REAL TIME UNMANNED AIR VEHICLE ROUTING

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

BY

NAİL KARABAY

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

JANUARY 2018

Approval of the thesis:

REAL TIME UNMANNED AIR VEHICLE ROUTING

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

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Yasemin Serin
Head of Department, **Industrial Engineering** _____

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

Assist. Prof. Dr. Diclehan Tezcaner Öztürk
Co-Supervisor , **Industrial Eng. Dept., TED University** _____

Examining Committee Members

Prof. Dr. Meral Azizoğlu
Industrial Engineering Dept., METU _____

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

Assist. Prof. Dr. Diclehan Tezcaner Öztürk
Industrial Engineering Dept., TED University _____

Assoc. Prof. Canan Ulu
McDonough School of Business, Georgetown University _____

Assist. Prof. Dr. Mustafa Kemal Tural
Industrial Engineering Dept., METU _____

Date: _____

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 : Nail Karabay

Signature :

ABSTRACT

REAL TIME UNMANNED AIR VEHICLE ROUTING

KARABAY, Nail

M.S., Department of Industrial Engineering
Supervisor : Prof. Dr. Murat Köksalan
Co-Supervisor : Assist. Prof. Dr. Diclehan Tezcaner Öztürk

January 2018, 67 pages

In this thesis, we study real-time routing of an unmanned air vehicle (UAV) in a two-dimensional dynamic environment. The UAV starts from a base point, visits all targets and returns to the base point, while all targets change their locations during the mission period. We find the best route for the route planner (RP) considering two objectives; minimization of distance and minimization of radar detection threat.

We develop a real-time algorithm to find the UAV's most preferred route for a RP who has an underlying linear or quadratic preference function. In this algorithm, we structure the nondominated frontiers of the trajectories between each target pair and find a route using these trajectories. The algorithm updates the route of the UAV each time the UAV arrives at a target. As the UAV must return to the base target at the end of its journey, we solve a multi-objective shortest Hamiltonian path problem to find a route rather than a multi-objective traveling salesperson problem each time the UAV visits a target. To reduce the computational burden, we develop k-closest heuristic. In this heuristic, instead of structuring the nondominated frontiers between all target pairs, for each target, we select k closest targets and structure only the nondominated

frontiers of these k targets. In addition, we develop an adaptive algorithm to determine the value of k. For the RP who has a quadratic preference function, we choose among n nondominated trajectories for each target pair to find a route. We consider the cases $n = 1$ and $n > 1$, seperately. We demonstrate all algorithms on different examples.

Keywords: unmanned air vehicle routing, real-time routing, multi-objective programming, linear preference function, quadratic preference function

ÖZ

İNSANSIZ HAVA ARAÇLARININ GERÇEK ZAMANLI ROTALAMASI

KARABAY, Nail

Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Murat Köksalan
Ortak Tez Yöneticisi : Yrd. Doç. Dr. Diclehan Tezcaner Öztürk

Ocak 2018, 67 sayfa

Bu tezde, iki boyutlu dinamik ortamda bulunan insansız hava araçlarının (İHA) rotalamasını çalıştık. İHA üs noktasından ayrılarak bütün hedeflere gidiyor ve üs noktasına geri dönüyor. Bu esnada bütün hedefler yerlerini değiştiriyorlar. Rota planlayıcısı için toplam mesafeyi en aza indirmeyi ve toplam radara yakalanma tehdidini en aza indirmeyi amaçlayarak en iyi rotayı buluyoruz.

Lineer ve ikinci dereceden tercih fonksiyonuna sahip rota planlayıcısı için en tercih edilen turu bulmak için gerçek zamanlı bir algoritma geliştirdik. Bu algoritmada, her hedef arası etkin sınırları yapılandırıyoruz ve her ikili arasındaki uçuş güzergahlarına göre en iyi rotayı buluyoruz. Algoritma, İHA bir hedefe ulaşınca rotayı güncelliyor. İHA'nın yolculuğunun sonunda üs noktasına dönmesi gerektiği için, her hedef ziyaretinde çok amaçlı gezgin satıcı problemi yerine çok amaçlı en kısa Hamiltonian yol problemi çözdürüyoruz. Hesaplama yükünü azaltmak için, k-en yakın sezgisel algoritmayı geliştirdik. Her hedef çifti için bütün etkin sınırları bulmaktansa, her hedef için en yakın k hedefi seçip sadece onların etkin sınırlarını yapılandırıyoruz. Buna ek olarak, k'nın değerini belirlemek için uyarlanabilir bir algoritma geliştirdik. İkinci

dereceden tercih fonksiyonuna sahip rota planlayıcısı için, her hedef çifti için n tane etkin uçuş güzergahı seçip rota bulduk. Burada, $n = 1$ ve $n > 1$ olduğu durumları ayrı ayrı ele aldık. Bütün algoritmaları farklı örneklerde gösterdik.

Anahtar kelimeler: insansız hava aracı rotalama, gerçek zamanlı rotalama, çok amaçlı programlama, k-en yakın sezgisel algoritma, doğrusal tercih fonksiyonu, ikinci dereceden tercih fonksiyonu

ACKNOWLEDGEMENTS

I wish to thank my supervisors Prof. Dr. Murat Köksalan and Assist. Prof. Dr. Diclehan Tezcaner Öztürk for their valuable contributions and guidance during this study. I am also very thankful for their support and understanding.

I would like to thank Air Force Office of Scientific Research for their funding. This material is based upon work supported by the Air Force Office of Scientific Research, Air Force Material Command, USAF under Award No. FA9550-16-1-0333. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the Air Force Office of Scientific Research, Air Force Materiel Command, USAF.

To my family…

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Unmanned Air Vehicles (UAVs) are unpiloted aircrafts that have been used for both military and civilian purposes such as reconnaissance, surveillance against crimes, agricultural applications, weather forecast, hurricane detection, transportation and minimizing hazardous effects of natural disasters. UAVs have substantial advantages over conventional aircrafts. The most noteworthy of them are listed below. Firstly, there is no need for a qualified on-board pilot to operate UAVs. As there is no pilot, human fatigue is not an issue in determining the flight duration. Secondly, UAVs are mostly equipped with high-resolution video-recording capabilities to get aerial video of distant locations that would be risky for the pilots. Moreover, in some military applications, UAVs are equipped with weapons to destroy enemy targets. Lastly, the initial costs and operational costs (fuel and maintenance) of UAVs are much cheaper than those of conventional aircrafts. Owing to these advantages, use of UAVs, where practical, has increased significantly over the years.

Route planning for these vehicles comprises determining the path of the UAV visiting the targets. Many criteria could be considered in these problems such as distance traveled, radar detection threat and risk posed by a UAV to third parties on the ground. Although a single objective, that is an aggregation of multiple objectives, is used in many applications, there is usually a tradeoff between the objectives that needs to be taken into account. Minimizing distance traveled and minimizing radar detection threat are two meaningful conflicting objectives in many military applications. In the

literature, this problem is studied in different domains. Either the terrain is discretized and the move of the UAV is limited or a continuous terrain is used where the UAV is allowed to move to any point in the terrain. Different solution methods are developed for static environments where the terrain structure is fixed and for dynamic environments where the terrain structure changes in time.

In this thesis, we develop algorithms to find routes for a UAV in real-time under a dynamic environment where the targets change their locations in time. The UAV moves in continuous terrain, and the routes are structured considering two criteria; minimizing distance traveled and minimizing radar detection threat. We develop algorithms that find the most preferred solution of the route planner (RP) whose underlying preference functions are linear and quadratic. For computational efficiency, we also develop heuristic algorithms addressing both preference functions. Additionally, we establish an adaptive algorithm for underlying linear preference functions. We demonstrate all algorithms on different examples.

The thesis unfolds as follows: in Chapter 2, we present the literature on the UAV route planning problem. We give the background and the problem definition in Chapter 3. In Chapter 4, we explain our solution methods for underlying linear and quadratic preference functions and demonstrate our algorithms on examples. In Chapter 5, we give our conclusions.

CHAPTER 2

LITERATURE REVIEW

There are many studies on the multi-objective route planning of UAVs in the literature. These studies can be classified according to whether the environment is static or dynamic. In a static environment, it is assumed that the conditions of the environment do not change in time. Therefore, when the route of the UAV is determined, this route is valid until the end of the mission of the UAV. We first explain the studies where the UAV visits a single target in a static environment. Bortoff (2000) developed a two-step algorithm to generate an optimal UAV path over a hostile territory in a two-dimensional static environment. In the first step, a suboptimal rough-cut path is generated through the radar sites. In the second step, a set of nonlinear ordinary differential equations are solved to minimize the objective function that is a weighted sum of the two objectives: path length, and average distance from the radar sites. Kuwata et al. (2004) also developed a two-phase algorithm. In the first phase, a linear programming model is solved to find visibility of graph to avoid exposure to threats in a three-dimensional static environment. In the second phase, the Dijkstra's algorithm is used to design a detailed trajectory. Xu et al. (2006) applied Particle Swarm Optimization to find a path for the UAV in a two-dimensional static environment. They consider two objectives (the flight time and safety). Allaire et al. (2009) developed a Genetic Algorithm to find a minimum path within the safety zone for the UAV in a three-dimensional static environment. Swartzentruber et al. (2009) used Particle Swarm Optimization to find a path for the UAV in a three-dimensional static environment. The resulting path is optimized with a preference towards

maximum safety, minimum fuel consumption, and target reconnaissance. Rudnick-Cohen et al. (2016) presented four different optimization approaches (Network Optimization Approach, Local Improvement Approach, Greedy Improvement Approach, and Non-Network Approach) to find a path for the UAV in a three-dimensional static environment. All algorithms minimize the risk posed by the UAV if it crashes during its operation and flight time.

The following studies incorporate a dynamic environment to the UAV routing problem in different aspects. Zheng et al. (2005) developed an evolutionary algorithm to find a path for the UAV in a three-dimensional dynamic environment visiting a static target. They can handle unforeseeable changes in the environment like pop-up threats, and they also consider different kinds of mission constraints such as minimum route leg length, flying low altitude to avoid radar detection, maximum turning angle, and fixed approach vector to the goal position. Another study, Cruz et al. (2008) developed an Evolutionary Algorithm (EA) to find a path for multiple UAVs in a three-dimensional environment where the target is stable whereas there can be pop-up threats. The EA tries to minimize the path length, the probability of kill, the flight altitude and the radar detection probability. For real-time route planning, the algorithm tries to avoid the new threats without considering two of the objectives, the flight altitude, and the path length. Ruz et al. (2008) used Mixed Integer Linear Program to find a path in three-dimensional dynamic environments where threat zones called popups are present. They take into account constraints such as maximum turning force which causes a minimum turning radius, maximum flying speed, and radar's detection. Their objective function contains different measures of the quality in the solution of this problem, and the most important criterion is the minimization of the total flying time. Peng et al. (2012) developed Model Predictive Control and Particle Swarm Optimization Algorithm to find a path for the UAV in a three-dimensional dynamic environment where there are pop-up threats. Their cost function includes voyage cost, altitude cost, threat cost, and error cost that comes from deviation from

the shortest distance. Their algorithm first finds the reference route before the UAV starts its flight, based on the known information of the terrain and threats. During the UAV's flight, as the UAV detects the environment and the threats, the route is replanned online. Chen et al. (2014) used the Voronoi diagram for path planning of the UAV in a two-dimensional dynamic environment. Initially, the radar threat field based on the Voronoi diagram is created, and the track performance indicators are established based on the radar threat and fuel costs. Then, an improved version of the Dijkstra's algorithm is applied to reduce the planning time and to hit a moving target effectively.

We next explain some studies on the route planning of the UAV visiting multiple targets. O'Rourke (1999) developed a reactive tabu search heuristic suitable for solving traveling salesperson problem (TSP) and vehicle routing problem (VRP). His objective function minimizes travel cost and penalties for violation of time windows, load capacity, and duration. If a pop-up target appears, the UAV goes to this target directly rather than following a potentially sub-optimal route. Location of the pop-up target becomes a new starting point, and the remaining targets are processed in a route that returns the UAV to the depot. Another study, Kinney (2000) extended O'Rourke's efforts in two main areas. He added the ability to route vehicles so as to avoid restricted time windows or time walls. He also provided a quicker solution using a jump search/tabu search (JTS) hybrid algorithm. Another study, Zhenjua et al. (2008) developed a Multiobjective Ant Colony System (MACS) algorithm to the UAVs route planning problem based on the Voronoi diagram in a two-dimensional static environment. Their objective function minimizes the route length and danger exposure.

Kim et al. (2007) developed exact and non-exact methods for the target assignment problem for the cases where the UAV returns to the base and stays at the last target in a three-dimensional dynamic environment. They considered single and multiple

UAVs. Their objective function minimizes operating time and risk exposure. They first showed how the problem can be exactly formulated in MILP which is likely to be cumbersome as the problem size increases. They then showed theoretically as well as numerically that the non-exact methods perform well regarding optimality and computational complexity. The non-exact methods are alternative MILP formulations which are computationally less intensive and, therefore, suited for real-time purposes.

Tezcaner and Köksalan (2011) developed an algorithm to find the most preferred solution of a decision maker who has an underlying linear preference function considering two objectives in a discretized terrain. For the same problem structure, Tezcaner Öztürk and Köksalan (2016) developed an algorithm to find the most preferred solution of a decision maker who has an underlying quasiconvex preference function. Tezcaner Öztürk (2013) developed methods to generate the nondominated frontier of the routes in a continuous terrain. For the same problem structure, Türeci (2017) developed algorithms to find the best solution for a decision maker having linear and quasiconvex preference functions. We explain these studies in detail in the following chapter. In this thesis, we study the dynamic version of this problem. Different than the previous studies in the literature where pop-up threats are included in the dynamic environment, in our problem setting, the locations of the targets change in time.

CHAPTER 3

BACKGROUND & PROBLEM DEFINITION

The multi-objective route planning problem of a UAV is studied well in the literature with different terrain structures, considering different objectives in a static or a dynamic environment. In this chapter, we give the specifics of our problem. We use the same problem structure and objectives as in Tezcaner Öztürk (2013) and Türeci (2017). We extend their studies by adding a dynamic environment to the problem.

3.1 Definitions

We present some notation and definitions that are directly taken from Tezcaner Öztürk and Köksalan (2016), and Smith (2003) before explaining our problem structure. We specify where we take each part from at the part we introduce it.

Assume, without loss of generality, we have p objectives to minimize. Let x be the decision variable vector, and X be the feasible set. The image of the feasible set in objective function space is denoted with Z. The objective function vector for decision vector x is $z(x) = \left(z_1(x), z_2(x), \dots, z_p(x)\right)$, where $z_k(x)$ is the performance of solution $x \in X$ in objective k.

We take definitions 3.1-3.4 directly from Tezcaner Öztürk and Köksalan (2016).

Definition 3.1 A solution $x \in X$ is efficient if there does not exist $x' \in X$ such that $z_k(x') \leq z_k(x)$ for $k = 1,2, \dots, p$ and $z_k(x') < z_k(x)$ for at least one objective. If there is such an x', x is said to be inefficient. All efficient solutions constitute the efficient frontier (set).

Definition 3.2 If a solution $x \in X$ is efficient, then $z(x)$ is said to be nondominated, and if x is inefficient, and then $z(x)$ is said to be dominated. All nondominated points constitute the nondominated frontier (set).

Definition 3.3 A nondominated solution $z(x)$ is a supported nondominated solution if and only if there exists a positive linear combination of objectives minimized by x. If $z(x)$ is a supported nondominated solution then x is supported efficient solution. Otherwise, $z(x)$ is an unsupported nondominated solution and x is unsupported efficient solution.

Definition 3.4 An extreme nondominated point $z(x)$ is a supported nondominated point that has the minimum possible value in at least one of the objectives.

Definition 3.5 (Smith, 2003) A multigraph is formed from two parts. First, there is a set of points, called nodes or vertices, and second there is a set of lines which join pairs of these points. These lines are known as arcs or edges. For the convenience, this combination will be referred to as *G= (V, E)* with a vertex set *V* and an edge set *E*.

