MULTI-OBJECTIVE ROUTE SELECTION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DİCLEHAN TEZCANER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

JULY 2009

Approval of the thesis:

MULTI-OBJECTIVE ROUTE SELECTION

submitted by **DİCLEHAN TEZCANER** in partial fulfillment of the requirement for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Nur Evin Özdemirel
Head of Department, **Industrial Engineering** _____

Prof. Dr. Murat Köksalan
Supervisor, **Industrial Engineering Dept., METU** _____

Examining Committee Members

Assist. Prof. Dr. Pelin Bayındır
Industrial Engineering Dept., METU _____

Prof. Dr. Murat Köksalan
Industrial Engineering Dept., METU _____

Dr. Umut Durak
TÜBİTAK - SAGE _____

Prof. Dr. Nesim Erkip
Industrial Engineering Dept., Bilkent University _____

Dr. Orhan Karasakal
OR Division, Turkish Naval Forces Command _____

Date: **21.07.2009**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Diclehan TEZCANER

Signature :

ABSTRACT

MULTI-OBJECTIVE ROUTE SELECTION

TEZCANER, Diclehan

M.S., Department of Industrial Engineering
Supervisor: Prof. Dr. Murat Köksalan

July 2009, 101 pages

In this thesis, 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. For this case, there are multiple combinatorial problems to be considered. First, the paths to be followed between any pairs of targets should be determined. This part can be considered as a multi-objective shortest path problem. Additionally, we need to determine the order of the targets to be visited. This in turn, is a multi-objective traveling salesperson problem. The overall problem is a combination of these two combinatorial problems.

The route selection for UAVs has been studied by several researchers, mainly in the military context. They considered a linear combination of the two objectives; minimizing distance traveled and minimizing radar detection threat; and proposed heuristics for the minimization of the composite single objective problem. We treat these two objectives separately. We develop an evolutionary algorithm to

determine the efficient tours. We also consider an exact interactive approach to identify the best paths and tours of a decision maker. We tested the two solution approaches on both small-sized and large-sized problem instances.

Keywords: Multi-objective Decision Making; Combinatorial Optimization; Evolutionary Algorithms; Interactive Method

ÖZ

ÇOK AMAÇLI ROTA SEÇİMİ

TEZCANER, Diclehan

Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Murat Köksalan

Temmuz 2009, 101 Sayfa

Bu tezde, birden çok amaç gözetilerek, İnsansız Hava Araçları (İHA)'nın rotasının belirlenmesi incelenmiştir. Bu problemin genel bir durumu olan İHA'nın birden çok hedefe uğrayıp başlangıç noktasına döndüğü durum ele alınmıştır. Bu durum için iki farklı kombinatoryal problem incelenmelidir. İlki, İHA'nın hedefler arasındaki gidiş yolunun bulunmasıdır. Bu kısım çok amaçlı en kısa yolu bulma problemi olarak incelenebilir. Bunun yanı sıra, hedeflere hangi sırayla gidileceğine de karar verilmelidir. Bu kısım da çok amaçlı gezgin satıcı problemidir. Genel problem bu iki kombinatoryal problemin birleşimidir.

İHA için rota seçimi, genelde askeri alanda birkaç araştırmacı tarafından çalışılmıştır. Araştırmacılar, toplam katedilen mesafenin ve toplam radara yakalanma olasılığının minimizasyonu olan iki amacın doğrusal kombinasyonunu incelemişlerdir. Bu birleştirilmiş amacın minimizasyonu için sezgisel yöntemler önermişlerdir.

Biz bu çalışmada amaçları ayrı bir şekilde inceledik. Etkin turları bulmak için bir evrimsel algoritma geliştirdik. Bunun yanı sıra, karar vericinin en çok tercih ettiği çözümleri bulmak için bir etkileşimli algoritma geliştirdik. İki çözüm yaklaşımını da hem küçük hem büyük boyutlu problemlerde test ettik.

Anahtar Kelimeler: Çok Amaçlı Karar Verme; Kombinatoryal Optimizasyon; Evrimsel Algoritmalar; Etkileşimli Metod

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my supervisor Prof. Dr. Murat Köksalan. I am grateful to him for his support and motivation throughout my M.S. study and for his guidance in this study and academic life. It was a great experience working with him.

I would like to express my deepest thanks to my parents, Olcay Tezcaner and Tevfik Tezcaner, and my younger brother, Batuhan Tezcaner, for their endless patience, affirmative point of views and motivation. Feeling their support is the main encouragement in my life.

I would like to thank my friends Gülşah Karakaya, Aslı Gül Buğdacı and Burcu Özsayın for sharing all the great moments during my M.S. study. I would also like to thank Alp Oğuz for adding value to my thesis from an electrical engineer perspective and Tuğçe Yüksel, Çiğdem Cihangir and Eda Göksoy for their lovely friendship.

I would like to acknowledge Dr. Umut Durak for introducing us this subject and Prof. Dr. Altunkan Hızal for sharing his valuable knowledge on radars. Also, I would like to thank my examining committee members; Prof. Dr. Nesim Erkip, Assist. Prof. Dr. Pelin Bayındır, Dr. Umut Durak and Dr. Orhan Karasakal for their valuable comments and inputs.

I would like to thank TÜBİTAK for the funding they have provided during my M.S. study.

Lastly, I am grateful to Emrah Ünalan for his unconditional support and guidance in this study and in my life. His presence has always motivated me and will motivate me throughout my life.

To my family…

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

LIST OF TABLES

TABLES

CHAPTER 1

INTRODUCTION

Unmanned Air Vehicles (UAV) are unpiloted aircrafts that are used for both military and civilian purposes. Their missions are mainly espionage and reconnaissance. In the military context, route planning for UAVs; i.e. finding the "best" route that the UAV should follow through a defended area; is very essential. This "best" route can be determined based on several criteria. These can be listed as total distance, fuel consumption, total flight time, detection threat avoidance and navigation performance. In this study, we consider two objectives: minimization of the distance traveled and minimization of the radar detection threat. These two objectives approximately represent most of the remaining criteria.

This problem has been studied by a number of researchers in the literature, mainly in the military context. They considered a linear combination of the two objectives and used heuristics to minimize the composite objective. In our approach, we treat the two objectives separately.

Route planning problem can have two forms; route planning with a single target and route planning with multiple targets. In the presence of only one target, the problem is a Multi-Objective Shortest Path Problem (MOSPP). However, the problem becomes more complicated when more than one target is involved. In the presence of many targets, two problems need to be considered. Firstly, we need to determine the efficient paths between each target pair, which is a MOSPP. Afterwards, we need to determine the order of visit to the targets and the paths to be used between the targets. This part can be considered as a Multi-Objective

Traveling Salesperson Problem (MOTSP). The overall problem is a combination of these two combinatorial problems.

We use both a heuristic and an exact approach for solving different versions of the problem. In the exact interactive approach we find the best paths and the best tour of a Decision Maker (DM). We assume that the DM has a linear utility function. By interacting with the DM, we learn about the possible values of the weights of the underlying utility function and reach the best paths and the best tour of the DM.

In the heuristic approach, we develop an Evolutionary Algorithm (EA); a modification of elitist non-dominated sorting genetic algorithm (NSGA-II). In this algorithm, we generate the potential efficient tours. Knowing the potential efficient tours, we come up with the approximation of the efficient frontier of the original problem.

In Chapter 2, we review the literature related with our study. The problem structure is explained in Chapter 3. In Chapter 4, we present an interactive approach to find the most preferred paths and the most preferred tour of a DM based on a linear utility function. In Chapter 5 we find the potential efficient tours using an EA, a modification of NSGA-II. We present our conclusions in Chapter 6.

CHAPTER 2

LITERATURE SURVEY

Route planning for air vehicles is a combinatorial problem. This problem has been studied by a number of researchers in the literature. In the military, this subject is referred as Mission Route Planning (MRP).

Gudaitis (1994) considers the MRP for the air vehicles from a multi-objective perspective. In his problem, the aircraft starts from an initial point and tries to reach a single target in a three dimensional terrain. In determining the route to be followed, he considers two objectives; minimization of the total distance and minimization of the radar detection threat. He combines these two objectives linearly and tries to minimize the composite single objective. With this transformation, the problem turns to be a Single Objective Shortest Path Problem (SOSPP). In the solution, he uses A* algorithm. In this algorithm, the cost of any route is calculated as the summation of the lowest cost path from starting node to an intermediate node and the estimated cost of the shortest path from that intermediate node to the target. He makes experiments over a 250 square kilometer area. He claims that he has made efficient improvements to A* algorithm and the execution time has greatly improved.

Olsan (1993) uses a genetic algorithm in the solution of an MRP problem with only one target. The two objectives are common with Gudaitis; minimization of distance traveled and minimization of radar detection threat. Olsan combines the two objectives into a single objective and tries to minimize the resulting objective using a genetic algorithm.

In their studies, Olsan (1993) and Gudaitis (1994) restrict the move of the aircraft. They assume that the aircraft can make moves only within 45^0 from the horizontal. For instance, while going to north, it cannot make a direct 90^0 turn to the west. It must make two moves to turn to west; first to the northwest and the other to the west. Both of these moves are 45^0 from the horizontal. This restriction makes the next move of the aircraft dependent on its previous move. Therefore, it is not straightforward to transform the grid structure to be used by a shortest path problem solution approach. In the shortest path problem, the nodes that one can go from a node can be represented by a graph. However, with move restrictions, the successor nodes of one node cannot be determined easily. Gudaitis suggests representing a node with its predecessor node that is used before coming to that node. Knowing the previous and the current node, it is straightforward to determine the possible successor nodes. Gudaitis argues that Dijkstra's algorithm is not computationally efficient in the solution since the number of grids to be evaluated becomes very large after the transformation. Instead he uses heuristics in his solution approach without any transformation.

Yavuz (2002) considers the MRP problem in a different way than the previous two researchers. In his problem, there exist multiple targets that the aircraft should visit. The best order of visit and the paths used between these consecutive targets are determined based on their performance under the same two criteria with the previous theses; the total distance traveled and radar detection threat. The two objectives are combined linearly and this resulting objective function is minimized. Yavuz divides the solution process into two; firstly he determines the best order of visit to targets. To do this, he generates the cost matrix between each target point using "best first search technique". In best first search technique, from the potential next nodes, the node that increases the objective function the least is selected as the next node. After creating the cost matrix, the best order of visit is found using an algorithm called Particle Swarm Optimization and Ant System (PSO_AS). Yavuz claims that this algorithm is a synthesis of Particle Swarm Optimization and Ant System. In the second part of the algorithm, Yavuz finds the paths to be used between each consecutive target using the algorithm

PSO_AS. He makes three sets of experiments on different terrains and he claims that this algorithm is effective for MRP with multiple targets.

Route planning problem is a combinatorial problem. Gudaitis (1994) and Olsan (1993) refer to this problem as shortest path problem. However, since both of the authors combine the two objectives under a single objective, they use single objective algorithms in the solution approach. If the objectives are treated separately, multi-objective solution approaches should be employed.

MOSPP is NP-complete and intractable (Ehrgott, 2000). There are lots of solution approaches for MOSPP in the literature.

Gandibleux et al. (2006) state that there are mainly two classes of algorithms for solving MOSPP with linear functions; labeling based and ranking path based algorithms. Labeling based algorithms may be classified in two headings; label correcting and label setting.

- Label correcting method: In this method, in each iteration one node's successors' values (distance, cost, etc.) are recalculated until there is no label remaining in the label set.

- Label setting method: Before we discuss this method, we give a brief explanation on the Dijkstra's algorithm (Winston, p. 416); an exact algorithm used in the solution of SOSPP. In Dijkstra's algorithm, all the nodes have either permanent or temporary labels that indicate their shortest distance from the initial node. When the label is set permanent for one node, it implies that the shortest distance to that node has been found and it is equal to the label's value. However, if the label of a node is temporary, its shortest distance to the initial node has not been found yet. In each iteration, the node with the minimum label is labeled permanently, and all the remaining temporary labeled nodes' values are recalculated. In each iteration, one node is set permanent. The algorithm continues until

the destination node is set permanent. The value of the destination node gives the shortest path value from the initial node to the destination node. The label setting method is an extension of Dijkstra's algorithm to multi-objective case. In each iteration, one label is set permanent and the remaining labels are recalculated based on this permanent label. Martin's algorithm is an example of this method which deals with objectives that are of sum type; i.e. the values are summed as the path proceeds. Gandibleux et al. (2006) makes an extension to Martin's algorithm to include one max-min function in addition to the sum type objectives.

Ranking path based methods are the following;

- Near shortest path method: This method needs a composite objective function. All the paths that are within a constant deviation from the optimal path length are calculated.

- Two phase method: In the first phase, the extreme supported solutions are found and in the second phase the remaining unsupported and nonextreme supported solutions are found. In the second phase, either label correcting or label setting method is used.

Skriver and Andersen (2000) argue that labeling algorithms outperform path based algorithms because labeling algorithms store nondominated values whereas path based algorithms store the efficient paths whose number is always greater than or equal to the number of nondominated values. Therefore, the memory requirement for labeling algorithms is less than that of path based algorithms. Besides, by storing the values of the objectives in label based algorithms, only the paths which the DM prefers are backtracked. In the article, they improve the label correcting algorithm by modifications in the algorithm's code, and it is argued that this approach is the fastest for solving bicriteria shortest path problems.

Raith and Ehrgott (2009) compare four methods; label setting, label correcting, ranking – near shortest path and the two phase methods for biobjective shortest

path problems by their solution speeds with respect to three networks; grid network, road network and a specialized network called NetMaker network. For the grid networks, label correcting method seems to be the most efficient method, for road networks the two phase method with label setting as the second phase is the best, and for NetMaker networks both the two phase method and the near shortest path method give compatible solutions.

Gabrel and Vanderpooten (2002) propose an interactive approach for a MOSPP; scheduling an earth observing satellite. Considering three objectives (demand satisfaction, priority and satellite use), the whole efficient frontier is generated with a label setting algorithm. Afterwards, by a Tchebycheff-like function the preferred portion of the efficient frontier is found. The DM is given the opportunity to either choose the path that minimizes the weighted deviation from the ideal point or guide the search to the preferred region. If the DM does not prefer the proposed solution, he/she can define desirable values on some or all criteria or denote which objectives should be improved. An inefficiency of this method is the need to develop the whole efficient frontier. Without generating the whole frontier, some of the efficient paths could be used in the interactive method that guides the search direction.

Granat and Guerriero (2003) do not generate the whole efficient frontier in the algorithm they propose. Between the aspiration and the reservation levels, a solution that minimizes the weighted deviations from the aspiration levels of each objective is generated. According to the DM's preferences, the aspiration and reservation levels can be changed to have higher satisfaction. This algorithm continues until the DM is satisfied.

Yavuz (2002) considers the MRP problem as a combination of MOTSP and MOSPP. The Traveling Salesperson Problem (TSP) is defined as finding the shortest tour that passes through all m nodes. The MOTSP is finding the tour that "minimizes" the p objectives; $f_1(\pi), f_2(\pi),..., f_p(\pi)$.

In the literature, for the p-objective case, to our best knowledge, between any two nodes only one path is defined with the objective vector $[c_1, c_2, ..., c_p]$ where c_i is the value in objective i. Then, p cost matrices are made; each matrix containing the values for one of the objectives. The problem turns to be a TSP with p "cost" matrices.

Generally, from the multi-objective perspective, there may exist many efficient paths between two nodes; each better than any other path in at least one objective. This problem can be redefined as finding the efficient tours that are composed of these paths. The "one path between any two nodes" version is a special case of this problem in which it is assumed that there exists only one efficient path between any two nodes. In our solutions, we consider the MOTSP with multiple efficient paths. However, since this problem has not been defined in the literature, there are no studies on this subject to the best of our knowledge.

The general MOTSP is proven to be intractable and the problem is NP-hard for both single objective and multi-objective cases (Ehrgott, 2000).

There are a few studies on the MOTSP. Karademir (2008) and Berube et al. (2009) consider the biobjective TSP, the TSP with profits. In this problem, one objective is the minimization of the distance and the other objective is the maximization of the benefit from visiting nodes.

In his thesis, Karademir (2008) proposes a genetic algorithm for the solution of the biobjective TSP with profits. He proposes some improvements over an EA, NSGA-II. With these improvements, he indicates that he managed to solve large TSP instances in shorter durations than the durations present in the literature.

Berube et al. (2009) try to generate the efficient frontier of a TSP with profits with the ε-constraint method and they propose improvement heuristics for this method.

In his book Ehrgott (2000) suggests solving the MOTSP by single objective TSP algorithms. Firstly, the cost matrices of the different objectives are combined to a single cost matrix using l_1 norm. Then, the single objective TSP is solved using heuristics like Tree Heuristic and Christofides' Heuristic.

Hansen (2000) proposes two scalarizing functions to be used for solving MOTSP. Based on his experiments on MOTSP instances, he argues that using substitute scalarizing functions for local search optimization procedures may produce better results than using the Tchebycheff scalarizing function.

CHAPTER 3

PROBLEM DEFINITION

3.1. Unmanned Air Vehicles (UAVs)

UAVs are air vehicles that were firstly designed for military purposes. Presently, they are used for both civilian and military purposes. In military, the main missions of UAVs are espionage, annihilation and reconnaissance. These are missions that are hard for manned air vehicles. Surveillance against crimes, studies to minimize hazardous effects of natural disasters can be considered as the civilian usage areas of UAVs.

Based on our discussions with experts on UAVs, it is possible to classify these vehicles based on their flight durations. There are four UAV types that can fly different durations. Mini UAVs can fly for 1 hour with a maximum altitude of 5000 ft. Tactical UAVs can fly between 6 and 8 hours with an altitude within 5000 and 15000 ft. The third UAV kind is operative UAV which can fly between 8 and 18 hours within an altitude of 15000 and 30000 ft. The last UAV is strategic UAV which flies above 30000 ft with a duration of at least 24 hours. Generally, UAVs do not fly below 1000-2000 m and the reconnaissance area can be as large as 200 square kms.

Route planning for these air vehicles is essential since they have important roles both in the military and civilian context. This route planning problem for UAVs is a combinatorial problem. It could take one of the following two forms;

- Route planning with a single target in which the aim is to find the "best" path(s) between the starting point and the target.
- Route planning with multiple targets in which the aim is to find the "best" tour(s) that starts from the starting point, passes through all targets and ends at the starting point.

There are many path options between any pair of targets, and the "best" paths are determined by their performance in certain criteria. The possible criteria are total distance, fuel consumption, visual or radar detection avoidance, navigation performance, total flight time, etc. In this study, we consider two criteria; distance traveled and detection avoidance. Distance traveled is a proxy for fuel consumption or total flight time. Detection avoidance represents both radar and visual detection avoidance.

3.2. Representation of the Real World

The following representations are necessary for the route planning problem to represent the real world.

Terrain representation

We consider two dimensional terrain representation for simplicity. However, it is also possible to analyze the problem in three dimensions. This may require substantially increased computations. The two dimensional terrain area is divided into equidistant grids (Olsan, 1993). Each grid point is represented by two dimensions (x,y) corresponding to latitude and longitude respectively. Figure 3.1 illustrates the grid representation. The terrain is divided into 10 equidistant horizontal and vertical grids. The triangles represent the target points. The circles represent the radar threat areas, where the radar is in the center, and radar detection threat decreases from the center to the circumference.

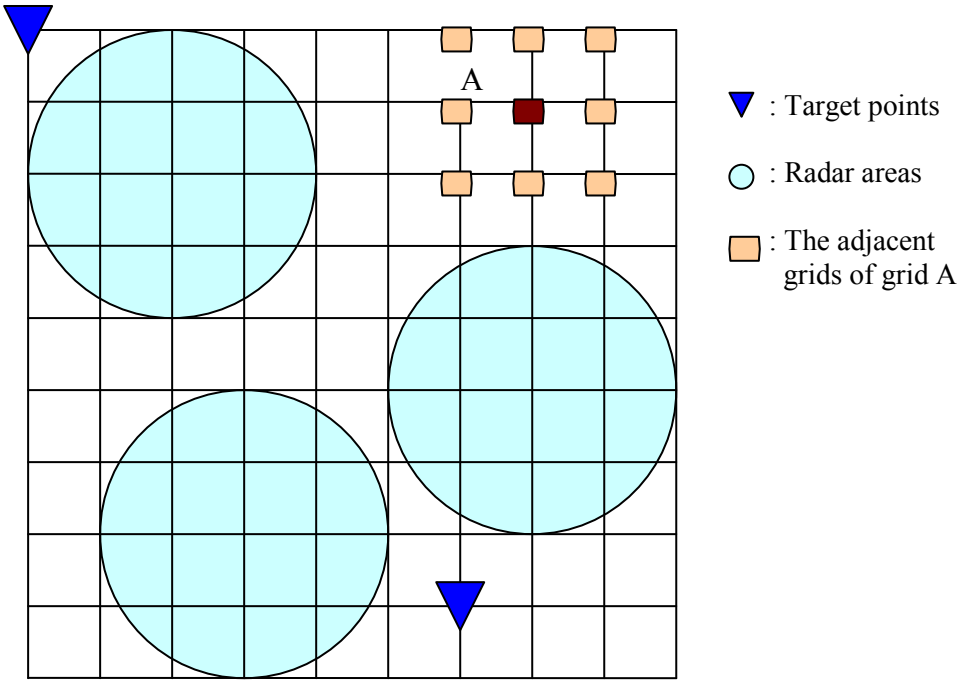


Figure 3.1 Representation of the terrain of the UAV route selection problem

UAV movement representation

Gudaitis (1994) assumes in his research that the aircraft can make at most 45^{o} turns from a horizontal movement. However, it is possible for UAV to move to all its adjacent grids. The adjacent grids of a grid are shown in Figure 3.1.

