UNMANNED AERIAL VEHICLE ROUTING
IN THE PRESENCE OF
THREATS

by

KAMIL A. ALOTAIBI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2014

Acknowledgements

First of all, I am truly thankful and grateful to my supervising professor, Dr. Jay M. Rosenberger. I extremely admire and appreciate all of his consistent help, support, guidance, understanding, patience, encouragement, and insightful advice during my research, course work, and the entire time in my doctoral journey. It is my honor to be one of his PhD students. I would like to extend my sincere appreciation to Dr. Stephen P. Mattingly for his help, his invaluable remarks during our meetings, and being one of my dissertation committee members. I would also like to express my deep gratitude to Dr. Bill W. Corley and Dr. Brian L. Huff for their interest in my research, being members of my dissertation committee, and teaching me during my course work in my doctoral studies. I am also highly grateful to Dr. Victoria Chen for teaching me during my course work in my doctoral studies. I am especially thankful to Dr. Olena Shevchenko for her help and encouragement early in my research.

I highly appreciate Dr. Sheik Imrhan, Julie Estill, and Richard Zercher for their time and cooperation. I would like to especially thank Dr. Nadia Martinez Cepeda and John Dickson for their valuable help and support. I want to thank Karthik S. Easwar for his help and friendship. I would like to thank Antonio Alanis Pena and Na Wang for their support and kindness.

My deep gratitude goes to Taibah University and to the government of Saudi Arabia for funding me during the entire course of my doctoral studies. Finally, I am extremely thankful and grateful to my parents, wife, children, siblings, especially my two older brothers, Dr. Faisal A. Alotaibi and Dr. Mamdouh A. Alotaibi, and my all friends for their consistent help, moral support, especially during my hard times, encouragement, and love.

November 17, 2014

Abstract

UNMANNED AERIAL VEHICLE ROUTING

IN THE PRESENCE OF

THREATS

KAMIL A. ALOTAIBI, PhD

The University of Texas at Arlington, 2014

Supervising Professor: Jay M. Rosenberger

The use of Unmanned Aerial Vehicles (UAVs) and the importance of its role have evolved and increased recently in both civilian and military operations. In this research, we study the routing of Unmanned Aerial Vehicles (UAVs) in the presence of the risk of enemy threats. The main goal for this research is to find optimal routes that consider the targets visited, the expected fuel burn, the threat exposure, and the travel time. We formulate two mixed integer linear programs. In the first formulation, we minimize the total expected fuel burn for multiple UAVs in order to visit multiple targets while maintaining the total threat exposure level for all UAVs to a predetermined constant parameter. In the second formulation, we maximize the total number of visited targets for multiple UAVs while maintaining both the route travel time for a UAV and the total threat exposure level for all UAVs to predetermined constant parameters. Both formulations consider a set covering Vehicle Routing Problem (VRP), and some assumptions are made. The expected fuel burn, the risk of threat exposure, and the travel time are modeled and calculated for each edge and for each route. Several waypoint generation methods are proposed. In this research, waypoints are considered targets that do not need to be visited. However, a UAV

may or may not visit a waypoint while traveling from a target to another in order to reduce the threat level.

The Branch and Cut and Price (BCP) methodology is used to solve the problem. In the BCP, first, the linear programming relaxation of the problem is solved at each node of the branch-and-bound tree. A cut generation step is called to try to find Minimum Dependent Set (MDS) cuts and add them to the Restricted Master Problem (RMP) in order to cut some fractional solutions and encourage integrality. If the MDS cuts do not exist, the pricing step is called. In the pricing step, routes, variables, with negative reduced costs are generated using a Delayed Column Generation (DCG) algorithm and added to the RMP. In our sub problem, the Integer Programming Shortest Path (IPSP) algorithm is used as an engine for the DCG algorithm. A simple path heuristic (HEU) is used with the DCG algorithm in order to generate simple paths from negative cost cycles. Finally, bounds are updated and branching is carried out using a variant of Ryan and Foster branching logic.

A computational study for both formulations is done and results for different scenarios are presented. The results for the first formulation show that for the 10-target case, as the total threat level decreases, both the total expected fuel burn and the number of waypoints visited increase for both algorithms, the DCG-HEU and the DCG-HEU-MDS. In addition, only one UAV is used. Therefore, the problem is considered a Traveling Salesman Problem with waypoint generation. The results for the second formulation show that both algorithms, the DCG-HEU and the DCG-HEU-MDS, perform better when using fewer waypoints based on the 4-hour run time limit. For the small-sized problem, the DCG-HEU performs better than the DCG-HEU-MDS when using the same number of waypoints. For the large-sized problem, the DCG-HEU-MDS performs better than the DCG-HEU when using same number of waypoints.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

INTRODUCTION

The importance of Unmanned Aerial Vehicles (UAVs) has increased recently in both military and civilian operations. An important advantage of using UAVs, especially in dangerous activities, is not jeopardizing the lives of humans, since there is no pilot or crew on board. A UAV can reach places and areas where human survival is risky or impossible. UAVs can be used for various purposes. For example, in military applications, UAVs can be used in reconnaissance missions, spying, attacking targets by delivering ordinance and dropping bombs on them, or even search and rescue missions on the battlefield. While in civilian applications, UAVs can be used to monitor the environment (e.g. gathering information in inclement weather), monitor traffic congestion, monitor oil pipelines, and in disaster relief operations.