In this thesis, we use "target" and "trajectory" to refer to "node" and "edge," respectively.

Definition 3.6 A Hamiltonian path is a sequence of trajectories that visits each vertex exactly once. In this thesis, we use a Hamiltonian Path and a route interchangeably. Shortest Hamiltonian Path Problem (SHPP) is the problem of finding a shortest Hamiltonian Path.

Definition 3.7 A Hamiltonian tour is a sequence of trajectories that visits each vertex exactly once and returns to the starting vertex. A problem where a Hamiltonian tour that minimizes the total travel distance is found is called the TSP.

3.2 Terrain Structure

The UAV is located at a base point, aims to visit all targets once and return to the base point. During its travel to all targets, the vehicle is allowed to move to any point in a two-dimensional terrain. A number of radars are located in the terrain. An example terrain can be seen in Figure 3.1, where the circular regions are areas where the radars are effective, the triangles are the target points, and the dashed lines are example tours that visit all targets.

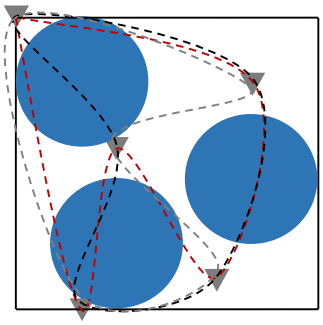


Figure 3.1 Terrain Structure

The radar is located at the center of the radar region, and it is ineffective in detecting the UAV outside its region. Inside the radar region, we have two parts: inner and outer regions. These two regions can be seen in Figure 3.2.

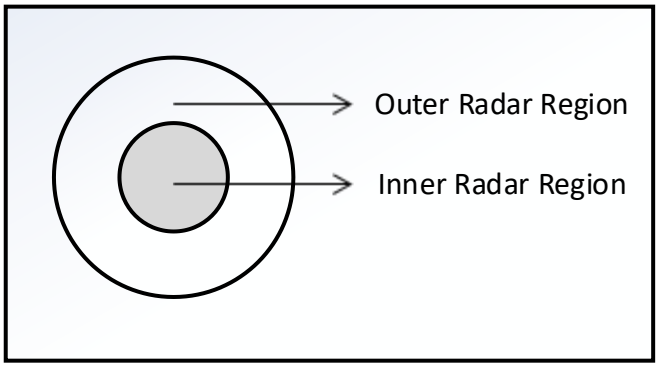


Figure 3.2 Radar Region

3.3 Objectives

The features of the UAV and its mission's purpose determine the objectives for the UAV route planning problem. Some objectives used in the literature are minimizing traveled distance, maximizing average distance from the radar sites, maximizing safety, minimizing the risk posed by the UAV if it crashes during its mission, minimizing flight altitude, minimizing radar detection probability, and minimizing error cost that comes from deviation from the possible shortest distance. In this thesis, to determine the best route the UAV follows, we consider two objectives: minimizing total distance traveled and minimizing radar detection threat.

To find the total distance traveled, we sum the lengths of all the paths the UAV follows as in Tezcaner Öztürk (2013). The total distance between the initial target (x_s, y_s) and the final target (x_f, y_f) can be calculated using (3.1). The expression ds in formula (3.1) represents the infinitesimal part in the movement of the vehicle betweeen the targets.

$$D = \int_{(x_s,y_s)}^{(x_f,y_f)} ds \tag{3.1}$$

For the second objective, we use the measure developed by Gudaitis (1994) and used in Tezcaner Öztürk (2013) and Türeci (2017). We sum the detection probabilities on the UAV's path inside the radar regions. The detection probability is 1 in the inner radar region. The detection probability reduces from 1 to 0 as we move from the circumference of the inner radar region towards the circumference of the outer radar region. There is no detection probability outside of the radar region. The summation of these probabilities is a measure that approximates how long the vehicle is exposed to that much radar detection probability. The detection probability of a point (x, y)($pd_{(x,y)}$) is calculated using (3.2) and it depends on the S/N (signal-to-noise ratio) of that point (Equation 3.3) inside a radar region with center at (k, l). Total radar detection threat (3.4) is the summation of all the detection probabilities over the

trajectory between the initial target (x_s, y_s) and final target (x_f, y_f). Realistic values are assigned to upper bound (UB) and lower bound (LB) to reach meaningful results.

Let,

P_t: Power transmitted by radar (Watts)

G_t: Power gain of transmitting antenna

L_t: Transmitting system loss

λ: Wave lenght of signal frequency (Meters)

T_s: Receive system noise temperature (Kelvin)

B_n:Noise bandwidth of receiver (Hertz)

K: Boltzman's constant (Joules/Kelvin)

σ: Aircraft radar cross section (RCS) (Square Meters)

R: Distance from the transmitter to aircraft's location (x, y) (Meters)

$pd_{(x,y)}$: Detection probability at location (x, y)

$$pd_{(x,y)} = \begin{cases} 1 & if\ S/N_{(x,y)} > UB_{S/N} \\ \frac{S/N_{(x,y)} - LB_{S/N}}{UB_{S/N} - LB_{S/N}} & if\ LB_{S/N} < S/N_{(x,y)} \leq UB_{S/N} \\ 0 & if\ S/N_{(x,y)} \leq LB_{S/N} \end{cases} \quad (3.2)$$

$$S/N_{(x,y)} = 10log_{10}\left(\frac{C}{R^4}\right) \text{ where } C = \frac{P_t {G_t}^2 \lambda^2 \sigma}{(4\pi)^3 K T_s B_n {L_t}^2} \ \&\ R = ((x-k)^2 + (y-l)^2)^{\frac{1}{2}} \quad (3.3)$$

$$RDT = \int_{(x_s,y_s)}^{(x_f,y_f)} pd_{(x,y)} ds \quad (3.4)$$

3.4. Problem Structure

The route planning problem with multiple targets to be visited is a TSP. In this problem, the aim is to find a tour that starts from a base, visits all targets and returns to the base while optimizing some objectives. The most widely used objective is minimizing the total distance. If there are multiple objectives to be optimized, the problem turns to a MOTSP.

In the literature, most studies on UAV route planning problem assume that the targets are connected with a single efficient trajectory. Although this might be the case for some target pairs, we expect to have multiple efficient trajectories between target pairs under the presence of conflicting objectives. MOTSP with multiple efficient trajectories is referred as generalized MOTSP and is studied in Tezcaner and Köksalan (2011), in Tezcaner Öztürk and Köksalan (2016), and in Köksalan and Tezcaner Öztürk (2017). The first two studies develop algorithms to find the most preferred solution of a decision maker with underlying linear and quasiconvex preference functions, respectively. The last study approximates the nondominated frontier of this problem using an evolutionary algorithm.

In the MOTSP with a single efficient trajectory between targets, the aim is to find the visiting order to the targets. In the generalized MOTSP, we find both the visiting order to the targets and (since there are multiple options for the trajectories that can be used between target pairs) the trajectory to use between target pairs. The studies on generalized MOTSP (Tezcaner and Köksalan, 2011, Tezcaner Öztürk and Köksalan, 2016, and Köksalan and Tezcaner Öztürk, 2017) assume that the terrain is discretized by grids, and the UAV is allowed to move only between the intersections points of the grids. We then have finite trajectory options between any two targets. However, in reality, the UAV can move to any point in the air so that there are infinitely many trajectory options between target pairs. In this study, we allow the UAV to move in a continuous terrain. Tezcaner Öztürk (2013) and Türeci (2017) are two studies on

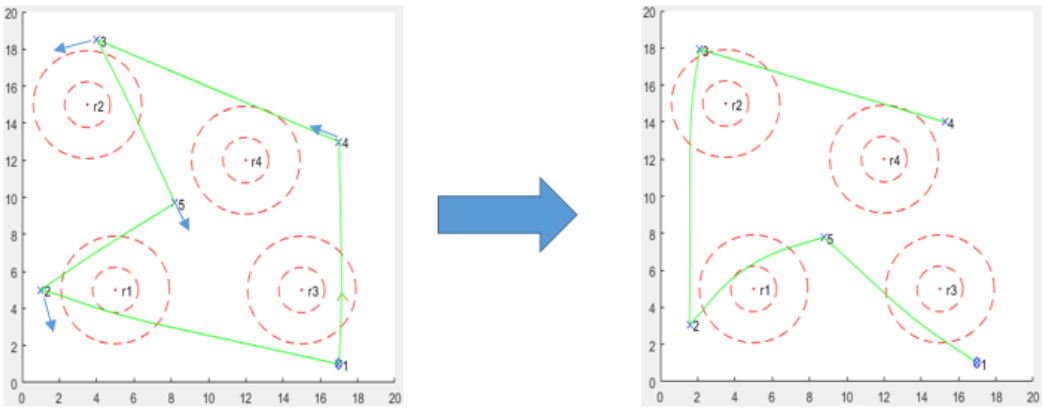
routing UAVs in continuous terrain. Tezcaner Öztürk (2013) develop approaches to find the nondominated frontier of the overall problem, and Türeci (2017) develop interactive algorithms that find the most preferred solution of RPs with linear and quasiconvex preference functions.

For the generalized MOTSP, the solution approach can be decomposed into two parts. Firstly, the efficient trajectories between target pairs should be found. This part is a multi-objective shortest path problem (MOSPP). After all efficient trajectories are found, the tour visiting all targets should be found. For the continuous terrain, Tezcaner Öztürk (2013) developed a nonlinear mathematical model that finds the optimal tour given one objective value.

MOTSP with a single efficient trajectory between target pairs is NP-hard, and MOSPP is NP-complete and intractable (Ehrgott, 2000). This means finding a solution to these problems requires substantial computational effort, and as the problem size gets larger, the computational effort required increases considerably. The generalized MOTSP is also NP-hard. The solution approaches generally require substantial computational time for these problems. Although the problem is NP-hard, it is not so critical for us since UAVs do not need to visit too many targets. In this thesis, we use problems that have up to 15 targets.

In this study, we relax the assumption that all locations of the targets are static. We allow the targets to change their locations during the travel of the UAV. We assume that the moves of the targets are not known in advance by the RP, but the exact locations of all the targets are known when the UAV visits a target. In a time between the departure of the UAV from any target and the arrival of the UAV to the next target, we assume that each target moves in a single direction with a constant velocity. As the UAV can detect the directions and the velocities of the targets, we find the point where the UAV catches the next target exactly.

With the assumptions stated above, the problem is still a generalized MOTSP with multiple efficient trajectories between target pairs. We can solve the generalized MOTSP before the UAV initiates its flight and let the UAV follow this initial route. However, this initial route of the UAV may become dominated as the targets change their locations, some get closer, and some get farther, and we might need to restructure the route. This update can be done more accurately when the new locations of the targets are known. Thus, we update the route as we visit each target on the route. Although the initial problem (with all targets to be visited) is a generalized MOTSP, as we visit targets, we only need to find a route that visits the targets that the UAV has not visited yet. This problem is a SHPP. Therefore, we need to solve a new SHPP after visiting each target in real time. In this problem, we need to start from the target the UAV is currently at; we need to visit all the remaining targets, and terminate at the base point. The resulting solution is not a tour for this problem but a path starting from one target, visiting all targets and going to a final target. SHPP is also NP-hard (Karger et al., 1995).



(a) The initial map (b) The map after visiting target 4

Figure 3.3 The location of the targets and the best routes initially and after visiting a target

Figure 3.3 demonstrates the dynamic routing problem. Figure 3.3 (a) shows the initial map. Firstly, MOTSP is solved according to the initial map, and let tour 1-4-3-5-2-1 is found as the most preferred solution of the RP. We assume that, while the UAV

moves to target 4 from target 1, all targets move in the directions shown by the arrows. When the UAV reaches target 4, it knows the exact locations of each target. Then, SHPP is solved for the updated map after visiting target 4, and the route 4-3-2-5-1 is found as the most preferred route of the RP (Figure 3.3 (b)). The RP prefers another route after visiting target 4, for the current locations of the targets that have not been visited yet.

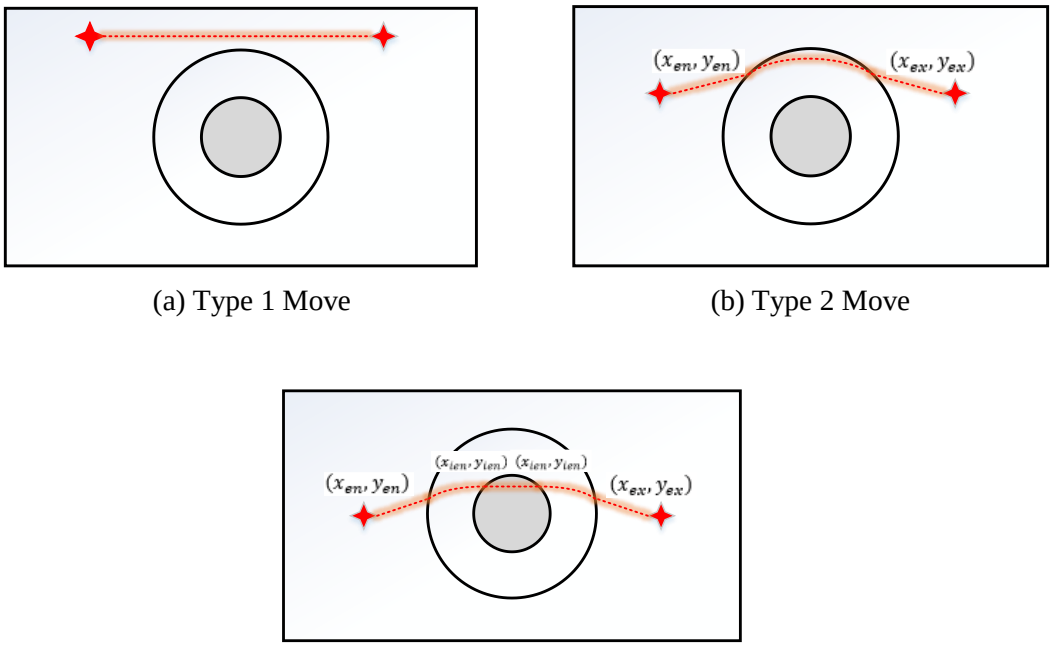
3.5 Movements of the UAV

As the UAV is allowed to move to any point in the continuous terrain, there are infinitely many trajectories between each target pair and the combinations of these trajectories make infinitely many tours visiting all targets. In our solution approach, we firstly find the efficient trajectories between each target pair. This problem is a MOSPP between target pairs. We then find a route made of these efficient trajectories. This problem is a MOTSP with multiple efficient trajectories between target pairs. Tezcaner and Köksalan (2011) refer this problem as generalized MOTSP. In this section, we first explain the movement types between two targets.

3.5.1 Movement between Two Targets

The properties of the movements between two targets are developed by Tezcaner Öztürk (2013). There are three different movement types between each target pair. If the shortest path between two targets does not intersect with a radar region, we have only one efficient trajectory (with the smallest distance and zero radar detection threat) between those targets. We refer this type of move as Type 1. For Type 2 moves, the shortest path between a target pair intersects a radar region and passes through only the outer radar region but not the inner radar region. For Type 3 moves, the shortest path between a target pair passes through both the outer and inner radar

regions. The three movement types can be seen in Figure 3.4.The points (x_{en}, y_{en}) and (x_{ex}, y_{ex}) represent the entrance and exit points from the outer radar region, respectively. The entrance and exit points from the inner radar region are the points (x_{ien}, y_{ien}) and (x_{iex}, y_{iex}) in Figure 3.4(c), respectively. If there is more than one radar between any target pair, we select the radar whose detection threat value is the highest as the effective radar, and ignore the detection threat from the other radars.



(a) Type 1 Move

(b) Type 2 Move

(c) Type 3 Move

Figure 3.4 Movement Types

In Type 1, the shortest trajectory between two targets does not pass through the radar region. As a result, there exists a single efficient trajectory between those targets with the shortest distance and zero radar detection threat. Assuming that one target is located at coordinates (x_s, y_s), and the other target located at coordinates (x_f, y_f), we calculate the total distance (D) with equation (3.4). This is the Euclidean distance which is shown in Figure 3.4(a).

$$D = \sqrt{\left(x_f - x_s\right)^2 + \left(y_f - y_s\right)^2} \tag{3.5}$$

In Type 2, as the shortest path between a target pair intersects the radar region, there are infinitely many efficient trajectories between those targets. At the same time, each efficient trajectory enters and leaves the radar region from different points and passes through only the outer radar region. One example of Type 2 trajectories can be seen in Figure 3.4(b).