3.3. Objectives

As stated in Section 3.1, we consider two criteria to model the route planning problem. These are distance traveled and radar detection avoidance.

3.3.1. Distance Traveled

There may be many criteria for the determination of the "best" route that an UAV should follow. The two criteria considered represent most of the remaining

criteria. Distance traveled is a proxy for fuel consumption and total flight duration. In our solution approach, we assume that the UAV moves with constant speed. However, it is possible to consider various speeds. For simplicity, a number of speed levels can be defined and the speed levels during the flight period can be determined.

In a grid structure, a route is represented by the consecutive grid points passed. For this structure, the total distance traveled is computed by summing the arc distances between the consecutive grids passed. As stated before, the UAV does not have move restrictions; it can move to all its adjacent grids provided that it does not pass beyond the boundaries of the terrain. However, in this structure we restricted the move to only adjacent points. We represent each grid point with (x,y) where x shows the row number and y shows the column number. The upper left extreme grid is (1,1). Despite the fact that in Figure 3.1, a move from (1,1) to (3,2) is possible in real life, this move is not allowed in our approach. Instead, the UAV may move from (1,1) to (2,1) and to (3,2) or it may move from (1,1) to (2,2) and then to (3,2). However, after finding the "best" tours with our approach, we may structure the final tours manually by allowing these movements. The movement representation we use in our solution approach is an approximation for the real life movements.

Let i and j be indices in the node set N. Then, the objective function that shows the total distance traveled is as follows;

$$d = \sum_{i \in N} \sum_{j \in N} c_{ij} * x_{ij}$$

where,

c_{ij} : distance of arc (i, j)

$$x_{ij} = \begin{cases} 1 & if\ arc(i,j)\ is\ on\ the\ path \\ 0 & otherwise \end{cases}$$

3.3.2. Radar Detection Threat

The second objective that we consider in route planning problem is the radar detection threat.

Gudaitis (1994) indicates that the radar threat can be calculated by two methods; namely, static and dynamic. In the static model, the radar threat is precalculated for each grid cell, it does not take into account the orientation of the aircraft or the angle between the UAV and the radar when it passes to that grid. However, in the dynamic model, radar threat is calculated as the UAV moves. In this model, the orientation of the UAV and the radar cross section are taken into account. Gudaitis mentions many advantages of the dynamic model over the static model like the memory requirements of the static model, easy updating capabilities of the dynamic model etc. However the static model is simpler and knowing only the radar sites define the radar threat values sufficiently. Therefore, we consider the static model in this study.

The signal-to-noise-ratio (S/N) is used in detecting the probability of detection at a grid. If the transmitter and receiver are separate, we consider a bistatic radar model which uses the following equation for S/N calculation as indicated by Gudaitis.

$$S/N = \frac{P_t G_t G_r \lambda^2 \sigma}{(4\pi)^3 K \; T_s B_n L_t L_r R_t^2 R_r^2} \tag{3.1}$$

where;

P_t: power transmitted by radar (watts)

G_t: power gain of transmitting antenna

G_r: power gain of receiving antenna

L_t: Transmitting system loss

L_r: Receiving system loss

σ : aircraft radar cross section (RCS) (square meters)

R_t : distance from the transmitter to aircraft (meters)

R_r : distance from the receiver to aircraft (meters)

λ : wavelength of signal frequency (meters)

K : Boltzman's constant (joules/Kelvin)

T_s : receive system noise temperature (Kelvin)

B_n : noise bandwidth of receiver (Hertz)

The first five parameters are related to the characteristics of the radar; whereas σ is related to the orientation of the UAV when it faces the radar signals. In the dynamic model, R_t, R_r and σ change as the aircraft moves to another grid. In the static model, only R_t and R_r change as the aircraft moves and a single σ value is used for all grids.

If the transmitter and the receiver are not separate, we consider the monostatic radar model. The S/N equation (3.1) is reduced to the following equation:

$$S/N = \frac{P_t G_t^2 \lambda^2 \sigma}{(4\pi)^3 K\ T_s B_n L_t^2 R^4}\,. \tag{3.2}$$

Gudaitis (1994) argues that there are many ways of representing the RCS. He indicates that both the orientation and the distance from the radar affect the RCS. He uses the dynamic model, therefore computes many RCS values for different angles of aircraft. Since we are using the static model, we assume that the UAV is a sphere; therefore has the same RCS from all angles. Skolnik (1980, p. 44) gives examples on the RCSs of different vehicles. The unmanned winged missile is considered to have 0.5 square meters of RCS. We consider this value for the RCS of the UAV.

With the constants combined under the heading C as follows, we obtain the following equation for S/N ratio.

$$C = \frac{P_t G_t^2 \lambda^2 \sigma}{(4\pi)^3 K \; T_s B_n L_t^2}$$

$$S/N = \frac{C}{R^4} \tag{3.3}$$

The S/N ratio is used in the calculation of the probability of detection, p_d. Gudaitis (1994) uses the following simplified relation between S/N and p_d.

$$p_d = \begin{cases} 1 & if \; S/N > 15 \\ ((S/N) - 5)/10 & if \; 5 < S/N \le 15 \\ 0 & if \; S/N \le 5 \end{cases}$$

In our approach, we changed the upper and lower bounds for the S/N ratio. The S/N ratios need to be converted into dB units before computing p_d values. After converting the S/N ratio in dB units (by taking $10 \times \log_{10}(S/N)$), we compute p_d with the following equation.

$$p_d = \begin{cases} 1 & if \; S/N > 30 \\ ((S/N) - 15)/15 & if \; 15 < S/N \le 30 \\ 0 & if \; S/N \le 15 \end{cases} \tag{3.4}$$

Olsan (1993) estimates the radar detection threat by the measure $r = \sum_{i=1}^{i=n} d_i * (p_d)_i$, where r is the total radar detection threat, d_i is the distance of edge i and $(p_d)_i$ is the radar detection probability of edge i. Firstly the radar detection probability of edge i is calculated by the S/N ratio. Then, this probability of detection of edge i is multiplied with the edge's distance. When these values are summed over all the edges passed, he obtains the overall radar detection threat.

Gudaitis (1994) introduces one more variable, w_i in the radar detection threat equation. The radar detection threat measure he suggests is the following.

$$r = \sum_{i=1}^{i=n} w_i * d_i * (p_d)_i$$

where;

$$w_i = \frac{(k+1)X}{k+X}$$

$$k = \begin{cases} 0 & if\ (p_d)_{i-1} = 0 \\ k+1 & otherwise \end{cases}$$

X is an upper bound on w_i defined by Gudaitis (1994).

Gudaitis indicates that the duration of radar exposure is also important. Therefore, if consecutive edges encountered have a probability of detection greater than zero, the value of w_i increases from 1. The maximum value w_i can take is X. He argues that taking X=4 gives satisfactory results.

In Section 3.3.1, we state that with constant speed, the distance traveled can be used to represent the duration of the flight. Multiplying the detection probability of one 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. Therefore, we use the radar exposure measure presented by Olsan in our solution approach. Next, we give details of the calculation of the radar exposure measure.

Firstly, we calculate the S/N ratios at grids. To calculate p_d of an edge, we take the arithmetic average of the S/N ratios of the two grids that are the end points of that edge. We compute the p_d value with equation (3.4). After computing the p_d value, we multiply it with the length of the edge using the following equation.

$$r_{ij} = c_{ij} * (p_d)_{ij}$$

where

rd_{ij} : the radar detection threat of arc (i, j)

c_{ij} : distance of arc (i, j)

$(p_d)_{ij}$: the radar detection probability of arc (i, j)

In this model, we do not consider the continuous exposure to radar. We assume that each movement can be detected by only a single radar. Therefore, we compute the S/N ratio with respect to the nearest radar site. The total radar detection threat is the summation of all radar threats at the edges that the UAV passes. We assume all the radar sites are identical, therefore they own the same values that are used in the equation (3.3).

The objective function that shows the total radar detection threat is as follows;

$$r = \sum_{i \in N} \sum_{j \in N} x_{ij} * c_{ij} * (p_d)_{ij}$$

where,

c_{ij} : distance of arc (i, j)

$$x_{ij} = \begin{cases} 1 & if\ arc(i,j)\ is\ on\ the\ path \\ 0 & otherwise \end{cases}$$

$(p_d)_{ij}$: the radar detection probability of arc (i, j)

In our solution approach, we consider a terrain without obstacles. However, if the obstacles are also included in the terrain model, the computation of the p_d value may be recalculated taking into account the location of the obstacle. This leads to asymmetric radar detection threat values around the radar areas and two grids having the same distance may have fundamentally different p_d values.

3.4. Definitions

As stated before, there exist two forms of route planning; with a single target and with multiple targets. Although the above three representations are common to both, the solution approaches are different.

Before we continue with the problem types, we give some definitions.

Let x be the decision variable vector, X be the feasible space and $z(x) = (z_1(x), z_2(x),..., z_p(x))$ where p is the number of objectives and $z_i(x)$ is the performance of solution x in objective i. Let us assume without loss of generality all objectives are to be minimized.

Definition 3.1. A solution $x \in X$ is said to be efficient if there does not exist any solution $x' \in X$ that is at least as good as x in all objectives, and better than x in at least one objective; i.e. there does not exist $x' \in X \ni z_j(x') \le z_j(x)$ $j = 1,..., p$ and $z_j(x') < z_j(x)$ *for at least one* j. Otherwise, x is said to be inefficient. The set of all efficient solutions constitute the efficient frontier of a problem.

Definition 3.2. If x is efficient, then $z(x)$ is nondominated. Otherwise, $z(x)$ is dominated.

Definition 3.3. An efficient solution x is a supported efficient solution if none of the convex combinations of other solutions dominates x. Otherwise, x is an unsupported efficient solution.

Definition 3.4. The extreme efficient solutions are the solutions that have the best value in one of the objectives among all solutions.

Figure 3.2 illustrates all the solution types. The solutions A, B, C, D, E and F are the efficient solutions and they form the efficient frontier of the problem. The

solutions G, H and I are inefficient solutions; G is dominated by B, C and D; H is dominated by C, D and E; I is dominated by A. Among the efficient solutions, A, C, E and F are supported efficient solutions and B and D are unsupported efficient solutions. The extreme efficient solutions are A and F.

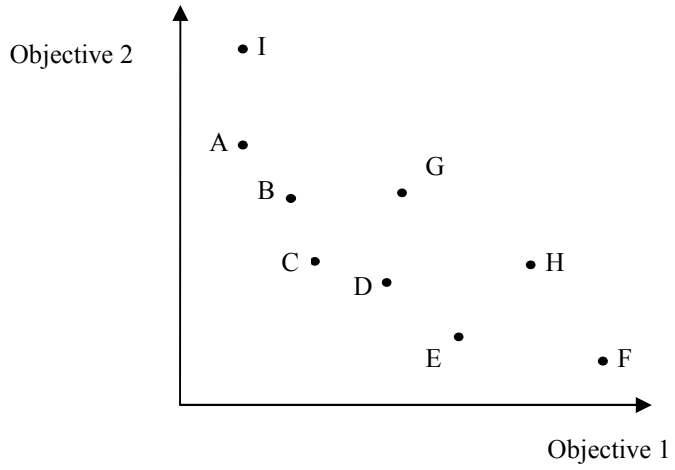


Figure 3.2 Solution Types

3.5. Route Planning With a Single Target

As the problem turns into finding the "best" path between the starting point and the target, it can be considered as a MOSPP. The formulation of the MOSPP is as follows;

Parameters:

N : node set

s : starting node

t : target node

c_{ij} : distance from node i to node j

Decision variables:

$$x_{ij} = \begin{cases} 1 & if\ arc(i,j)\ is\ on\ the\ path \\ 0 & otherwise \end{cases}$$

Model:

$$Min\ z_1 = \sum_{i \in N} \sum_{j \in N} c_{ij} * x_{ij} \tag{3.5}$$

$$Min\ z_2 = \sum_{i \in N} \sum_{j \in N} x_{ij} * c_{ij} * (p_d)_{ij} \tag{3.6}$$

$$\sum_{j \in N - \{s\}} x_{sj} = 1 \tag{3.7}$$

$$\sum_{j \in N - \{t\}} x_{jt} = 1 \tag{3.8}$$

$$\sum_{i \in N - \{t\}} x_{ij} - \sum_{i \in N - \{s\}} x_{ji} = 0 \quad \forall j \in N - \{s,t\} \tag{3.9}$$

$$x_{ij} \in \{0,1\} \tag{3.10}$$

The two objectives, total distance traveled and total radar detection threat, are minimized in equations (3.5) and (3.6) respectively. Equations (3.7), (3.8) and (3.9) ensure that one unit flow moves from node s to node t. With (3.10) we ensure that if arc (i, j) is traversed, the variable x_{ij} gets value 1; otherwise it gets value 0.

There are many solution approaches proposed in the literature for MOSPP. One approach is to combine the objectives linearly and to minimize the resulting composite single objective. Gudaitis (1994) and Olsan (1993) consider the mission route planning problem with a single target as SOSPP and use heuristics in the solution. As explained in Chapter 2, Gudaitis (1994) argues that the solution of this problem using single objective shortest path algorithms; for instance the greedy algorithm Dijkstra's algorithm; is both computationally inefficient and needs transformation. In his problem, Gudaitis assumes that the aircraft cannot make moves more than 45^0 from the horizontal. For instance, in

Figure 3.1, if the UAV comes to (2,2) from (2,1), it can only go to (1,3), (2,3) and (3,3). However, if it comes to (2,2) from (1,2), it can go to (3,1), (3,2) and (3,3). In other words, the next move of the aircraft is determined by its previous move. Therefore, he suggests using a transformed graph that indicates not only the present grid but also the previous grid that is used to reach that present grid. The next grid to be entered can be found this way. The problem becomes a general shortest path problem with this transformation; however Gudaitis argues that it would be computationally hard for Dijkstra's algorithm to solve this enlarged grid structure. In our problem structure, we do not need this transformation since the UAV is assumed to move to all its adjacent grids independent of the previous grid it passed.

A drawback of the approaches of Gudaitis (1994), Olsan (1993) and Yavuz (2002) is that they all consider the linear combination of the two objectives; distance traveled and radar detection threat. When the linear combination is considered, the problem turns to be a single-objective problem. With their single objective solution approach, only supported efficient solutions can be found. However, by multi-objective approaches both the supported and unsupported efficient solutions can be found. In the solution of multi-objective problems, the multi-objective approaches may need more time to generate the whole efficient frontier when compared to single-objective approaches. However, reaching both the supported and unsupported solutions is an advantage of multi-objective approaches.

The objectives for the MOSPP can be treated separately. There are many algorithms proposed in the literature that are used for solving MOSPP as explained in Chapter 2.

3.6. Route Planning With Multiple Targets

The problem becomes more complicated when more than one target is involved. In this problem, the UAV starts from the starting point, moves to all the targets and finally returns to the starting point. This problem can be considered as a MOTSP.

To our best knowledge, in the literature the MOTSP is assumed to have a single path between any pair of nodes, as discussed in Chapter 2. For a problem with p - objectives (where all objectives are to be minimized), each node pair is connected with a single path that is represented with its performance in all these objectives as $[c_1, c_2, ..., c_p]$ where c_i is the performance of that path in the i^{th} objective. Then, the tours that "minimize" the p objectives are tried to be found.

However, there may exist many efficient paths between any two nodes under the presence of p objectives. Like we are able to find efficient shortest paths between any two nodes; in MOTSP it is also possible to have at least two efficient connections between two nodes. Therefore, a generalized MOTSP can be defined by considering multiple efficient paths between nodes.

In other words, for the MOTSP defined in the literature, we only need to answer the question "What should be the order of visits to the nodes?". However, in the generalized MOTSP we need to answer two questions; "What should be the order of visits to the nodes?" and "Which path should be used between any two consecutive nodes?". Furthermore, these two questions are dependent on each other. This substantially increases the complexity of the problem.

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

Theorem 1. An efficient tour does not contain any inefficient paths.

Proof. Let's assume we have a MOTSP with N nodes. Let $x'_{kl} \in X$ be an inefficient path between nodes k and l. Since x'_{kl} is inefficient, we know that there exists at least one $x''_{kl} \in X$ such that the following inequalities hold.

$$z_j(x''_{kl}) \le z_j(x'_{kl}) \quad j = 1,..., p \tag{3.11}$$

$$z_j(x''_{kl}) < z_j(x'_{kl}) \quad \textit{for at least one } j \tag{3.12}$$

For any tour T that includes nodes k and l as consecutive nodes, we have the performance of tour T in objective i for $i = 1,..., p$ as follows;

$$z_i(T) = \sum_{(a,b)\in N-\{k,l\}} z_i(x_{ab}) + z_i(x_{kl})$$

For the path between nodes k and l, if we use the path x'_{kl} we obtain the performance in objective i for $i = 1,..., p$ as follows;

$$z'_i(T) = \sum_{(a,b)\in N-\{k,l\}} z_i(x_{ab}) + z_i(x'_{kl})$$

For the path between nodes k and l, if we use the path x''_{kl} we obtain the performance in objective i for $i = 1,..., p$ as follows;

$$z''_i(T) = \sum_{(a,b)\in N-\{k,l\}} z_i(x_{ab}) + z_i(x''_{kl})$$

If we keep the paths used between the other consecutive nodes constant; i.e. x_{ab} $(a,b) \in N - \{k,l\}$; we know from (3.11) and (3.12) that;

$z''_i(T) \le z'_i(T)$ $i = 1,..., p$

$z''_i(T) < z'_i(T)$ *for at least one* i

The solution $z'(T)$ is dominated by the solution $z''(T)$. Therefore any efficient tour T would have path x''_{kl} instead of path x'_{kl}. #

Remark 1. All combinations of efficient paths do not necessarily yield efficient tours.

Example. Consider a biobjective TSP with two nodes. We represent the paths and tours with (z, z_2) where z_1 is their performance in the first objective and z_2 is their performance in the second objective. Let's assume that there are two efficient paths when going from node 1 to node 2. The paths are (2,8) and (4,6). Similarly, there are two efficient paths when going from node 2 to node 1. These paths are (1,10) and (2,5). There are four ways of going from node 1 to node 2 and returning to node 1. The four possible tours are (3,18), (4,13), (5,16) and (6,11). Among these four tours, the three tours (3,18), (4,13) and (6,11) are efficient. The tour (5,16) is inefficient even though it is a combination of some efficient paths.

Theorem 2. A dominated subtour cannot be part of an efficient tour even if it is made up of efficient paths.

Proof. Let $y'_Q \in X$ be an inefficient solution for the subtour Q and is composed of the efficient paths $(y'_{ka}, y'_{ab}, ..., y'_{jl})$. Let S be the set of nodes included in subtour Q such that $S: \{k, a, b, ..., j, l\}$. Since y'_Q is dominated, we know that there exists at least one $y''_Q \in X$ such that the following inequalities hold.

$$z_j(y''_Q) \le z_j(y'_Q) \quad j = 1, ..., p \tag{3.13}$$

$$z_j(y''_Q) < z_j(y'_Q) \quad \textit{for at least one } j \tag{3.14}$$

Let tour T be composed of subtours Q and R. Then the performance of T in objective i for $i = 1, ..., p$ as follows;

$$z_i(T) = z_i(y_Q) + z_i(y_R)$$

For subtour Q, if we use the path y'_Q we obtain the performance in objective i for $i = 1, ..., p$ as follows;

$$z'_i(T) = z_i(y'_Q) + z_i(y_R)$$

For subtour Q, if we use the path y''_Q we obtain the performance in objective i for $i = 1,..., p$ as follows;

$z''_i(T) = z_i(y''_Q) + z_i(y_R)$

If we keep the paths used in subtour R constant; i.e. y_R; we know from (3.13) and (3.14) that;

$z''_i(T) \leq z'_i(T) \quad i = 1,..., p$

$z''_i(T) < z'_i(T) \quad for\ at\ least\ one\ i$

The solution $z'(T)$ is dominated by the solution $z''(T)$. Therefore any efficient tour T would have solution y''_Q instead of solution y'_Q for subtour Q. #

Consider the example problem shown in Figure 3.3. We try to identify the "best" tour(s) that starts from node A, passes through all nodes and returns to node A. There are two objectives for each arc; both to be minimized; written in parentheses next to the arcs.

