the same algorithm.

Under the presence of multiple targets, the route planning of UAVs turns out to be a Multi-Objective Traveling Salesperson Problem (MOTSP). The single-objective Traveling Salesperson Problem (TSP) searches for a tour that has the best objective function value. The MOTSP, on the other hand, may contain many tours, each performing relatively better than others in some objectives. The overall problem may then be defined as finding a tour that gives the best combination of the objectives for the decision maker (DM).

In the literature, MOTSP has been defined to have a single path between any two consecutive nodes, to the best of our knowledge. For p-objectives, p separate cost matrices are defined, each matrix representing a single value for each link corresponding to a specific objective. However, in a practical setting, there are usually multiple paths between any pair of nodes. For example, if cities are our nodes, there are typically multiple ways of traveling between any two cities and it is unlikely to have a single path that is best in all objectives under consideration. In general, in a problem with conflicting multiple objectives, it is not clear which path is best up front. A path that performs well in a criterion typically does not perform very well in some other criteria. Therefore, in a realistic representation of the MOTSP, we need to consider that multiple competing paths may exist between nodes. This leads to a problem of finding efficient paths between the nodes included in the TSP. Then there is the problem of finding efficient tours that are made up of these efficient paths. Therefore, the problem turns into a combination of an interrelated Multi-Objective Shortest Path Problem (MOSPP) and a MOTSP. The MOTSP having single paths between nodes, as defined in the literature so far, is a special case of this generalized MOTSP.

In the generalized MOTSP, we need to consider the efficient paths between the nodes while searching for a preferred tour of a DM. Naturally, such a tour depends on the specific paths chosen between the nodes.

MOSPP is a well-known NP-hard problem (see [4], p. 166–167). Furthermore, finding all efficient paths is intractable, since they may increase exponentially with the problem size. There are many approaches for MOSPP in the literature. Gandibleux et al. [5] discuss the algorithms for solving MOSPP with sum-type objectives under the categories of labeling-based and ranking path-based algorithms. Raith and Ehrgott [6] make a comparative study on the computational efficiency of some of the methods on two-objective problems. Gabrel and Vanderpooten [7] develop an approach for a MOSPP, scheduling an Earth observing satellite. Considering three objectives, they first generate the whole efficient frontier with a label setting algorithm. Then they try to identify the preferred portion of the efficient frontier by a Tchebycheff-like function. Granat and Guerriero [8] develop an interactive algorithm for the MOSPP based on the reference-point methodology.

The TSP is NP-hard for both single and multi-objective cases (see [4], p. 211). The number of efficient tours can grow exponentially with the problem size for the MOTSP problem and finding them all is intractable. The literature on MOTSP is limited. In [9] and [10], TSP with profits is considered, where all nodes need not be visited. The problem is to decide which nodes will be visited and in which order. Here, one objective is to minimize the distance traveled and the other objective is to maximize the benefit from the visited nodes. Karademir [9] develops a genetic algorithm to solve this problem. Berube et al. [10] try to generate the efficient frontier of this problem with the ε-constraint method and they propose improvement heuristics. Özpeynirci and Köksalan [11, 12], address special cases of MOTSP.

The rest of the paper unfolds as follows. We formally define the problem in Sect. 2. We develop an algorithm in Sect. 3 and demonstrate it on an example problem in Sect. 4. We present our conclusions in Sect. 5.

2 Problem Definition

Before we explain the problem in detail, we give some definitions.

Let $G = (V, E)$ be a graph having $|V| = N$ nodes and $|E| = M$ edges, where $E \subset \{(n_1, n_2) : n_1, n_2 \in N\}$. A path between nodes n_s and n_t is a sequence of edges $\{(n_s, n_1), (n_1, n_2), \ldots, (n_{t-1}, n_t)\}$ and a tour (π) is a cyclic permutation of the set N. Under the presence of multiple targets, the route planning of UAVs becomes a generalized MOTSP that aims to find a tour that gives the best combination of the p objectives, $z_1(\pi), z_2(\pi), \ldots, z_p(\pi)$, for the DM.

Let x denote the decision variable vector, X denote the feasible space, Z denote the corresponding feasible objective function space, and let $z(x) = (z_1(x), z_2(x), \ldots, z_p(x))$ be the objective function vector. We assume, without loss of generality, that all objectives are to be minimized.

Definition 2.1 A solution $x \in X$ is said to be *efficient* iff there does not exist $x' \in X$ such that $z_j(x') \leq z_j(x)$ $j = 1, \ldots, p$ and $z_j(x') < z_j(x)$ for at least one j. If there exists such an x', x is said to be *inefficient*. All efficient solutions constitute the efficient set.

Definition 2.2 If x is efficient, then $z(x)$ is said to be *nondominated*, and if x is inefficient, $z(x)$ is said to be *dominated*.

Definition 2.3 An efficient solution x is a *supported efficient solution* iff there exists a positive linear combination of objectives that is minimized by x. Otherwise, x is an *unsupported efficient solution*.

Definition 2.4 An *extreme efficient solution* is a supported efficient solution that has the minimum possible value in at least one of the objectives.