Inside the outer radar regions, we assume that the vehicle follows a circular path as if it is moving on a circle, $(x-a)^2 + (y-b)^2 = r^2$, centered at (a, b) with radius r. The distance and radar detection threat measure are calculated with equations (3.6) and (3.7), respectively.

$$D = \sqrt{(x_{en} - x_s)^2 + (y_{en} - y_s)^2} + 2.arcsin\left(\frac{\sqrt{(x_{ex}-x_{en})^2+(y_{ex}-y_{en})^2}}{2r}\right).r + \sqrt{\left(x_f - x_{ex}\right)^2 + \left(y_f - y_{ex}\right)^2} \tag{3.6}$$

$$RDT = \int_{x_{en}}^{x_{ex}} \left(\frac{10}{UB_{S/N} - LB_{S/N}} log_{10}\left(\frac{C}{\left(x^2 + \left(\sqrt{r^2-(x-a)^2}+b\right)^2\right)^2} \right) - \frac{LB_{S/N}}{UB_{S/N}-LB_{S/N}} \right) . \sqrt{\frac{r^2}{r^2-(x-a)^2}}\, dx \tag{3.7}$$

In distance calculation, as the UAV follows a straight line outside the radar region, the distance between the initial target (x_s, y_s) and the entrance point (x_{en}, y_{en}) to the radar region, and the distance between the destination target (x_f, y_f) and the exit point (x_{ex}, y_{ex}) from the radar region are the Euclidean distances. The second term is the length of the arc traveled inside the outer radar region. It gives the circular distance between the entrance and exit points.

Equation 3.7 gives the total radar detection threat value for the arc traveled inside the outer radar region. Details of this formula can be seen in Tezcaner Öztürk (2013).

In type 3, the efficient trajectories between two targets pass through both the outer and the inner radar regions. Inside the outer radar region, it is assumed that the UAV makes a circular movement as in Type 2 (centered at (a, b) with radius *r)*. Inside the inner radar region, as the detection probability is 1, there is only one objective: minimizing the distance traveled. Consequently, the UAV follows a straight path inside the inner radar region. One example of a Type 3 trajectory is illustrated in Figure 3.4(c).

Formulas for the total distance and total radar detection threat are given in (3.8) and (3.9), respectively. Here, (x_{ien}, y_{ien}) is the entrance point to the inner radar region and (x_{iex}, y_{iex}) is the exit point from the inner radar region.

$$D = \sqrt{(x_{en} - x_s)^2 + (y_{en} - y_s)^2} + 2.arcsin\left(\frac{\sqrt{(x_{ien}-x_{en})^2+(y_{ien}-y_{en})^2}}{2r}\right).r + \sqrt{(x_{iex} - x_{ien})^2 + (y_{iex} - y_{ien})^2} + 2.arcsin\left(\frac{\sqrt{(x_{ex}-x_{iex})^2+(y_{ex}-y_{iex})^2}}{2r}\right).r + \sqrt{\left(x_f - x_{ex}\right)^2 + \left(y_f - y_{ex}\right)^2} \quad (3.8)$$

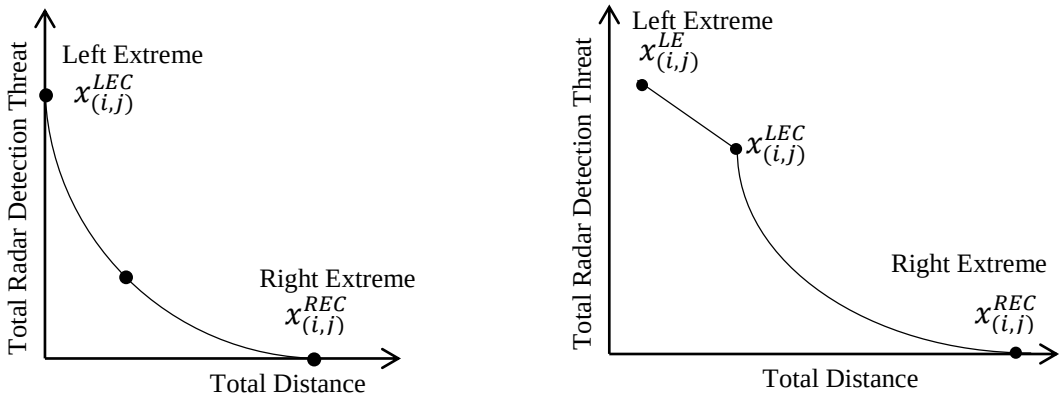
$$RDT = \int_{x_{en}}^{x_{ien}} \left(\frac{10}{UB_{S/N} - LB_{S/N}} log_{10}\left(\frac{C}{\left(x^2+\left(\sqrt{r^2-(x-a)^2}+b\right)^2\right)^2}\right) - \frac{LB_{S/N}}{UB_{S/N} - LB_{S/N}}\right).\sqrt{\frac{r^2}{r^2-(x-a)^2}}dx + \int_{x_{iex}}^{x_{ex}} \left(\frac{10}{UB_{S/N} - LB_{S/N}} log_{10}\left(\frac{C}{\left(x^2+\left(\sqrt{r^2-(x-a)^2}+b\right)^2\right)^2}\right) - \frac{LB_{S/N}}{UB_{S/N} - LB_{S/N}}\right).\sqrt{\frac{r^2}{r^2-(x-a)^2}}dx + \sqrt{(x_{iex} - x_{ien})^2 + (y_{iex} - y_{ien})^2} \quad (3.9)$$

In distance calculation, the first and the last terms are the lengths of straight paths outside the outer radar region. The middle term is the direct distance between the entrance point to the inner radar region and the exit point from the inner radar region. The rest are the lengths of the arcs traveled inside the outer radar region.

In radar detection threat value calculation, the first and the second terms give the radar detection threat values corresponding to the circular paths inside the outer radar region. The last term is the radar detection threat value of the movement inside the inner radar region.

3.5.2 Nondominated Frontiers of Type 1-2-3 Moves

The structures of the nondominated frontiers of each move type are developed by Tezcaner Öztürk (2013). They enumerate all possible trajectories between two targets and generate the nondominated frontier of the problem. They end up with three different move types. Specifically, for moves of Type 1 we have a single nondominated point, for moves of Type 2 (Figure 3.5(a)) we have a nondominated frontier that is curved, and for moves of Type 3 (Figure 3.5(b)) we have a two-piece nondominated frontier (one piece is a straight line and the other is curved). We end up with a straight line part, since the UAV follows a different path in the inner radar region and we use a different formula to estimate the detection probability. In the figure, i and j represent the targets.



(a) Nondominated Frontier of Type 2 (b) Nondominated Frontier of Type 3

Figure 3.5 Nondominated Frontiers of Type 2 and 3 Moves

To structure the curved part of the nondominated frontiers, we use *Lq* distance functions. *Lq* distance functions were first developed by Köksalan (1999) in a scheduling context, and it can be applied to the curved, convex, and continuous parts of the nondominated frontiers. Later, Köksalan and Lokman (2009) showed on many combinatorial problems that a *Lq* distance function fitted using only a few nondominated points is able to approximate the nondominated set well. In order to fit a *Lq* distance function, we need three points on the nondominated frontiers; two points on the extremes of the curved-line and one central point, as shown in Figure 3.5. These three nondominated points can be found using exact or heuristic methods developed by Tezcaner Öztürk (2013). In the exact method, a nonlinear programming model is solved. In the heuristic, the trajectory with the smallest radar detection threat value is approximated for a given distance value. The heuristic searches for the smallest radar detection threat value using the property that it is a convex function of the entrance and exit points' midpoint. Using Golden Section Search, we change that midpoint and converge to the midpoint that gives the best radar detection threat value.

Let (c_L^1, c_L^2) and (c_R^1, c_R^2) represent the first and second objective function values of the left and the right extreme points, respectively. Then, we find a central point $(c_{central}^1, c_{central}^2)$ between the extreme points by applying the heuristic method developed by Tezcaner Öztürk (2013). We then fit the *Lq* distance function using (3.10) below.

$$(1 - zf_1^m)^q + (1 - zf_2^m)^q = 1 \qquad (3.10)$$

where, $zf^m = (zf_1^m, zf_2^m) = \left(\frac{c_{central}^1 - c_L^1}{c_R^1 - c_L^1}, \frac{c_{central}^2 - c_R^2}{c_L^2 - c_R^2}\right)$

After finding these three nondominated points, we find the value of *q* in the equation (3.10) by using the bisection method.

For Type 2 moves, we have two extreme movements as shown in Figure 3.6(a). To find the trajectory with the minimum distance and maximum radar detection threat, we let the UAV follow a straight path between targets. This movement can be seen in Figure 3.6 (a). In Figure 3.6 (b), the UAV does not enter the radar region and follows a path around the circumference of the outer radar region. Thus, it gives the trajectory with the maximum distance and minimum radar detection threat (that is zero).

(a)Min D& Max RDT Trajectory (b) Max D & Min RDT Trajectory

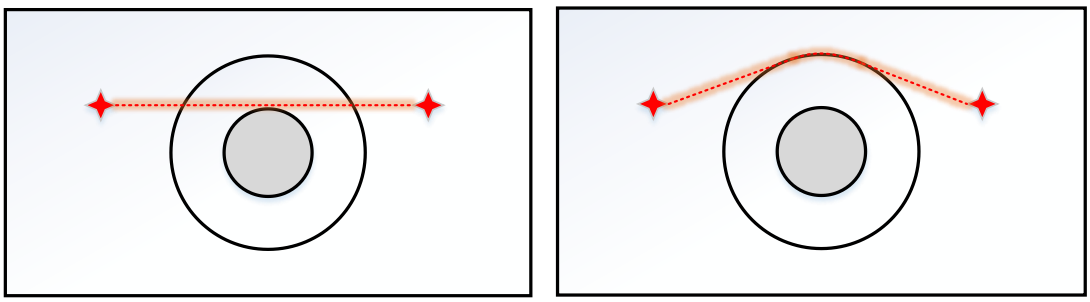
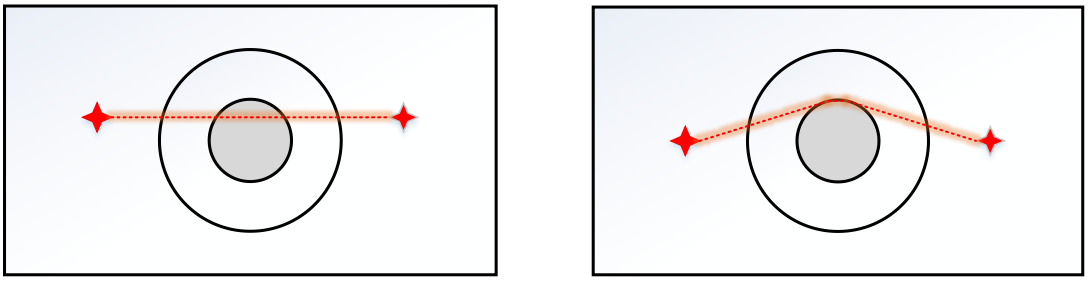


Figure 3.6 Extreme Movements of Type 2

The structure of the nondominated frontier for type 2 moves can be seen in Figure 3.5(a). The left extreme point of the curve $\left(x_{(i,j)}^{LEC}\right)$ corresponds to the trajectory which

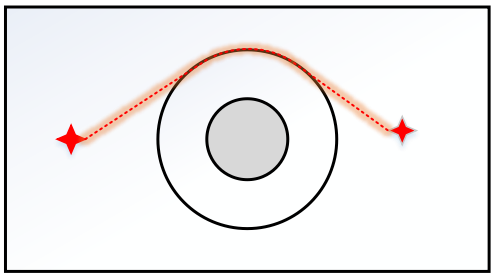
has the minimum distance and maximum radar detection threat between targets i and j (see Figure 3.6 (a)). The right extreme point of the curve $\left(x_{(i,j)}^{REC}\right)$ corresponds to the trajectory which has the maximum distance and minimum radar detection threat (Figure 3.6 (b)).

As the type 3 moves have a two-piece nondominated frontier, we need to find the extreme points of each piece of the nondominated frontier. For the straight line of the nondominated frontier, we need to find the shortest distance trajectory (Figure 3.7 (a)) and minimum distance trajectory (Figure 3.7 (b)) that does not enter the inner radar region. For the curved part, we additionally need to find the maximum distance trajectory (Figure 3.7 (c)) that does not enter the outer radar region.



(a) Min D & Max RDT Traj

(b) Min D traj without entering the inner radar



(c) Max D & Min RDT Trajectory

Figure 3.7 Extreme Movements of Type 3

In Figure 3.5(b), the structure of the nondominated frontier of type 3 moves is given. The left extreme solution of the nondominated frontier $\left(x_{(i,j)}^{LE}\right)$ represents the shortest path between targets (Figure 3.7 (a)). The left extreme of the curved part $\left(x_{(i,j)}^{LEC}\right)$ represents the trajectory that does not enter the inner radar region but passes around the boundary of inner and outer regions (Figure 3.7 (b)). The right extreme solution$\left(x_{(i,j)}^{REC}\right)$ represents the path with zero radar detection threat (Figure 3.7 (c)).

CHAPTER 4

SOLUTION APPROACH & RESULTS

In this chapter, we explain our solution approaches we developed for routing problem in a dynamic environment. We consider two different underlying preference functions for the RP, linear and quadratic. For these two functions, we develop algorithms to find (or approximate) the most preferred route of the RP. For both approaches, we solve MOTSP and MOSHPP. We first introduce the mathematical models of MOTSP and MOSHPP. We next explain our solution approaches and give their computational results.

4.1 Mathematical Models of MOTSP and MOSHPP

We first give some definitions that are common to both models. We then introduce the generalized MOTSP formulation. We change some constraints of the MOTSP model to obtain the MOSHPP model. For both formulations, we assume that there are a finite number of efficient trajectories (H_{ij}) connecting target pair (i,j). Although this is not the case for multi-objective routing problems in continuous terrain, we use these formulations to find the most preferred solutions of the RP due to the structures of the preference functions and our assumptions.

Sets:

T : Set of targets

Decision Variables:

x_{ij}^{h}: 1 if h^{th} efficient trajectory connecting targets (i, j) is used, 0 otherwise

u_i : auxiliary variable

Parameters:

H_{ij} : number of efficient trajectories between targets *i* and *j*.

c_{ijh}^{k}: objective *k* value of efficient trajectory *h* between targets *i* and *j*

We next introduce the generalized MOTSP formulation as in Tezcaner Öztürk and Köksalan (2016).

(MOTSP)

$$\text{Min } z_1 = \sum_{i\in T}\sum_{j\in T}\sum_{h=1}^{H_{ij}} c_{ijh}^{1} x_{ij}^{h} \quad (4.1)$$

$$\text{Min } z_2 = \sum_{i\in T}\sum_{j\in T}\sum_{h=1}^{H_{ij}} c_{ijh}^{2} x_{ij}^{h} \quad (4.2)$$

S.to:

$$\sum_{j\in T}\sum_{h=1}^{H_{ij}} x_{ij}^{h} = 1 \qquad \forall\, i \in T \quad (4.3)$$

$$\sum_{i\in T}\sum_{h=1}^{H_{ij}} x_{ij}^{h} = 1 \qquad \forall\, j \in T \quad (4.4)$$

$$u_i - u_j + \sum_{h=1}^{H_{ij}}(|T|-1)x_{ij}^{h} + \sum_{h=1}^{H_{ij}}(|T|-3)x_{ji}^{h} \geq |T|-2 \;\forall\, i,j \in T-\{1\}, i \neq j \quad (4.5)$$

$$1 \leq u_i \leq |T|-1 \qquad \forall\, i \in T-\{1\} \quad (4.6)$$

$$x_{ij}^{h} \in \{0,1\} \qquad \forall\, i,j \in T, h = 1,\dots,H_{ij} \quad (4.7)$$

We minimize first and second objectives in (4.1) and (4.2), respectively. Departure from and arrival to each target are satisfied with Equations (4.3) and (4.4), respectively. Constraints (4.5) and (4.6) are the subtour elimination constraints which are the strengthened version of subtour elimination constraints of Miller, Tucker, and Zemlin (1960). Desrochers and Laporte (1991) developed this version.