Early types of UAVs were not autonomous and typically required a group of operators to coordinate them [1]. The development of UAVs is evolving, and there has been an increased interest to make UAVs operate more and more autonomously. When UAVs operate fully autonomously, they can plan their own paths to move from a location to another and avoid any obstacles or threats encountered on their routes. This limits the need to only one operator to control a group of UAVs [2]. However, if the UAV is not operating autonomously, then pre-mission path planning is considered a very crucial step for routing the UAV. This is because usually UAV routing decisions are made before the mission begins. However, once the operation begins, changes to these pre-mission plans are hard to make.

A good mission path plan should solve an optimization problem that satisfies the mission objectives and restrictions. A UAV should complete its mission by traveling from where it is initially located and proceed as planned until reaching its desired final

destination. Mission objectives can include, for example, planning a mission with least distance traveled, which eventually leads to less fuel consumed, and ultimately making sure the mission can be completed. Similarly, minimizing time to complete the mission is very important if the mission is time sensitive. In military operations, minimizing the level of exposure to enemy threats or avoiding them entirely is of utmost importance for the planner and crucial for the UAV to complete its mission safely and successfully. Depending upon the type of the mission, a single or multiple UAVs may be required to visit a single or multiple targets. This needs to be taken into account when designing a plan for a UAV mission. The aerodynamic constraints of the UAV are of importance as well and should be taken into consideration.

In a military environment, a UAV could be detected by an enemy and become vulnerable to being shot down when traveling in threatened areas. Thus, the entire mission can be disrupted or lost. In this research, the UAV exposure to enemy threats is considered a major risk factor, which requires a crucial pre-mission plan. Therefore, routes must be designed efficiently and effectively, that not only optimize the expected fuel burn or the number of visited targets but also limit the threat level of exposure and the travel time. Since a UAV must maintain its preplanned schedule, an acceptable level of threat is dependent on the planner's preference. As the accepted level of exposure to threats decreases, it is expected that the UAV will try to avoid flying directly over these threats and be more likely to travel further away from them in order to reduce the threat exposure level. Thus, the UAV will travel a longer route, which results in more fuel burn. However, as the accepted level of exposure to threats increases, it is expected that the UAV will travel a shorter route, which results in less fuel burn.

In the following sections, we present our research objective and our contributions.

1.1 Research Objective

The main goal for this research is to find optimal routes that consider the targets visited, expected fuel burn, threat exposure, and travel time. We formulate two mixed integer linear programs. In first formulation, we minimize the total expected fuel burn for a group of UAVs to visit a given set of targets. In the second formulation, we maximize the total number of visited targets for multiple UAVs. In addition, we maintain the route travel time for a UAV to a predetermined constant parameter. In both formulations, we maintain the total threat level to a predetermined constant parameter. Furthermore, we generate waypoints that a UAV can visit while traveling from a target to another in order to reduce the threat level on high threat level links. When the maximum level of threat decreases, the UAVs visit more waypoints, and conversely as the maximum level of threat increases, the UAV is more likely to visit fewer waypoints.

Consider the following process. The pre-mission path planner receives information about the area of operation that includes the locations of a set of available UAVs, the set of targets that need to be visited, and the set of threat points, as shown in Figure 1-1. Then, the pre-mission path planner generates a set of waypoints, as shown in Figure 1-2, that a UAV can visit while traveling from target to target in order to reduce the total threat level. Finally, the pre-mission path planner needs to accept a certain level of threat. Depending on the accepted level of threat, the pre-mission path planner obtains one or several optimized routes. Figure 1-3 shows an optimized route in which a single UAV originates and terminates at a depot location, represented by a black circle around the center of the area of operation.