Definition 2.5 Two solutions are *adjacent efficient solutions* iff none of their convex combinations is dominated by any convex combination of other solutions in the objective function space. That is, x_j is adjacent efficient to x_i iff there does not exist $x_t \in X \ni \sum_{t \neq j} \mu_t z(x_t) \leq \lambda z(x_j) + (1-\lambda) z(x_i)$ where $\sum_{t \neq j} \mu_t = 1$, $0 \leq \mu_t \leq 1$ and $0 \leq \lambda \leq 1$, and $z(x)$ is the objective function vector. In a bicriteria problem, there are at most two distinct adjacent efficient solutions of a solution (see [13]).

A path is a solution of the shortest path problem and a tour is a solution of TSP. Therefore, the above definitions directly apply to paths and tours.

The single-target problem is a special case of the multiple-target problem. In this paper, we consider route planning with multiple targets. We represent the terrain by equidistant grids. Each grid point (node) can be represented by the coordinates (x, y) corresponding to the latitude and longitude, respectively. For simplification, we introduce our terrain structure on a two dimensional plane (as in [2]) as shown in Fig. 1. In this structure, the UAV can only move to its adjacent grid points.

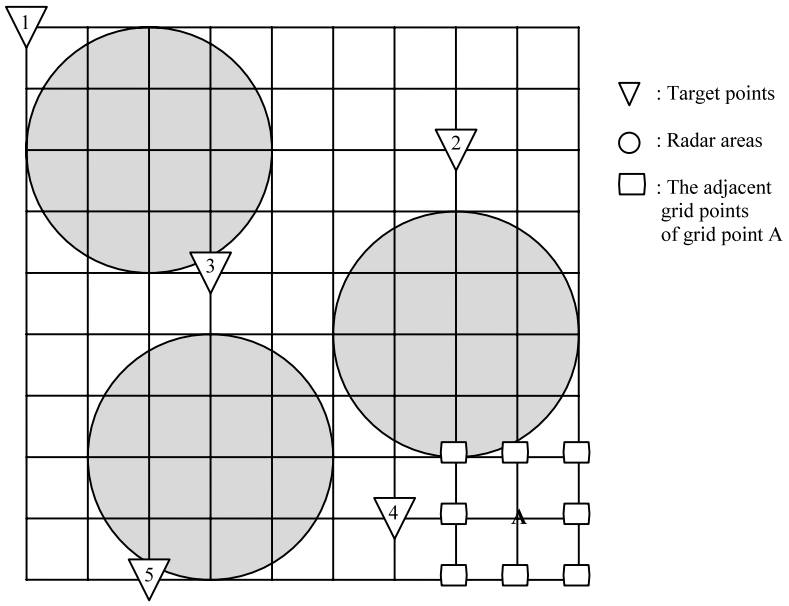


Fig. 1 Representation of the terrain of the UAV route selection problem

This problem is a generalized MOTSP with multiple efficient paths between nodes. If the aim is to find all efficient solutions, there are two problems to be considered. Firstly, a MOSPP is defined between each target pair and all efficient paths of each MOSPP are found. Secondly, the efficient tours of the MOTSP that are composed of these efficient paths should be found. For both parts, we consider two objectives: minimization of the total distance traveled and minimization of the radar detection threat.

Let the distance from node i to node j, c_{ij}, and the radar detection probability at any point between nodes i and j, $(p_d)_{ij}$ be the parameters of the problem. Let x_{ij} be the binary decision variable indicating whether or not link (i, j) is used and let N denote the node set.

Then we have the following two objectives to be minimized:

$$z_1 = \sum_{i \in N} \sum_{j \in N} c_{ij} \cdot x_{ij}, \tag{1}$$

$$z_2 = \sum_{i \in N} \sum_{j \in N} c_{ij} \cdot (p_d)_{ij} \cdot x_{ij}. \tag{2}$$

The total distance is found by summing all the edge lengths that are traversed. For the estimation of the total radar detection threat, we use the radar exposure measure presented in [1]. The details of this measure are given in the Appendix. We assume that the UAV moves with constant speed. With constant speed, the distance traveled can be used to represent the duration of the flight. Multiplying the detection probability of an edge with its distance is approximately equivalent to computing the duration that a UAV is exposed to that detection probability. Minimizing this objective is equivalent to minimizing the duration of total radar detection threat.

For this general structure of the MOTSP, we present some properties.

Theorem 2.1 *An efficient tour does not contain any inefficient paths.*

Proof Consider a MOTSP with node set $N = \{1, 2, \ldots, n\}$. Let $x'_{kl} \in X$ be an inefficient path between nodes k and l. Then there must exist at least one $x''_{kl} \in X$ such that the following inequalities hold.

$$z_j(x''_{kl}) \leq z_j(x'_{kl}) \quad j = 1, \ldots, p, \tag{3}$$

$$z_j(x''_{kl}) < z_j(x'_{kl}) \quad \text{for at least one } j. \tag{4}$$

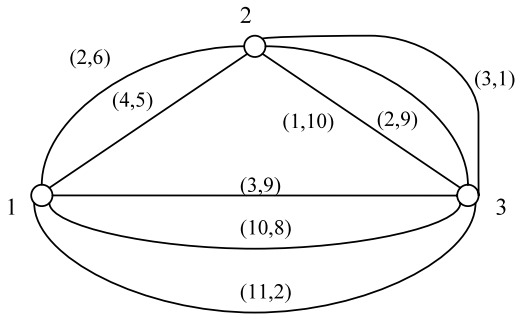
Let the set A_τ contain the node pairs (a, b) such that the path, x'_{ab}, between nodes a and b is included in tour T. The performance of tour T in objective j is:

$$Z_j(T) = \sum_{(a,b) \in A_\tau - (k,l)} z_j(x'_{ab}) + z_j(x'_{kl}) \quad \text{for } j = 1, \ldots, p.$$