For the MOSHPP, we find the path starting from target ℓ and reaching target 1 (base point). We change constraints (4.3) and (4.4) such that only the targets that are not visited yet should both be arrived to and departed from. We should not depart from the base point and should not arrive at the current target again. Let the set of unvisited targets be *M*.

$$\sum_{j\in M\cup\{1\}}\sum_{h=1}^{H_{ij}} x_{ij}^{h} = 1 \qquad \forall\, i \in M \cup \{\ell\} \qquad (4.8)$$

$$\sum_{i\in M\cup\{\ell\}}\sum_{h=1}^{H_{ij}} x_{ij}^{h} = 1 \qquad \forall\, j \in M \cup \{1\} \qquad (4.9)$$

We change constraints (4.3) and (4.4) with constraints (4.8) and (4.9), respectively. These constraints provide that there is no departure from visited targets except for where the UAV is currently located, and there is no arrival to visited targets except for the base target. In addition, constraints (4.5) and (4.6) are valid for all targets included in the set *M* for MOSHPP.

4.2 Real-time Algorithm

We develop a real-time algorithm that structures the route of the UAV each time a target is visited. Our algorithm consists of two phases. Before starting the algorithm, we assume that we interact with the RP and estimate the structure of the RP's preference function. We explain the details of this step in the following sections.

Phase 1 - Initialization

In the first phase, we find the initial route of the UAV starting from the base target. Let the base target be denoted as target 1, and assume that there are $|T| - 1$ targets to be visited. Let M be the set of targets that the UAV has not visited yet. At the start of the algorithm, $M = \{2,3, \dots, |T|\}$.

At this phase, we solve the MOTSP for the initial placement of the targets. We obtain a tour that starts from and ends at target 1, and we find the trajectories to use between consecutive target pairs in the tour. Let the tour be 1-2-3-4-5-1 for a 5-target example. Target 1 is linked to two different targets in the resulting tour (targets 2 and 5), and, in a symmetrical problem setting, we need to choose which of those two targets to visit next. We compare the preference function values of the trajectories between 1 and 2, and 1 and 5, using the equation of the preference function of the RP. We choose the trajectory that gives a better preference function value. Let ℓ indicate the selected target. We then update M as $M = \{2,3, \dots, |T|\} - \{\ell\}$.

Phase 2 - Real Time Phase

In this phase, we solve the MOSHPP for the remaining targets that have not been visited yet. In the first execution of this phase, the UAV is at target ℓ, and there are $|T| - 2$ targets to be visited. We obtain the new terrain setting for the current locations of the targets and based on this information, we derive the nondominated frontiers between each target pair in set $M \cup \{1, \ell\}$ except for the nondominated frontier between the target 1 and the target ℓ if it is not the last iteration. We then find the trajectory to be used between each target pair in set $M \cup \{1, \ell\}$, and input this trajectory information to solve the MOSHPP. After the solution, we obtain a path that starts from target ℓ, ends at target 1, and visits all targets in M. Let the next target to be visited be denoted as ℓ'. We update M as $M \leftarrow M - \{\ell'\}$. We let the UAV follow

this path and when the UAV reaches target ℓ', we follow the steps explained above. We repeat this phase until set M becomes empty.

We next give the steps of the two phases.

PHASE 1

Step 1: Solve MOTSP using the initial locations of the targets.

Step 2: Find ℓ and set $M = T\text{-}\{1, \ell\}$

PHASE 2

Step 1: Approximate the nondominated frontiers between all target pairs (i, j), $i, j \in M \cup \{1, \ell\}$, $(i, j) \neq (1, \ell)$, for the updated map.

Step 2: Solve MOSHPP starting from ℓ and ending at 1, visiting all targets in M. Let ℓ' be the target following ℓ in the MOSHPP.

Step 3: Update $\ell \leftarrow \ell'$ and $M \leftarrow M\text{-}\{\ell'\}$. If $M = \varnothing$, stop. Otherwise, go to Step 1.

With our algorithm, we do not guarantee finding the optimal solution for the whole problem since there are many unknowns that we cannot account for. However, for given locations of targets, the solutions of MOTSP and MOSHPP give the optimal routes.

In our problem, we assume that the locations of radars are static but our algorithm is applicable even if they are dynamic. For dynamic radars, at each execution of Phase 2, we need to structure the nondominated frontiers between all target pairs for the new locations of the targets/radars.

4.2.1 Linear Preference Function (RT-L)

In this section, we explain how we use our real-time algorithm for the dynamic routing problem to find the most preferred route of a RP whose underlying preference function is linear.

The linear preference function (4.10) minimizes the summation of two objectives aggregated with weight w. Here, w and 1- w represent the importance the RP gives to the first and the second objectives, respectively. We minimize both objectives, therefore the most preferred solution of the RP is the solution that minimizes $U(x)$.

$$U(x) = w\, z_1(x) + (1 - w)\, z_2(x) \text{ where } 0 < w < 1 \quad (4.10)$$

Here, $z_i(x)$ is the i^{th} objective's value, and the objective values are normalized between 0 and 1 so that the weights (w and $1 - w$) roughly represent the relative importance the RP associates with the corresponding objectives.

In our solution approach, we assume that we interact with the RP before starting our algorithm and estimate the value of w^* that structures the RP's preference function. We use Türeci (2017)'s algorithm and find a narrow value range for w^*. We then set w to the middle point of that range and start our algorithm.

During the algorithm, each time we search a new solution, we solve a MOTSP or a MOSHPP. However, we only input one efficient trajectory between each target pair to the model. Due to the structure of linear preference functions, the best tour optimizing (4.10) for a *w* uses the trajectories that optimize (4. 10) for that *w* between target pairs (Tezcaner and Köksalan, 2011). In order to find this trajectory, we first check the movement type of the target pair. If the movement type is 1, then there is only one efficient trajectory which is the shortest distance trajectory, and we use this trajectory between that target pair. If the movement type is 2, we find the point where the derivative of the Lq distance function is equal to the derivative of the linear

objective function (4.10) on the Lq distance function. As the Lq distance function is nonlinear, we employ rood2d function in Matlab to solve this nonlinear equation. This is the trajectory which optimizes (4.10) for w between target pair that has movement type 2. If movement type of pair is 3, as type 3 has a two-piece nondominated frontier, we check both of the pieces. We find the point where the straight line part of the nondominated frontier intersects the linear objective function in addition to the point optimizing (4.10) on the curved part. We select the smaller one as the trajectory which optimizes (4.10) for w between the target pair that has movement type 3. Therefore, we reduce the infinitely many trajectory options between each target pair to one. The resulting problem turns into the classical TSP, where a single trajectory is assumed between each target pair. The mathematical model for this problem has $H_{ij} = 1$ for all target pairs.

Before starting the route, we solve a MOTSP, and at each visit to the targets, we solve a MOSHPP. As we visit a new target, the cardinality of set M decreases by 1 and we have a smaller problem to be solved. Even though our problem size decreases at each execution of Phase 2, the input preparation step (finding the nondominated frontiers of the trajectories between each target pair, and then, finding the best trajectory between each target pair) for the new setting requires considerable computational time. Also, we still solve NP-hard problems as explained above, and they require substantial computational time for large-sized problems. For our application, typically, the number of targets a UAV can visit is not too many and therefore the main computational burden is not in solving these problems to optimality. However, our approach should be employed in real-time, and we need to obtain results in short durations in order not to delay the flight of the UAV. To reduce the computational burden in Phase 2, we develop a heuristic that reduces the required computations for the input preparation step. In this heuristic, we may sacrifice from optimality for a gain in computational time. We next explain this heuristic.

4.2.2 k-Closest Heuristic

As explained above, we develop a heuristic to reduce the computational burden. We first partitioned the algorithm into stages and recorded their solution times. The input preparation phase turns out to be the most time-consuming stage. In this stage, we structure the nondominated frontiers for all target pairs. For this, we first find the extreme nondominated points. Then we find solutions in the middle regions using the heuristic developed by Tezcaner Öztürk (2013). This heuristic employs Golden Section Search and partitions the solution space to find the solution that gives the smallest radar detection threat value for a given distance value. We use this heuristic once for each target pair that has more than one nondominated point. Then, we fit *Lq* distance function to their nondominated frontiers. On this function, we find the solution that minimizes equation (4.10) for a *w* value. That is, we find the intersection of the *Lq* distance function with the assumed preference function by equating their slopes. We use binary search to fit the *Lq* distance function, and solve two nonlinear equations simultaneously to find the solution minimizing (4.10).

We employ these steps for all target pairs, even if they are very distant and using the trajectories between these pairs in our solution is unlikely. To exploit this observation, for each target, we choose *k* closest targets, and structure only these *k* pairs' nondominated frontiers. To decide on these closest targets, we evaluate their extreme nondominated points' preference function values. For each target, the *k* targets with the smallest preference function values are selected and the nondominated frontiers between those targets are structured. From these nondominated frontiers, the best solutions for a given *w* are selected. If one target is not among the *k* closest targets of another target, then the extreme nondominated point, with the smallest preference function value, is used as the best solution, although another solution on the frontier could be minimizing the preference function. We next give the steps of the *k*-closest heuristic.

Let $M' = M \cup \{1, \ell\}$, $(i, j) \neq (1, \ell)$.

For each target $i \in M'$:

Step 1: Find the nondominated points (A_{ij} and B_{ij}) minimizing each objective, for each $j \in M', j \neq i$.

Step 2: Let U_{ij} =min ($U(A_{ij})$, $U(B_{ij})$). Choose k closest neighbors (k smallest U_{ij} values) of target i and place them in set Q_i.

Step 3: Fully structure the nondominated frontiers of (i, j) pairs, $j \in Q_i$.

Step 4: Find solution x that minimizes $U(x)$ on the nondominated frontier of (i, j), $j \in Q_i$.

Step 5: Update $U_{ij} = U(x)$ for (i,j), $j \in Q_i$.

With this heuristic, we have the possibility of missing some good solutions of some target pairs. However, overall, we avoid computation of the nondominated frontiers for many target pairs that are not used in the resulting tours.

To obtain the best solution, we set k to $|T| - 1$. For $k = k', k' < |T| - 1$, we expect to obtain solutions that are at most as good as the results for $k = k' + 1$.

<u>k=0-Closest Heuristic</u>

This is the special case of k-closest heuristic. In this case, we do not structure any nondominated frontier. For each pair, the extreme nondominated point having the smaller preference function value, is used as the best solution. Using these trajectories, MOTSP and MOSHPP are solved.

4.2.3 Adaptive Algorithm

The aim of the k-closest heuristic is to decrease the solution times. The smaller the k, the shorter the solution times. However, in theory, the value of k is inversely proportional to the quality of the solution. Thus, when we determine k, we should take both objectives (the solution quality and the solution time) into consideration. Our motivation in this algorithm is to increase the value of k (that is set initially to 0) if the solution of MOSHPP has the possibility of forming a different route. Thus, we develop a mechanism that firstly checks if higher k values have the possibility of changing the route. If so, we increase the value of k and structure more nondominated frontiers between target pairs. In each execution of Phase 2, we compare the values of a threshold and a gap to decide on the value of k. Although it is better to find new values of the threshold and gap each time the comparison is made, we set an initial value to the threshold and use it for all iterations. We recompute the value of gap each time we execute Phase 2. We explain the details of our algorithm below.

The adaptive algorithm that determines the value of k for each iteration contains two phases. In the initialization phase, we structure the nondominated frontiers of all target pairs, find the best trajectory between each target pair and solve TSP using these best trajectories. We refer this version as k=(|T|-1)-closest heuristic, which actually gives the best solution for the current location of the targets. The objective function coefficient of the TSP model for target pair (i,j) is the weighted combination of c_{ijh}^1 and c_{ijh}^2 values in (4.1) and (4.2) for the best trajectory h. We also find the objective function coefficients of the TSP using k-closest heuristic for k =0 (k=0-closest heuristic). Afterwards, for each decision variable, we subtract the objective function coefficients of the TSP for k=(|T|-1)-closest heuristic from the objective function coefficients of the TSP for k=0-closest heuristic. Then, we set the maximum objective function coefficient difference value as a threshold. This threshold is used in deciding whether k=0-closest heuristic is sufficient or not for further iterations. We do not

update this threshold in the following iterations because calculating a new threshold is a time-consuming process. In the following iterations, we apply k=0-closest heuristic, and we find the best and the second best route by solving SHPP. To find the second best route, we add a constraint that prevents model to reach the best route of the previous model. Afterwards, we calculate a gap which is the difference between the preference function values of the best route and the second best route. If this gap is greater than the threshold, we use the solution of k=0-closest heuristic. Otherwise, we repeat this iteration for a new k value.

We next explain our motivation for updating the value of k. We assume that the trajectory with maximum difference (for the k=(|T|-1) and k=0-closest heuristics) is not used in the solution of the SHPP. If the best trajectory for that target pair was used, we have the possibility of including that trajectory to the SHPP and improving the solution of the SHPP by that difference. If this difference is large enough to change the best route (by making the second best route the best route if threshold is greater than gap), we calculate better coefficient values by increasing the value of k. The maximum value that k can take depends on the number of unvisited targets. When we are setting the new value of k, we initially partition the range [0, threshold] into equal-length intervals. The number of intervals equals the number of unvisited targets. We then set $k = |M| - \left\lfloor \frac{gap}{\text{interval-length}} \right\rfloor$.

To illustrate this method, we assume that there are 6 unvisited targets in a problem with 15 targets. In the ninth iteration, assume that the threshold is equal to 0.66 and the gap is equal to 0.2. As the gap is lower than the threshold, we need to determine a new k. We divide [0, 0.66] into 6 intervals of length 0.66/6=0.11. The intervals can be seen in Table 4.1. We then set k to 6-floor(0.2/0.11)=5.

Table 4.1 Representation of the intervals in the adaptive algorithm

Interval	1	2	3	4	5	6
Range	0-0.11	0.11-0.22	0.22-0.33	0.33-0.44	0.44-0.55	0.55-0.66

The gap falls in the second interval in Table 4.1. If the gap was higher than 0.66, we would accept the solution of k=0-closest heuristic. If the gap was 0.6, k=1-closest heuristic would be employed. Smaller values of the gap result in the need for higher precision in the solutions.

We next give the steps of the adaptive algorithm.

PHASE 1

Step 1: (k=(|T|-1)-closest heuristic) Structure the nondominated frontiers between all target pairs, find the best trajectory between each target pair, and find their corresponding objective function coefficients for the TSP. Let this be matrix C_1 of dimensions 1 x |T|.|T|+|T|-1. Solve the TSP for C_1.

Step 2: Find the objective function coefficients of TSP according to the best extreme nondominated points. Let this be matrix C_2 of dimensions 1 x |T|.|T|+|T|-1.

Step 3: Find the threshold = max $(c_{2,1} - c_{1,1}, c_{2,2} - c_{1,2}, \dots, c_{2,|T|.|T|+|T|-1} - c_{1,|T|.|T|+|T|-1})$

PHASE 2

Step 1: Solve SHPP using k=0-closest heuristic and let the preference function value of the best route be Y_1.

Step 2: Find the second best route of SHPP using k=0-closest heuristic and let the preference function value of the second best route be Y_2.

Step 3: Set $\text{gap} = Y_2 - Y_1$.

Step 4: If $\text{gap} \geq \text{threshold}$ then the solution of SHPP using k=0-closest heuristic is acceptable and go to step 1 for the next iteration. Otherwise, go to step 5.

Step 5: Find the $i\text{nterval} - \text{length} = \frac{\text{threshold}}{|M|}$.