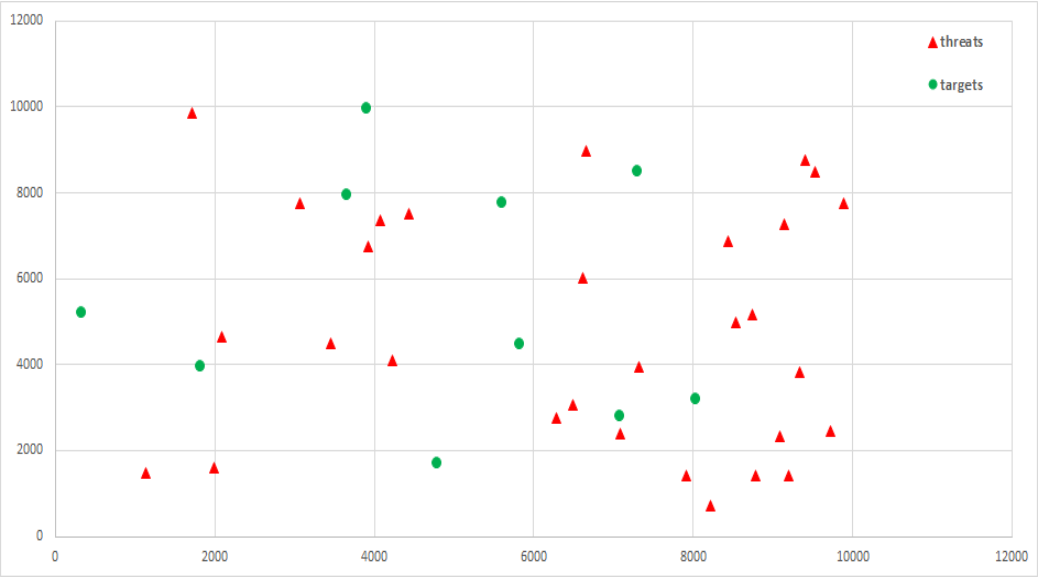


Figure 1-1 An Area of Operation Containing Targets and Threat Points

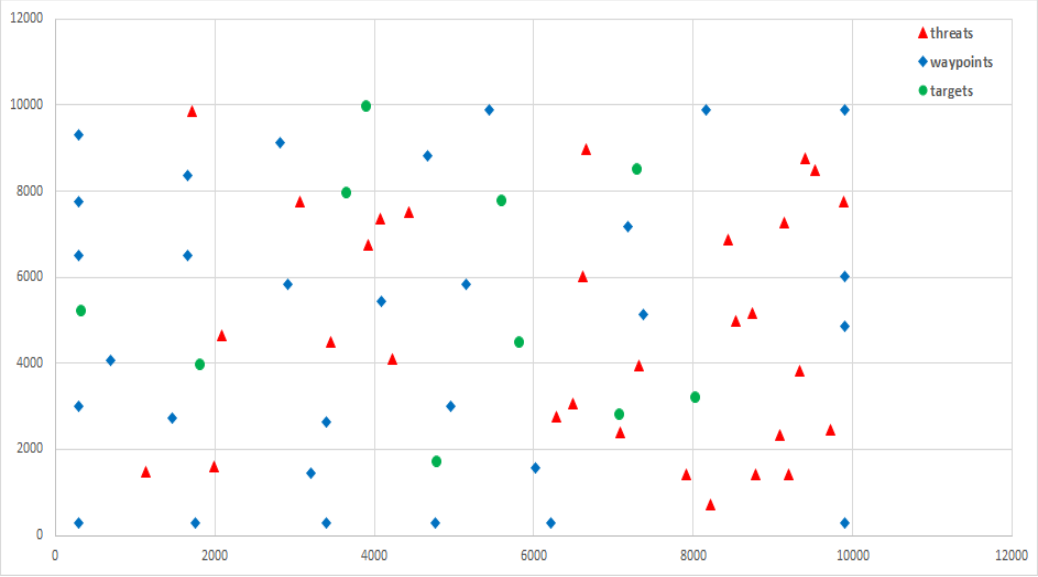


Figure 1-2 An Area of Operation Containing Targets, Threat Points, and Waypoints

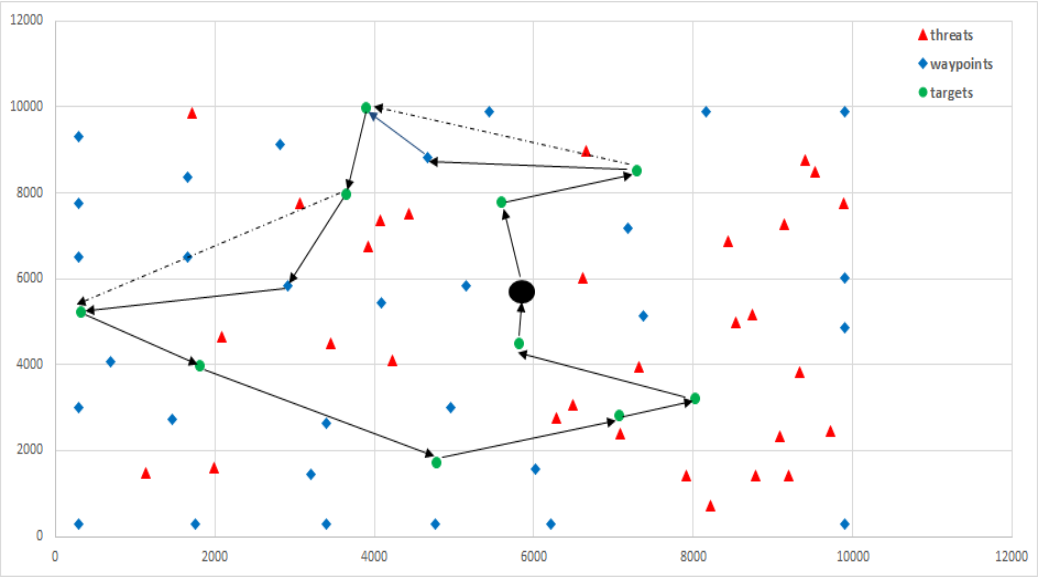


Figure 1-3 An Optimized Route

In the following chapter, we present the literature review in Vehicle Routing Problem (VRP) and in routing the Unmanned Aerial Vehicle (UAV) in the presence of threats, as well as contributions of this dissertation.