If we replace x'_{kl} in tour T with x''_{kl}, we obtain tour T' having performance in objective j as:

$$Z_j(T') = \sum_{(a,b) \in A_\tau - (k,l)} z_j(x'_{ab}) + z_j(x''_{kl}) \quad \text{for } j = 1, \ldots, p.$$

Fig. 2 Example MOTSP



Based on (3) and (4), we have

$$Z_j(T') \leq Z_j(T) \quad j = 1, \ldots, p,$$
$$Z_j(T') < Z_j(T) \quad \text{for at least one } j.$$

Therefore, T cannot be an efficient tour. □

Remark All combinations of efficient paths do not necessarily yield efficient tours and there may be efficient paths that are not parts of any efficient tours.

Example 2.1 Consider the bicriteria TSP with three nodes shown in Fig. 2, where each objective is to be minimized. We would like to identify the efficient tours. Each edge corresponds to an efficient path, where the first and second values in parenthesis correspond to the value of the path in the first and second objective, respectively. Since the TSP is symmetric, there is only a single tour, 1–2–3–1, for a three-node problem. However, there are multiple efficient paths between each node pair as shown in the figure. Permuting these paths, we can form tour 1–2–3–1 in 18 ($= 2 \times 3 \times 3$) different ways, yielding the values of the two objectives in the resulting tours as follows: (6, 25), (7, 24), (8, 16), (8, 24), (9, 23), (10, 15), (13, 24), (14, 18), (14, 23), (15, 15), (15, 17), (15, 23), (16, 9), (16, 17), (16, 22), (17, 14), (17, 16), and (18, 8). Only six of these objective vectors, (6, 25), (7, 24), (8, 16), (10, 15), (16, 9), and (18, 8), correspond to efficient tours. The remaining tours are inefficient even though they are combinations of some efficient paths. Furthermore, the efficient path having the objective values (10, 8) does not appear in any of the efficient tours.

Theorem 2.2 *A dominated subtour cannot be part of an efficient tour even if it is made up of efficient paths.*

The proof is similar to that of Theorem 2.1, and we omit it.

3 An Interactive Approach

Generating the whole efficient frontier of a multi-objective combinatorial optimization problem would be neither practical nor meaningful, especially as the problem

size increases. A meaningful approach is to generate solutions from the preferred part of the efficient set under the guidance of the DM. In this section, we introduce such an interactive approach, where we incorporate the preferences of the DM in finding the best solution assuming an underlying linear utility function.

3.1 The Solution Approach

With the guidance of a DM, our aim is to find the most preferred solution by effectively utilizing the available preference information.

In our solution approach, we assume that the DM has a linear utility function. That is, the DM has a utility function of the following form:

$$U(z) = w_1 z_1(x) + \cdots + w_p z_p(x),$$

where p is the number of objectives, w_i is the weight of ith objective and $z_i(x)$ is the performance of x in the ith objective. We consider two objectives and, without loss of generality, we assume that U is to be minimized and the weights are normalized. The following is the utility function to be minimized:

$$U(z) = wz_1(x) + (1 - w)z_2(x), \quad \text{where } 0 \leq w \leq 1.$$

We utilize two properties of a linear utility function in our solution approach. Firstly, it is well known that the most preferred solution of a DM is a supported efficient solution. Secondly, a supported efficient solution, that is preferred to all its adjacent solutions, is the most preferred solution [14]. In an interactive approach, we would like to keep the information requirement from the DM at a reasonable level. Therefore, it is important to ask questions that will help converge towards the preferred solutions quickly.

In our algorithm, we ask the DM to compare an efficient solution with its adjacent efficient solutions. We utilize the responses of the DM to restrict the possible weights of his/her underlying utility function. This reduces the candidates that are eligible to be the best solution. Our approach adapts the approach of Zionts [14] developed for a set of readily available discrete alternatives to the case of multi-objective combinatorial optimization problems.

3.2 The Interactive Algorithm

We develop an interactive algorithm, *BestSol*, that finds the most preferred solution of a DM for a bicriteria problem. Let ε be a small positive constant, $w_L = 1 - \varepsilon$, and $w_R = \varepsilon$.

Step 1. Set $k = 0$. Find the extreme efficient solutions of the problem using w_L and w_R. Let the solutions be x_L and x_R, respectively. If $x_L = x_R$, the problem has only one solution in the preferred region, namely $x^* = x_L = x_R$. Go to Step 8. Otherwise, go to Step 2.

Step 2. Let

$$w' = \frac{z_2(x_L) - z_2(x_R)}{(z_2(x_L) - z_2(x_R)) + (z_1(x_R) - z_1(x_L))}.$$

Find the solution that minimizes the utility function formed with weight w'.