Step 6: Find $k' = |M| - \left\lfloor \frac{gap}{i\text{nterval}-\text{length}} \right\rfloor$.

Step 7: Solve SHPP using $k = k'$-closest heuristic.

4.2.4 Results

4.2.4.1 Five-Target UAV Route Planning Problem

Our first setting has one base point and 4 targets to be visited. In total, we have 5 targets and 4 radar regions in a 400 km^2 terrain. We are inspired from the setting of Tezcaner Öztürk (2013) but changed the locations of some targets and radars to reach different tours after solving TSP for the original problem and our algorithms. The initial locations of the targets and radars can be seen in Figure 4.1. The first target is the base point where the UAV starts and terminates its route.

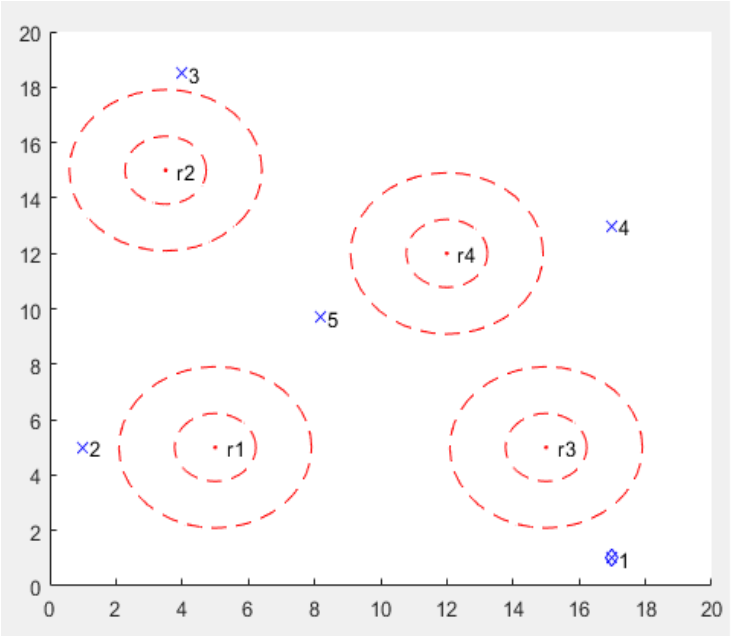


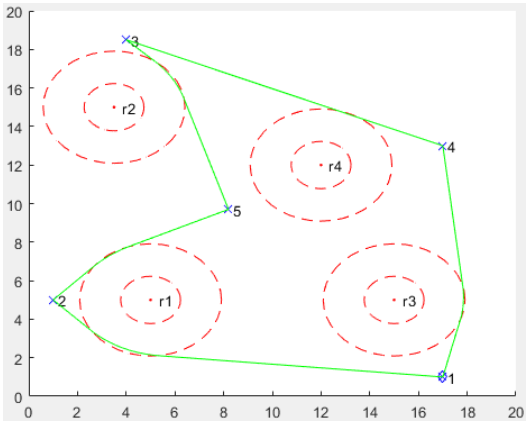
Figure 4.1 5-Target UAV Route Planning Problem

We change the locations of the targets randomly by satisfying that there is no target in any radar region. We also assume that the targets can move with a speed that is one-sixth of the speed of the UAV. We coded the algorithm in Matlab and used optimization package CPLEX to solve TSP and SHPP.

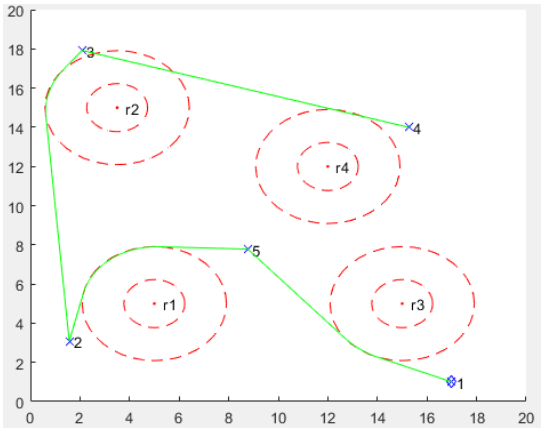
Illustration of the Algorithm, k=4, w*=0.2

We illustrate the real-time algorithm ($k = 4$) in Figure 4.2. The base point is target 1, and the UAV is at that target at time point 0. We assume that $w^* = 0.2$ is the importance the RP gives to our first objective, distance traveled, and 0.8 is the importance of the second objective, radar detection threat. We solve the TSP for the initial setting and obtain tour 1-4-3-5-2-1 (Figure 4.2 (a)). We compare the preference function values of the trajectories between targets 1-2 and 1-4, and the trajectory between 1-4 gives smaller preference function value. Therefore, the UAV follows the trajectory between targets 1 and 4. When we find this trajectory, we use the final location of target 4 as the point where the UAV catches target 4.

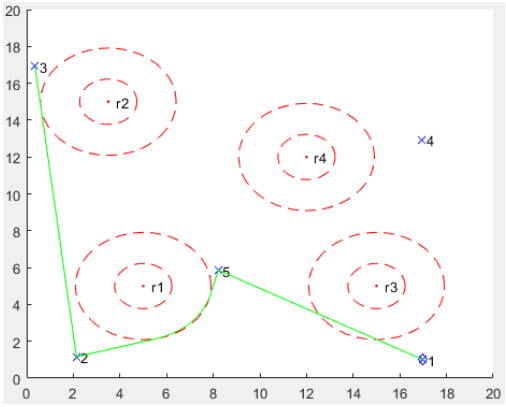
In Figure 4.2 (b), the new locations of the targets at the time the UAV visits target 4 is given. The SHPP is solved for the targets that have not been visited yet, and the path 4-3-2-5-1 is found. We then visit target 3 and observe the new locations of the targets not visited yet. The new path at target 3 is found as 3-2-5-1. In Figure 4.2 (c), we show the new path starting from target 3. Figure 4.2 (d) is the final path going to the base point.



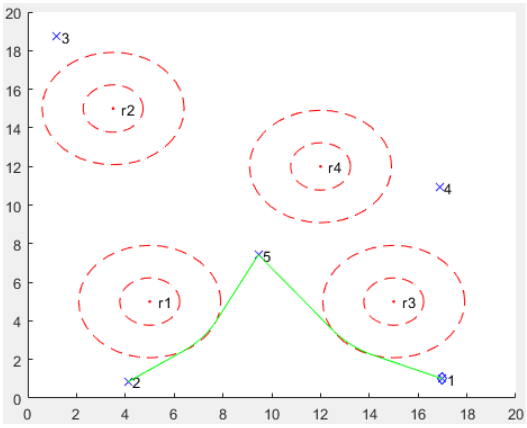
(a) The initial tour



(b) Hamiltonian path after visiting target 4



(c) Hamiltonian path after visiting target 3



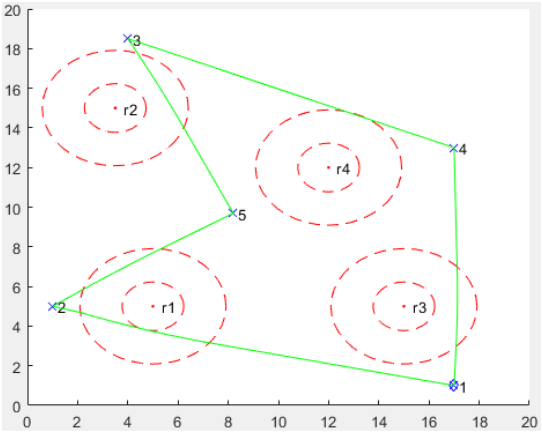
(d) Hamiltonian path after visiting target 2

Figure 4.2 Illustration of the Algorithm, $\boldsymbol{k = 4}$, $\boldsymbol{w^* = 0.2}$

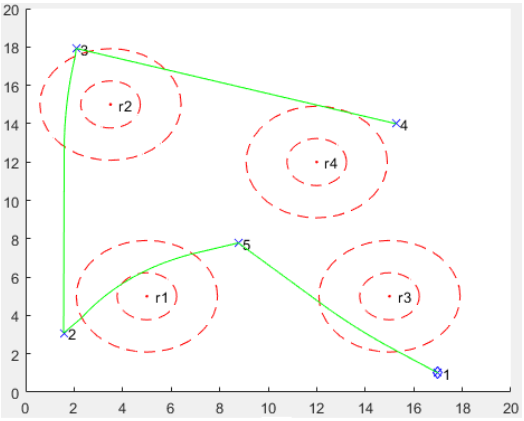
As can be seen from this example, the initial tour (1-4-3-5-2-1) is changed after the UAV visits target 4. The resulting tour is 1-4-3-2-5-1.

Illustration of the Algorithm, k=4, w*=0.8

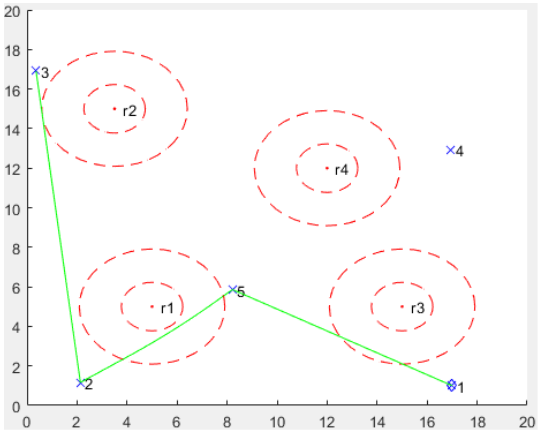
In this part, we illustrate the solution approach for $w^* = 0.8$. The importance of the first objective, distance traveled, is greater than the importance given to the second objective, radar detection threat. The results can be seen in Table 2. The initial tour is 1-4-3-5-2-1 (Figure 4.3 (a)) but the visiting order after target 4 changes in the second iteration (Figure 4.3 (b)). The final route followed by the UAV is 1-4-3-2-5-1.



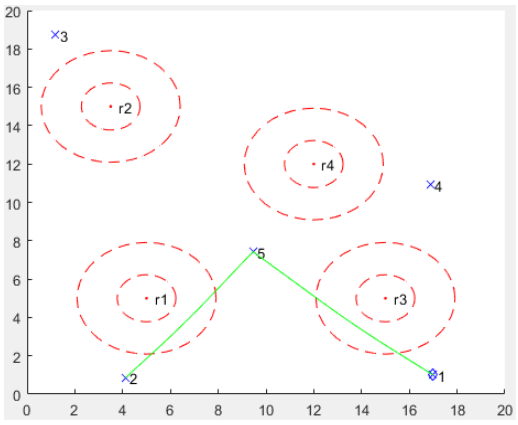
(a) The initial tour



(b) Hamiltonian path after visiting target 4



(c) Hamiltonian path after visiting target 3



(d) Hamiltonian path after visiting target 2

Figure 4.3 Illustration of the Algorithm, $\boldsymbol{k = 4}$, $\boldsymbol{w^* = 0.8}$

Computational Results of the Algorithm

We next give the results of the algorithm for $k = 0, 2, 3$ and 4. We expect $k = 0, 2$ and 3 to result in a final tour that is at most as good as $k = 4$.

We solve the algorithm RT-L for the following values of w; 0.2, 0.4, 0.6, and 0.8. The results can be seen in Table 4.2. For all w values, we obtain the same results when $k = 0, 2$ and, 3 and when $k = 4$, therefore we report their results in a single column. In the first three columns, we report the results if the UAV follows the initial route that we obtain in Phase 1 and do not update the path as new targets are visited. In the second three columns, we apply our algorithm and update the path as new targets are visited. The distance traveled (D) and radar detection threat (RDT) values of the trajectories are given for both cases, and in the second last row, the overall distance and radar detection threat values of the route of the UAV is given. For all w values, until the third iteration, the visit order is same. At the third iteration, the real-time algorithm changes the initial route. Overall, the routes of the real-time algorithm dominates the initial routes for w=0.2, 0.4, 0.6, and 0.8. As we give more importance

to the first objective (higher *w* values), in general, the total distance traveled decreases and the total radar detection threat increases. The preference function value of each tour is given in the last row. We calculate the preference function values of the final routes by multiplying each objective with their weights to compare the resulting tours. For all *w* values, updating the initial route results in better preference function values compared to using the initial route of Phase 1.

We report the solution times for $k = 0,2,3,$ and 4, after each target visit in Table 4.3. The solution durations generally decrease at each iteration since we have a smaller problem after each visit to one of the targets. The solution durations increase as the value of *k* increases, as expected.

Table 4.2 – Results of RT-L - 5-Target Problem

w	Initial Route			Real-Time Routing (*k=0,2,3,4*)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.2	1-4	13.4460	0.0515	1-4	13.4460	0.0515
	4-3	15.7020	0.0131	4-3	15.7020	0.0131
	3-5	13.9026	0.0219	3-2	16.7000	0.0052
	5-2	9.6470	0.1473	2-5	9.7541	0.0000
	2-1	14.4556	0.0000	5-1	9.0479	0.0977
	Tour Total	67.1532	0.2338	Tour Total	64.6500	0.1675
	Preference Function Value	13.6177		Preference Function Value	13.0640	
w	Initial Route			Real-Time Routing (|*k=0,2,3,4*)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.4	1-4	13.3080	0.4644	1-4	13.3080	0.4644
	4-3	15.6070	0.1516	4-3	15.6070	0.1516
	3-5	13.6957	0.3669	3-2	16.6690	0.0556
	5-2	8.9639	1.7954	2-5	9.7541	0.0000
	2-1	14.4556	0.0000	5-1	8.6754	1.0171
	Tour Total	66.0302	2.7783	Tour Total	64.0135	1.6887
	Preference Function Value	28.0791		Preference Function Value	26.6186	

Table 4.2 Continued

w	Initial Route			Real-Time Routing (*k=0,2,3,4*)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.6	1-4	13.1670	1.3758	1-4	13.1670	1.3758
	4-3	15.4030	0.8162	4-3	15.4030	0.8162
	3-5	13.3144	1.7154	3-2	16.6000	0.3129
	5-2	8.7273	2.9412	2-5	9.7541	0.0000
	2-1	14.4556	0.0000	5-1	8.4240	2.3089
	Tour Total	65.0673	6.8486	Tour Total	63.3481	4.8138
	Preference Function Value	41.7798		Preference Function Value	39.9344	
w	Initial Route			Real-Time Routing (*k=0,2,3,4*)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.8	1-4	13.1180	2.0708	1-4	13.1180	2.0708
	4-3	15.2200	2.1095	4-3	15.2200	2.1095
	3-5	13.1655	2.8071	3-2	16.5160	1.0035
	5-2	8.7083	3.1464	2-5	9.7541	0.0000
	2-1	14.4556	0.0000	5-1	8.3780	2.8281
	Tour Total	64.6674	10.1338	Tour Total	62.9861	8.0119
	Preference Function Value	53.7607		Preference Function Value	51.9913	

Table 4.3 – CPU Times for RT-L (seconds) - 5-Target Problem

k	Iteration				Total
	1	2	3	4	
0	2.7737	2.3909	2.7884	2.5944	10.547
2	26.209	18.617	7.065	2.823	54.713
3	28.387	23.556	7.152	2.635	61.730
4	28.028	27.335	7.798	2.610	65.771

In this problem, the initial route (IR) is dominated for all cases when compared to real-time routing (RT-L) (Figure 4.4). In addition, as the importance the RP gives to the first objective increases, the distance travelled decreases and the radar detection threat increases.