Chapter 2

LITERATURE REVIEW AND CONTRIBUTIONS

2.1 Vehicle Routing Problem (VRP)

UAV routing is considered a vehicle routing problem (VRP), which is a popular combinatorial optimization problem that is extensively used in the fields of distribution and transportation. The VRP is considered a generalization of the traveling salesman problem (TSP). The VRP's main objective is to determine a set of optimized cost routes for a group of homogenous vehicles, which originate and terminate at a single or multiple depots, to deliver service or goods to a given number of customers. Each customer must be visited exactly once by exactly one vehicle. In addition, vehicle capacity must not be exceeded by the total demand of customers in the path.

In general, the vehicle routing problem is usually defined on a network graph $G = (V, E)$, where $V = \{v_0, v_1, \ldots, v_n\}$ is a set of all vertices, nodes. Vertex v_0 is a common depot, source. The rest of the vertices are customers or targets. $E = \{e_{ij} = (v_i, v_j) : i \neq j \; and \; v_i, v_j \in V\}$ is a set of all links, edges. There is an associated nonnegative cost, distance, c_{ij}, for each edge e_{ij} when travelling from target v_i to target v_j. The VRP can be formulated as an integer set-covering problem as follows

$$\min \; c^T x \tag{2.1}$$

$$s.t. \;\; Ax \geq 1, \tag{2.2}$$

$$x \in \{0, 1\}^n, \tag{2.3}$$

The cost c represents a constant $n \times 1$ vector. The decision variable x represents a binary $n \times 1$ vector that indicates paths that are selected in the solution. The constant 1 represents an $m \times 1$ vector of ones. In the constant 0-1 $m \times n$ matrix A, each row represents a customer or target, while each column represents a path, and an entry a_{ij} equals to 1 if customer i is serviced by route j, while it is 0 otherwise.

The VRP is considered a NP-hard problem in combinatorial optimization [7]. This means that as the size of the problem increases, its computational complexity exponentially increases. Even with medium-sized VRP problem data, a large number of variables can be used. Therefore, a column generation algorithm is traditionally used to solve these type of problems.

There are many VRP survey papers in the literature. A classification scheme that was proposed and applied to fourteen problems from the VRP's literature can be found in Desrochers et al. [3]. A survey involving only exact algorithms such as branch-and-bound, dynamic programming, integer and linear programming for solving VRP can be found in Laporte and Nobert [4]. The book by Toth and Vigo [5] surveyed many heuristic and exact algorithms that were used to solve the VRP and its variation problems such as Capacitated Vehicle Routing Problem (CVRP), Vehicle Routing Problem with Time Windows (VRPTW), Vehicle Routing Problem with Pickup and Delivery (VRPPD), and Vehicle Routing Problem with Backhauls (VRPB). A survey involving only VRPTW can found in Desrochers et al. [6]. Exact and heuristic algorithms in the fields of Inventory Routing Problems (IRP) and Stochastic Vehicle Routing (SVR) problems are among other VRP problems surveyed by Cordeau et al. [7].

The VRP is widely believed to be first introduced in Dantzig and Ramser [8] in 1959, which involves trucks delivering gasoline to a number of service stations. Clarke and Wright [9] used the same formulation as Dantzig and Ramser [8] and developed a fast heuristic approach that assesses the savings on the distance that can result when a truck services two pairs of customers on the route instead of servicing them individually from the depot. Their problem considered finding minimum distance routes for trucks with different capacities that start and end at a depot and scheduled to carry loads to delivery points.

Many papers in the literature have proposed solution methods for solving the VRPTW. Solomon [10] extended several heuristics to solve the VRPTW, such as the saving heuristic proposed in Clarke and Wright [9]. He examined their performance with test sets and concluded that heuristics with insertion gave better results for problems involving VRPTW. An algorithm for solving the VRPTW, in which each customer is required to be visited within a time window, using dynamic programing and branch-and-bound was first proposed by Kolen et al. [11]. Desrochers et al. [12] formulated VRPTW as an integer set partitioning problem, and the linear programing relaxation was solved using column generation. Dynamic programing was used to solve the shortest path sub-problem, and the solution found was used as a lower bound for the branch-and-bound algorithm in order to solve the integer formulation of the master problem. The algorithm was able to solve problems with 100 customers. In a paper by Gambardella et al. [13], a VRPWT with multiple objective functions including the total travel time and the number of vehicles was optimized using a multiple ant colony optimization algorithm. The idea was to coordinate the activities of different ant colonies that minimize different objectives and use independent pheromones but cooperatively share information among them.

Many papers in the literature have proposed solution methods for solving CVRP. Laporte et al. [14] introduced a formulation for the CVRP. Exact algorithms using integer linear programming and subtour elimination constraints were proposed. Solutions were obtained for problems with up to sixty cities. Balinski and Quandt [15] were the first to introduce an integer programming formulation of the CVRP based on set partitioning and covering models. A tabu search heuristic for CVRP that allows infeasible solutions in order to reduce the possibility of local minima and uses a generalization insertion procedure by removing a node from its current path and inserting it into another path was proposed by Gendreau et al. [16]. Osman [17] proposed approximated methods based on simulated

annealing, tabu search, and descent algorithms for solving CVRP. The paper tested the methods with an existing test set from the literature and showed that the computational time was reduced by more than 50% for the proposed tabu search algorithm.