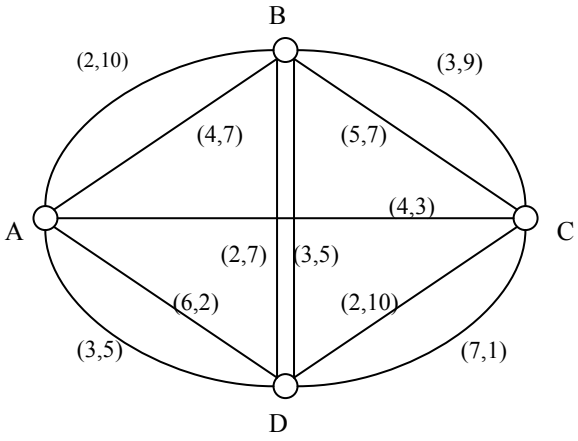


Figure 3.3 An Example MOTSP

In a 4-target problem, there are three different orders of visits since the TSP is symmetric.

Order 1: A – D – C – B – A

Order 2: A – B – D – C – A

Order 3: A – C – B – D – A

If we enumerate all the tours for order 1, we come up with 16 different tours. [2(paths between A and D) × 2(paths between D and C) × 2(paths between C and B) × 2(paths between B and A)].

Only 9 of the 16 tours turn out to be efficient as shown in Table 3.1.

Table 3.1 The Efficient Tours of A – D – C – B – A Order

	Efficient Tours								
Objective 1	10	12	14	15	17	18	19	20	22
Objective 2	34	31	27	25	23	22	20	19	17

When the same analysis is conducted for orders A – B – D – C – A and A – C – B – D – A; and all the resulting tours are compared, we find 8 efficient tours overall, as shown in Table 3.2. None of the tours of order A – D – C – B – A turn out to be efficient when compared with the tours of the other orders.

Table 3.2 The Efficient Tours of the Example Problem

	A – B – D – C – A				A – C – B – D – A			
	Tour 1	Tour 2	Tour 3	Tour 4	Tour 5	Tour 6	Tour 7	Tour 8
Obj. 1	10	11	17	18	12	13	15	16
Obj. 2	30	28	18	16	24	22	20	19

For the MOTSP, if the objectives are linearly combined to form a single objective, the problem becomes a single-objective TSP. To solve this problem, we first need to prepare the cost matrix for the composite objective; i.e. we need to find the shortest paths between every pair of targets and between the starting point and all the targets. Following the determination of the cost matrix, the problem reduces to a regular TSP. The second part is the determination of the shortest path that the UAV should follow based on the composite cost matrix.

Yavuz (2002) considers route planning with multiple targets and uses a solution approach different than the one outlined above. Firstly, by combining the objectives linearly, he generates the cost matrix by the best first search technique. In this technique, the next node that increases the objective function (to be minimized) the least is selected. In three dimensional space, there are totally 26 adjacent grids of one grid, 9 grids in front, 9 grids in the back and eight grids on the same level. Backward moves are not allowed; therefore from the next 17 nodes (in three dimensional space) the next node is selected as the one increasing the composite objective the least. The distances of these paths are used to construct the cost matrix. This cost matrix is used to determine the visiting order of targets. Then the shortest path between each consecutive target is calculated by partial swarm optimization-ant system (PSO-AS).

This is a myopic heuristic and it may not yield near optimum solutions. The following example demonstrates that this technique may produce poor results. The area shown in Figure 3.4 is divided to 5×5 equidistant grid points. The extreme upper left point has coordinates (1,1). The rows show the latitude; x coordinate; and the columns show the longitude; y coordinate. In the example, four targets (shown with triangles) are placed at grid points (1,1), (2,4), (3,3) and (5,5). The numbers on the edges represent the composite objective value to be minimized. For simplicity, we do not allow diagonal moves.

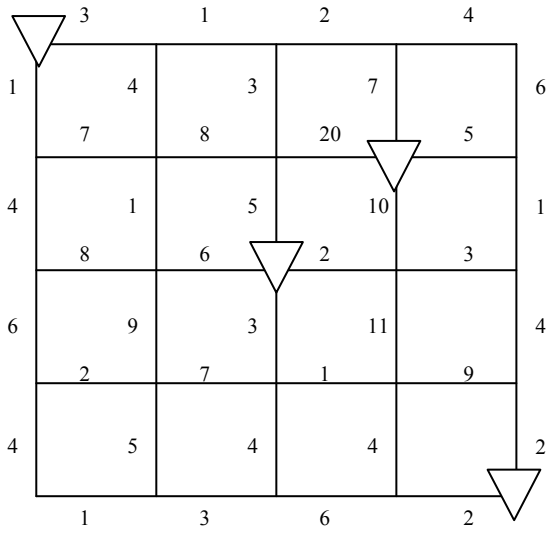


Figure 3.4 An Example for the Best First Technique

Yavuz (2002) assumes that the aircraft can only make forward moves. We use this assumption in generating the cost matrix. We refer the target at point (1,1) as Target 1, at point (2,4) as Target 2, at point (3,3) as Target 3 and at point (5,5) as Target 4. We find the entries for all target pairs to construct the cost matrix by the best first search technique.

Shortest path between Target 1 and Target 3

We assume that the aircraft can only make forward moves. From Target 1 to Target 3, the aircraft should move to the south east direction; therefore we do not allow moves to north or west. From point (1,1), there are two options; either move to (2,1) or (1,2). Since the cost of moving to (2,1) is smaller than the cost of moving to (1,2), the aircraft moves to (2,1). At (2,1) there are again two options, either move to (3,1) or (2,2). The aircraft moves to (3,1). At point (3,1), the aircraft must move to (3,2). If it passes beyond Target 3 to points that have a latitude value larger than 3, to reach Target 2 the aircraft should make moves to north. However, because of move restrictions on aircraft it should move in the

rectangle that contains the two targets on its opposite corners. Therefore, the aircraft moves from (3,1) to (3,2) and to (3,3). The total cost is 19 (1+4+8+6).

We find all the entries of the cost matrix similarly. One should also note that the results may change based on the starting target; for instance, the entry between Target 1 and Target 3 is 19 if we start from Target 1 and move to Target 3; however it is 12 if we start from Target 3 and move to Target 1. In the calculations, we start from Target i and moved to Target j where i<j. The resulting cost matrix is shown in Table 3.3.

Table 3.3 Cost Matrix Using Best First Search Technique

	Target 1	Target 2	Target 3	Target 4
Target 1	0	36	19	29
Target 2	36	0	12	12
Target 3	19	12	0	11
Target 4	29	12	11	0

For a problem with four targets, we have three different orders. When we enumerate all tours we have;

Tour 1: Target 1 - Target 2 - Target 3 - Target 4 - Target 1 Total Cost: 88
Tour 2: Target 1 - Target 2 - Target 4 - Target 3 - Target 1 Total Cost: 78
Tour 3: Target 1 - Target 3 - Target 2 - Target 4 - Target 1 Total Cost: 72

We calculate the tour costs for the three tours using the cost matrix in Table 3.3. The best tour is Tour 3 with total cost 72. Tour 2 has a total cost of 78 and Tour 1 has a total cost of 88.

However, if we have generated the cost matrix according to the real shortest paths between the targets, we would have the cost matrix shown in Table 3.4.

Table 3.4 Cost Matrix using Shortest Path Algorithms

	Target 1	Target 2	Target 3	Target 4
Target 1	0	13	12	22
Target 2	13	0	12	12
Target 3	12	12	0	10
Target 4	22	12	10	0

The tours and their total costs are as follows;

Tour 1: Target 1 - Target 2 - Target 3 - Target 4 - Target 1 Total Cost: 57
Tour 2: Target 1 - Target 2 - Target 4 - Target 3 - Target 1 Total Cost: 47
Tour 3: Target 1 - Target 3 - Target 2 - Target 4 - Target 1 Total Cost: 58

Tour 2 is the best tour with this approach. Although Tour 3 is the best tour according to the best first search technique, it has the highest cost among the three tours when we use the real shortest distances.

As stated before, the main difference between the MOTSP with multiple efficient paths between nodes and the original TSP is that; instead of one path we consider all the efficient paths between each node pair. As a result, we do not have any straightforward cost matrix. The approach of Yavuz (2002) is inadequate to find optimal or near optimal solutions from many points. First of all, the cost matrix constructed by using best first search technique is not a good estimator of the real cost matrix since it does not show the true shortest paths between the targets. A visiting order depending on these wrongly estimated values would not yield optimal results.

Additionally, in the second part of the solution approach he proposes; i.e. finding the shortest paths between the consecutive targets of the TSP tour; Yavuz uses a search technique similar to the best first search technique. The result of this search technique again does not give optimal values.

Improvement for this thesis could be done in both determining the cost matrix and calculating the shortest paths between the targets. If the cost matrix is prepared by taking the real shortest distances between the targets, then we would not need a second phase after determining the visiting order of targets. Since the cost matrix reflects the real shortest distances between the targets, we would come up with the optimal visiting order of targets; therefore, we would not need a second phase in which we try to determine the shortest paths between the consecutive targets. Finding the visiting order would be the result of this problem.

In conclusion, this algorithm is not very useful for solving MOTSP with multiple efficient paths between nodes and it may give suboptimal results.

In our approach, we consider route planning with multiple targets and treat the two objectives separately. In Chapter 4, we present an interactive approach to find the most preferred paths and most preferred tour of a DM based on a linear utility function. In Chapter 5 we find the potential efficient tours of the problem using an EA, a modification of NSGA-II.

CHAPTER 4

AN INTERACTIVE APPROACH

Generating the whole efficient frontier of a multi-objective combinatorial optimization problem would not be practical or meaningful. Typically, there are too many such solutions and solving for each solution is computationally hard. Instead, under the guidance of the DM's preferences, we generate the most preferred region of the efficient frontier. In this chapter, we introduce a solution approach in which we incorporate the preferences of the DM in finding the best solution assuming that the DM has a linear utility function.

4.1. The Solution Approach

With the guidance of a DM, our aim is to find the most preferred solution. In order to find the most preferred solution of the DM, we effectively utilize the available preference information.

In our solution approach, we assume that the DM has a linear utility function. In a linear utility function, the objectives are linearly combined and we try to minimize (or maximize) the composite objective function. It can be shown as follows;

$$U(z) = w_1 z_1(x) + \ldots + w_n z_n(x)$$

where n is the number of objectives, w_i is the weight of i^{th} objective and $z_i(x)$ is the performance of x in the i^{th} objective. Typically, the weights are normalized to sum to 1.

In our case, we have only two objectives.

$z(x) = (z_1(x), z_2(x))$

We restrict the value of the weights between 0 and 1. Consequently, the utility function becomes;

$U(z) = w_1 z_1(x) + w_2 z_2(x)$

$w_1 + w_2 = 1$

or

$U(z) = w z_1(x) + (1-w) z_2(x)$

where $0 \le w \le 1$

The knowledge of w; i.e. weight of the first objective; is enough to come up with the composite objective function since the weight of the second objective can be found by $1-w$. Therefore, we only refer to w throughout the text.

At this stage, prior to the description of the properties of a linear utility function, we give some definitions on MCDM Literature.

Definition 4.1. Two solutions are adjacent efficient solutions if none of their convex combinations are dominated by any convex combinations of other solutions. In a problem where all objectives are to be minimized, x_j is adjacent efficient to x_i if there does not exist $x_t \in X \ni \sum_{t \ne j} \mu_t x_t \le \lambda x_j + (1-\lambda) x_i$ where $\sum_{t \ne j} \mu_t = 1$, $0 \le \mu_t \le 1$ and $0 < \lambda \le 1$.

Adjacent efficient solutions of x are shown in Figure 4.1. The left and right adjacent solutions of x are x_L and x_R, respectively. In a bicriteria problem, there are at most two distinct adjacent efficient solutions of a solution.

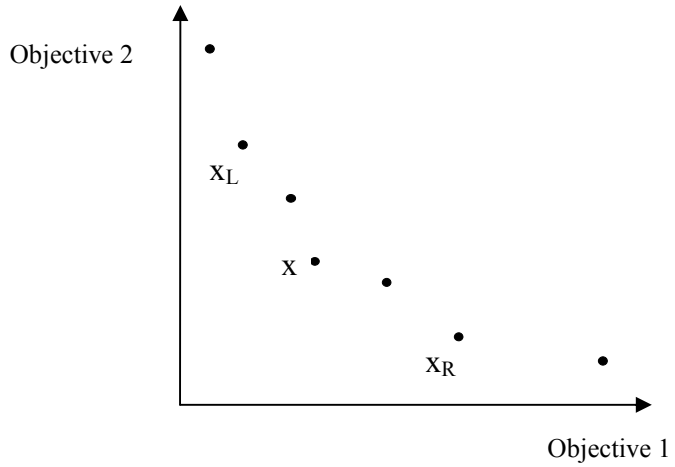


Figure 4.1 Adjacent Efficient Solutions of x

There are two properties of a linear utility function that can be used in our solution approach. Firstly, it is well known that the most preferred solution of a DM is a supported efficient solution. Therefore, we present only supported efficient solutions to the DM and do not try to find unsupported efficient solutions.

In an interactive approach, we do not want to ask too many questions to the DM. Therefore, we need to determine suitable solution pairs to be presented to the DM. The second property of a linear utility function that we use is that the solution that is preferred to all its adjacent solutions is the most preferred solution (Zionts, 1981). Using this property, we only present the adjacent solutions to the DM.

In our algorithm, we utilize the responses of the DM to restrict the possible weights of his/her underlying utility function. This decreases the number of questions asked. Initially, we do not have any restriction on the weights of the objectives and they range between 0 and 1. However, the stated preferences of the DM help us reduce the weight space and we update the upper and lower bounds on w.

4.2. An Interactive Algorithm

The following is an interactive algorithm we develop that finds the most preferred solution of a DM for a bicriteria (both objectives to be minimized) problem.

Step 0. Let $w_L = 1 - \varepsilon$ and $w_R = \varepsilon$ (Here, ε is a small positive number).

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. $x^* = x_L = x_R$. Go to Step 8. Oherwise, go to Step 2.

Step 2. Let $w' = \dfrac{(z_2(x_L) - z_2(x_R))}{(z_2(x_L) - z_2(x_R)) + (z_1(x_R) - z_1(x_L))}$

Find the middlemost solution minimizing the utility function formed with weight w'.

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

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.

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

  where ε is a very small positive number.

  $x^* = x_L$

  Go to Step 8.

- If x_R is preferred to x_L, update w_L.

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

  where ε is a very small positive number.

$x^* = x_R$

Go to Step 8.

Step 4. Find the left adjacent solution of x'. 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.

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

$x' = x_L$

Set $k = 1$.

Go to Step 4.

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

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

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'. 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.

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

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

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

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

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

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

$U(z) = wz_1(x) + (1-w)z_2(x)$

where $w_R \leq w \leq w_L$

The algorithm first finds the two extreme efficient solutions. If we can not 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 the middlemost solution using the function that passes 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 the middlemost solution in Step 2, we go to Step 4 where we search for the left adjacent solution of the middlemost solution. In Step 5, the DM compares the middlemost solution 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.

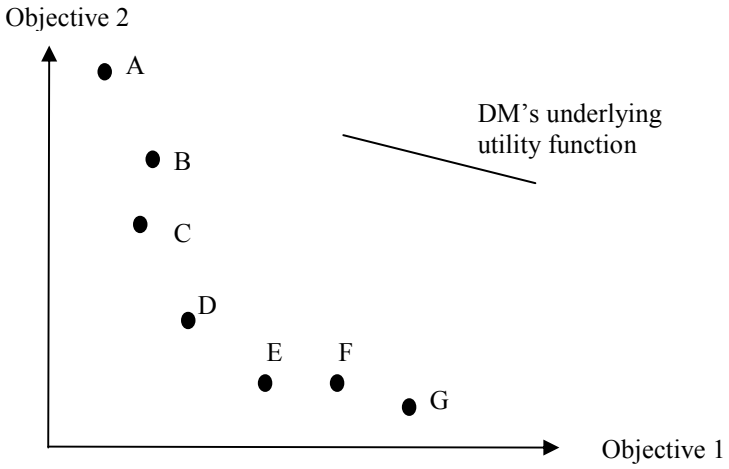
If, the middlemost solution is preferred to its left adjacent solution the first time Step 5 is encountered, we go to Step 6 where we find the right adjacent of the middlemost solution. If the DM prefers the middlemost solution to the right adjacent x_R we conclude that the best solution is the middlemost solution and we terminate the algorithm. This is because of the fact that the middlemost solution is

also preferred to its left adjacent before we get to Step 6. However, if x_R is preferred to the middlemost solution, 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.

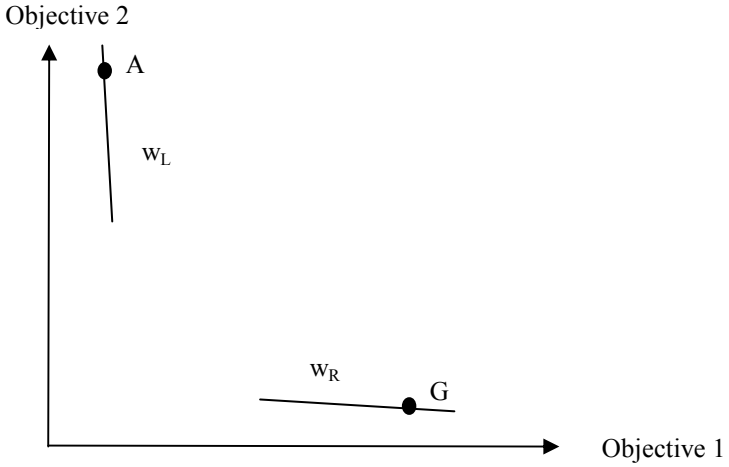
During the algorithm, if we find more than one efficient solution; i.e. if we can pass from Step 1 to Step 2; then we guarantee that either lower bound or upper bound or both of the bounds of w will be updated.

Figure 4.2 illustrates how the interactive algorithm proceeds. In Figure 4.2 (a), we show the efficient frontier of an example discrete bicriteria problem (both objectives to be minimized) and the DM's underlying utility function. We should note that the solution that minimizes the utility function is solution E.

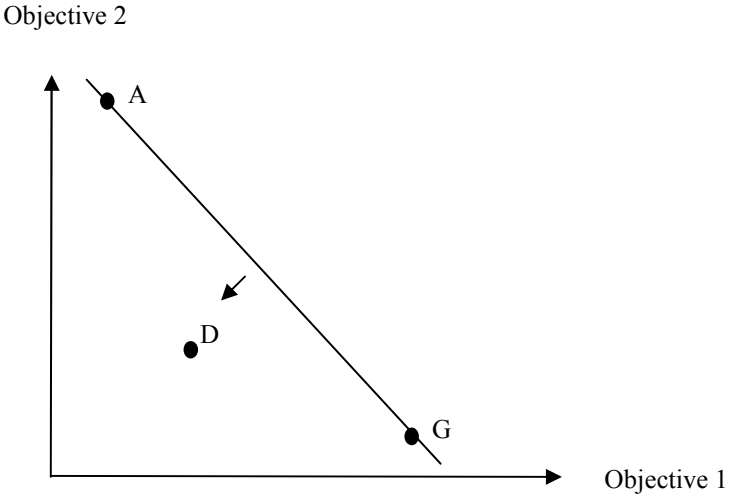
We run the interactive algorithm to find the most preferred solution of the DM iteratively. We start with finding the extreme efficient solutions of the problem; left extreme solution A with weight w_L and right extreme solution G with weight w_R as shown in Figure 4.2 (b). Since these two solutions are different, we find the middlemost solution that minimizes the function passing through the extreme efficient solutions. We come up with solution D as shown in Figure 4.2 (c). We continue with finding the left adjacent of D; which is solution C. We ask the DM to compare D and C and the DM prefers D to C since D gives a lower utility value. We update the upper bound on w; w_L as shown in Figure 4.2 (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. 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 Figure 4.2 (e) and conclude that the best solution is E. Now we have a better estimate on the favorable weight range of the DM that ranges from w_R to w_L.