Min $U'(x) = w'z_1(x) + (1 - w')z_2(x)$.

Let the solution be x'. If $x' = x_L$ or $x' = x_R$, go to Step 3. Otherwise, go to Step 4.

Step 3. Ask the DM x_L versus x_R.

- If x_L is preferred to x_R, update w_R.

$$w_R = \frac{z_2(x_L) - z_2(x_R)}{(z_2(x_L) - z_2(x_R)) + (z_1(x_R) - z_1(x_L))} + \varepsilon.$$

$x^* = x_L$. Go to Step 8.

- If x_R is preferred to x_L, update w_L.

$$w_L = \frac{z_2(x_L) - z_2(x_R)}{(z_2(x_L) - z_2(x_R)) + (z_1(x_R) - z_1(x_L))} - \varepsilon,$$

$x^* = x_R$. Go to Step 8.

Step 4. Find the left adjacent solution of x' (by the method presented in Sect. 3.4). Let the solution be x_L. Go to Step 5.

Step 5. Ask the DM x' versus x_L.

- If x_L is preferred to x', update w_R. Set

$$w_R = \frac{z_2(x_L) - z_2(x')}{(z_2(x_L) - z_2(x')) + (z_1(x') - z_1(x_L))} + \varepsilon,$$

$x' = x_L$. Set $k = 1$.

Go to Step 4.

- If x' is preferred to x_L, update w_L. Set

$$w_L = \frac{z_2(x_L) - z_2(x')}{(z_2(x_L) - z_2(x')) + (z_1(x') - z_1(x_L))} - \varepsilon.$$

If $k = 0$; go to Step 6.

If $k \neq 0$; $x^* = x'$. Go to Step 8.

Step 6. Find the right adjacent solution of x' (by the method presented in Sect. 3.4). Let the solution be x_R. Go to Step 7.

Step 7. Ask the DM x' versus x_R.

- If x' is preferred to x_R, update w_R. Set

$$w_R = \frac{z_2(x') - z_2(x_R)}{(z_2(x') - z_2(x_R)) + (z_1(x_R) - z_1(x'))} + \varepsilon,$$

$x^* = x'$. Go to Step 8.

- If x_R is preferred to x', update w_L. Set

$$w_L = \frac{z_2(x') - z_2(x_R)}{(z_2(x') - z_2(x_R)) + (z_1(x_R) - z_1(x'))} - \varepsilon,$$

$x' = x_R$. Go to Step 6.

Step 8. The most preferred solution is x^*. The utility function of the DM is

$$U(z) = wz_1(x) + (1-w)z_2(x), \quad \text{where } w_R \leq w \leq w_L.$$

The algorithm first finds the two extreme efficient solutions. If we cannot find two distinct extreme efficient solutions, we conclude that the problem has only one efficient solution in Step 1. Then we go to Step 8 and the algorithm terminates. However, if we can find two distinct extreme efficient solutions, we go to Step 2. Here, we search for a solution x' that minimizes the function passing through the two extreme efficient solutions. If we do not find a different solution than the two extreme efficient solutions, we conclude that the problem has only two supported efficient solutions and we go to Step 3. In Step 3, we ask the DM to choose between the two extreme efficient solutions. Based on the preference of the DM, we update the lower or upper bound on the favorable weight set and go to the last step. If we can find another solution in Step 2, we go to Step 4 where we search for its left adjacent solution. In Step 5, the DM compares the solution x' and its left adjacent solution, x_L. If x_L is preferred to x', we set $k = 1$. This indicates that we will only search the portion of the efficient frontier that is on the left side of x_L. We continue finding the left adjacent solution of the most preferred solution of the DM. Whenever a solution in this region is preferred to its left adjacent, we terminate the algorithm. This is because of the fact that when we set $k = 1$, we indicate that the best solution found up to that point of the algorithm is preferred to its right adjacent solution. If this solution is also preferred to its left adjacent, we say that this is the most preferred solution of the DM.

If a solution is preferred to its left adjacent solution the first time Step 5 is executed, we go to Step 6 where we find the right adjacent of the preferred solution. If the DM prefers the solution to the right adjacent x_R as well, we conclude that the current solution is the best solution and we terminate the algorithm. However, if x_R is preferred, we continue finding right adjacent solutions that lie on the right side of x_R. Whenever a solution is preferred to its right adjacent solution, we conclude that it is the best solution and we terminate the algorithm.

During the algorithm, if we find more than one efficient solution, i.e., if we can pass from Step 1 to Step 2, we guarantee that the lower bound or the upper bound or both bounds of w will be updated.

Figure 3 illustrates how the interactive algorithm (*BestSol*) proceeds. In Fig. 3(a), we show the efficient frontier of a hypothetical discrete bicriteria problem as well as the DM's underlying utility function. It can be seen that solution E minimizes the utility function.

We run *BestSol* to find the most preferred solution of the DM iteratively. We first find the extreme efficient solutions; the left extreme solution A with weight w_L and right extreme solution G with weight w_R as shown in Fig. 3(b). Since these two solutions are different, we find the solution that minimizes the function passing through the extreme efficient solutions. The resulting solution is D as shown in Fig. 3(c). We next find the left adjacent solution of D which is solution C. We ask the DM to compare D and C and the DM prefers D to C. We update the upper bound on w, w_L, as shown in Fig. 3(d) and search for the right adjacent of D. The right adjacent of D is solution E. We ask the DM to choose between D and E.