2.2 The Unmanned Aerial Vehicle (UAV) Routing in The Presence of Threats

2.2.1 Overview

Many papers in the literature have studied UAV mission path planning and tried to assess the risk associated with the routes that the UAV can take in a hostile area of operation. The risk can be caused by stationary or moving sources of threats or obstacles such as the enemy's surface-to-air missiles, radar detection zones, terrain, or other vehicles that the UAV could collide with on its route. Different numbers of UAVs, targets, threats or obstacles were used. Threats were modeled in two-dimensional or three-dimensional areas of operation. Different objectives were proposed and many approaches and methods were used to find optimized routes. The methods can be classified as graph-based approaches, probabilistic approaches, and deterministic approaches. Some proposed algorithms that are suitable for real time applications. Most of the optimization problems were restricted to UAV dynamics. Threat avoidance constraints and methods were proposed in many papers.

Many papers used a single UAV and a single target, such as in Rathbun et al. [2], Foo et al. [18], Wang et al. [19], Schouwenaars et al. [20], Zhenhua et al. [23], Ruiz et al. [24], Xinzeng et al. [25], Bortoff [26], Dogan [27], Blackmore et al. [29], Zengin and Dogan [32], Jun and D'Andrea [34], Pelosi et al. [35], Helgason et al. [36], Zabarankin et al. [37], McManus et al. [38], and Pfeiffer et al. [40]. A single UAV and multiple targets were used in a paper by Dogan [28]. Papers that used multiple UAVs and a single target include Schouwenaars et al. [20], Jun and D'Andrea [34], and Carlyle et al. [39]. Few papers in the

literature studied multiple UAVs and multiple targets. In Richards et al. [1], multiple targets were ordered and assigned to a group of UAVs with time dependency and logical assignments constraints. In Krishna et al. [22], ten UAVs located at the same location in the middle point of one side of the area of the operation were required to arrive to ten targets located on the same vertical line on the other side. Beard et al. [30] used five UAVs and six targets. A single target was assigned to a team of UAVs, where some targets can be unassigned, and the time over each target that is assigned to a team of UAVs was estimated in order to coordinate simultaneous arrivals for attacking. Maddula et al. [31] used eight UAVs and different numbers of targets for each scenario ranging from eight to forty-eight targets. Targets were assigned to UAVs and divided equally among them.

Some papers used stationary or moving threats and obstacles or a combination of both. A similar approach was applied to targets. Stationary threats were used in many papers including Richards et al. [1], Foo et al. [18], Wang et al. [19], Zhenhua et al. [23], Ruiz et al. [24], Xinzeng et al. [25], Dogan [27] and [28], Blackmore et al. [29], Beard et al. [30], Maddula et al. [31], Zengin and Dogan [32], Pelosi et al. [35], Helgason et al. [36], Zabarankin et al. [37], McManus et al. [38], and Pfeiffer et al. [40]. Rathbun et al. [2] considered moving obstacles representing other vehicles. Bortoff [26] considered fixed threat zones representing radar sites. De Filippis et al. [33] used stationary obstacles representing mountains. Carlyle et al. [39] used stationary threat zones representing surface-to-air missiles guided by radar. A few papers used only moving threats or obstacles such as Jun and D'Andrea [21] and [34]. Both moving and fixed threats were used in Schouwenaars et al. [20]. Similarly, almost all papers used stationary targets. Moving targets were used by Zengin and Dogan [32], but we could not find a paper that used both stationary and moving targets.

Different numbers of threats, obstacles, and no fly zones were used in the literature. Rathbun et al. [2] used two obstacles. Wang et al. [19] used a single threat and three obstacles, mountains. Xinzeng et al. [25] used five threat zones. Bortoff [26] designated ten radar sites as threat zones. Dogan [27] and [28] used seventeen threat zones. Zengin and Dogan [32] used seventeen threat areas and three no fly zones. Blackmore et al. [29] used three obstacles. Beard et al. [30] used thirty-six threat points that represent radar locations. Maddula et al. [31] used forty threat points that represent radars sites. Pelosi et al. [35] used two radar areas. Helgason et al. [36] used five obstacles. Zabarankin et al. [37] presented algorithms for a single radar and for two radars and tested them with different examples. Carlyle et al. [39] presented two different case studies to test the proposed algorithms. One used fifteen surface-to-air missiles to test the problem without the turn radius constraint, and the other used four of them with the turn radius constraint. In one example, Pfeiffer et al. [40] used three threat zones of the same shape while in another example, two different shapes that represent the threat zones were used.