Objective 2
A
DM's underlying
utility function
B
C
D
E
F
G
Objective 1

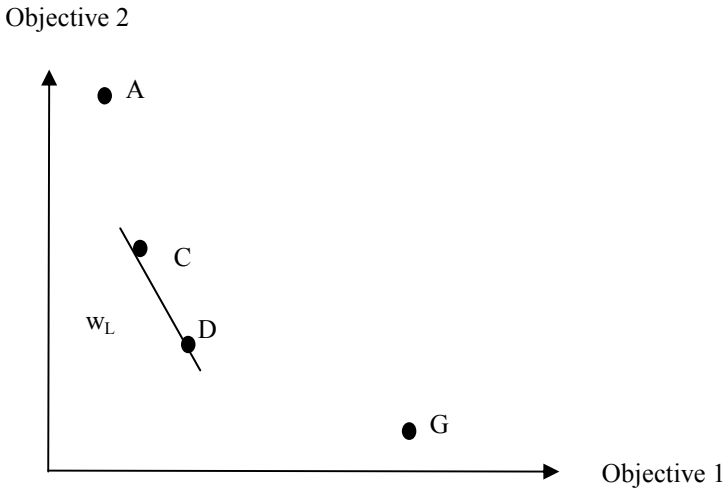
(a)


Objective 2
A
w_L
w_R
G
Objective 1

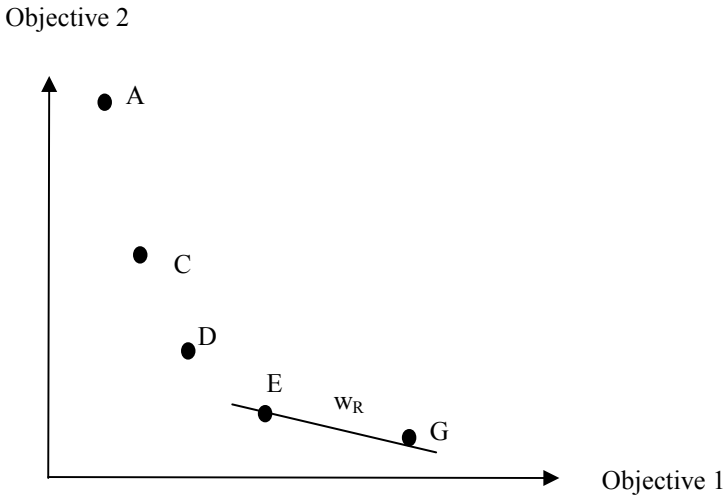
(b)


Objective 2
A
D
G
Objective 1

(c)



(d)



(e)

Figure 4.2 The Interactive Algorithm

In steps 4 and 6, we find the left and right adjacent solutions of an efficient solution using the algorithm we propose in Section 4.4. This algorithm is a modification of the algorithm explained in Section 4.3, Finding the Supported Efficient Solutions. This algorithm is proposed by Aneja and Nair (1979) and

Cohon (1978). In Section 4.4, we give the details of the algorithm for finding the adjacent efficient solutions.

4.3. Finding the Supported Efficient Solutions

This method generates the supported efficient solutions of a discrete multicriteria problem. It is developed by Aneja and Nair (1979) and Cohon (1978) in close time periods. In Figure 4.3, we illustrate how the algorithm proceeds for a bicriteria (both objectives to be minimized) problem.

Firstly, 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 (Here, ε is a very small positive number). In Figure 4.3 (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 Figure 4.3 (b), we obtain solution C and we update our current adjacent solution pairs. (A, C) and (B, C) are adjacent solution pairs in Figure 4.3 (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 Figure 4.3 (c).

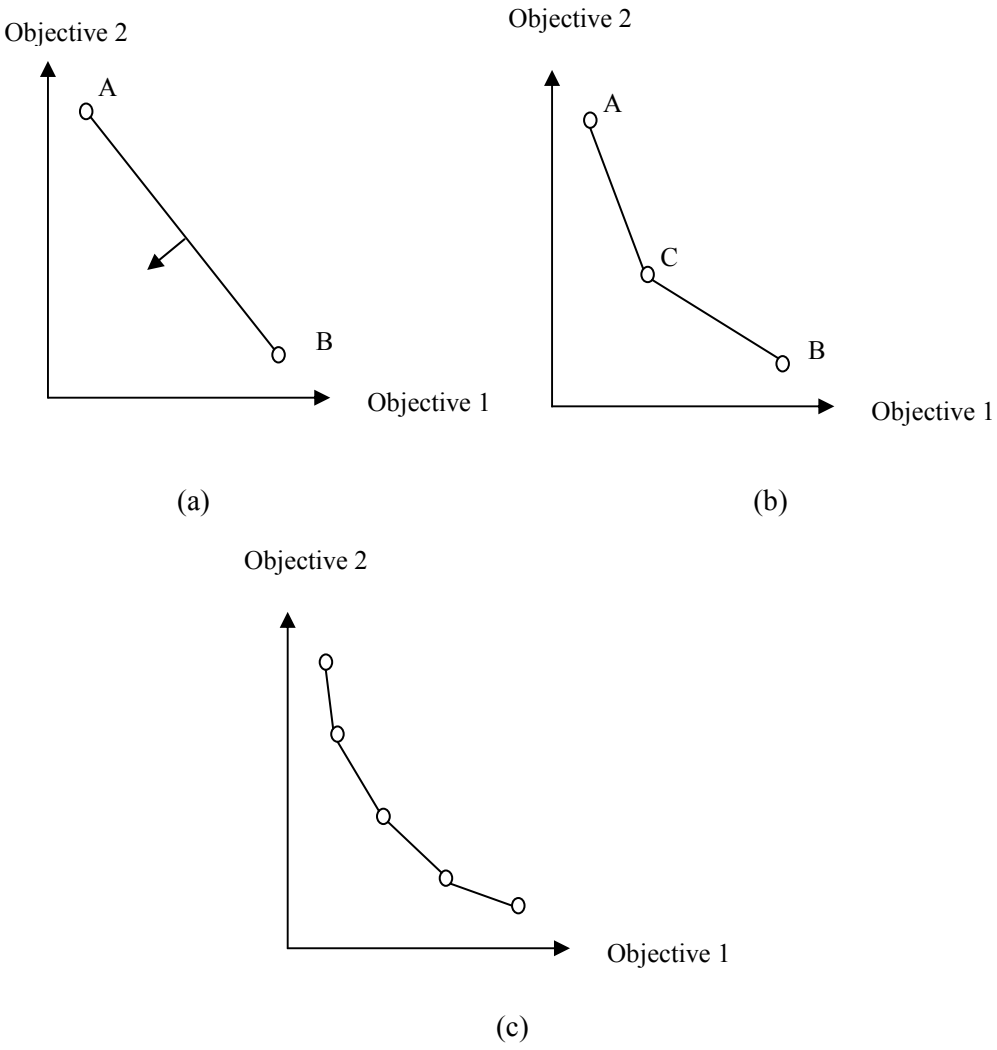


Figure 4.3 Finding the Supported Efficient Solutions

For a linear utility function, we only need supported efficient solutions; therefore we modified this algorithm to obtain the adjacent efficient solutions of a solution.

4.4. An Algorithm to Find the Adjacent Efficient Solutions

The following is an algorithm to find the left and right adjacent solutions of a supported efficient solution in a bicriteria (both objectives to be minimized) problem. Before we start the algorithm for finding the left adjacent solution, we assume that we know the left extreme efficient solution x_L; and before starting

the algorithm for finding the right adjacent solution, we assume that we know the right extreme efficient solution x_R. If we do not know the left extreme solution, we can find it by minimizing the composite objective $U(z) = (1-\varepsilon)z_1(x) + \varepsilon\ z_2(x)$ where ε is a small positive number. Similarly, we can find the right extreme efficient solution by minimizing the composite objective $U(z) = \varepsilon\ z_1(x) + (1-\varepsilon)z_2(x)$ where ε is a small positive number. We must make sure that we are not searching for the left adjacent of a left extreme efficient solution or searching for the right adjacent of a right extreme efficient solution.

Finding the Left Adjacent Solution

Step 0. Select a supported efficient solution x whose left adjacent solution is to be found.

Step 1. Let $w' = \dfrac{(z_2(x_L) - z_2(x))}{(z_2(x_L) - z_2(x)) + (z_1(x) - z_1(x_L))}$

Find the solution minimizing the utility function formed with weight w'.

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

Let the solution be x'. If $x' = x_L$ or $x' = x$, go to Step 3. Otherwise, go to Step 2.

Step 2. Let $x_L = x'$. Go to Step 1.

Step 3. The left adjacent of solution x is x_L.

In Step 1, we minimize the composite objective that passes through the current left adjacent of x; x_L. If we find another supported efficient solution between these two solutions, we update the current left adjacent of x with this solution and go back to Step 1. If we cannot find another solution between x and its current left adjacent, we conclude that the current left adjacent is the left adjacent 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 $w' = \dfrac{(z_2(x) - z_2(x_R))}{(z_2(x) - z_2(x_R)) + (z_1(x_R) - z_1(x))}$

Find the solution minimizing the utility function formed with weight w'.

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

Let the solution be x'. If $x' = x_R$ or $x' = x$, go to Step 3. Otherwise, go to Step 2.

Step 2. Let $x_R = x'$. Go to Step 1.

Step 3. The right adjacent of solution x is x_R.

The algorithm proceeds in a manner similar to finding the left adjacent solution. We minimize the composite objective that passes through x and its current right adjacent solution; x_R and update the current right adjacent with the new solution we find. When we do not obtain a new solution, we conclude that the current right adjacent is the right adjacent of x and we terminate the algorithm.

4.5. MOTSP with multiple efficient paths between nodes

We use the interactive algorithm proposed in Section 4.2 for the route selection of UAVs. It is explained in Chapter 3 that our problem is a combination of the MOSPP and the MOTSP. 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 of these most preferred paths.

4.5.1. Interactive Algorithm – Part I

In this part, we find the most preferred path between each target pair. Firstly, we number the targets from 1 to N; N being the number of targets. To find the most preferred path between a target pair (i, j) we run the interactive algorithm explained in Section 4.2. In each run of the algorithm, we not only find the most preferred path between the selected targets; 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 is the summary of the algorithm we use for finding the best path between each node pair.

Step 0. Set $t = 0$.

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

Step 2. If $t = 0$; find the most preferred path between target pair (i, j) using the interactive algorithm. Update the lower (w_R) and upper bounds (w_L) of w.
If $t \neq 0$; find the most preferred path between target pair (i, j) using the interactive algorithm skipping Step 0.
Update the lower (w_R) and upper bound (w_L) of w.
Set $t = t + 1$.

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.

The most preferred path between the first target pair is found using the algorithm in Section 4.2. Since we do not know at that stage the preferences of the DM, the favorable w ranges from ε to $1-\varepsilon$. However, based on the responses of the DM, we update the bounds of w. For the following pairs of targets, we have a tighter range for w, so we skip Step 0 of the interactive algorithm that defines the bounds of w.

In Step 2, we solve the shortest path problem for finding the extreme efficient solutions, middlemost solution and the adjacent solutions. Since we combine the objectives to a composite objective, we solve the SOSPP. We use an exact algorithm; Dijkstra's Algorithm (Winston, p. 416). Briefly, in this algorithm we have nodes that have permanent or temporary labels; labels indicating the shortest path from initial node to that node. If one node has a temporary label, it means that its shortest path to the initial node has not been found yet. When its shortest path to the initial node is calculated, its label is set permanent. In each step of this algorithm the node that has the smallest label is set permanent. Based on the label of this node, the labels of the nodes that have temporary labels are recalculated. Until the destination node is set permanent, the algorithm continues.

In the second part of the problem, we determine the best tour of the DM. We use the information that is obtained in Part I; the knowledge of the most preferred path between each target pair and the knowledge of the upper and lower bounds of w.

4.5.2. Interactive Algorithm – Part II

At the end of Part I of our algorithm, we have a single best path between each target pair and a range for w. The resulting problem turns to be a regular MOTSP

with a single connection between each target pair. Having this information at hand, we run the interactive algorithm to find the most preferred tour of the DM.

We run the interactive algorithm for the MOTSP skipping Step 0 since we use the updated range of w; $[w_R, w_L]$. In Steps 1, 2, 4 and 6, we use the TSP solver CONCORDE to find the optimal visiting order of targets. CONCORDE is a single objective TSP solver. We need to give the node-node adjacency matrix as input to CONCORDE to find the optimal visiting order. The node-node adjacency matrix is found for a weight w as follows;

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

As a result, CONCORDE gives the optimal order of visits to targets which is the best tour for the DM.

4.6. Computational Results

In this section, we give computational results on how our algorithm works. The first problem we consider is a terrain approximated by 10×10 grid structure. In this problem, we have five targets to be visited and three radar sites. Firstly, we use a simplified approach in terms of radar detection threat. Instead of using the radar equation presented in Section 3.2, we assigned radar detection threat values for each grid point depending on the distance of that point to the nearest radar. We use this simplified approach for demonstration purposes. However, we also solve the same problem using the equation presented in Section 3.2 in finding the radar detection threat values. Simplified radar detection threat values and calculated counterparts have similar difficulty values. The second problem we consider is a larger problem. We approximate the terrain by 100×100 grid structure and we place eight targets and 40 radar sites.

We generated all the problem instances. In the generation of the problem instances, we tried to ensure that we obtain more than one efficient path between the targets. Also, we wanted to have different efficient tours. The solutions are sensitive to the positions of the radars and the targets. We tried many combinations of radar and target placements until we obtain these two problems.

In Sections 4.6.1 and 4.6.2, we mention the first problem and the second problem respectively.

4.6.1. 5-Target MOTSP with Simplified Radar Threat Values

The first problem is a 10×10 terrain with five targets. The structure of the terrain can be seen in Figure 4.4. We refer the nodes as (x,y) where x shows the latitude; i.e. the rows; and y shows the longitude; i.e. the columns. The triangles represent the target points and Target 1 is located at (1,1), Target 2 at (3,8), Target 3 at (5,4), Target 4 at (9,7) and Target 5 at (10,3). The three radar sites are represented by circles. The radars are located in the center of the circles at (3,3), (6,8) and (8,4). The radar detection threat decreases from the center to the circumstance of the circle. We assign values to represent the radar detection threat inside the radar sites. For instance, for the first radar the radar threat is 7 at node (3,3), 5 at nodes (2,3), (3,2) (3,4) and (4,3), 3 at nodes (2,2), (2,4), (4,2) and (4,4) and 1 at nodes (1,3), (3,1), (3,5) and (5,3). The same assignment is also applied to the other radar sites. We assume there is no radar detection threat for the remaining nodes since the radar would not be effective beyond a certain range. The UAV does not have any move restrictions as stated in Chapter 3. However, from one node it can move only to its eight adjacent nodes. The horizontal and vertical distances are 1 unit, and the diagonal distances are $\sqrt{2}$ units. For example, moving from (1,1) to (1,2) or (2,1) has a distance of 1 unit and moving from (1,1) to (2,2) has a distance of $\sqrt{2}$ units.

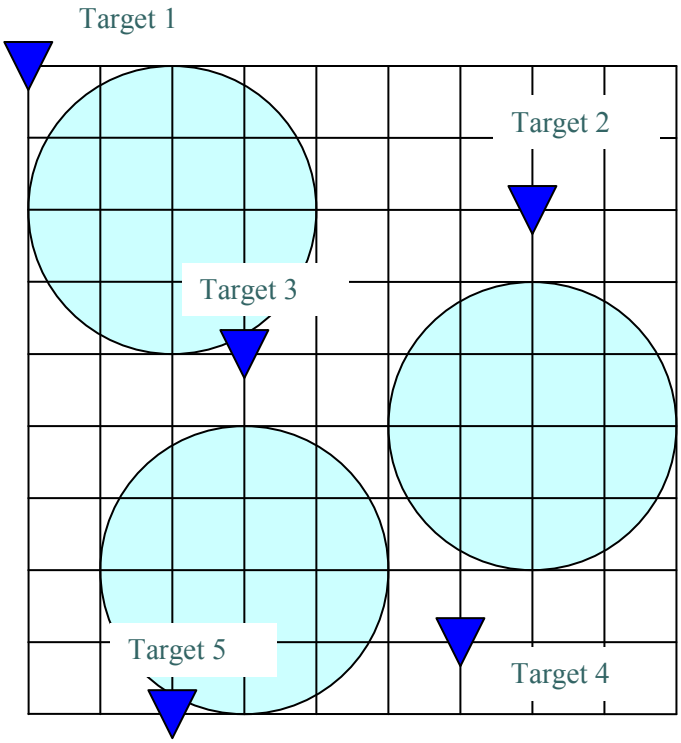


Figure 4.4 Terrain structure of 5-Target MOTSP (Simplified Radar Threat Values)

We have written a C code for computing the most preferred paths and the most preferred tour of a DM. The radar detection threat values are given as input to the code. The distances are calculated as the algorithm proceeds. Dijkstra's algorithm finds the shortest distance from one target to another target given the weight w. Each supported efficient solution found for the same target pair reduces the favorable weight space range. Finally, we come up with the best paths and tour of a DM in a few questions.

For this problem, we also find all efficient solutions. We firstly compute all the efficient paths between targets. We use the LP solver GAMS, and utilize the ε-constraint method. In this method, for a bicriteria problem (all objectives are to be minimized) one objective is treated as a constraint. While the other objective is minimized, the objective written as a constraint is restricted to values lower than ε. Changing the value of ε for each run of the linear program gives the whole

efficient frontier of the problem. In Table 4.1, the objective values of all the efficient paths between the targets can be seen. For this problem instance, by coincidence all the efficient solutions are also supported solutions. Some target pairs are connected by many efficient paths like targets 1 and 3 or targets 2 and 5; whereas there are target pairs that have only one connection like targets 1 and 2 or 1 and 5.

The efficient tours that are made up of these efficient paths are indicated in Table 4.2. The efficient frontier of the whole problem is shown in Figure 4.5. With five targets we have at most (5-1)!/2 = 12 different tours. For each tour, we have a number of efficient solutions depending on the number of efficient paths between each consecutive target. For instance, if we assume there are two efficient paths between each target, we have $12 \times 2^5 = 384$ potential efficient solutions. Of course, some of the tours and some of the combinations inside the tours would yield inefficient solutions. In the example problem, we have eight efficient tours. Six of the efficient solutions correspond to tour 1-2-4-5-3, one efficient solution corresponds to tour 1-2-3-4-5 and one efficient solution corresponds to tour 1-2-4-3-5. The composition of the efficient tour 1-2-4-5-3 is different in all the cases. Table 4.3 shows the path used between the consecutive targets for each efficient tour. For the same tour, at least one path between each consecutive target is different in distinct efficient solutions.

Table 4.1 The Efficient Paths Between the Target Pairs of 5-Target MOTSP (Simplified Radar Threat Values)

Target Pair (i,j)	# of Efficient Paths	Total Distance	Total Radar Detection Threat
(1,2)	1	7.828	1
(1,3)	4	7.242	1
		6.414	2
		5.828	5
		5.242	10
(1,4)	4	12.484	1
		11.656	2
		11.070	5
		10.484	10
(1,5)	1	9.828	1
(2,3)	1	4.828	0
(2,4)	3	8.070	0
		7.242	1
		6.414	11
(2,5)	5	11.898	0
		11.070	1
		10.242	3
		9.656	6
		9.070	11
(3,4)	1	5.242	0
(3,5)	3	7.070	0
		6.242	1
		5.414	11
(4,5)	2	12.312	0
		4.414	1

Table 4.2 Efficient Tours of 5-Target MOTSP (Simplified Radar Threat Values)

Tour	Total Distance	Total Radar Detection Threat
1-2-4-5-3	29.312	34
1-2-4-5-3	29.898	29
1-2-4-5-3	30.140	24
1-2-4-5-3	30.726	19
1-2-4-5-3	30.968	14
1-2-4-5-3	31.554	9
1-2-3-4-5	32.140	3
1-2-4-3-5	38.038	2

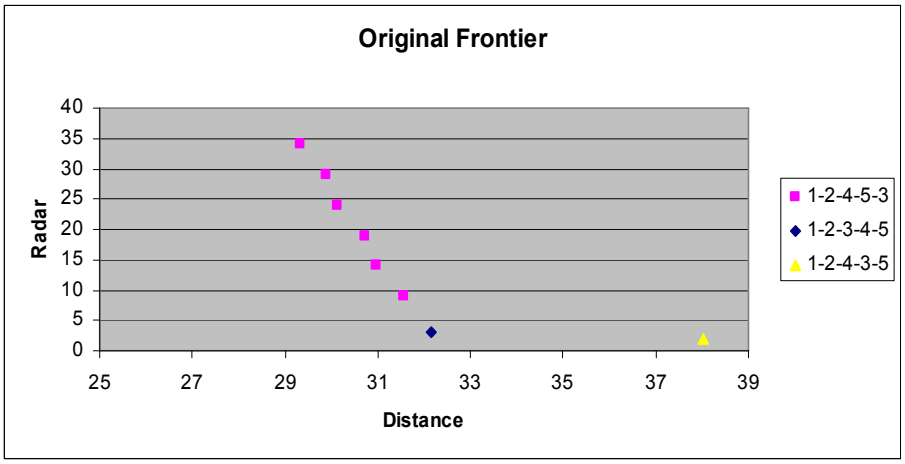


Figure 4.5 Efficient Frontier of 5-Target MOTSP (Simplified Radar Threat Values)

Table 4.3 Composition of Efficient Tours of 5-Target MOTSP (Simplified Radar Threat Values)

Tour	Target Pair	Distance	Radar Detection Threat	Total Distance	Total Radar Detection Threat
1-2-4-5-3	1-2	7.828	1	29.312	34
	2-4	6.414	11		
	4-5	4.414	1		
	5-3	5.414	11		
	3-1	5.242	10		
1-2-4-5-3	1-2	7.828	1	29.898	29
	2-4	6.414	11		
	4-5	4.414	1		
	5-3	5.414	11		
	3-1	5.828	5		
1-2-4-5-3	1-2	7.828	1	30.140	24
	2-4	7.242	1		
	4-5	4.414	1		
	5-3	5.414	11		
	3-1	5.242	10		
1-2-4-5-3	1-2	7.828	1	30.726	19
	2-4	6.414	11		
	4-5	4.414	1		
	5-3	6.242	1		
	3-1	5.828	5		
1-2-4-5-3	1-2	7.828	1	30.968	14
	2-4	7.242	1		
	4-5	4.414	1		
	5-3	6.242	1		
	3-1	5.242	10		

Table 4.4 Composition of Efficient Tours of 5-Target MOTSP (Simplified Radar Threat Values) (Continued)

Tour	Target Pair	Distance	Radar Detection Threat	Total Distance	Total Radar Detection Threat
1-2-4-5-3	1-2	7.828	1	31.554	9
	2-4	7.242	1		
	4-5	4.414	1		
	5-3	6.242	1		
	3-1	5.828	5		
1-2-3-4-5	1-2	7.828	1	32.140	3
	2-3	4.828	0		
	3-4	5.242	0		
	4-5	4.414	1		
	5-1	9.828	1		
1-2-4-3-5	1-2	7.828	1	38.038	2
	2-4	8.070	0		
	4-3	5.242	0		
	3-5	7.070	0		
	5-1	9.828	1		

We run the interactive algorithm for all target pairs and obtain the best path for each pair based on the preferences of the DM. Since the total distance traveled and total radar threat exposed do not depend on the initial or final nodes; we evaluate a node pair only once. For instance, if we compute the best path between target 2 and target 5, we do not compute the best path from target 5 and target 2 again since they result in the same best path. The algorithm starts finding the most preferred paths from target 1 to all other targets. After computing the best paths from target 1 to all other targets, the algorithm computes all the best paths from target 2 to the remaining targets 3, 4 and 5. Finally, the best path between targets 4 and 5 is computed.