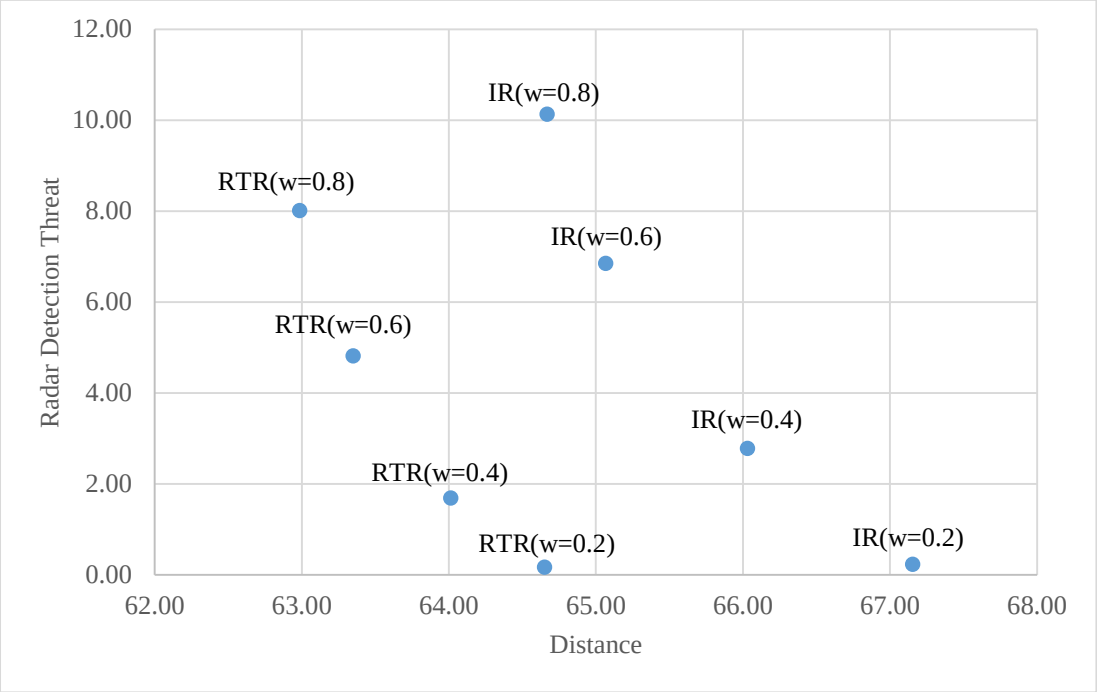


Figure 4.4 Results of IR and RT-L for the 5-target problem

4.2.4.2 Nine-Target UAV Route Planning Problem

Our second setting has one base point and 8 targets to be visited. In total, there are 9 targets and 4 radar regions in a 400 km^2 terrain. We are inspired from the setting of Türeci (2017). Target 1 is the base point again. We employ the same approach as in the 5-target example and change the locations of the targets randomly assuming that the targets can move with a speed that is one-sixth of the speed of the UAV.

Results of the Algorithm

We solve this problem for $k = 0,2,3$, and 8. We assume four different w values; 0.2, 0.4, 0.6, and 0.8. We give the results in Table 4.4. All the routes of the heuristics with $k = 0,2,3$ are the same as those of $k = 8$. The initial route changes at some point during the algorithm for $w = 0.2, 0.4, 0.6$, and we obtain solutions that have better

preference function values than the initial route by using our algorithm. Only for $w = 0.8$, we obtain the same tour even if we update the tour at each visit to the targets. We give a summary of the results in Figure 4.5. All solutions except for $w = 0.8$ found by the initial route (IR) are dominated by our algorithm RT-L. For $w = 0.8$ we find the same solution with IR and RT-L.

We report the solution times in Table 4.5. We gain in computational times considerably by using the heuristic, and our solution quality is not worsened.

Table 4.4 – Results of RT-L - 9-Target Problem

w	Initial Route			Real-Time Routing (*k=0,2,3,8*)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.2	1-5	5.5289	0.0000	1-5	5.5289	0.0000
	5-8	9.3764	0.0089	5-8	9.3764	0.0089
	8-3	7.3557	0.0059	8-3	7.3557	0.0059
	3-9	2.4750	0.0000	3-9	2.4750	0.0000
	9-4	5.5480	0.0000	9-4	5.5480	0.0000
	4-7	8.6491	0.0840	4-2	6.2181	0.0000
	7-2	5.9977	0.0191	2-7	6.0560	0.0138
	2-6	9.7032	0.0000	7-6	10.8670	0.0000
	6-1	10.7644	0.0133	6-1	10.7640	0.0133
	Tour Total	65.3984	0.1312	Tour Total	64.1891	0.0419
	Preference Function Value	13.1846		Preference Function Value	12.8713	
w	Initial Route			Real-Time Routing (*k=0,2,3,8*)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.4	1-5	5.5289	0.0000	1-5	5.5289	0.0000
	5-8	8.0256	2.0079	5-8	8.0256	2.0079
	8-3	7.3124	0.0750	8-3	7.3124	0.0750
	3-9	2.4750	0.0000	3-9	2.4750	0.0000
	9-4	5.5480	0.0000	9-4	5.5480	0.0000
	4-7	8.3567	0.8155	4-2	6.2181	0.0000
	7-2	5.9102	0.2150	2-7	5.9927	0.1563
	2-6	9.7032	0.0000	7-6	10.8670	0.0000
	6-1	10.6849	0.1523	6-1	10.6850	0.1523

Table 4.4 Continued

	Tour Total	63.5449	3.2657	Tour Total	62.6527	2.3915
	Preference Function Value	27.3774		Preference Function Value	26.4960	
w	Initial Route			Real-Time Routing (*k=0,2,3,8*)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.6	1-5	5.5289	0.0000	1-5	5.5289	0.0000
	5-8	7.3942	3.4674	5-8	7.3942	3.4674
	8-3	7.2261	0.3786	8-3	7.2261	0.3786
	3-9	2.4750	0.0000	3-9	2.4750	0.0000
	9-4	5.5480	0.0000	9-4	5.5480	0.0000
	4-7	8.1451	1.9291	4-2	6.2181	0.0000
	7-2	5.8293	0.5949	2-7	5.9253	0.4771
	2-6	9.7032	0.0000	7-6	10.8670	0.0000
	6-1	10.5306	0.7541	6-1	10.5310	0.7538
	Tour Total	62.3804	7.1241	Tour Total	61.7136	5.0769
	Preference Function Value	40.2779		Preference Function Value	39.0589	
w	Initial Route			Real-Time Routing (*k=0,2,3,8*)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.8	1-5	5.5289	0.0000	1-5	5.5289	0.0000
	5-8	7.3892	3.4926	5-8	7.3892	3.4926
	8-3	7.1698	0.7996	8-3	7.1698	0.7996
	3-9	2.4750	0.0000	3-9	2.4750	0.0000
	9-4	5.5480	0.0000	9-4	5.5480	0.0000
	4-7	8.1003	2.4475	4-7	8.1003	2.4475
	7-2	5.8098	0.7940	7-2	5.8098	0.7940
	2-6	9.7032	0.0000	2-6	9.7032	0.0000
	6-1	10.4160	1.7120	6-1	10.4160	1.7120
	Tour Total	62.1402	9.2457	Tour Total	62.1402	9.2457
	Preference Function Value	51.5613		Preference Function Value	51.5613	

Table 4.5 – CPU Times for RT-L (seconds) - 9-Target Problem

k	Iteration								Total
	1	2	3	4	5	6	7	8	
0	1.1474	4.3572	2.9009	0.5953	0.5935	1.0255	4.8634	3.3068	18.7899
2	12.1974	16.4278	8.6395	16.1356	8.8076	9.3296	7.5073	3.2480	82.2928
3	35.2190	32.9728	28.7157	16.1758	16.3130	6.9202	9.3035	3.0500	148.6700
8	60.6453	57.6695	33.4395	24.7435	17.3754	9.5500	7.3270	3.1666	213.9168

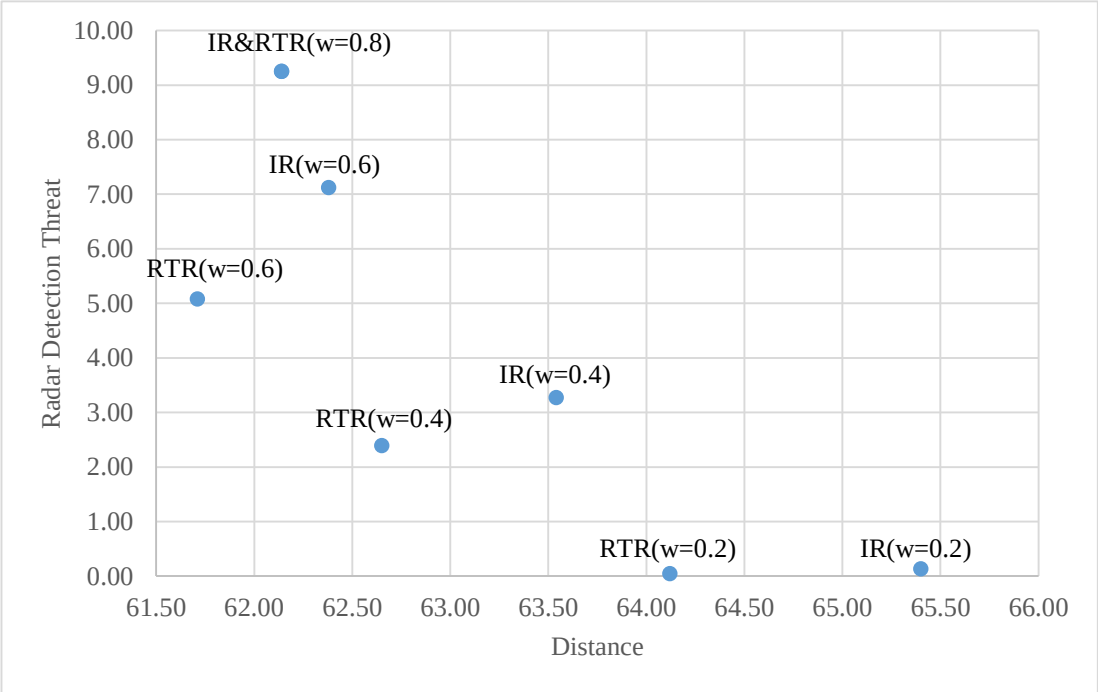


Figure 4.5 Results of IR and RT-L for the 9-target problem

Effects of *w* Estimation

We find a narrow range for the value of *w* using the interactive algorithm developed by Türeci (2017). We then set *w* to the middle value of this range and execute our algorithm. However, there is a possibility that we are using a wrong value for *w* and thus finding a route that is not preferred by the RP. To estimate the error in the

estimation of *w*, we also compute the routes as if *w* equals the end points of its range. We report the results for the 9-target problem.

Suppose we obtain the range [0.6, 0.8] covering *w*. We thus set *w* to 0.7 and execute our algorithm. In Table 4.6, we give the results for the case where actual *w* is 0.6. We find the same route for both cases and, since we use different trajectories between target pairs, the preference function value of our estimation is barely worse than the preference function value of the actual *w*. In Table 4.7, we assume that the actual *w* is 0.8 when our estimation of *w* is 0.7. By coincidence, our estimation results in a better route. For these two cases, the preference function values deviate by at most 2.5% which can be tolerable for the RP.

Table 4.6 Effects of $w = 0.7$ estimation if actual $w = 0.6$

	Actual (0.6)	Estimation (0.7)
Preference Function Value	39.06	39.33
Route	1-5-8-3-9-4-2-7-6-1	Same
Change	-0.70%	

Table 4.7 Effects of $w = 0.7$ estimation if actual $w = 0.8$

	Actual (0.8)	Estimation (0.7)
Preference Function Value	51.56	50.46
Route	1-5-8-3-9-4-7-2-6-1	1-5-8-3-9-4-2-7-6-1
Change	2.14%	

4.2.4.3 Fifteen-Target UAV Route Planning Problem

Our third setting has one base point and 14 targets to be visited. In total, there are 15 targets and 4 radar regions in a 400 km^2 terrain. Target 1 is the base point again. We employ the same approach as in the 5-target example and change the locations of the targets randomly assuming that the targets can move with a speed that is one-sixth of the speed of the UAV.

Results of the Algorithm

We solve this problem for k = 0,2,3, and 14. We assume four different *w* values; 0.2, 0.4, 0.6, and 0.8. We give the results in Table 4.8. All the routes of the heuristic with k = 0,2,3 are the same as those of k = 14. The initial route changes at some point during the algorithm for $w = 0.2, 0.4, 0.6$, and 0.8, and with our algorithm we obtain solutions that have better preference function values than the initial route. We give a summary of the results in Figure 4.6.

We report the solution durations in Table 4.9. We again gain in computational times considerably by using the heuristic, and our solution quality is not worsened.

Table 4.8 – Results of RT-L - 15-Target Problem

w	Initial Route			Real-Time Routing (*k=0,2,3,14*)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.2	1-4	15.7160	0.0141	1-4	15.7160	0.0141
	4-5	2.0998	0.0000	4-5	2.0998	0.0000
	5-6	5.6690	0.0000	5-6	5.6690	0.0000
	6-7	7.2094	0.0000	6-7	7.2094	0.0000
	7-8	6.3537	0.0000	7-8	6.3537	0.0000
	8-9	8.0721	0.0000	8-9	8.0721	0.0000
	9-10	5.8702	0.0000	9-12	2.8374	0.0000
	10-11	2.5173	0.0000	12-13	1.4225	0.0000
	11-3	1.9984	0.0000	13-10	2.6470	0.0000
	3-12	6.6119	0.0000	10-11	3.8408	0.0000
	12-13	2.8764	0.0000	11-3	2.8324	0.0000
	13-14	7.1553	0.0000	3-2	7.8616	0.0186
	14-15	3.5492	0.0000	2-14	4.5913	0.0000
	15-2	7.5457	0.0937	14-15	4.5459	0.0000
	2-1	9.6113	0.0598	15-1	1.7877	0.0000
	Tour Total	92.8557	0.1676	Tour Total	77.4866	0.0327

Table 4.8 Continued

	Objective	18.7052		Objective	15.5235	
w	Initial Route			Real-Time Routing (*k=0,2,3,14*)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.4	1-4	15.6530	0.1437	1-4	15.6530	0.1437
	4-5	2.0998	0.0000	4-5	2.0998	0.0000
	5-6	5.6690	0.0000	5-6	5.6690	0.0000
	6-7	7.2094	0.0000	6-7	7.2094	0.0000
	7-8	6.3537	0.0000	7-8	6.3537	0.0000
	8-9	8.0721	0.0000	8-9	8.0721	0.0000
	9-10	5.8702	0.0000	9-12	2.8374	0.0000
	10-11	2.5173	0.0000	12-13	1.4225	0.0000
	11-3	1.9984	0.0000	13-10	2.6470	0.0000
	3-12	6.6119	0.0000	10-11	3.8408	0.0000
	12-13	2.8764	0.0000	11-3	2.8324	0.0000
	13-14	7.1553	0.0000	3-2	7.7352	0.2390
	14-15	3.5492	0.0000	2-14	4.5913	0.0000
	15-2	7.1176	1.0202	14-15	4.5459	0.0000
	2-1	9.3388	0.6609	15-1	1.7877	0.0000
	Tour Total	92.0921	1.8248	Tour Total	77.2972	0.3827
	Objective	37.9317		Objective	31.1485	
w	Initial Route			Real-Time Routing (*k=0,2,3,14*)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.6	1-4	15.5340	0.7011	1-4	15.5340	0.7011
	4-5	2.0998	0.0000	4-5	2.0998	0.0000
	5-6	5.6690	0.0000	5-6	5.6690	0.0000
	6-7	7.2094	0.0000	6-7	7.2094	0.0000
	7-8	6.3537	0.0000	7-8	6.3537	0.0000
	8-9	8.0721	0.0000	8-9	8.0721	0.0000
	9-10	5.8702	0.0000	9-12	2.8374	0.0000
	10-11	2.5173	0.0000	12-13	1.4225	0.0000
	11-3	1.9984	0.0000	13-10	2.6470	0.0000
	3-12	6.6119	0.0000	10-11	3.8408	0.0000
	12-13	2.8764	0.0000	11-3	2.8324	0.0000
	13-14	7.1553	0.0000	3-2	7.5418	0.9628
	14-15	3.5492	0.0000	2-14	4.5913	0.0000
	15-2	6.8495	2.2177	14-15	4.5459	0.0000