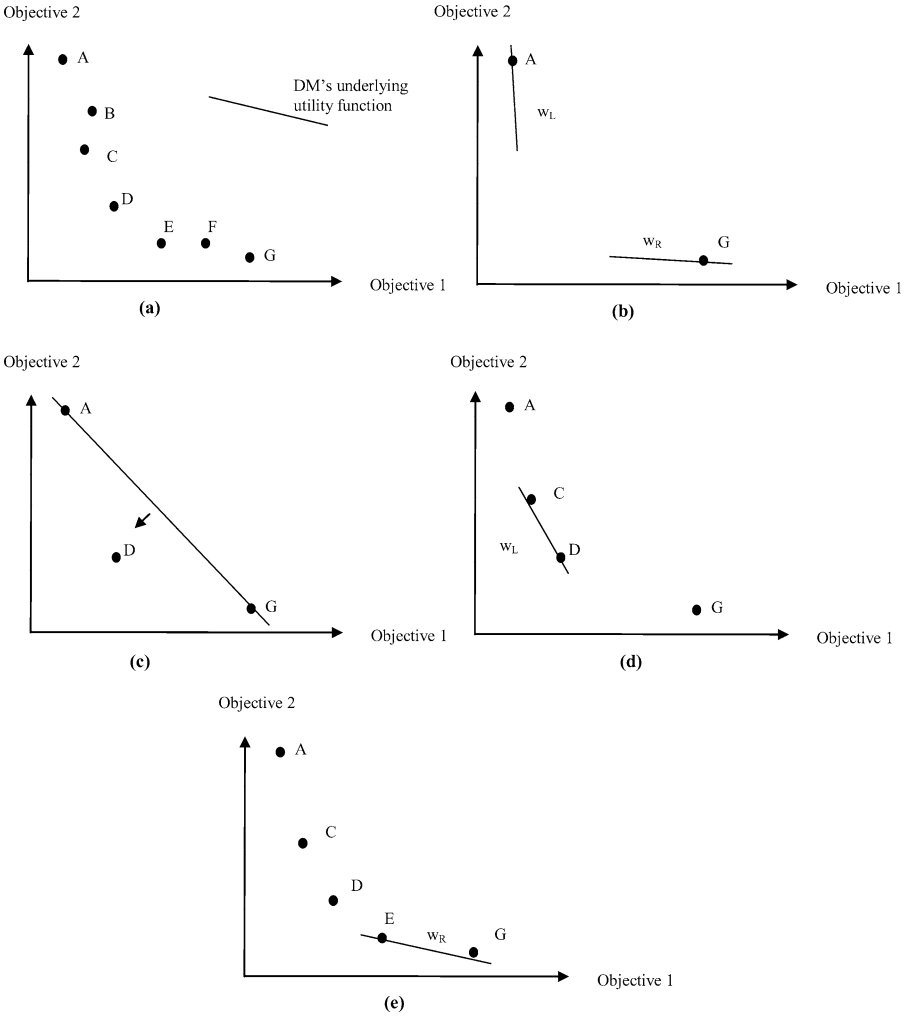


Fig. 3 The interactive algorithm

Since E gives a lower utility value, the DM prefers E. We update the upper bound on w and continue with finding the right adjacent of E. Solution G is the right adjacent of E. The DM now prefers E to G. We update the lower bound on w, w_R, as in Fig. 3(e) and conclude that the best solution is E. Now we have a narrower interval that contains the true weight of the DM's utility function: $w_R \leq w \leq w_L$.

3.3 Finding Supported Efficient Solutions

This method generates all supported efficient solutions of a discrete bicriteria problem. It has been developed by Aneja and Nair [15] and Cohon ([16], p. 127–133) independently. In Fig. 4, we illustrate how the algorithm proceeds.