After the computation of the best paths between all target pairs, we derive the input for the CONCORDE, the single objective TSP solver. CONCORDE follows the interactive approach with the reduced weight space and finally gives the best tour for the DM.

We define the underlying linear utility of the DM by specifying the w before we run the algorithm. The following is a brief explanation of how the algorithm works. All the solutions found are represented by their performances in the two objectives as (z_1, z_2) where z_1 is the total distance traveled and z_2 is the total radar detection threat exposed.

At first we define the best weight of the DM; w_{best} as 0.9000. Since we do not have any information about the preferences of the DM, the favorable weight space we know in the beginning is between 0.0001 and 0.9999. The algorithm evaluates the paths between targets 1 and 2 and reports that there is only one favorable path between these targets that is (7.828, 1). When there is only one path, the algorithm cannot reduce the weight space. It then searches for the best path between targets 1 and 3. It finds the two extreme solutions (5.242, 10) for the first extreme and (7.242, 1) for the second extreme. To find the middlemost solution, the weight is set to 0.8182, and the middlemost solution is found as (6.414, 2). Then, the left adjacent of this solution is found as (5.828, 5). The DM compares these two adjacent solutions and concludes that solution (5.828, 5) whose weighted objective is 5.7452 is better than solution (6.414, 2) whose weighted objective is 5.9726. The algorithm continues with the left adjacent of (5.828, 5) which turns out to be (5.242, 10). Now, (5.242, 10) whose weighted objective is 5.7178 is preferred to (5.828, 5) whose weighted objective is 5.7452. The best path between 1 and 3 is found as (5.242, 10). These two comparisons reduce the weight space between 0.8951 and 0.9999. Since all the solutions are preferred to their right adjacent solutions, the lower bound of the favorable weight space is updated. After this reduction, the algorithm finds only one efficient path between targets 1 and 4 as (10.484, 10), between targets 1 and 5 as (9.828, 1), between targets 2 and 3 as (4.828, 0). Between targets 2 and 4, the algorithm finds

two extreme solutions [(6.414, 11) and (7.242, 1)] that are adjacent to each other. The solution (7.242, 1) is preferred to its left adjacent (6.414, 11); therefore, the upper bound for the favorable weight space is updated to 0.9235. From this step until the end of the Part I of the interactive algorithm, only one path for each of the remaining target pairs is found. Best paths between targets 2 and 5 is (9.070, 11), targets 3 and 4 is (5.242, 0), targets 3 and 5 is (6.242, 1) and targets 4 and 5 is (4.414, 1).

After the computation of the best path between each target pair and reduction of the weight space to (0.8951, 0.9235), we solve the TSP. For the two extreme weights; 0.8951 and 0.9235, we compute the best tour. For the weight 0.8951, we find the tour 1-2-4-5-3 which is (30.968, 14), and for the weight 0.9235, we find the tour 1-2-3-4-5 with objective values (32.140, 3). There are not other supported efficient solutions in between these two solutions therefore the DM chooses from these two adjacent solutions. Since the weighted objective of solution 1-2-3-4-5 is smaller, he/she prefers this solution.

During the algorithm, we ask three questions in Part I and one question in Part II; four questions in total. With the first three questions, we manage to reduce the weight to a narrow range (0.8951, 0.9235) around the w_{best} ; 0.9000. In Part II, we do not reduce the weight space since we do not utilize from the answers after the computation of the best tour.

We also run the algorithm for w_{best} = 0.6000. Totally, three questions are asked to the DM and the favorable weight space is reduced to the range 0.5470 and 0.7072. In the end, only one tour turns out to be efficient within this favorable weight space; that is tour 1-2-3-4-5 with objective values 32.140 and 3 for total distance traveled and total radar detection threat, respectively.

In the route planning problem for UAVs, typically there are few targets to be visited. Therefore, the duration is not long in finding the best paths and the best tour of a DM. The second part; the solution of TSP generally takes longer time

compared to the first part, the MOSPP. However, we convert the MOTSP to a single objective TSP and solve the TSP with CONCORDE which can solve very large single objective TSP instances in short time intervals. Therefore, we reach the results of this problem in a short duration.

4.6.2. 8-Target MOTSP

In this problem, we approximate the terrain with 100×100 equidistant grids. We randomly place eight target points and 40 radar sites. The targets are placed at (20,27), (6,46), (52,41), (35,89), (41,17), (29,38), (59,57) and (77,43). The radar sites are at (12,45), (68,47), (71,42), (58,48), (48,35), (18,55), (43,42), (30,20), (24,74), (98,93), (55,29), (47,27), (50,20), (28,59), (53,54), (28,49), (11,32), (25,33), (31,24), (67,34), (43,22), (35,75), (63,40), (8,39), (8,70), (39,53), (66,53), (62,59), (49,12), (43,82), (30,30), (23,44), (41,30), (15,36), (52,68), (6,40), (74,51), (38,36), (46,57) and (56,79). The terrain structure can be seen in Appendix B, Figure B.1. We have a larger terrain in this problem; therefore, we assume that the UAV responsible for this terrain has a bigger size and similarly the radars have a larger power gain. We define the RCS as 5 square meters, G_t as 40 dB and the constant C as $2.27\text{x}10^{17}$ m^4.

In this problem, we did not derive the efficient frontier of the problem and we do not know the efficient paths between the nodes. This problem can be used to represent real world situation in detail; therefore, the duration of obtaining the answers is important.

We run the algorithm for this problem as mentioned in Section 4.6.1. We run the algorithm for many w_{best} values. Next, we give the steps of the algorithm when the best weight of the DM, w_{best} is defined as 0.5000. The computations and the reduced weight space are summarized in Table 4.4.

The best tour is 1-5-8-3-7-6-4-2-6 with a traveled distance of 280.572 units and a radar detection threat of 97.043.

Table 4.5 Computations and Reduced Weight Range for 8-Target MOTSP

Target Pair	Best Path		Weight Range
	z_1	z_2	
(1,2)	24.796	19.779	(0.3786, 0.9999)
(1,3)	37.796	27.507	(0.4408, 0.9999)
(1,4)	71.522	29.796	(0.4942, 0.5479)
(1,5)	35.382	1.777	(0.4942, 0.5124)
(1,6)	14.726	11.808	(0.4942, 0.5124)
(1,7)	52.592	31.153	(0.4942, 0.5124)
(1,8)	98.388	2.774	(0.4942, 0.5015)
(2,3)	50.554	32.520	(0.4942, 0.5015)
(2,4)	59.694	3.967	(0.4942, 0.5015)
(2,5)	69.006	10.275	(0.4942, 0.5015)
(2,6)	26.312	18.602	(0.4942, 0.5015)
(2,7)	70.280	13.180	(0.4942, 0.5015)
(2,8)	75.554	51.392	(0.4942, 0.5015)
(3,4)	55.866	19.573	(0.4942, 0.5015)
(3,5)	29.382	24.586	(0.4942, 0.5015)
(3,6)	24.242	15.315	(0.4942, 0.5015)
(3,7)	18.898	12.361	(0.4942, 0.5015)
(3,8)	25.828	19.988	(0.4942, 0.5015)
(4,5)	79.452	33.828	(0.4942, 0.5015)
(4,6)	56.796	18.019	(0.4942, 0.5015)
(4,7)	44.866	15.517	(0.4942, 0.5015)
(4,8)	82.662	2.876	(0.4942, 0.5015)
(5,6)	25.968	15.397	(0.4942, 0.5015)
(5,7)	47.452	37.457	(0.4942, 0.5015)
(5,8)	54.866	13.021	(0.4942, 0.5015)
(6,7)	37.866	19.345	(0.4942, 0.5015)
(6,8)	54.210	29.645	(0.4942, 0.5015)
(7,8)	23.796	21.802	(0.4942, 0.5015)

During the algorithm, we ask eight questions in Part I and no questions are needed in Part II. In the seventh target pair; i.e. targets 1 and 8; the weight space is reduced to a narrow range around the w_{best}; 0.5000. For the remaining target pairs, the algorithm finds only one solution within the reduced weight space range therefore we do not need to ask any questions to the DM. Also, we do not need to ask any questions to the DM in the second part of the problem; finding the best tour. The reduction of the weight space is quick. The reason for this is that we have many efficient paths between the target pairs under the presence of many radar sites. Therefore, to determine the best path to be followed, the DM needs to make most comparisons in the initial target pairs reducing the weight space to a narrow range around w_{best}.

This problem can be used in the representation of real world data; for instance this problem can be used to represent the terrain of a country. Therefore, the time is critical since the tour to be followed may be needed quickly. According to the runs we have made, the running time of the algorithm is approximately 100 seconds. We can say that, under a linear utility function assumption, finding the most preferred tour of a DM is done in a short time interval and this algorithm can be used for situations where the route to be followed is needed urgently.

4.7. Discussion on the Interactive Approach

In this chapter, we explain an interactive algorithm to be used for a linear utility function. As a part of this algorithm, we also propose an algorithm for finding the adjacent solutions of one solution.

In an interactive approach, we want to reach the most preferred solution of a DM asking few questions. In order to accomplish this, we use two properties of a linear utility function. The first one is that, the most preferred solution of a DM whose underlying utility function is linear is a supported efficient solution. Secondly, if a solution is preferred to its adjacent solutions, then it is the most

preferred solution. To use these two properties, we generate only the supported efficient solutions and ask for comparison only between adjacent solutions.

Route planning for UAVs is a combination of two combinatorial problems: MOSPP and MOTSP. In these two problems, the DM should make a number of decisions. In MOSPP, the DM should determine the best path between each target pair. After determining the best paths between each target pair, he/she should determine the best tour. On the whole, this makes at most $\sum_{i=1}^{N-1} i+1$ decisions to be made for $N-$target problem. There may seem many decisions to be made to reach the final solution; however, in each decision the DM makes; we learn more of his/her w_{best} and we reduce the search region for the following decisions to be made. Consequently, we ask fewer questions by learning the favorable weight space of the DM. For the problems where the DM faces many distinct comparisons, this algorithm is useful in both learning the relevant weight range and asking fewer questions in reaching the final answer. This also reduces the computation effort in finding the final solution each time by focusing on a smaller portion of the efficient frontier.

The first part of the problem; route planning for UAVs; is finding the best path between each target pair. Some computations may seem unnecessary in this part. For instance, one may think that if two targets are too distant, no TSP solution would make these two targets consecutive; therefore one should not try to find the best path between these two targets. Instead, a high cost value could be assigned between these two targets. However, two distant targets often are connected by many efficient paths. The DM usually makes comparisons in finding the best path between these two targets and we reduce the weight space based on these decisions. Therefore, a distant target pair may not help in finding the best tour of the DM, but may help in reducing the favorable weight space. This way, the questions we ask to the DM may decrease for the following iterations.

We consider two problems for the evaluation of the algorithm. The first problem has a small terrain with five targets. The second problem has a larger terrain that can be used in the representation of real world situations. We obtain the results in less than 2 minutes which is a short duration for the determination of the tour to be followed for real life situations.

We assume that the targets are fixed. If we had moving targets, we could start with a solution and update it dynamically after each visit to any target.

To sum up, we conclude that the interactive algorithm we propose is an efficient tool for finding the best solution of a DM whose underlying utility function is linear and this algorithm can be used in problems where the DM faces many distinct comparisons.

In the following chapter, we address the problem of finding efficient solutions. We propose an EA, a modification of NSGA-II that finds the potential efficient tours of the problem.

CHAPTER 5

AN EVOLUTIONARY ALGORITHM TO APPROXIMATE THE EFFICIENT FRONTIER

In Chapter 4, we find the most preferred paths and the most preferred tour of a DM whose underlying utility function is linear. However, for some cases we may need to generate the whole efficient frontier of the problem. In this chapter, we present an EA that finds the potential efficient tours, which eventually leads to the efficient frontier of the problem.

5.1. Evolutionary Algorithms

EAs have been inspired from natural evolution. In these algorithms, a population is maintained throughout the algorithm and this population gradually evolves to nondominated solutions. In the beginning of the algorithm, we start with an initial population. This population creates offspring by the genetic operator; crossover. Also, during offspring generation, some genes may undergo transformation and this is called mutation. As new offspring enter the population, an elimination process starts. With this process, the fittest population members survive and are maintained in the population and the worst solutions are discarded from the population. Through generations, the population moves towards the best solutions and finally the population contains only the nondominated solutions that form the efficient (Pareto-optimal) frontier.

In the literature, there are many EAs proposed. Deb (2001) argues that most of the multi-objective EAs are modifications of the single objective EAs. There are two aims of multi-objective EAs; converging to the Pareto-optimal frontier and

maintaining diversity among the population. In the literature, there are two types of multi-objective EAs, non-elitist multi-objective EAs and elitist multi-objective EAs. Elitist algorithms keep an elite preserving mechanism. In these algorithms, elite members; i.e. better solutions in terms of their performances in the objectives; of the population are sent to the next generation easier than the other solutions. On the contrary, in non-elitist multi-objective EAs, all the population members have the same probability of being carried over the next generation. Deb (2001) notifies the importance of elitism towards reaching the nondominated solutions, therefore we modified one of the most commonly used elitist algorithm; NSGA-II.

5.2. Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II)

NSGA-II is suggested by Deb et al. in 2000. This algorithm gives importance to the solutions that are better than the other solutions by introducing an elitism preserving mechanism. Meanwhile, it maintains diversity in the population. This way, the algorithm satisfies the two objectives of a multi-objective evolutionary algorithm; converging to the Pareto-optimal frontier and maintaining diversity among the solutions.

NSGA-II starts with an initial population of size N. All the members in the population undergo crossover to have offspring population of size N. Then, the two populations; the parent population and the offspring population; are combined to have a global population of size 2N. The global population is sorted in fronts according to domination. This front separation for solutions is referred to as ranking. The solutions from the best fronts are selected to the next population until the population size becomes N. Figure 5.1 shows the front generation for the NSGA-II procedure.

The solutions in one front are dominated by the solutions in a better front. The first front in Figure 5.1 is made up of the nondominated solutions. The solutions in the second front are dominated only by the first front. The solutions that are

dominated by both the first and the second front are grouped in the last front. The solutions in the first front have rank one, the solutions in the second front have rank two etc.

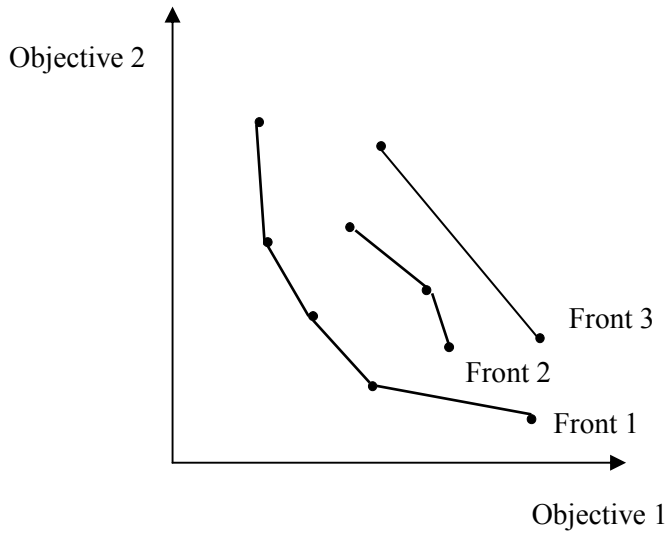


Figure 5.1 Front Separation for Solutions in a Population

In a global population with 10 solutions like in Figure 5.1, we would choose five solutions for the next generation. Since we give priority to the elite solutions, we would choose the five solutions in the first front. However, if we had to choose four solutions for the next generation, we would choose the diversely distributed four solutions in the first front based on some diversity measure. This is the niching mechanism that chooses a subset of solutions among the solutions in one front based on their crowdedness. If a solution is in a crowded region, then it would not be preferred to a solution that is in a least crowded region. The crowding distance is a measure that shows distance of the closest solutions to one solution.

The algorithm continues with the new population. The parents that produce offspring population are selected by crowded tournament operator. In this operator, two solutions that are candidates for the first parent are compared with

respect to their fronts and their crowdedness. If one solution is in a better front, then it is chosen as the first parent. If both of the candidates are in the same front, the solution that is in a less crowded region is selected as the first parent. The second parent is selected the same way. The algorithm continues producing offspring population with crossover and mutation until a predefined condition is met.

In our algorithm, we modified NSGA-II. In Section 5.3, we describe our algorithm and the changes we applied to NSGA-II.

5.3. The Modified NSGA-II

In route planning with multiple targets, there may be many efficient tours; each tour using different efficient paths between consecutive targets. In our algorithm, we want to find the tours that can be efficient. The initial step is the approximation of the efficient frontiers of the tours and the algorithm compares these efficient frontiers. As a result, the algorithm gives the tours that contain nondominated solutions.

5.3.1. Representation

The solutions in our population represent the order of visit to the targets. Larranaga et al. (1999) suggest that in the literature on TSP solutions by EAs, there are many representation schemes. The most straightforward representation is the *path representation*. In this scheme, if node *i* is the *j*th node to be visited, it is placed in the *j*th position in the chromosome. For instance, the solution 1-5-4-2-3 simply shows that node 1 is visited first, then node 5 is visited and so on. We use path representation in our algorithm.

5.3.2. The Modifications on NSGA-II

The steps of NSGA-II are not changed in this study; however, certain modifications are done to make the algorithm work for MOTSP with multiple efficient paths between nodes. Each step of the algorithm and the modifications are outlined below.

Initialization

In the initialization step, we generate the solutions in the initial population of size N. We refer to each solution as chromosome. The chromosomes represent which target order we should follow. Each bit of the chromosome shows the target visited at that time; therefore in each chromosome all targets should be present, and they should be represented only once. To have this structure in the chromosome, random numbers are generated for each target in a chromosome. Then these random numbers are sorted in ascending order. The rank of the target gives its order of visit.

Evaluation of the Population

We need to determine the objective values; distance and radar detection threat for each of the chromosome generated in the initialization. However, it is not straightforward since one tour can be composed of many different paths leading to many different objective values. To account for each tour, we make an approximation of the efficient frontier of that tour. Köksalan and Soylu (2007) proposes an Lq function-based approach for the approximation of a frontier. In this approach, a function passing through some efficient points of a frontier is used to approximate that frontier.

The Lq function is as follows;

$$(1-zf_1^2)^q + (1-zf_2^2)^q = 1 \tag{5.1}$$

where;

$zf^j = (zf_1^j, zf_2^j) = \left(\frac{z_1^j - z_1^1}{z_1^3 - z_1^1}, \frac{z_2^j - z_2^3}{z_2^1 - z_2^3} \right)$ is the normalized objective function values for the j th point. (Köksalan and Soylu, 2007)

z_k^j: the k th objective function value of the j th solution

j = 1: left extreme solution

j = 2: the third efficient solution

j = 3: right extreme solution

In order to fit the Lq function, we need to find the value of q. However, since the computation of q is not straightforward, binary search is used to approximate it. For a given q and one objective function value of a solution, the other objective function value could be computed by substituting each value in equation (5.1).