Many papers modeled threat zones, obstacles, or no fly areas in two- or three-dimensional areas of operation and formulated their problems accordingly. Richards et al. [1], Schouwenaars et al. [20], and Ruiz et al. [24] defined obstacles as rectangles. Xinzeng et al. [25] modeled each threat zone as two-nested circles. The inner circle represents the no-fly zone, and the outer one represents the can-fly zone. Dogan [27] and [28] characterized each source of threat position and area by two-dimensional Gaussian probability density function. In addition to threat zones, Zengin and Dogan in [32] added no-fly areas with two-dimensional uniform distributions. Beard et al. [30] and Maddula et al. [31] constructed polygons around threat points, which represent radar sites. De Filippis et al. [33] divided an area of operation containing obstacles, mountains, into cells in a two-dimensional grid. Jun and D'Andrea [34] divided an area of operation, involving

moving threats, into hexagonal cells in a two-dimensional grid. Helgason et al. [36] modeled threat areas as circles and polygons. Carlyle et al. [39] modeled threat zones as circles, which represent surface-to-air missiles that are guided by radars. Pfeiffer et al. [40] modeled threat zones as nested polygons and nested circles. Some papers used three-dimensional models in order to incorporate low or high altitude UAV flight. In Foo et al. [18], a three-dimensional virtual path planner that can aid the human operator to choose alternative routes was introduced. The threat zone was defined as a sphere. Wang et al. [19] proposed a three-dimensional path planning model and modeled the threat area as a sphere as well. Pelosi et al. [35] presented a three-dimensional model that uses UAV capability to fly in different altitude ranges. McManus et al. [38] introduced a mission and pilot planner that used three-dimensional graphics to enable the UAV with onboard situational awareness.

2.2.2 Graph Based Approaches

Many papers in the literature used graph based approaches to obtain optimal paths for UAVs in the presence of threats. In one of the graph approaches, two major steps are considered. In the first step, based on the known location of the threat sources, a graph is constructed and weights such as threat, distance traveled, or flight time are assigned to the edges of the graph. Then, a search algorithm is used to find an initial suboptimal path for a UAV to travel from a starting position to a target location. In the second step, the suboptimal initial path obtained in the first step is made flyable by improving and refining it in order to meet the UAV dynamic constraints. A VORONOI diagram is a very popular graph approach that uses a Delaunay triangulation procedure in order to construct a set of edges, around known threat point locations, that can form potential paths from a starting point to a target location. Zhenhua et al. [23] constructed a VORONOI diagram and used a

multiobjective ant colony algorithm to find the best routes. Bortoff [26] constructed a VORONOI diagram based on known radar locations, but the graph was searched using a dynamic programing search to find an initial route. Beard et al. [30] constructed a VORONI diagram, and the K-best paths algorithm was used as a searching tool to find the initial best routes. Maddula et al. [31] constructed a VORONI diagram. The edges of the graph that are greater than a threat threshold were deleted. Then, the reduced graph was searched to find the K-best shortest paths from UAV locations to target locations.

Another graph approach was found in a paper by De Filippis et al. [33], where the area of operation was divided into cells, and a risk function for each cell was calculated. The risk map was transformed into a digraph and searched using an A* algorithm to obtain an initial path. Similarly, in a paper by Jun and D'Andrea [34], the area of operation was divided into hexagonal cells, and the Bellman-Ford algorithm was used to find an initial shortest path. McManus et al. [38] brought some techniques from the robotics field such as the cube space and octree algorithms, which divide a three-dimensional area of operation into spaces. Each space that contains an obstacle entity was marked as occupied, and free spaces were assigned a value equal to the distance to the final destination. The UAV starts from its current space, scans its adjacent spaces, and moves from a free space to another until reaching a destination.

Carlyle et al. [39] also introduced a method that reduces the two-dimensional network size by identifying edges and vertices that cannot be part of the solution and deleting them. Then, the airspace was discretized into a grid of potential waypoints. Finally, these waypoints were connected with line segments to obtain possible routes.

Helgasonet al. [36] drew triangles around the threat areas. Then, a heuristic branch-and-bound algorithm was used to find the shortest route that composed of line segments form the starting position to final destination that coincided with the sides of the

triangles that are tangent to the threats and did not intersect with the any interior points of any threat.

2.2.3 Probabilistic Approaches

Many papers in the literature used probabilistic approaches to obtain optimal paths for UAVs in the presence of threats. The most common probabilistic method used was building a probability map of the area of operation. Dogan [27] presented an online probabilistic approach for UAV mission planning in an area of known threats. Each source of threat position and area was characterized by a two-dimensional Gaussian probability density function. Based on this probabilistic map, the conditional probability of a UAV being disabled by at least one source of threat if it followed a particular path was modeled to be the integration of the probability density function along the route. The paper further defined an upper limit on this conditional probability and used it as a cost function. In addition, based on local information of both target location and the probabilistic map, the paper introduced a strategy to generate routes with different threshold values on the probability of being disabled. A trade-off between the length of the path and the probability of a UAV being disabled by any source of threat on its route was proposed. UAV dynamics such as speed and turning angle were also taken into consideration. Dogan in [28] extended his probabilistic map approach in [27] from a single UAV and a single target to a single UAV and multiple targets in the presence of multiple threats. In addition to minimizing the probability of exposure to threat using the same probabilistic threat map used in [27][28], Zengin and Dogan in [32] added no fly areas with a uniform probability distribution function and developed a strategy that requires the UAVs to avoid them while chasing moving targets. If the target is within the UAV's sensor circle range, then the UAV is considered following the target, and the strategy tries to estimate the speed, heading, and future

location of the target. Otherwise, the strategy tries to estimate the target's current position parameters based on its recent locations.