Table 4.8 Continued

	2-1	9.0636	1.9453	15-1	1.7877	0.0000
	Tour Total	91.4298	4.8641	Tour Total	76.9848	1.6639
	Objective	56.8035		Objective	46.8564	
w	Initial Route			Real-Time Routing (*k=0,2,3,14*)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.8	1-4	15.4310	1.7556	1-4	15.4310	1.7556
	4-5	2.0998	0.0000	4-5	2.0998	0.0000
	5-6	5.6690	0.0000	5-6	5.6690	0.0000
	6-7	7.2094	0.0000	6-7	7.2094	0.0000
	7-8	6.3537	0.0000	7-8	6.3537	0.0000
	8-9	8.0721	0.0000	8-9	8.0721	0.0000
	9-13	4.6981	0.0000	9-12	2.8374	0.0000
	13-12	1.4533	0.0000	12-13	1.4225	0.0000
	12-10	2.5115	0.0000	13-10	2.6470	0.0000
	10-11	3.8408	0.0000	10-11	3.8408	0.0000
	11-3	2.8324	0.0000	11-3	2.8324	0.0000
	3-14	11.9797	0.2210	3-2	7.4626	1.5914
	14-15	3.5492	0.0000	2-14	4.5913	0.0000
	15-2	6.8063	2.6430	14-15	4.5459	0.0000
	2-1	8.9879	2.7093	15-1	1.7877	0.0000
	Tour Total	91.4942	7.3289	Tour Total	76.8026	3.3470
	Objective	74.6611		Objective	62.1115	

Table 4.9 – CPU Times for RT-L (seconds) - 15-Target Problem

k	Iteration							
	1	2	3	4	5	6	7	
0	7.5060	1.3846	1.2349	0.9994	0.8557	0.8280	0.7513	
2	36.7091	16.9697	9.9140	8.2834	9.4224	7.5305	8.2567	
3	44.3867	35.3451	27.9331	23.4369	24.7633	20.8668	32.5523	
14	146.5959	129.3704	112.5418	89.9264	64.8744	56.1650	49.1579	

Table 4.9 Continued

k	Iteration							Total
	8	9	10	11	12	13	14	
0	0.8657	0.7436	0.7156	0.6961	2.8312	0.6208	0.5328	20.5658
2	14.2409	11.7810	7.2457	9.2997	11.6953	2.7654	0.5259	154.6398
3	25.6776	17.9709	17.2812	18.5023	13.3634	2.7619	0.5142	305.3557
14	43.1888	29.3928	24.3432	19.0597	14.3084	3.3948	0.5582	782.8776

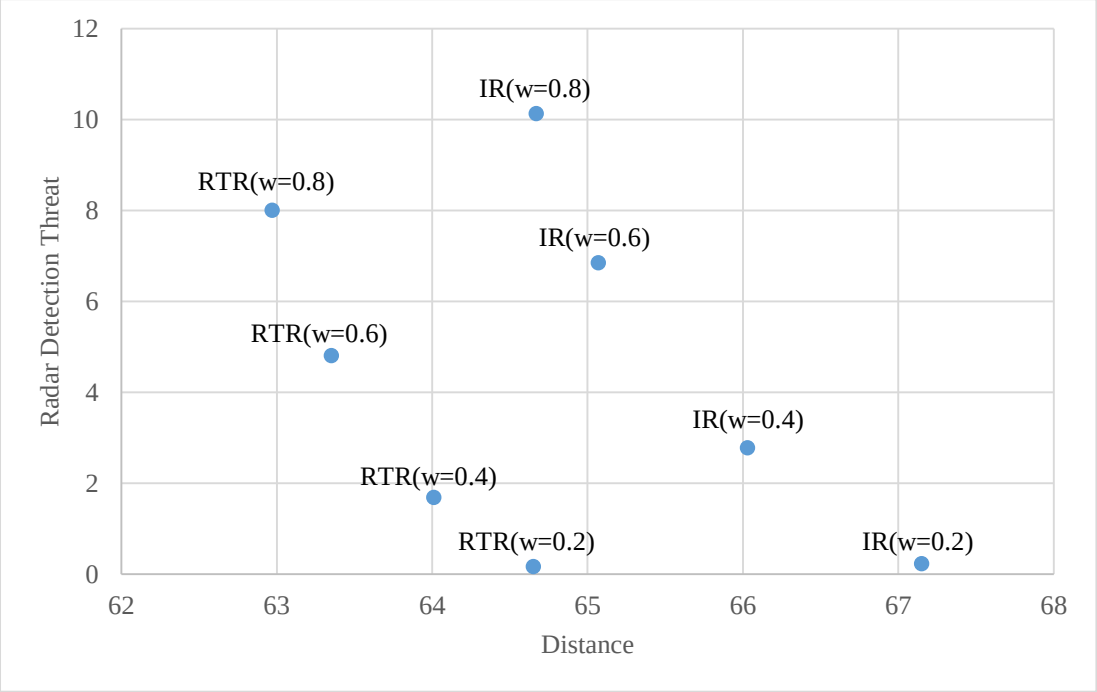


Figure 4.6 Results of IR and RT-L for the 15-target problem

4.3 Quadratic Preference Function (RT-Q)

We apply our algorithms to find the most preferred solutions of RPs with quadratic preference functions. In this chapter, we explain the structure of quadratic preference functions and how we apply our algorithms for these functions.

Quadratic preference functions are one type of quasiconvex preference functions (where all objectives are minimized). Quasiconvex preference functions cover a large set of preference function. They are considered to represent human behavior well. For these functions, the marginal rate of substitution decreases. The structure of the quadratic preference function with two objectives is given in (4.11), where the square of two objectives are aggregated with weight w. The importance the RP gives to the first objective is w and $1-w$ is the importance of the second objective. As we minimize both objectives, the solution that minimizes $U(x)\ is$ the most preferred solution of the RP.

$$U(x) = w\, z_1(x)^2 + (1-w)\, z_2(x)^2 \text{ where } 0 < w < 1 \qquad (4.11)$$

In $U(x)$ function, $z_i(x)$ is the i^{th} objective's value, and the objective values are normalized between 0 and 1 so that the weights (w and $1-w$) roughly represent the relative importance the RP associates with the corresponding objectives.

We apply the real-time algorithm to these functions. Different than the linear preference functions, the most preferred tour is not necessarily made of the most preferred trajectories between target pairs. This is shown with an example in Tezcaner Öztürk (2013). Therefore, to find the most preferred tour, we need to input all efficient trajectories to the model. For our problem, this requires including infinitely many efficient trajectories to the model and solving a mixed integer nonlinear program (with nonlinear objective function and constraints) each time the UAV visits one target, which is not computationally favorable for dynamic routing. We thus select a subset of the efficient trajectories and introduce those trajectories to the models. We next explain the different approaches we employ for selecting a subset of efficient trajectories.

4.3.1 n -Trajectories between All Target Pairs

In our first method, we select n efficient trajectories for each target pair on their nondominated frontier, and we use these n trajectories for each target pair in the solution of TSP or SHPP. The value of n affects the problem size drastically because the number of decision variable in TSP or SHPP is $|T|.|T|.\ n$ +($|T|$-1).

- Choosing n Trajectories ($n > 1$)

We reduce the infinitely many trajectory options between each target pair to n which is greater than 1. In other words, we discretize the continuous frontier with n different points. Thus, we set H_{ij} to n for all target pairs in the models for TSP and SHPP. We select n trajectories in two different ways.

- $n - 1$ equally-dispersed trajectories & best trajectory

In this method, we select n trajectories for each target pair including the most preferred trajectory for that target pair. The remaining $n - 1$ trajectories are equally-dispersed points on the nondominated frontier of that target pair, where two of them are the extreme nondominated points.

To find the most preferred trajectory for a target pair for a w, we find the trajectory that optimizes (4.11) for w between that target pair. For this, we solve a minimization problem. Our objective function is the quadratic objective function (4.12), and our constraint finds the most preferred solution on the nondominated frontier. In the formulation, d and r represent the distance and radar detection threat values for that target pair, respectively. $\mathrm{d}'$and r' are the normalized versions of d and r. We normalize these values by subtracting their minimum possible values and then dividing with their range.

$$\min(\mathrm{d}', \mathrm{r}') = w\ \mathrm{d}'^2 + (1 - w)\ \mathrm{r}'^2 \tag{4.12}$$

S.to:

$$(1 - \frac{(\mathrm{d}-\mathrm{dmin})}{(\mathrm{dmax}-\mathrm{dmin})})^q + (1 - \frac{(\mathrm{r}-\mathrm{rmin})}{(\mathrm{rmax}-\mathrm{rmin})})^q = 1 \tag{4.13}$$

$$\mathrm{r} - (\mathrm{slopeofline}) * \mathrm{d} = \mathrm{constantofline} \tag{4.14}$$

Constraint (4.13) makes sure that the solution optimizing (4.12) is on the *Lq* distance function and constraint (4.14) makes sure that the solution optimizing (4.12) is on the straight line part of the nondominated frontier, which has a slope *slopeofline* and an intersection point with the y-axis *constantofline*. When movement type of pair is 2, we only use constraint (4.13) since the nondominated frontier is only made of the *Lq* distance function. When the movement type of pair is 3, we optimize (4.12) twice; once subject to (4.13) and once subject to constraint (4.14). After solving both models, we select the solution which has the smaller objective function value as the trajectory which optimizes (4.11) for *w* between that target pair. As the *Lq* distance function is nonlinear, we employ confuneq and objfun functions in Matlab to solve this problem.

In order to find $(n - 1)$ equally-dispersed points, we firstly scale the nondominated frontier between 0 and 1 for all objectives, such that every distance and radar detection threat value are proportionally declined to a value between 0 and 1. The graph that has normalized distance and radar detection threat values can be seen in Figure 4.7. We then find the nadir point, which is the point with the worst objective values. For the normalized objective values, the nadir point is (1, 1). We divide the right angle at (1, 1) to $n - 2$ to find the angle θ structuring the location of the equally-dispersed points on the nondominated frontier.To determine the coordinates of equally-dispersed points on the nondominated frontier, from point (1,1), $n - 3$ straight lines are sent to the nondominated frontier as the angle between each neighbour line is θ.

Afterwards, we find all intersection points of these straight lines and nondominated frontier. As nondominated frontier is nonlinear, we employ rood2d function, and this function finds us distance and radar detection threat value which satisfy both of the equations (4.15) and (4.16). At last, we have n trajectories for each pair to use in the mathematical models of MOTSP or MOSHPP.

$$(1 - \mathrm{d}')^q + (1 - \mathrm{r}')^q = 1 \qquad (4.15)$$

$$\tan(\theta)=(1-\mathrm{r}')/(1-\mathrm{d}') \qquad (4.16)$$

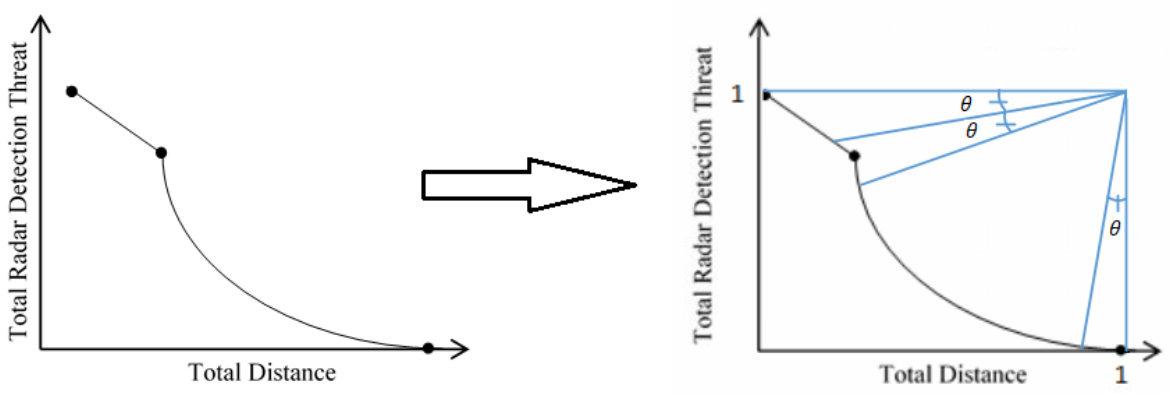


Figure 4.7 Normalized graph and equally-dispersed lines

- n equally-dispersed Trajectories

In this case, as finding the best trajectory for each target pair is time-consuming, we find n equally-dispersed trajectories. The calculation of angle θ and the number of straight lines to be sent change in this method as θ is determined by dividing the right angle to $n - 1$ and we have $n - 2$ lines.

- Choosing n Trajectories ($n = 1$)

As there are no known polynomial-time algorithms for the TSP (Dasgupta et al., 2006), reduction of the number of the decision variables is very crucial to reduce the computation times. Therefore, we developed the version that only one trajectory for

each pair is sent to the TSP or SHPP. This decreases the number of the decision variables to $|T|.|T|+(|T|-1)$ from $|T|.|T|.n+(|T|-1)$.

- Using the best trajectory ($n = 1/B$)

The best trajectory for each target pair is found using the method explained above. This trajectory for each target pair is used in the solution of MOTSP or MOSHPP.

- Using the best extreme trajectory ($n = 1/E$)

To find the best trajectory, we need to find the structure of the nondominated frontiers between each target pair, which requires substantial computational effort. In order to further reduce the computation times, we use the extreme trajectory with the smaller quadratic preference function (4.11) value for each target pair.

4.3.2-Results

4.3.2.1-Five-Target UAV Route Planning Problem

We solve the problem introduced in Section 4.2.4.1 for a RP who has quadratic preference function. We assume four different w values; 0.2, 0.4, 0.6, and 0.8. We give the results in Table 4.10. For all w values, the best routes for the cases $n = 1$ with the best trajectory (n=1/B), $n = 1$ with the best extreme trajectory (n=1/E),$n = 5$ equally-dispersed (n=5/WB), and $n = 5 - 1$ equally–dispersed with the best trajectory (n=5/B) are the same. We thus report their results in a single column. For all cases and for all w values, until the third iteration, the visiting order is the same with the initial route. At the third iteration, RT-Q follows a different path compared to the initial route. Overall, the results of the real-time algorithm are better than the initial routes since they dominate the initial routes (Figure 4.8). Again as we give more

importance to the first objective (higher *w* values), in general, the total distance traveled decreases and the total radar detection threat increases. We calculate the preference function values of the final routes by multiplying each squared objective with their weights to compare the resulting tours. For all *w* values, updating the initial route results in better preference function values compared to using the initial route of Phase 1.

We report the solution durations in Table 4.11. When n=5, solving nonlinear models of MOTSP or MOSHPP takes a lot of time due to the presence of higher number of decision variables. If the best trajectory is included in n, it takes more time because finding the best trajectory takes more time than finding equally distributed points. We again gain in computational times considerably by using one trajectory between each target pair, and our solution quality is not worsened.