In Figure 5.2, we approximate the efficient frontier of an example tour that passes through three efficient solutions; A, B and C. The structure of the Lq function is very close to the original efficient frontier. After fitting the Lq function, equi-spaced dummy points are taken on the function to represent the tour as in Köksalan (1999). In Figure 5.3, the equi-spaced point generation for the Lq function in Figure 5.2 is shown. In this figure, we select five equi-spaced points on the Lq function to represent that tour. The two extreme points A and C are included in the representative points; however, solution B is not included.

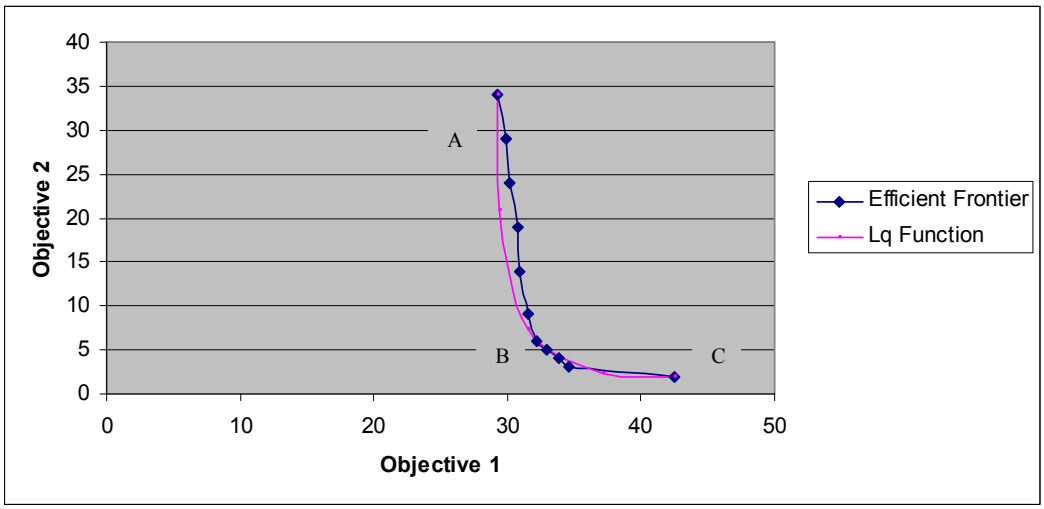


Figure 5.2 Lq Function Approximation for Efficient Frontier of a Tour

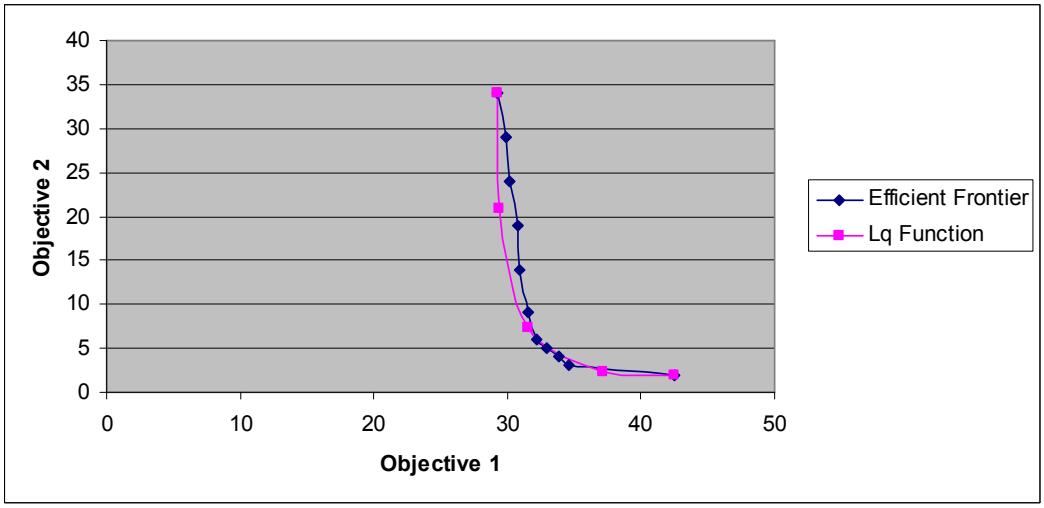


Figure 5.3 Equi-spaced Point Generation on Lq Function

Both the initial population number and the number of equi-spaced points are user defined. After this point, we have (initial population number × equi-spaced point number) solutions in our population. Each solution is represented by the objective function values corresponding to its position on the Lq function.

In our approach, before starting the algorithm, we prepare the input necessary for the computation of the three efficient points. We combine the two objectives linearly as $U(z) = wz_1(x) + (1-w)z_2(x)$ where $z_1(x)$ is the first objective, $z_2(x)$ is the second objective and $0 \le w \le 1$. For each target pair, we compute the shortest paths by minimizing the composite objective $U(z)$ for different values of w. We compute shortest paths between each target pair using w values ε, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1-ε (here ε is a very small number). The computation of the extreme points is straightforward with these inputs. For the left extreme solution of any tour, the shortest paths computed with w=1-ε are summed up for the target pairs that make up that tour. Similarly, for the right extreme solution, the shortest paths computed with w=ε are summed up. For the third efficient solution, we use the middlemost solution. After the computation of the two extreme points, we derive the function that passes through these points.

For this function, the w' can be found as follows;

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

where;

x_L : left extreme solution

x_R : right extreme solution

$z_i(x)$: the ith objective value of solution x

Then we find the w value closest to w' from the set ε, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1-ε. We use the shortest paths found with the chosen w as the third point. After having these three efficient solutions, we derive the Lq function.

The next step is the selection of the representative points on this Lq function. There is one point that is important in the selection of the representative points. These points should be as close to the real efficient solutions as possible for a better approximation of the frontier of a tour. Therefore, we include also the

middlemost solution that we used in the approximation of the Lq function in our representative solutions since it is already an efficient solution. We make a change in the representative point selection such that this third solution replaces one of the representative solutions. After the derivation of the whole representative points, we select the representative point that is closest to the middlemost solution in terms of one objective and replace that representative solution with the middlemost solution.

Using approximate efficient solutions of the tours has many advantages compared to using the real efficient solutions of these tours. Firstly, the tours that can be efficient are found without evaluating all the efficient paths between each target in that tour. This way, some tours are eliminated from consideration. For a problem with 10 targets, there are 181,440 ((10-1)!/2) different tours. If each consecutive target pair in each tour contains two efficient paths, there are 185,794,560 ($181{,}440 \times 2^{10}$) different alternatives. Finding all of the alternatives and evaluating each is computationally hard and most of the time it is unnecessary. However, with this approach we consider only three efficient solutions of one tour and approximate the remaining efficient solutions with some pseudo solutions. If a tour turns out to be inefficient, it is not evaluated even from the initial generations. Consequently, we would not need to compute all the efficient solutions of an inefficient tour.

Secondly, we use 11 weights to find a set of efficient paths between each target pair. These are computationally easy problems. When some tours are found to be inefficient and discarded from further evaluation, so are the paths between some targets. For example, if the targets i and j are not consecutive in any of the resulting potential efficient tours, then there is no need to determine all the efficient paths between these targets.

As a result, there is no need for the determination of all the efficient paths before the program execution and this saves computational time.

Selection

Until this step, we have a population of size (initial population number × equi-spaced point number). We refer to this population as the secondary population. In the solution of this problem, we need to determine a secondary population since we cannot evaluate the objective values of the solutions in the initial population. After creating the representative points for each of the solutions in the population, we find their objective function values. The next step is finding the ranks of each solution. The objective function values are used in ranking the solutions in the secondary population. If we let the number of equi-spaced points to be q, in the secondary population we have N × q solutions. For a solution in the initial population, we have q representatives in the secondary population with different objective values and consequently different ranks. However, since we perform all the operations in the initial population, we should have totally N solutions. To reduce the number of solutions from N × q to N, for each solution in the initial population we select its best representative point since we do not want to lose a tour in the following generations if that tour can yield efficient solutions. Then, we have N solutions in the initial population each represented by its best representative point. In choosing the best representative point, we consider their ranks. If more than one solution has the same best rank, we choose one of them randomly. As a result, we obtain a population whose size is the same as the initial population's size. All solutions in this population are represented with their best representative point.

The next step is the selection of the mating pool that is to be used in producing the following generation. For the mating pool, we need to select N solutions. Two solutions are randomly selected from the initial population and they are evaluated by their ranks. If their ranks are equal, they are evaluated by their crowding distance values. The one with a lower rank; or with same rank but with higher crowding distance is selected as the first parent in the mating pool. The other parents are also chosen in the same way.

Genetic Operators

- Crossover

The genetic operators for TSP are generally representation specific. For path representation, there are many crossover operators in the literature. For a review of the operators, the reader is referred to Larranaga et al. (1999). One of the crossover operators used for TSP is "position based crossover" (Larranaga et al., 1999). In POS, the offspring takes the order of visits from both parents. For any two parents, some positions are randomly selected. The nodes visited at those positions are directly passed to the offspring (from parent 1 to offspring 2, from parent 2 to offspring 1). The maximum number of positions that can be selected is previously defined by the user; and each position is selected with crossover probability, pcross. The remaining bits of the chromosome for offspring i (i=1,2) are filled by parent i (i=1,2). For example, let parent 1 be (2-4-3-5-1) and parent 2 be (5-1-2-4-3). Let pcross be 1. The positions 1, 3 and 4 are selected randomly; so offspring 1 becomes (5-*-2-4-*) and offspring 2 becomes (2-*-3-5-*). Then offspring 1 is filled by the parent 1 as; (5-3-2-4-1) and offspring 2 becomes (2-1-3-5-4). This operator takes the order of visit from both parents; therefore it is used in our solution approach.

- Mutation

The mutation operator used is also specific to path representation. To keep feasibility, two nodes are randomly selected and their places are exchanged with mutation probability (pmut). This is the exchange mutation operator (Larranaga et al., 1999).

Evaluation of the Offspring

The evaluation of the offspring is similar to the evaluation of the population outlined above. For each offspring, the Lq function is derived and new solutions

are generated with new objective function values. Then, the representative point which has the best rank or both best rank and best crowding distance is selected as the original solution. The algorithm continues this way.

Termination

The algorithm stops when a predetermined number of generations are produced.

We applied the modifications to the C code provided in the website of Deb Kalyanmoy, http://www.iitk.ac.in/kangal/deb.shtml

In Section 5.4., we define the steps to be followed for the generation of the efficient frontier. In Section 5.5, we give simulation results of two problems with different specifications.

5.4. Further Computations

In the EA we propose, we find the potential efficient tours. In order to generate the whole efficient frontier, we need to apply the following steps:

Step 1. Determine all the consecutive target pairs that are included in the potential efficient tours.

Step 2. Find the efficient paths between each consecutive pair using a multi-objective shortest path algorithm.

Step 3. Enumerate all the efficient paths between the targets. For each potential efficient tour, generate its efficient frontier by an assignment procedure; i.e. select the path to be used between each consecutive target pair.

Step 4. Combine all the efficient frontiers of the potential efficient tours. Select the solutions that are nondominated from the solution set. These solutions make the efficient frontier of the overall problem.

After the execution of the EA, we follow a few steps to obtain the efficient frontier of the problem. Generating the whole efficient frontier is computationally easy after finding the potential efficient tours. In Step 1, we determine all the consecutive target pairs in the potential efficient tours. This means that some target pairs will not be considered if they are not present in any of the potential efficient tours. For a problem with $N-$ nodes, if we compute all the efficient shortest paths between all node pairs, we make $\sum_{i=1}^{N-1} i$ computations. However, in some problems, some nodes may never be consecutive. Finding the order of visit in the first place helps in decreasing the number of target pairs to be evaluated. In Step 2, we find the efficient paths between all consecutive target pairs using any of the multi-objective shortest path algorithms discussed in Chapter 2. In Step 3 we find the paths to be used for a given tour using an assignment procedure. We know the consecutive targets to be visited, and we choose the path to be used between each target pair. This turns to be an assignment problem. We can use ε-constraint method to sequentially generate the efficient frontier of the tour. In ε-constraint method, one objective is treated as a constraint instead of an objective and a certain upper limit is determined for that objective. The objective that is not treated as a constraint is minimized while the other objective is restricted to values below that upper limit. In each time the problem is solved, the upper limit is relaxed by a small amount. The problem gives each time one efficient solution. We continue solving the problem until we do not obtain a new efficient solution. The whole efficient solutions make the efficient frontier of the problem. In Step 4, we combine all the efficient frontiers of the potential efficient tours. From the whole solutions, we select the ones that are nondominated. These solutions make the efficient frontier of the overall problem.

5.5. Simulation Results

In this section, we give simulation results on how the algorithm works. The first problem mentioned in Section 4.6.1 and the second problem mentioned in Section 4.6.2 is solved in parts 1 and 2 using the EA, respectively.

5.5.1. 5-Target MOTSP with Simplified Radar Threat Values

In this problem, the terrain is divided into 10×10 equidistant grids. There are five targets randomly placed on the terrain and three radar sites. For calculating the radar threat detection, we assigned values to each node inside the radar sites. We again sum up the radar detection threat of the nodes that are on the path of the UAV. The terrain can be seen in Figure 4.4 and for more detailed explanation on the problem; the reader is referred to Section 4.6.1.

In this problem, we have (5-1)!/2 = 12 different tours. The efficient frontiers of each tour are shown in Figure 5.4. It can be seen from the figure that 9 of the tours are inefficient; i.e. none of their solutions lie in the efficient frontier. There are only three tours that are efficient. The overall efficient frontier of the problem can be seen in Figure 4.5.

The algorithm is run for different number of equi-spaced points and initial population sizes. The crossover probability is taken as 0.9 and the mutation probability is taken as 0.1. The maximum number of crossover positions is taken as 3.

We run the algorithm for 30000 generations. Each time we run the algorithm, we change the total population size by changing both the initial population size and the equi-spaced point number. We tried three different ways of choosing the initial population size and number of equi-spaced points. These alternatives are listed in Table 5.1.

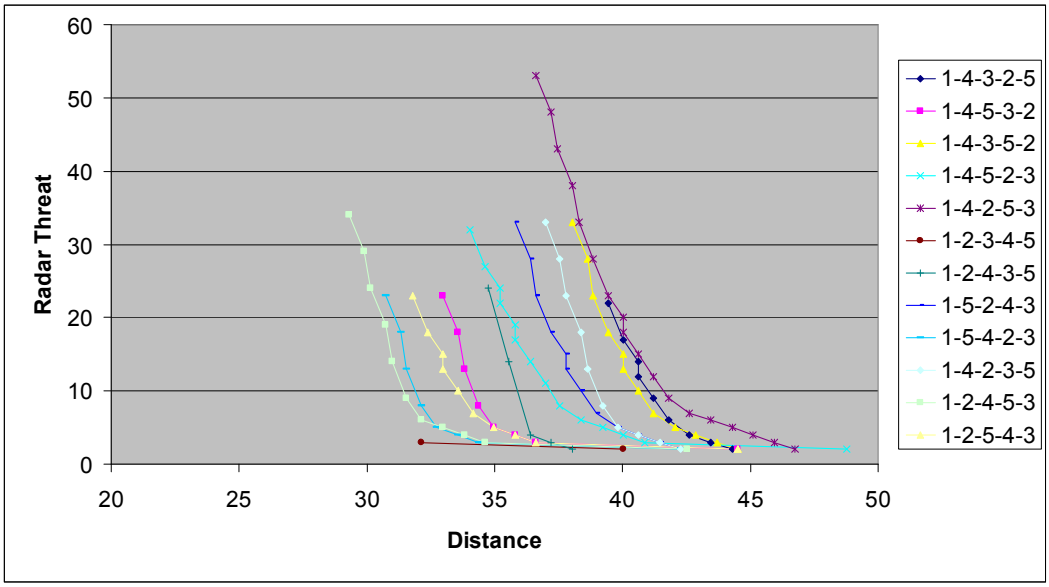


Figure 5.4 The Efficient Frontiers of Each Tour of 5-Target MOTSP (Simplified Radar Threat Values)

Table 5.1 Alternatives of Solution for 5-Target MOTSP (Simplified Radar Threat Values)

Alternative	**1**	**2**	**3**
Initial population size	10	15	30
No. of equi-spaced points	10	15	8
Total population size	100	225	240

In this problem, we expect to obtain the three efficient tours; 1-2-4-5-3, 1-2-3-4-5 and 1-2-4-3-5; at the end of the algorithm. When we run the algorithm setting both initial population size and number of equi-spaced points to 10, we obtain four potential efficient tours. We obtain the three efficient tours; 1-2-4-5-3, 1-2-3-4-5 and 1-2-4-3-5 and one more potential efficient tour; 1-3-2-4-5. The objective function values of these tours can be seen in Figure 5.5.

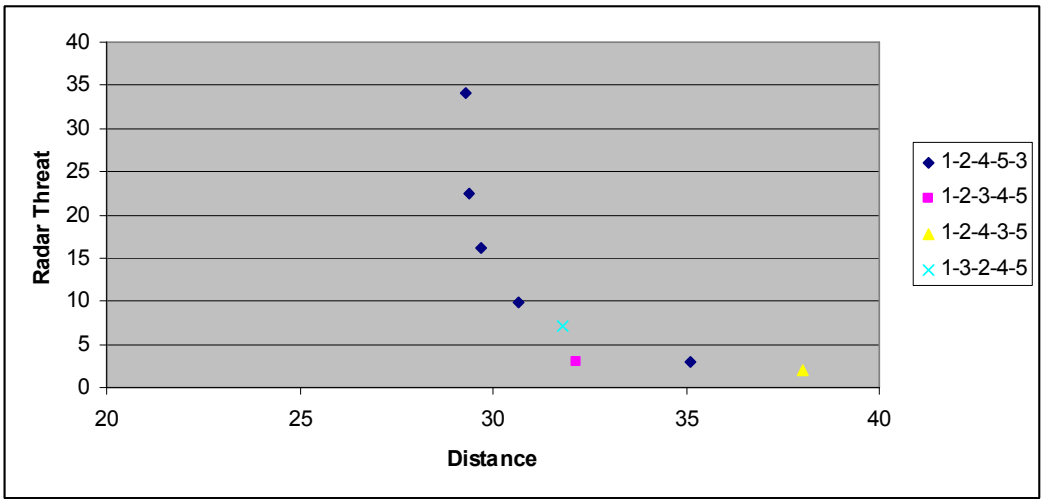


Figure 5.5 The Efficient Frontier – Alternative 1 (5-Target MOTSP - Simplified Radar Threat Values)

For the second alternative, we set both the initial population size and number of equi-spaced points to 15. We again find four potential efficient tours like alternative 1, however we obtain more solutions from the potential efficient tour 1-3-2-4-5 when compared to alternative 1. The reason is that, we are increasing the number of intermediate solutions in alternative 2, therefore we obtain more dummy solutions. More of these dummy solutions yield more efficient solutions. The objective function values of the solutions and their spread can be seen in Figure 5.6.

In the third alternative, we decrease the number of intermediate solutions to 8 and increase the initial population size to 30. The results are similar to the previous alternatives. We again obtain the same four potential efficient tours. The objective function values of these tours can be seen in Figure 5.7.

For the three alternatives, we obtain the three real efficient tours; 1-2-4-5-3, 1-2-3-4-5 and 1-2-4-3-5 and this is a promising result. However, for each alternative we also obtain the inefficient tour 1-3-2-4-5 that seems efficient. To have a closer look at why this tour can be efficient, we draw the tour 1-3-2-4-5 vs. the efficient

frontier of the problem in Figure 5.8. The efficient frontier of tour 1-3-2-4-5 is very close to the original efficient frontier. Since we cannot find all of the efficient solutions of each tour, we cannot find all the solutions that dominate the tour 1-3-2-4-5. Therefore, this tour may seem efficient.