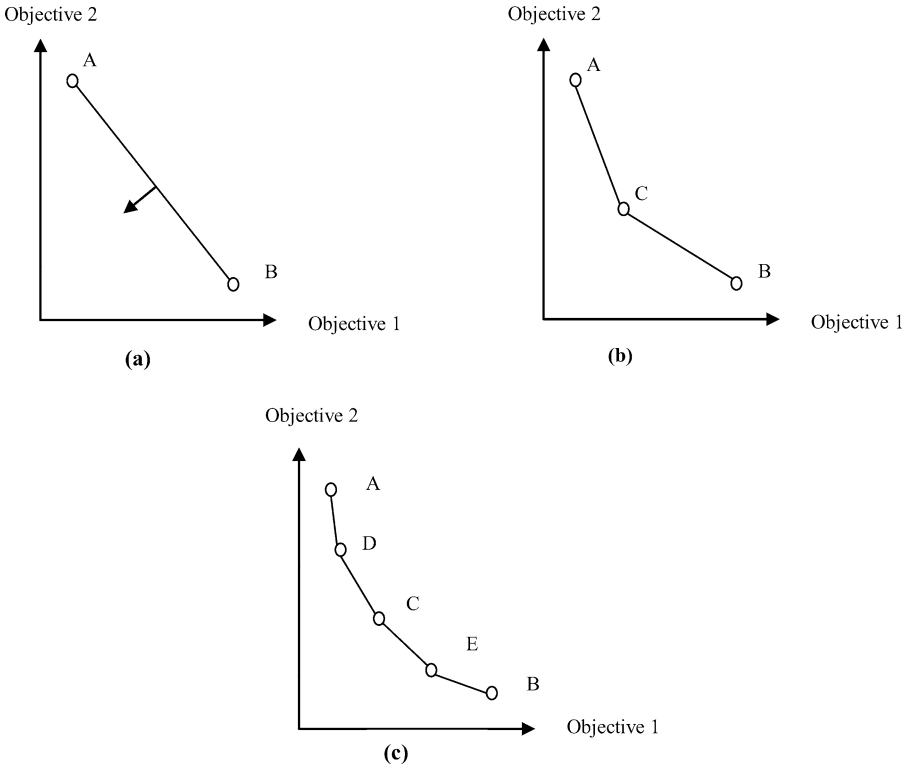


Fig. 4 Finding the supported efficient solutions

First, the extreme efficient solutions are found by minimizing the composite objectives formed by setting w to $1-\varepsilon$ and ε for the left extreme and the right extreme solutions, respectively, where ε is a small positive constant. In Fig. 4(a), A and B are the extreme efficient solutions found with weights $1-\varepsilon$ and ε, respectively. In the absence of any other solutions, A and B are adjacent solutions. Then, the linear function that passes through these solutions is found and this function is minimized. In Fig. 4(b), we obtain solution C and we update our current adjacent solution pairs. (A, C) and (B, C) are adjacent solution pairs in Fig. 4(b). The supported solutions between these adjacent solutions are found by minimizing the linear function that passes through these solutions. The same procedure is repeated for each adjacent point pair until we do not obtain any new solution. Finally, we find all the supported efficient solutions of the problem as shown in Fig. 4(c).

3.4 Finding the Adjacent Efficient Solutions

The following is an algorithm to find the left and right adjacent solutions of a supported efficient solution, x, in a bicriteria problem. We first find the left and right extreme efficient solutions, x_L and x_R, by respectively minimizing $U(x) = (1-\varepsilon)z_1(x) + \varepsilon z_2(x)$ and $U(x) = \varepsilon z_1(x) + (1-\varepsilon)z_2(x)$, where ε is a small positive constant used to guarantee that the resulting solution is efficient.

Finding the Left Adjacent Solution Step 0. Let x be the supported efficient solution whose left adjacent solution is to be found.

Step 1. Let

$$w' = \frac{z_2(x_L) - z_2(x)}{(z_2(x_L) - z_2(x)) + (z_1(x) - z_1(x_L))}.$$

Find the solution minimizing $U'(x) = w'z_1(x) + (1 - w')z_2(x)$ and denote it as x'. If $x' = x_L$ or $x' = x$, go to Step 2. Otherwise, let $x_L = x'$ and go to Step 1.

Step 2. The left adjacent solution of x is x_L.

In Step 1, we minimize the linear objective function that has the slope of the line passing through x and its current left adjacent x_L. If we find a new supported efficient solution between these two solutions, we update the current left adjacent of x with the new solution and go back to Step 1. If we cannot find another supported efficient solution between x and its current left adjacent, we conclude that the current left adjacent solution is the true left adjacent solution of x.

The right adjacent of any solution can be found similarly. The following is the algorithm to find the right adjacent solution of a solution for a bicriteria problem.

Finding the Right Adjacent Solution Step 0. Select a supported efficient solution x whose right adjacent solution is to be found.

Step 1. Let

$$w' = \frac{z_2(x) - z_2(x_R)}{(z_2(x) - z_2(x_R)) + (z_1(x_R) - z_1(x))}.$$

Find the solution minimizing $U'(x) = w'z_1(x) + (1 - w')z_2(x)$ and denote it as x'. If $x' = x_R$ or $x' = x$, go to Step 2. Otherwise, let $x_R = x'$ and go to Step 1.

Step 2. The right adjacent solution of x is x_R.

3.5 MOTSP with Multiple Efficient Paths between Nodes

We utilize the *BestSol* algorithm for the route selection of UAVs. In our solution approach, we first find the most preferred path between each target pair. In the second part, we find the most preferred tour that is made up of a subset of these most preferred paths.

3.5.1 Interactive Algorithm—Part I

In this part, we find the most preferred path between each target pair. We consider M targets. In each run of *BestSol*, we not only find the most preferred path between the selected target pair, but we also update the upper and lower bounds on w. In the beginning of the algorithm, w ranges from ε to $1 - \varepsilon$. As the algorithm progresses, if we have more than one supported efficient path for any target pair (i, j) that minimizes a linear utility function for some w in the range $[w_R, w_L]$, we ask the DM to compare some of these solutions. Based on the responses of the DM, we either update w_L or w_R or both. Then, for the next target pair, we find the extreme points using these updated weights. The responses of the DM help in generating a smaller portion of the set of supported efficient solutions of the succeeding paths to be evaluated.

The following are the steps of the algorithm we use for finding the best path between each target pair.