Krishna et al. [22] presented an on-board cooperative approach controlling UAVs in an area of unknown threat locations. The area of operation was divided into cells. The number of "hostile agents" was assumed to be known, but their locations were unknown beforehand. As the UAVs move, information about a cell being occupied by a hostile agent are gathered using sensors and shared amongst them. The probability that a UAV being shot was calculated. By using an A* search algorithm, waypoints were generated for UAVs to reach to their final goal. The objective function was to minimize a combination of the probability of the UAV being attacked by a threat and the distance traveled. In addition, multiple UAVs were controlled to arrive at a target location with a restricted upper bound on time. De Filippis et al. [33] presented two approaches dealing with path planning in the presence of stationary obstacles, e.g. mountains. In the first approach, a probabilistic risk function was defined as a function of the UAV's altitude. The area of operation was divided into cells in a two-dimensional grid, and the probabilistic risk function for each cell was calculated. Then, the risk map was transformed into a digraph, by connecting the centers of any adjacent cells in the path, and an A* algorithm was used to obtain an initial path with minimum risk of collision with obstacles and length traveled. Finally, by using Dubins curves, the initial path was smoothed to obtain a flyable path that accounts for the UAV dynamics such as speed and turn radius. The second method used the same probabilistic risk map, but was based on a genetic algorithm. Jun and D'Andrea [34] presented an online algorithm for UAV path planning with uncertainty in the information of moving threats encountered. The paper built a probability map for the hostile area of operation. First, the area of operation was divided into hexagonal cells in a two-dimensional grid. Second, each cell occupancy was calculated based on data gathered by the UAV sensors and by

applying Bayes' rule. The map was then converted into a digraph by connecting the centers of any adjacent cells in the grid. Finally, the Bellman Ford algorithm was used to find an initial shortest path. The obtained initial path was smoothed to become flyable to meet UAV dynamics restrictions.

Rathbun et al. [2] introduced an online probabilistic evolutionary algorithm for path planning. Based on how accurate the prediction of the parameters such as location, velocity, acceleration, and direction that define the moving obstacles, the paper developed a model that can account for the uncertainty in the moving obstacles' locations that change over time. The UAV probability of collision with other moving obstacles, whose future motion is uncertain, was modeled. The generated paths were a chain of jointed spline curves. The multi-objective cost function balanced the probability of collision with obstacles, the path length, and fuel consumption. The model also considered the UAV speed and minimum turn radius restrictions. Jun and D'Andrea [21] built an online probabilistic map algorithm based on Bayesian estimation for an uncertain dynamic environment of multiple moving obstacles. The probability distribution function was updated based on the limited data measurements received from sensors as the vehicle moves. In case the vehicle loses track of a moving obstacle, the probabilistic map was updated based on *a priori* statistical information of its previous location and movements. Blackmore et al. [29] presented an online probabilistic approach to avoid obstacles with uncertainty about the future state of the UAV. The paper modeled the sources of uncertainty as additive Gaussian noise distributions. The method predicts the future distribution for the UAV's position under the restriction that the probability of collision with any obstacle at a given time step should be below a threshold. By using linear chance constraints, the paper showed that the problem could be formulated as a Disjunctive Linear Program. A trade-off between fuel burn and the probability of collision was presented.

Xinzeng et al. [25] used different types of threats such as weather, terrain, missile, and radar and modeled them with different probability functions. Each threat zone was modeled using two nested circles. The inner circle represents the no fly zone with threat probability of one, and the outer one represents the can-fly zone with probability between zero and one. The problem optimized the probability of threat with time constraints. Additional restrictions on the UAV capabilities were taken into account such as the UAV minimum turning radius, minimum and maximum altitude, flying direction, and maximum range. Pelosi et al. [35] modeled the radar detection probability as a function of the UAV radar cross section and the distance between the UAV and the radar location. The paper presented a three-dimensional model that uses terrain masking chances and UAV capability to fly on different altitude ranges in order to find optimal paths with minimal radar detection probability and minimal distance traveled. Carlyle et al. [39] introduced a probabilistic approach for UAV mission path planning that guided a group of UAVs to avoid risk of threats such as terrain and surface-to-air missiles, which are guided by radar. The probability that the group is being attacked by at least one surface-to-air missile was calculated for each edge on the route. The objective was to minimize the probability that a group of UAVs is being attacked by at least one surface-to-air missile while imposing constraints that limited the fuel consumption or flight time. The problem was formulated as a constrained shortest path problem, and UAV dynamics such as turn radius were considered directly in the formulation.