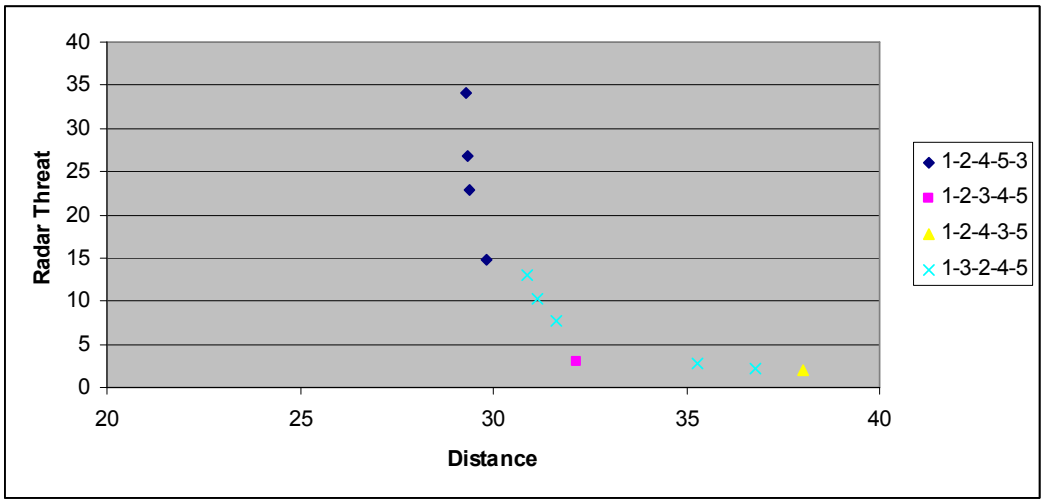


Figure 5.6 The Efficient Frontier – Alternative 2 (5-Target MOTSP - Simplified Radar Threat Values)

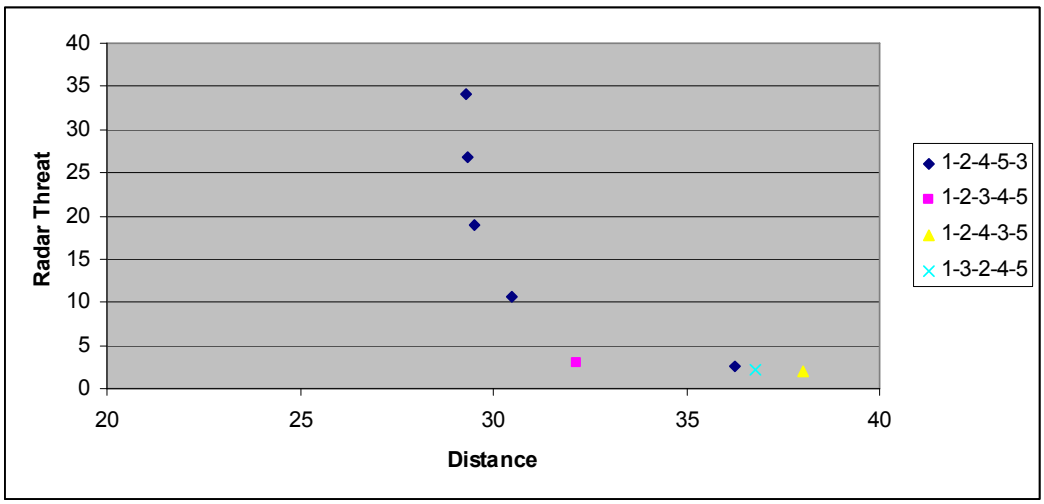


Figure 5.7 The Efficient Frontier – Alternative 3 (5-Target MOTSP - Simplified Radar Threat Values)

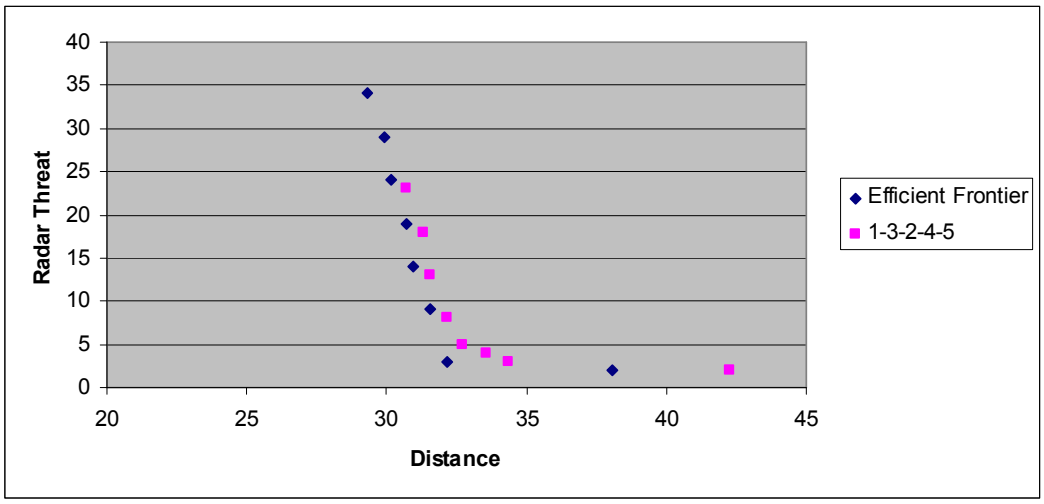


Figure 5.8 Tour 1-3-2-4-5 and the Efficient Frontier of 5-Target MOTSP (Simplified Radar Threat Values)

The 10×10 terrain with five targets with calculated radar threat values is also solved using the EA. The details of the computations are given in Appendix C.

5.5.2. 8-Target MOTSP

The second problem that we test our algorithm on is the 8-Target MOTSP that is defined in Section 4.6.2. In this problem, the terrain is approximated with 100×100 equidistant grids. Eight targets and 40 radar sites are randomly placed on the terrain. The structure of the terrain can be seen in Appendix B, Figure B.1.

We made preliminary runs to determine the most suitable pair of initial population size and number of equi-spaced points. We decided to continue with the results of initial population of size 25 and number of equi-spaced points as 10 since the results of this case were favorable. However, more detailed analysis could be conducted to obtain the best combination of the pair of initial population size and number of equi-spaced points. We test this problem with five different seeds.

In Table 5.2, we show the tours that are considered as efficient for each of the five seeds. The check marks indicate that the tour is efficient with respect to that seed. Totally there are 11 tours that seem efficient. Only the third seed misses two potential efficient tours compared to other seeds. However, for the generation of the efficient frontier, all of the tours that are produced as a result of each of the seeds should be checked to see whether they are really efficient or not.

Table 5.2 Potential Efficient Tours of Each Seed for 8-Target MOTSP

	Seed				
Tours	**1**	**2**	**3**	**4**	**5**
1-2-4-7-8-3-5-6	√	√	√	√	√
1-2-4-7-8-3-6-5	√	√	√	√	√
1-2-4-7-6-3-8-5	√	√	√	√	√
1-2-4-8-7-3-6-5	√	√	√	√	√
1-5-2-4-8-7-3-6	√	√	√	√	√
1-5-3-8-7-4-2-6	√	√	√	√	√
1-5-4-2-6-3-7-8	√	√	√	√	√
1-5-7-3-6-2-4-8	√	√		√	√
1-5-8-3-7-4-2-6	√	√	√	√	√
1-5-8-4-2-7-3-6	√	√	√	√	√
1-5-8-2-4-7-3-6	√	√		√	√

We do not know the efficient frontier of the problem. Therefore, to generate the efficient frontier we follow the steps outlined in Section 5.4. We find the efficient paths between the consecutive targets and generate the efficient frontiers of the potentially efficient tours. Combining all the efficient frontiers of these tours, we find the overall approximate efficient frontier. The objective function values of the efficient tours can be seen in Figure 5.9. Among the 11 potential efficient tours, 10 turn out to be efficient with respect to each other. Only one tour (1-2-4-7-6-3-8-5) is inefficient with respect to some of the other tours.

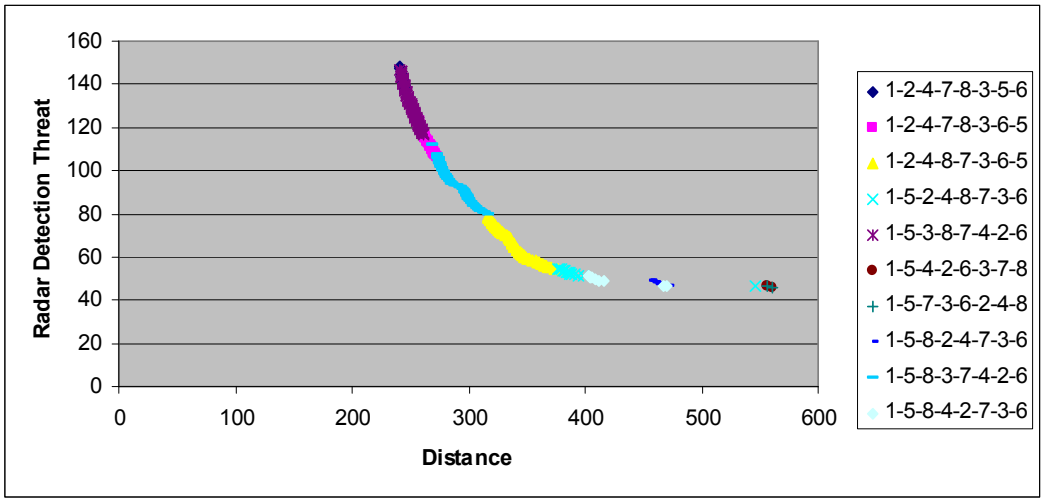


Figure 5.9 Efficient Solutions of 8-Target MOTSP

In order to check whether we obtain the supported efficient tours, we generate all the supported efficient tours. These tours are listed in Table 5.3. The distribution of the supported efficient tours can be seen in Figure 5.10. All of the eight supported efficient tours are found by the EA.

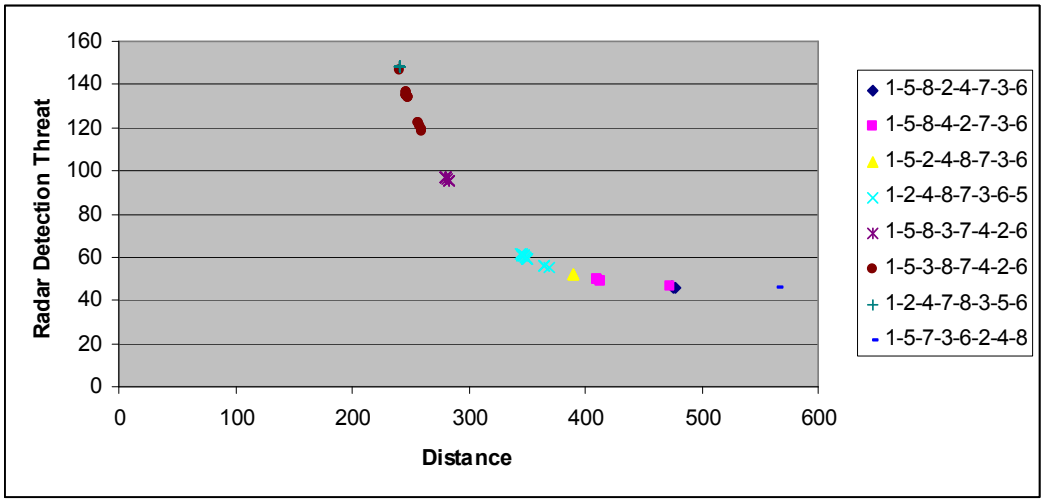


Figure 5.10 Supported Efficient Solutions of 8-Target MOTSP

Table 5.3 Supported Efficient Tours of 8-Target MOTSP

Tours	Distance	Radar Detection Threat
1-5-7-3-6-2-4-8	564.430	45.945
1-5-8-2-4-7-3-6	477.628	46.053
	476.456	46.068
	474.214	46.116
1-5-8-4-2-7-3-6	472.800	46.163
	413.794	49.081
	410.482	49.329
1-5-2-4-8-7-3-6	389.170	51.917
1-2-4-8-7-3-6-5	368.788	55.323
	365.374	55.957
	350.992	59.270
	350.164	59.485
	348.508	60.002
	347.336	60.403
	346.750	60.646
	346.164	60.897
	345.336	61.318
	344.508	61.764
1-5-8-3-7-4-2-6	283.158	95.295
	282.330	95.839
	280.572	97.043
1-5-3-8-7-4-2-6	259.986	118.049
	259.158	118.919
	257.744	120.549
	256.572	121.933
	247.744	133.631
	247.158	134.506
	246.572	135.567
	245.986	136.696

Table 5.4 Supported Efficient Tours of 8-Target MOTSP (Continued)

Tours	Distance	Radar Detection Threat
1-5-3-8-7-4-2-6	241.298	146.907
1-2-4-7-8-3-5-6	240.610	148.444

5.6. Discussion on the Evolutionary Algorithm

In this chapter, the elitist genetic algorithm NSGA-II is modified to solve the MOTSP with multiple efficient paths between nodes. In this problem, there exist two questions to be answered: what should be the order of visits to nodes and which path should be used in between the nodes. This algorithm answers the first question and approximates the tours that can yield efficient solutions.

After determining the possible efficient tours, the steps in Section 5.4 can be followed to generate the whole efficient frontier of the problem.

Two problems are solved using this approach. The first problem is a 5-target MOTSP whose efficient frontier is known. The results seem promising as all of the settings find all the tours that are efficient. The extreme points are not lost in any of the settings for the two problems, but intermediate solutions are generated with varying parameters; initial population size and number of equi-spaced points. In the first problem we consider, all the settings result in the true efficient tours and one additional inefficient tour. In the second problem, we try to obtain all the efficient tours for a problem whose efficient frontier is unknown. To obtain the efficient frontier of the problem, all of the potential tours should be considered.

It is not straightforward to select the best pair of initial population size and number of equi-spaced points. If we increase the number of equi-spaced points, we obtain many dummy solutions which have the possibility of dominating real efficient solutions. However, if we do not have sufficient equi-spaced points, we

may not wholly represent the tour. Therefore, many combinations of initial population size and number of equi-spaced points should be tried and all of the results of these combinations should be considered.

The algorithm may show inefficient solutions as efficient. This is because we are approximating each tour with dummy solutions that may not exist and we are not generating all the efficient solutions of each tour. The dummy solutions may appear as efficient even they do not exist. Also, the absence of any of the efficient solutions may make other inefficient solutions seem efficient if these inefficient solutions are only dominated by that efficient solution. However, this should not be considered as a drawback of the algorithm. During the generation of the efficient frontier of the whole problem as depicted in Section 5.4, we combine all the efficient frontiers of the potential efficient tours. The tours that are inefficient would not be present in the final efficient frontier; therefore having these inefficient tours in the potential efficient tour set would not change the results of the algorithm.

This approach considerably decreases the problem size. For the first problem, among the 12 possible tours at most four are shown as efficient. For the second problem, 11 tours seem efficient in 2520 tours. The proportion of tours to be considered decreases significantly. Secondly, there is no need to calculate all the efficient paths between the nodes. For example, if node i and j are not consecutively visited in any of the efficient paths, then we do not need to find the efficient paths between these nodes. Also, in the algorithm if the efficient paths are not given as input to the program, solving for only three solutions is sufficient for each of the node pairs. The two solutions are the extreme efficient solutions and the third solution is found by minimizing the function that passes through the two extreme solutions. The weight for the first objective can be found by the equation (5.1) and single objective shortest path algorithms can be used in finding these three solutions. Again, there is no need to calculate all of the efficient paths between the nodes during the program execution. The same procedure can also be applied with 10 input matrices; 5 distance and 5 radar threat matrices. These

weights partition the [0,1] weight interval into four equidistant parts as follows; [1-ε, ε], [0.75, 0.25], [0.5, 0.5], [0.25, 0.75] and [ε, 1-ε].

As a future study, the algorithm should be extended to solve the second question; i.e. which efficient path to use. Another algorithm can be run for the resulting efficient tours to find which paths can be used between the targets. Another alternative solution for the second part of the problem is using approximate algorithms for determining the efficient paths between the nodes. For large-scale problems, in order to find the efficient paths between the target pairs, the terrain can be approximated more loosely or smaller regions can be defined that include the two targets of interest. As an alternative, the properties of the problem can be used to apply exact algorithms in finding the efficient paths between targets.

The algorithm is yet only evaluated by how well it approximates the frontier and which tours it proposes as efficient. Since the aim of the algorithm is to give the efficient tours, these suffice. However, if the algorithm is extended, different performance measures could be used to evaluate the final solutions. One performance measure used in the literature is hypervolume metric (Zitzler and Thiele, 1998). This metric measures the total volume dominated by the nondominated solutions in the final population with respect to a reference point. When problems with known efficient frontiers are solved using the EA, the results could be compared with the original efficient frontiers using different performance measures like the hypervolume metric.

CHAPTER 6

CONCLUSIONS

Route planning for UAV can be defined as finding the "best" route that the UAV should follow through a defended terrain. Under the presence of multiple targets, the UAV should visit all the targets and return to the base. Route planning with multiple targets is a combination of two combinatorial problems. Determining the efficient paths between the targets is a MOSPP. Using these efficient paths, finding the best visiting order to targets is a MOTSP. In the literature for the MOTSP, it is assumed that all the targets have only one connection. However, in this problem the targets have multiple connections. Therefore, the solution to this problem should answer two questions; "Which visiting order of targets should be followed?" and "Which path should be used between the consecutive targets?".

We propose two algorithms for the MOTSP with multiple efficient paths between nodes. The first algorithm is an exact interactive algorithm that is used to find the most preferred paths and the most preferred tour of a DM. We assume the DM has a linear utility function. The algorithm is demonstrated on two different problems. As a future research, the interactive algorithm can be modified to handle more general utility functions.

The second algorithm is an EA which is a modification of NSGA-II. In this algorithm, we find the potential efficient tours. Using the information we obtain from the results, we generate the whole efficient frontier of the problem. This algorithm is demonstrated on two problem instances.

As a future research, the EA can be extended to compute also the efficient paths to be used between the targets within a tour. Knowing the efficient paths between the consecutive targets in the potential efficient tours helps in finding the overall efficient frontier of the problem. As a result, the answers to the two questions could be obtained; “Which visiting order should be followed?” and “Which path should be used between the consecutive targets?”.

The UAV route planning problem can be improved to solve different real life problems. The terrain can be modeled to represent the obstacles; like mountains; that the UAV cannot pass through. In addition, the problem can also be defined in a different way. The UAV may not have the chance to visit all the targets. Then the problem turns to a MOTSP with profits. In this problem, one objective becomes maximization of the total benefit that is gained from each target visited. For some missions, the MOTSP with profits representation would be more appropriate. Another problem can be defined when some targets need immediate visit and some targets do not have a restriction on their visit times. This makes the problem harder and heuristics can be developed for the solution.

We consider the biobjective route planning problem. However, there may be more objectives in the determination of the route to be followed. In our approach, we consider a constant speed throughout the flight. A more detailed analysis on the dependence on speed can be conducted.

In conclusion, we have redefined the MOTSP that has not been considered in the literature yet. We proposed two algorithms that can be used in the solution. One algorithm finds the most preferred paths and most preferred tour of a DM. The second algorithm which is an EA, results in the potential efficient tours. This information can be used in the generation of the whole efficient frontier.

REFERENCES

Aneja Y.P., Nair K.P.K., (1979) *Bicriteria Transportation Problem.* Management Science, **25** (1), 73-78.

Berube J.F., Gendreau M., Potvin J.Y., (2009) *An Exact ε-constraint Method for Bi-objective Combinatorial Optimization Problems – Application to the Traveling Salesman Problem with Profits*. European Journal of Operational Research **194** (1), 39-50.

Cohon, J.L., (1978) *Multiobjective Programming and Planning*, 127-140, Academic Press, New York.

Deb, K. (2001) *Multi-Objective Optimization using Evolutionary Algorithms.* Wiley, Chichester, UK

Ehrgott M. (2000) *Multi Objective Optimization, Chapter 7. Combinatorial Problems with Multiple Objectives, The Travelling Salesman Problem*, 211-220, Wiley.

Gandibleux X., Beugnies F., Randriamasy S., (2006) *Martins' Algorithm Revisited for Multi-objective Shortest Path Problems with a MaxMin Cost Function,* A Quarterly Journal of Operations Research **4** (1), 47-59.

Gabrel V., Vanderpooten D., (2002) *Enumaration And Interactive Selection of Efficient Paths in a Multiple Criteria Graph for Scheduling an Earth Observing Satellite.* European Journal of Operational Research **139**, 533-542.

Georgia Institute of Technology (2009), Concorde TSP Solver. http://www.tsp.gatech.edu/concorde

Granat J., Guerriero F., (2003) *The Interactive Analysis of the Multicriteria Shortest Path Problem by the Reference Point Method.* European Journal of Operational Research **151,** 103-118.

Gudaitis M.S. (1994) *Multi criteria Mission Route Planning Using a Parallel A* Search.* MS thesis, Air Force Institute of Technology (AU).

Hansen M.P., (2000) *Use of Substitute Scalarizing Functions to Guide a Local Search Based Heuristic: The Case of moTSP.* Journal of Heuristics **6**, 419-413.

Karademir S., (2008) *A Genetic Algorithm for the Biobjective Traveling Salesman Problem with Profits*. M.S. thesis, Department of Industrial Engineering, Middle East Technical University.

Köksalan, (1999) *A Heuristic Approach to Bicriteria Scheduling, Naval Research Logistics* **46**, 777-789.

Köksalan, M. and Soylu, B., (2007) *Bicriteria p-hub Location Problem and Evolutionary Algorithms*, Technical Report, IE, METU.

Larranaga, P., Kuijpers C.M.H., Murga R.H., Inza I. and Dizdarevic S., (1999) *Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators*. Artificial Intelligence Review **13**, 129-170.

Olsan J.B., (1993) *Genetic Algorithms Applied to a Mission Routing Problem.* MS thesis, Air Force Institute of Technology (AU).

Raith A, Ehrgott M., (2009) *A Comparison of Solution Strategies for Biobjective Shortest Path Problems*, Computers and Operations Research **36** (4), 1299-1331.

Skolnik M., (1980) *Introduction to Radar Systems*. *Chapter 2. The Radar Equation*, 15-67, McGraw-Hill.