Step 0. Let $w_L = 1 - \varepsilon$ and $w_R = \varepsilon$.

Step 1. Select a target pair (i, j) that has not been evaluated.

Step 2. Find the most preferred path between target pair (i, j) using BestSol. Update the lower (w_R) and upper bounds (w_L) of w.

Step 3. If there are target pairs not evaluated, go to Step 1. Otherwise, terminate the algorithm. All the most preferred paths between the target pairs are found.

In Step 2, we solve the shortest path problem for finding the extreme efficient solutions, the supported efficient solutions and their adjacent solutions. Since we take a linear combination of the objectives, we solve a single-objective shortest path problem. We use Dijkstra's exact algorithm ([17], p. 416).

3.5.2 Interactive Algorithm—Part II

In Part I of our algorithm, we identify the best path between each target pair and a reduced range of w values that reflects the preferences of the DM. The resulting problem becomes a regular MOTSP with a single connection between each target pair. Having this information at hand, we next run *BestSol* to find the most preferred tour of the DM.

We run *BestSol* using the updated range of w; $[w_R, w_L]$. In Steps 1, 2, 4, and 6, we use a single objective TSP solver, CONCORDE (see http://www.tsp.gatech.edu/concorde), to find the optimal visiting order of targets for a given w. Using w, we set the weighted cost of using edge x_{ij} to

$$e_{ij} = \begin{cases} 0 & \text{if } i = j \\ wz_1(x_{ij}) + (1 - w)z_2(x_{ij}) & \text{if } i \neq j \end{cases}.$$

BestSol terminates after finding the best tour of the DM.

4 An Example

We demonstrate our approach on a problem where the terrain is represented by grids of size 10×10 and there are five targets. We located the targets and the radars suitably at different parts for illustration purposes. We denote a node by its (x, y) coordinates where x and y show horizontal and vertical locations of the corner points of the grids, respectively. We start numbering from the northwest corner point assigning its coordinates the values of $(1, 1)$. Targets 1 to 5 are located at $(1, 1)$, $(3, 8)$, $(5, 4)$, $(9, 7)$, and $(10, 3)$, respectively. The radars are located at the centers of the circles at coordinates $(3, 3)$, $(6, 8)$, and $(8, 4)$. The terrain structure, the threat areas of the radars, and the five targets of this problem are shown in Fig. 1.

We give the details of the execution of the algorithm in Table 1 using an underlying weight representing the true preferences of the DM, $w_{\text{best}} = 0.5$. We use this information to simulate the DM whenever we need preference information. The first column in Table 1 shows the target pair (i, j) that is evaluated. The second column

Table 1 Computational results for the example problem ($w_{\text{best}} = 0.5$)

Target pair (i, j)	Number of efficient paths	Number of comparisons	Weight range
$(1, 2)$	1	–	$[0.0001, 0.9999]$
$(1, 3)$	7	2	$[0.2004, 0.5160]$
$(1, 4)$	7	2	$[0.4620, 0.5160]$
$(1, 5)$	2	–	$[0.4620, 0.5160]$
$(2, 3)$	1	–	$[0.4620, 0.5160]$
$(2, 4)$	6	–	$[0.4620, 0.5160]$
$(2, 5)$	10	–	$[0.4620, 0.5160]$
$(3, 4)$	2	–	$[0.4620, 0.5160]$
$(3, 5)$	6	–	$[0.4620, 0.5160]$
$(4, 5)$	1	–	$[0.4620, 0.5160]$

shows the actual number of efficient paths between target pair (i, j). The third column shows the number of comparisons the DM has to make between efficient paths in order to find the best path connecting target pair (i, j). Notice that we only need to identify Number of Comparisons $+ 1$ of the total efficient paths between target pair (i, j) to find the best path between those targets. The last column shows the reduced weight space as a result of the comparisons made so far. In the early stages of the algorithm, different efficient paths may be best for different weights within the reduced weight range for the considered target pairs. However, after the evaluation of target pair $(1, 4)$, the weight space is reduced to a sufficiently narrow range around $w_{\text{best}} = 0.5$ that a unique efficient path turns out to be the best for each of the succeeding target pairs. Therefore, the DM is not asked for any preference information for those target pairs and all of the best paths are obtained by a total of four pairwise comparisons made by the DM in this example problem.

In the second part of the problem, the reduced weight range is again sufficiently narrow that there is a unique tour, 1–2–3–4–5, that minimizes U for any $w \in [0.4620, 0.5160]$. The total distance of the corresponding tour, and the total radar detection threat are 32.140 and 5.306, respectively.

There were totally four questions asked to the DM. The weight space was quickly reduced to a narrow range around 0.5 and no additional questions were needed for determining most of the best paths and the best tour during the rest of the algorithm.

5 Conclusions

In this paper, we develop an interactive algorithm for the case where the DM's preferences can be represented by a linear function of the two objectives. As a part of this algorithm, we also develop an algorithm for finding the adjacent efficient solutions of a supported efficient solution.

It is important for an interactive approach to reach the most preferred solution of a DM without requiring excessive preference information from the DM. In order to accomplish this, we use two properties of a linear utility function: the most preferred

solution of a DM is a supported efficient solution and a solution that is preferred to its adjacent efficient solutions is the most preferred solution. Hence we generate only the supported efficient solutions and ask for comparison only between adjacent efficient solutions.