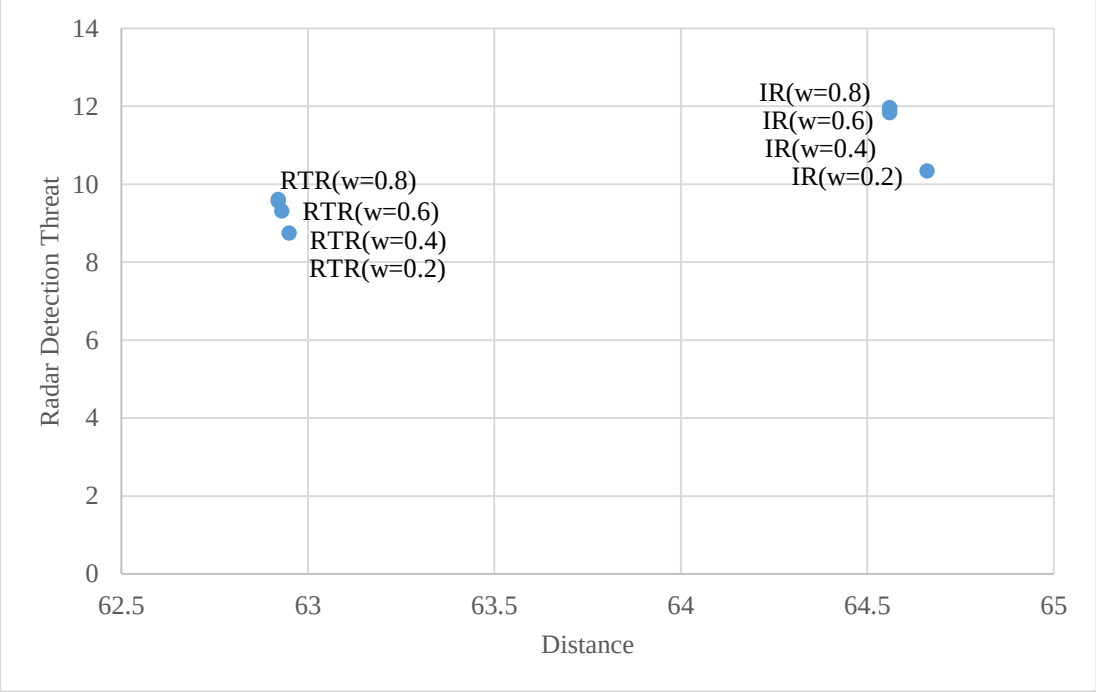


Figure 4.8 Results of IR and RT-Q for the 5-target problem

Table 4.10 – Results of RT-Q- 5-Target Problem

W	Initial Route			Real-Time Routing(n=1 & B/E, n=5(B&WB))		
	Trajectory	D	RDT	Trajectory	D	RDT
0.2	1-4	13.1130	2.2477	1-4	13.1130	2.2477
	4-3	15.2060	2.3020	4-3	15.2060	2.3020
	3-5	13.1651	2.8134	3-2	16.4980	1.3699
	5-2	8.7228	2.9798	2-5	9.7541	0.0000
	2-1	14.4556	0.0000	5-1	8.3779	2.8295
	Tour Total	64.6625	10.3429	Tour Total	62.9490	8.7491
	Objective	921.8282		Objective	853.7527	
W	Initial Route			Real-Time Routing(n=1 & B/E, n=5(B&WB)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.4	1-4	13.1120	2.2980	1-4	13.1120	2.2980
	4-3	15.1900	2.6382	4-3	15.1900	2.6382
	3-5	13.1392	3.3655	3-2	16.4960	1.4692
	5-2	8.6624	3.5351	2-5	9.7541	0.0000
	2-1	14.4556	0.0000	5-1	8.3751	2.9136
	Tour Total	64.5592	11.8368	Tour Total	62.9272	9.3190
	Objective	1751.2220		Objective	1636.0393	
w	Initial Route			Real-Time Routing(n=1 & B/E, n=5(B&WB)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.6	1-4	13.1120	2.3088	1-4	13.1120	2.3088
	4-3	15.1880	2.7279	4-3	15.1880	2.7279
	3-5	13.1392	3.3655	3-2	16.4960	1.4940
	5-2	8.6624	3.5351	2-5	9.7541	0.0000
	2-1	14.4556	0.0000	5-1	8.3736	3.0420
	Tour Total	64.5572	11.9373	Tour Total	62.9237	9.5727
	Objective	2557.5789		Objective	2412.2898	
w	Initial Route			Real-Time Routing(n=1 & B/E, n=5(B&WB)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.8	1-4	13.1120	2.3123	1-4	13.1120	2.3123
	4-3	15.1880	2.7576	4-3	15.1880	2.7576
	3-5	13.1392	3.3655	3-2	16.4960	1.5026
	5-2	8.6624	3.5351	2-5	9.7541	0.0000
	2-1	14.4556	0.0000	5-1	8.3736	3.0420
	Tour Total	64.5572	11.9705	Tour Total	62.9237	9.6145
	Objective	3362.7642		Objective	3186.0013	

Table 4.11 – CPU Times for RT-Q (seconds) - 5-Target Problem

n	Iteration				Total
	1	2	3	4	
1 (E)	3.5978	2.9364	2.9693	2.6884	12.1919
1 (B)	32.1996	25.9158	7.6884	2.8327	68.6365
5 (WB)	28.8324	24.7993	11.4747	7.2165	72.3229
5 (B)	166.3001	26.0832	11.9992	2.9802	207.3627

4.3.2.2-Nine-Target UAV Route Planning Problem

We solve the 9-target problem we introduced in Section 4.2.4.2 for the cases n=1/B and n=1/E. Cplex could not find a solution to the 9 target problem for n values greater than 1. We assume four different w values; 0.2, 0.4, 0.6, and 0.8. We give the results in Table 4.12. The initial route changes at some point during the algorithm for $w = 0.4, 0.6, 0.8$, and we obtain solutions that have better preference function values than the initial route by using our algorithm. Only for $w = 0.2$, we obtain the same initial tour even if we update the tour at each visit to the targets.

We report the solution times in Table 4.13. We gain in computational times considerably by using the best extreme trajectory rather than the best trajectory, and our solution quality is not worsened.

Table 4.12 – Results of RT-Q- 9-Target Problem

w	Initial Route			Real-Time Routing(n=1 & B/E)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.2	1-5	5.5289	0.0000	1-5	5.5289	0.0000
	5-8	7.0578	3.8862	5-8	7.0578	3.8862
	8-3	7.1645	0.8881	8-3	7.1645	0.8881
	3-9	2.4750	0.0000	3-9	2.4750	0.0000
	9-4	5.5480	0.0000	9-4	5.5480	0.0000
	4-7	8.0993	2.4709	4-7	8.0993	2.4709
	7-2	5.8085	0.8274	7-2	5.8085	0.8274
	2-6	9.7032	0.0000	2-6	9.7032	0.0000
	6-1	10.4080	1.8521	6-1	10.4080	1.8521
	Tour Total	61.7932	9.9247	Tour Total	61.7932	9.9247
	Objective	842.4796		Objective	842.4796	
w	Initial Route			Real-Time Routing(n=1 & B/E)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.4	1-5	5.5289	0.0000	1-5	5.5289	0.0000
	5-8	7.0578	3.8862	5-8	7.0578	3.8862
	8-3	7.1629	0.9385	8-3	7.1629	0.9385
	3-9	2.4750	0.0000	3-9	2.4750	0.0000
	9-4	5.5480	0.0000	9-4	5.5480	0.0000
	4-7	8.0971	2.5491	4-2	6.2181	0.0000
	7-2	5.8083	0.8368	2-7	5.9044	0.7228
	2-6	9.7032	0.0000	7-6	10.8670	0.0000
	6-1	10.3989	2.1083	6-1	10.4000	2.0663
	Tour Total	61.7801	10.3189	Tour Total	61.1621	7.6138
	Objective	1590.6001		Objective	1531.1030	
w	Initial Route			Real-Time Routing(n=1 & B/E)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.6	1-5	5.5289	0.0000	1-5	5.5289	0.0000
	5-8	7.0578	3.8862	5-8	7.0578	3.8862
	8-3	7.1627	0.9485	8-3	7.1627	0.9485
	3-9	2.4750	0.0000	3-9	2.4750	0.0000
	9-4	5.5480	0.0000	9-4	5.5480	0.0000
	4-7	8.0969	2.5626	4-2	6.2181	0.0000
	7-2	5.8083	0.8383	2-7	5.9044	0.7241
	2-6	9.7032	0.0000	7-6	10.8670	0.0000

Table 4.12 Continued

	6-1	10.3987	2.1321	6-1	10.3990	2.1193
	Tour Total	61.7795	10.3677	Tour Total	61.1609	7.6781
	Objective	2333.0197		Objective	2267.9747	
w	Initial Route			Real-Time Routing(n=1 & B/E)		
	Trajectory	D	RDT	Trajectory	D	RDT
0.8	1-5	5.5289	0.0000	1-5	5.5289	0.0000
	5-8	7.0578	3.8862	5-8	7.0578	3.8862
	8-3	7.1627	0.9513	8-3	7.1627	0.9513
	3-9	2.4750	0.0000	3-9	2.4750	0.0000
	9-4	5.5480	0.0000	9-4	5.5480	0.0000
	4-7	8.0969	2.5660	4-2	6.2181	0.0000
	7-2	5.8083	0.8386	2-7	5.9044	0.7244
	2-6	9.7032	0.0000	7-6	10.8670	0.0000
	6-1	10.3987	2.1395	6-1	10.3990	2.1364
	Tour Total	61.7795	10.3816	Tour Total	61.1609	7.6983
	Objective	3074.9208		Objective	3004.3773	

Table 4.13 – CPU Times for RT-Q (seconds) - 9-Target Problem

n	Iteration								Total
	1	2	3	4	5	6	7	8	
1 (E)	195.5762	14.2673	3.8205	1.3335	1.1978	0.733	5.5267	2.4164	224.8714
1 (B)	155.2906	68.1673	35.8789	25.8589	18.0768	7.2304	9.9618	2.5387	323.0034

CHAPTER 5

CONCLUSIONS

In this thesis, we consider the real-time routing problem of UAVs in a two-dimensional dynamic environment where the locations of the targets change in time. The UAV is located at a base point and is supposed to visit some targets in a continuous terrain where the UAV is allowed to move to any point. We aim to minimize two criteria: distance traveled and radar detection threat.

We develop a real-time algorithm to find the most preferred routes for a RP with underlying linear and quadratic preference functions. The algorithm updates the route of the UAV when the UAV arrives at a target. For the linear preference function case, the best trajectory for each target pair is calculated and the best routes are structured using these trajectories. To reduce the computational burden, we develop k-closest heuristic. We choose k closest targets and structure only these k pairs' nondominated frontiers for each target. We also develop an adaptive algorithm that determines the value of k each time the UAV visits a new target. For quadratic preference functions, we use the same algorithm, but we select the efficient trajectories to be used in the models using different approaches. In general, we select n trajectories for each target pair to find a route for the UAV. We consider the cases $n = 1$ and $n > 1$, seperately. To observe the changes in the computation times and the quality of the solutions, we also include the best trajectory in n solutionsfor each case.

We apply the algorithms on different examples in Chapter 4. We develop 5, 9, and 15 target problems with 4 radars distributed in 400 km^2 terrains. We compare the

solutions proposed by our algorithms and the initial tours that are found before the flight of the UAV. For these instances, the routes found by our algorithms (RT-L and RT-Q) generally dominate the initial routes. The results show that we reduce the computation times considerably using the heuristics without sacrificing from the quality of solutions.

For future works, this study can be extended to the three-dimensional dynamic environment. In this case, the altitude of the UAVs and the ground structure are considered. Also, our calculation of total distance and total radar detection threat measures must be revised according to the three-dimensional environment. More dynamic components can be added to the problem structure, like changing radar locations, having pop-up targets or radars, etc. Also, the problem of routing multiple UAVs to multiple targets can be considered. This problem is a vehicle routing problem and both static and dynamic environments can be considered for this problem. We considered this problem as a node routing problem, where targets need to be visited. If instead some edges need to be visited, the problem turns to be an arc routing problem.

REFERENCES

Allaire, F., Tarbouchi, M., Labonté, G., & Fusina, G. (2009). FPGA Implementation of Genetic Algorithm for UAV Real-Time Path Planning. *Journal of Intelligent and Robotic Systems*, 495-510.

Bortoff. (2000). Path planning for UAVs. *American Control Conference* (pp. 364-368). Chicago: IEEE.

Chen, X., & Chen, X. (2014). The UAV dynamic path planning algorithm research based on Voronoi diagram. *Chinese Control and Decision Conference* (pp. 1069-1071). Changsha: IEEE.

Cruz, J., Eva, B.-P., Torre-Cubillo, L., Andres-Toro, B., & Lopez-Orozco, J. A. (2008). Evolutionary path planner for UAVs in realistic environments. *10th annual conference on Genetic and evolutionary computation* (pp. 1477-1484). Atlanta: ACM.

Desrochers, M., & Laporte, G. (1991). Improvements and Extensions to the Miller-Tucker-Zemlin Subtour Elimination Constraints. *Operations Research Letters 10*, 27-36.

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

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

Karger, D., Arora, S., and Karpinski, M. (1995). Polynomial Time Approximation Schemes for Dense Instances of NP-Hard Problems. Proc. *27th ACM STOC.* (pp. 284-293).; the full version appeared in *J. Comput. System Sciences 58* (1999),193-210.

Kim, Y., Gu, D., & Postlethwaite, I. (2007). Real-Time Optimal Mission Scheduling and Flight Path Selection. *IEEE Transactıons On Automatıc Control*, 1119-1123.

Kinney, G. (2000). *A Hybrıd Jump Search And Tabu Search Metaheurıstıc For The Unmanned Aerıal Vehıcle (Uav) Routıng Problem.* Ohio: Aır Force Instıtute Of Technology.

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

Köksalan, M., & Lokman, B. (2009). Approximating the nondominated frontiers of multi-objective combinatorial optimization problems. *Naval Research Logistics (NRL)* , 191-198.

Kuwata, Y., & How, J. (2004). Three Dimensional Receding Horizon Control for UAVs. *Navigation, and Control Conference and Exhibit Providence* (p. 24). Rhode Island: AIAA.

Miller, C.E.,Tucker, A.W., and Zemlin, R.A. (1960). Integer Programming Formulation of Traveling Salesman Problems. *Journal of the Association for Computing Machinery 7 (4)*, 326-329.

O'Rourke, K. (1999). *Dynamıc Unmanned Aerıal Vehıcle (Uav) Routıng Wıth A Java-Encoded Reactıve Tabu Search Metaheurıstıc*. Ohio: Air Force Institute of Technology.

Öztürk, D. T. (2013). *Heurıstıc And Exact Approaches For Multı-Objectıve Routıng*. Ph.D. thesis. Middle East Technical University.

Peng, Z., Li, B., Chen, X., & Wu, J. (2012). Online Route Planning for UAV Based on Model Predictive Control and Particle Swarm Optimization Algorithm. *World COngress on Intelligent Control and Auomation*, (pp. 397-401). Beijing.

Rudnick-Cohen, E., W. Herrmann, J., & Azarm, S. (2016). Risk-Based Path Planning Optimization Methods for Unmanned Aerial Vehicles Over Inhabited Areas. *Journal of Computing and Information Science in Engineering*, 7.

Ruz, J., Arevalo, O., Pajares, G., & Cruz, J. (2007). Decision making among alternative routes for UAVs in dynamic environments. *Emerging Technologies and Factory Automation* (pp. 997-1004). Patras: IEEE.

Smith, D. K. (2003). *Networks and graphs : techniques and computational methods.* Chichester: Horwood Pub.

Swartzentruber, L., Foo, J. L., & Winer, E. H. (2009). Three-Dımensıonal Multı-Objectıve Uav Path Planner Usıng Terraın Informatıon. *Structures, Structural Dynamics, and Materials Conference* (p. 19). Palm Springs: AIAA.

Tezcaner D., Köksalan M. (2011). An Interactive Algorithm for Multi-objective Route. *Journal of Optimization Theory and Applications (150)*, 379–394.

Tezcaner Öztürk D., Köksalan M. (2016). An interactive approach for biobjective integer programs under quasiconvex preference functions. *Ann Oper Res (2016) 244*, 677–696.

Tezcaner Öztürk, D., Köksalan M. (2017). An Evolutionary Approach to Generalized Biobjective Traveling Salesperson Problem. *Computers and Operations Research 79*, 304-313.

Türeci, H. (2017). *Interactıve Approaches For Bı-Objectıve Uav Route Plannıng In Contınuous Space.* M.S. thesis. Middle East Technical University.

Xu, Y., Xiuxia, S., & Li, S. (2006). Particle Swarm Optimization for Route Planning of Unmanned Aerial Vehicles. *International Conference on Information Acquisition* (p. 6). Shandong: IEEE.

Zheng, C., Li, L., Xu, F., Sun, F., & Ding, M. (2005). Evolutionary Route Planner for Unmanned Air Vehicles. *IEEE Transactıons On Robotıcs*, 12.

Zhenhua, W., Weiguo, Z., Jingping, S., & Ying, H. (2008). UAV Route Planning using Multiobjective Ant Colony System. *IEEE Conference on Cybernetics and Intelligent Systems* (pp. 797-800). Chengdu: IEEE.