Skriver A.J.V, Andersen K.A., (2000) *A Label Correcting Approach for Solving Bicriterion Shortest Path Problems.* Computers and Operations Research **27**, 507-524.

Winston W.L., (2004) *Operations Research - Applications and Algorithms. Chapter 8, Network Models, Shortest Path Problems*, 414-419, Duxbury Press Cal., 4th edition.

Yavuz K., (2002) *Mission Route Planning Using Particle Swarm Optimization.* M.S thesis, Air Force Institute of Technology (AU).

Zionts, S., (1981) *A Multiple Criteria Method for Choosing Among Discrete Alternatives*. European J. Oper. Res., **7** (1), 143-147.

Zitzler, E., Thiele, L. (1998) *Multiobjective Optimization Using Evolutionary Algorithms – A Comparative Case Study*. Parallel Problem Solving from Nature - V., 292-301.

APPENDIX A

ALGORITHM FOR THE SOLUTION OF 5-TARGET MOTSP WITH CALCULATED RADAR THREAT VALUES

In this problem, we have five targets and three radar sites. In order to estimate the radar detection threat, we assume that the radar operates at 10 GHz; i.e. x-band. The values of the variables in the equation (3.2) is as follows;

P_t = 1 kW = 1000 watts

G_t = 35 dB

L_t = 1

$$\lambda = \frac{c}{f} = \frac{3x10^8\, m/\sec}{10x10^9\, Hertz} = 0.03m$$

$A = K\ T_s B_n$=-90 dBm = 1.10^{-12} watts

Then the constant C in equation (3.3) becomes $2.27x10^{15}$ m^4.

We assume that the distance between each consecutive grid point is 1 km.

The efficient paths between the targets can be seen in Table A.1. There are target pairs with only one connection and target pairs with many connections.

The efficient tours are listed in Table A.2. The efficient frontier can be seen in Figure A.1. There are three efficient tours; 1-2-3-4-5, 1-2-4-5-3 and 1-3-2-4-5.

Table A.1 The Efficient Paths Between the Target Pairs – 5-Target MOTSP (Calculated Radar Threat Values)

Target Pair (i,j)	# of Efficient Paths	Total Distance	Total Radar Detection Threat
(1,2)	1	7.828	1.078
(1,3)	7	9.828	1.294
		8.414	1.679
		7.828	1.726
		7.000	1.987
		6.414	2.080
		5.828	2.705
		5.242	3.765
(1,4)	7	14.828	2.156
		13.656	2.263
		12.828	2.524
		12.242	2.978
		11.656	3.482
		11.070	4.106
		10.484	4.985
(1,5)	2	10.414	1.247
		9.828	1.701
(2,3)	1	4.828	0.216
(2,4)	6	9.828	1.078
		8.656	1.185
		7.828	1.446
		7.242	1.900
		7.000	3.352
		6.414	3.438
(2,5)	10	13.070	0.986
		12.484	1.440
		11.898	1.894
		11.656	2.158

Table A.2 The Efficient Paths Between the Target Pairs – 5-Target MOTSP (Calculated Radar Threat Values) (Continued)

Target Pair (i,j)	# of Efficient Paths	Total Distance	Total Radar Detection Threat
(2,5)	10	11.070	2.252
		10.828	2.821
		10.484	2.976
		10.242	3.275
		9.656	3.847
		9.070	4.959
(3,4)	2	5.828	1.307
		5.242	1.401
(3,5)	6	8.242	0.770
		7.656	1.224
		7.070	1.678
		6.828	1.942
		6.242	2.036
		5.414	3.574
(4,5)	1	4.414	0.909

We give the algorithm details for the problem when we define the best weight of the DM; w_{best} as 0.5000.

The algorithm reports there is only one favorable path between target 1 and target 2, that is (7.828, 1.079).

Between targets 1 and 3, the two extreme points are (5.242, 3.765) and (9.828, 1.295). With weight 0.3500, the middlemost solution (6.414, 2.081) is found. Firstly, the left adjacent of the middlemost solution is found as (5.828, 2.706). These two solutions are compared by the DM. The weighted sum of solution (6.414, 2.081) is better than solution (5.828, 2.706). The algorithm continues with

finding the right adjacent of solution (6.414, 2.081). The right adjacent of (6.414, 2.081) is (7.828, 1.727).The weighted sum of solution (6.414, 2.081) is better than solution (7.828, 1.727). Therefore solution (6.414, 2.081) is the best solution. The favorable weight range of the DM is reduced between 0.2004 and 0.5160.

Table A.3 Efficient Tours of 5-Target MOTSP (Calculated Radar Threat Values)

Tour	Total Distance	Total Radar Detection Threat
1-2-4-5-3	29.312	12.764
1-2-4-5-3	29.898	11.704
1-2-4-5-3	30.140	11.226
1-2-4-5-3	30.484	11.079
1-2-4-5-3	30.726	10.166
1-2-4-5-3	30.968	9.688
1-2-4-5-3	31.312	9.541
1-2-4-5-3	31.554	8.628
1-3-2-4-5	31.898	8.344
1-2-3-4-5	32.140	5.305
1-2-3-4-5	32.726	4.851
1-2-3-4-5	33.312	4.757
1-3-2-4-5	39.312	4.744

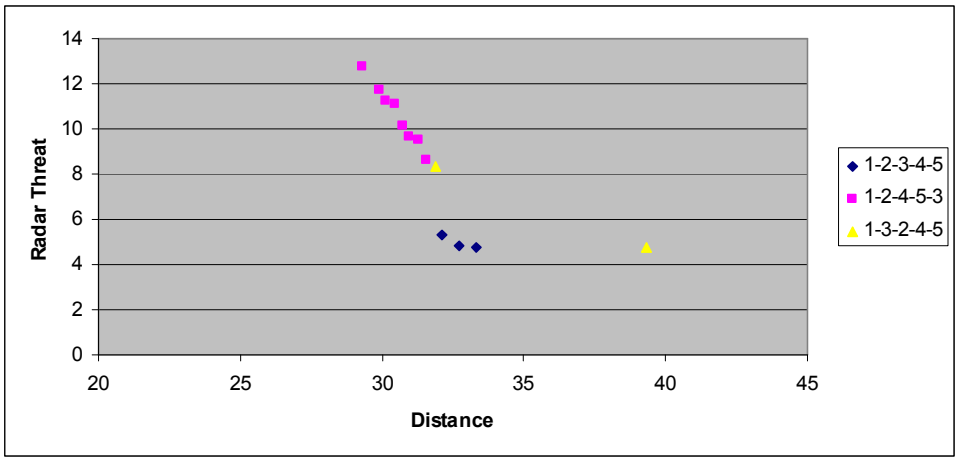


Figure A.1 Efficient Frontier of 5-Target MOTSP (Calculated Radar Threat Values)

The algorithm finds the two extreme solutions (11.656, 3.481) and (13.656, 2.263) between targets 1 and 4. The middlemost solution (12.828, 2.524) is found with weight 0.3785. Left adjacent of solution middlemost solution is (12.242, 2.978). These two solutions are compared and solution (12.242, 2.978) is preferred. Then the left adjacent solution of solution (12.242, 2.978) is found as (11.656, 3.481) which is also the left extreme solution found in the beginning of the evaluation of the path between these two targets. Solution (11.656, 3.481) is preferred to solution (12.242, 2.978); therefore solution (11.656, 3.481) is the best solution. With the comparisons, the favorable weight range is reduced between 0.4620 and 0.5160. We should note that only the lower bound on the weight space is changed since we always preferred the left solution to the right solution between the adjacent solutions.

For the rest of the target pairs, the algorithm finds only one solution within the favorable weight range of the DM. Between targets 1 and 5 solution (9.828, 1.702) is found, between targets 2 and 3 solution (4.828, 0.216) is found, between targets 2 and 4 solution (7.242, 1.900) is found, between targets 2 and 5 solution (11.07, 2.252) is found, between targets 3 and 4 solution (5.242, 1.400) is found, between targets 3 and 5 solution (6.242, 2.035) is found and between targets 4 and 5 solution (4.414, 0.909) is found. The final range of the favorable weight space of the DM is (0.4620, 0.5160).

The algorithm finds only one tour within the favorable weight range of the DM which is 1-2-3-4-5. The total distance of the tour is 32.140 and the total radar detection threat is approximated as 5.306.

There are totally four questions asked to the DM during the determination of the most preferred paths between the target pairs. However, no questions are asked during the determination of the best tour since the weight space is reduced to a narrow range around 0.5000.

APPENDIX B

STRUCTURE OF TERRAIN OF 8-TARGET MOTSP

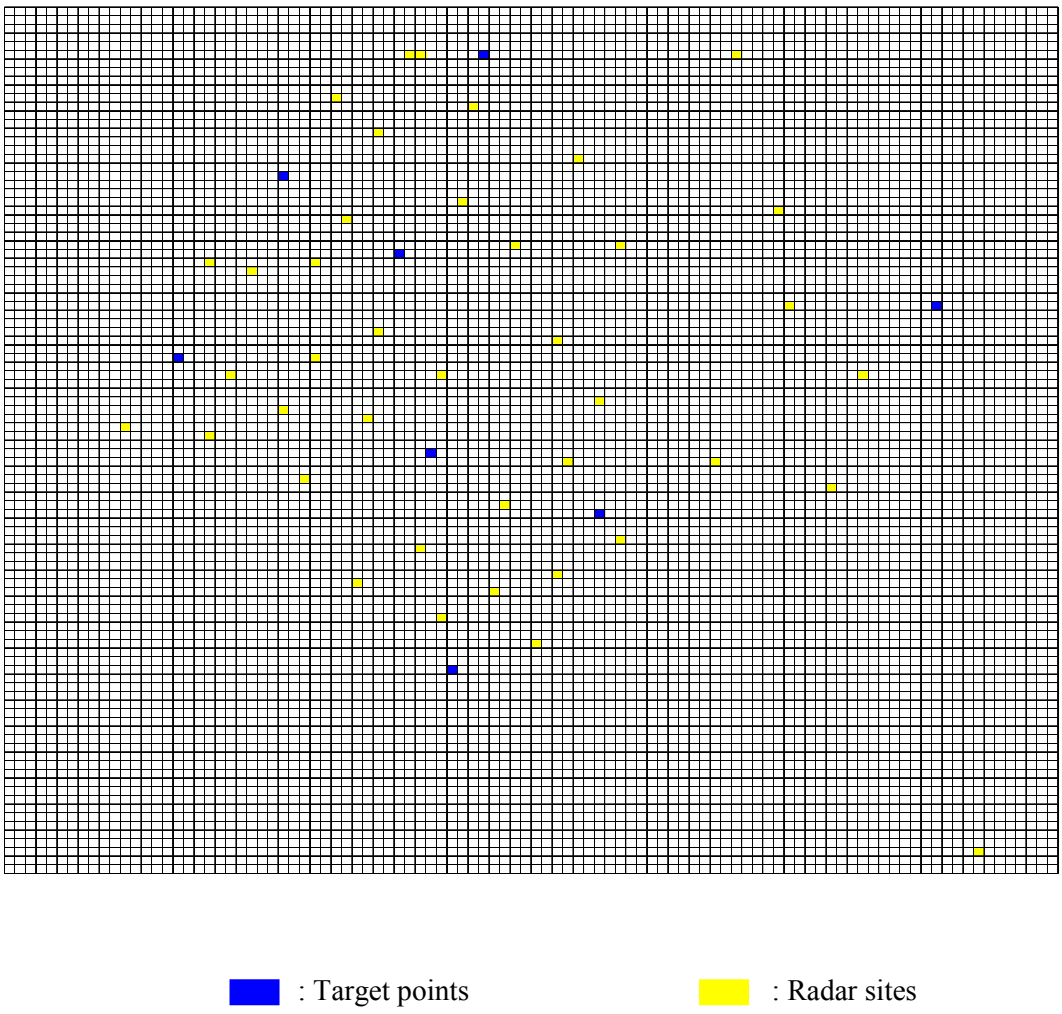


Figure B.1 Structure of Terrain of 8-Target MOTSP

APPENDIX C

THE SOLUTION OF 5-TARGET MOTSP WITH CALCULATED RADAR THREAT VALUES USING AN EVOLUTIONARY ALGORITHM

The structure of this problem is the same with the problem mentioned in Section 4.6.1. The terrain is divided into 10 equidistant grids in both horizontal and vertical. The places of the targets and the radar sites are as shown in Figure 4.4. We have five targets and three radar sites. We calculate the radar detection threat values by the equation (3.3).

The efficient frontier of the problem can be seen in Figure C.1. We have three efficient tours; 1-2-3-4-5, 1-2-4-5-3 and 1-3-2-4-5.

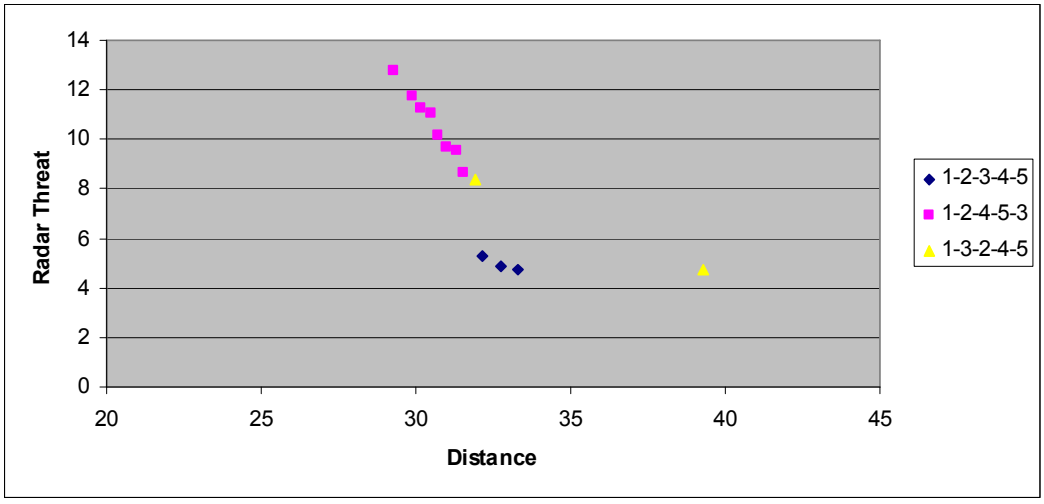


Figure C.1 The Efficient Frontier of 5-Target MOTSP (Calculated Radar Threat Values)

The algorithm is run for different number of equi-spaced points and initial population sizes. The crossover probability is taken as 0.9 and the mutation probability is taken as 0.1. The maximum number of crossover positions is taken as 3.

We run the algorithm for 30000 generations. We try three different ways of choosing the initial population size and number of equi-spaced points as in Chapter 5. These alternatives are listed in Table C.1.

Table C.1 Alternatives of Solution for 5-Target MOTSP (Calculated Radar Threat Values)

Alternative	**1**	**2**	**3**
Initial population size	10	15	30
No. of equi-spaced points	10	15	8
Total population size	100	225	240

When we run the algorithm setting both initial population size and number of equi-spaced points 10, we obtain three potential efficient tours; 1-2-3-4-5, 1-2-4-5-3 and 1-3-2-4-5. The objective function values of these tours can be seen in Figure C.2.

For the second alternative, we set both the initial population size and number of equi-spaced points to 15. We obtain the three efficient tours as a result; however this time the tours are represented by more dummy solutions. The solutions and their spread can be seen in Figure C.3.

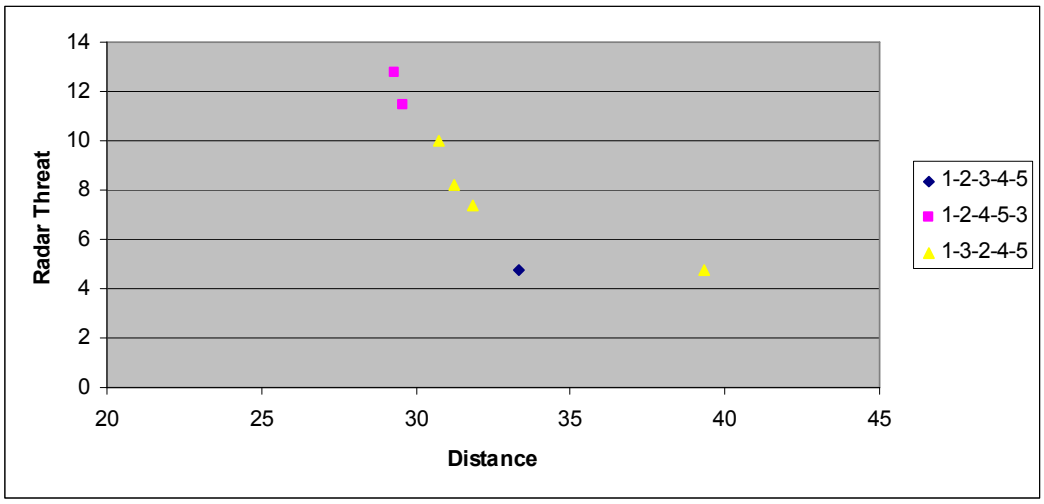


Figure C.2 The Efficient Frontier – Alternative 1 - 5-Target MOTSP (Calculated Radar Threat Values)

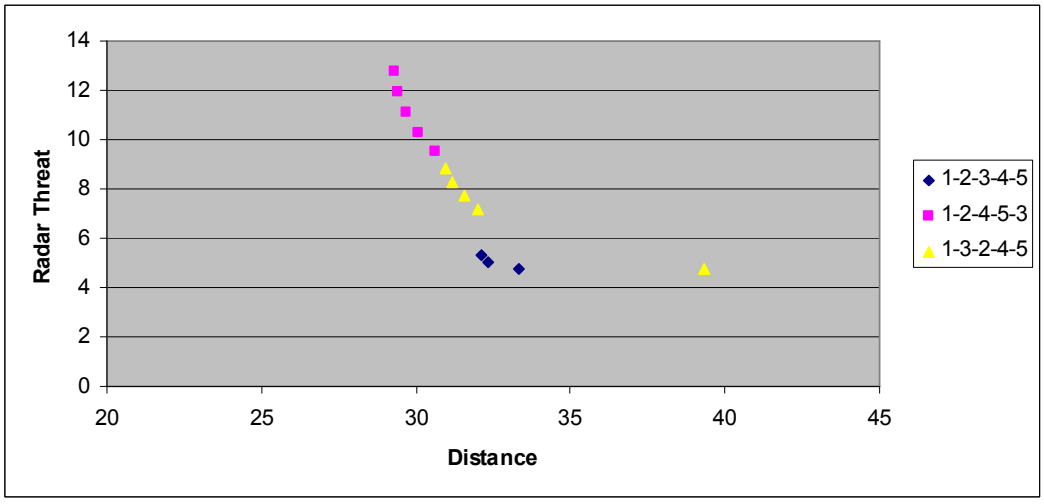


Figure C.3 The Efficient Frontier – Alternative 2 - 5-Target MOTSP (Calculated Radar Threat Values)

In the third alternative, we set the number of intermediate solutions to 8 and the initial population size to 30. We again obtain the same three potential efficient tours. The objective function of these tours can be seen in Figure C.4.

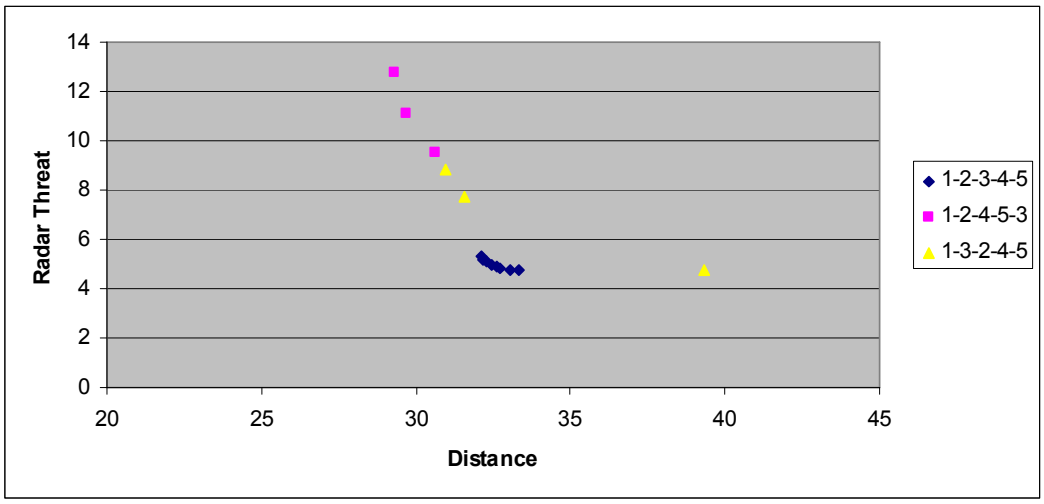


Figure C.4 The Efficient Frontier – Alternative 3 - 5-Target MOTSP (Calculated Radar Threat Values)

All of the alternatives yield the same potential efficient tours; which are also the efficient tours of the problem. We do not obtain any inefficient tour seen as potential efficient in these solutions.