In this paper, we define a generalized MOTSP, and apply it to the problem of route planning for UAVs. The generalized MOTSP accounts for multiple efficient paths between pairs of nodes and the dependence of the tours on these paths. The problem can be modeled as a combination of two combinatorial problems: MOSPP and MOTSP. In MOSPP, we need to determine the best path between each target pair. After determining the best path between each target pair, we need to determine the best tour. The preference information used in early stages help reduce the weight range quickly, and the best solution can be found after a small amount of preference information is obtained. This also reduces the computational effort by focusing on a smaller portion of the efficient set.

An area for future research is to consider more general utility functions than a linear function. Another future research area is to consider a dynamic version of the problem where the targets and/or radars may change their locations. In this case, a real time implementation of the route selection problem is necessary. The problem becomes more complex and the paths and routes need to be modified dynamically throughout the movement of the UAV.

Appendix: Computation of the Radar Exposure Measure

In this measure, we first calculate the signal-to-noise ratios, S/N, at grid points using the monostatic radar formula (5). To calculate p_d of an edge, we take the arithmetic average of the S/N ratios of the two grid points that are the end points of that edge. We then calculate the p_d value using (6). We modify the upper and lower bounds of [1] in (6) in order to obtain values that differentiate the grid points and demonstrate the problem well. After finding the p_d value, we multiply it with the length of the edge to come up with a threat measure that accounts for the length of the exposure:

$$S/N = \frac{P_t G_t^2 \lambda^2 \sigma}{(4\pi)^3 K T_s B_n L_t^2 R^4}, \tag{5}$$

where

P_t is the power transmitted by radar (watts),
G_t is the power gain of transmitting antenna,
L_t is the transmitting system loss,
σ is the aircraft radar cross-section (RCS) (square meters),
R_t is the distance from the transmitter to the aircraft (meters),
λ is the wave length of signal frequency (meters),
K is Boltzman's constant (joules/kelvin),
T_s is the receiving system noise temperature (kelvin),

B_n is the noise bandwidth of the receiver (hertz),

$$p_d = \begin{cases} 1 & \text{if } S/N > 30, \\ ((S/N) - 15)/15 & \text{if } 15 < S/N \leq 30, \\ 0 & \text{if } S/N \leq 15. \end{cases} \tag{6}$$

References

1. Gudaitis, M.S.: Multi criteria mission route planning using a parallel A* search. M.S. Thesis, Air Force Institute of Technology (1994)
2. Olsan, J.B.: Genetic algorithms applied to a mission routing problem. M.S. Thesis, Air Force Institute of Technology (1993)
3. Yavuz, K.: Mission route planning using particle swarm optimization. M.S. Thesis, Air Force Institute of Technology (2002)
4. Ehrgott, M.: Multi Objective Optimization. Springer, Berlin (2000)
5. Gandibleux, X., Beugnies, F., Randriamasy, S.: Martins' Algorithm revisited for multi-objective shortest path problems with a MaxMin cost function. 4OR **4**(1), 47–59 (2006)
6. Raith, A., Ehrgott, M.: A comparison of solution strategies for biobjective shortest path problems. Comput. Oper. Res. **36**(4), 1299–1331 (2009)
7. Gabrel, V., Vanderpooten, D.: Enumeration and interactive selection of efficient paths in a multiple criteria graph for scheduling an Earth observing satellite. Eur. J. Oper. Res. **139**, 533–542 (2002)
8. Granat, J., Guerriero, F.: The interactive analysis of the multicriteria shortest path problem by the reference point method. Eur. J. Oper. Res. **151**, 103–118 (2003)
9. Karademir, S.: A genetic algorithm for the biobjective traveling salesman problem with profits. M.S. Thesis, Department of Industrial Engineering, Middle East Technical University (2008)
10. Berube, J.F., Gendreau, M., Potvin, J.Y.: An exact ε-constraint method for bi-objective combinatorial optimization problems–application to the traveling salesman problem with profits. Eur. J. Oper. Res. **194**(1), 39–50 (2009)
11. Özpeynirci, Ö., Köksalan, M.: Multiobjective traveling salesperson problem on Halin graphs. Eur. J. Oper. Res. **196**, 155–161 (2009)
12. Özpeynirci, Ö., Köksalan, M.: Pyramidal tours and multiple objectives. J. Glob. Optim. **48**(4), 569–582 (2010)
13. Ramesh, R., Karwan, M.H., Zionts, S.: An interactive method for bicriteria integer programming. IEEE Trans. Syst. Man Cybern. **20**, 395–403 (1990)
14. Zionts, S.: A multiple criteria method for choosing among discrete alternatives. Eur. J. Oper. Res. **7**(1), 143–147 (1981)
15. Aneja, Y.P., Nair, K.P.K.: Bicriteria transportation problem. Manag. Sci. **25**(1), 73–78 (1979)
16. Cohon, J.L.: Multiobjective programming and planning. Academic Press, New York (1978)
17. Winston, W.L.: Operations research—applications and algorithms, 4th edn., Brooks/Cole–Thomson, Belmont (2004)