Pfeiffer et al. [40] introduced a probabilistic approach and investigated several scenarios for UAV path planning in the presence of threat zones. Using bicriteria optimization, a trade-off between the threat levels, which represent the probability of detection, and travel time was presented. The paper defined threat parameters for each threat zone and for the area outside the threat zones. Each potential route consisted of

disjunctive subpaths. The average number of detections along a route was calculated as the summation of each threat parameter multiplied by its corresponding subpath length. The optimization problem was reformulated as a weighted shortest path problem. In addition, the paper investigated cases where the risk level increases when the UAV is detected. The paper also calculated the expected time until the first detection by assuming that it is a random variable following an exponential distribution and used it to select a more preferable route among the ones with the same threat level. The paper assumed that the random variable that counts the number of detections follows a Poisson distribution, and the probability of K or fewer detections along the path was calculated. The paper also explored the time delay between detection and an attack.

2.2.4 Deterministic Approaches

Many papers in the literature used deterministic approaches to obtain optimal paths for UAVs in the presence of threats. In both Foo et al. [18] and Wang et al. [19], an initial straight path from the UAV location to the target location that can violate the threat zones is generated. Then, a particle swarm algorithm with digital pheromones was used to optimize. The resulting optimal paths were B-spline curves that try to avoid the threat zones. The threat cost function was assessed based on the distance from a point along the curve and the source of threat location. If the distance is less than the radius of the threat zone, then the UAV is considered traveling inside the threat zone, and the cost function returned a cost value; or zero otherwise. Wang et al. [19] analyzed and tested the convergence ratio of its algorithm based on a parameter selection approach and claimed that it performs better than the one in [18]. Both papers proposed objective functions that showed the trade-off between minimizing both the risk caused by the UAV exposure to enemy threats and the distance traveled. However, [19] added an extra term that

maximized the effect of observing targets. The resulting optimal paths for both papers satisfied the UAV's minimum turn radius and terrain constraints.

In Zhenhua et al. [23], threat intensity for each edge in the path was modeled deterministically as a function of the summation of the distances from predefined sampling points along the edge and the corresponding nearest threat points. The length of the route travelled and the UAV threat intensity level were minimized. A set of Pareto optimal solutions was obtained. Ruiz et al. [24] presented an online model to find optimal paths in the presence of radar detection regions with variable Radar Cross Section (RCS) of the UAV. The UAV was modeled as a moving object with time discrete dynamic states. The nonlinear radar detection risk function was approximated with a piecewise linearization function with hyper planes using 0-1 integer variables. The radar detection function was assumed to be proportional to the RCS of the UAV and reciprocal to the fourth power of the distance between the location of the UAV and the location of the radar. The objective function minimized the risk of radar detection and total flight time. The UAV maximum speed and minimum turning radius constraints were taken into account. The problem formulated by using a combination of mixed integer linear programing and avoided collision with obstacles by directly imposing constraints in the formulation. The CPLEX software package was used to solve the problem. Bortoff [26] proposed an online implementation consisting of two-step path planning methods for stationary and pop-up threats. The optimal path was described by a chain of masses being acted on by spring restoring forces and pushed away by a field of virtual forces from each radar site. By simulating the motions of these virtual masses, which were represented by a system of ordinary differential equations, a steady-state equilibrium solution for the masses' locations was found to represent an optimal stealthy path. The virtual forces from each radar acting on the chain of masses were modeled to be proportional to the RCS of the UAV and reciprocal to the

fourth power of the distance between the location of UAV and the location of the radar. A trade-off between a stealthy route and distance traveled was presented. The UAV maximum speed and minimum turning radius were taken into account.

Beard et al. [30] introduced a system architecture for multiple UAVs and multiple targets in the presence of known threat locations. The approach consisted of assigning a team of UAVs to a single target, where some targets can be unassigned, estimating time over each target that was assigned to a team of UAVs in order to coordinate simultaneous arrivals for attacking. The threat cost for traveling an edge was calculated deterministically by assuming that the UAV Radar Cross Sectional (RCS) was uniform and was inversely proportional to the distance from the UAV to the threat to the fourth power. Instead of calculating the integration along each edge, an approximation was used to calculate the threat cost at three points along the edge, at one-sixth, one half, and five-sixths of the edge length. The objective was to balance the risk from exposure to threats and length traveled. In addition, the paper introduced a trajectory generation algorithm that uses a nonlinear filter to model the kinematics of the vehicle, such as speed, acceleration, heading, and turning radius. Maddula et al. [31] introduced a problem that requires assigning targets to UAVs and balancing the number of targets that need to be visited equally among them in such a way that minimized the total distance traveled for all UAVs and limited the threat for each UAV under a threshold. The risk of threat was calculated similar to the threat risk calculation in [30]. Three algorithms to divide the targets among the UAVs equally were introduced and compared.

Richards et al. [1] presented two algorithms for an optimization problem that required both task assignments and path planning for a group of UAVs. The UAVs were modeled as moving objects with dynamic states in time steps. The goal was to minimize the total completion time of the mission that was restricted to avoid obstacles and no-fly

zones, satisfy UAV dynamics, and other constraints such as time dependency and logical assignments constraints. In the first method, the problem of task allocation and route planning was formulated entirely as a single mixed integer linear programing problem. The UAV dynamics such as maximum speed and maximum turning rate were satisfied by directly imposing mixed integer linear programing constraints in the problem formulation. CPLEX was used to solve the problem. In the second method, all assignments and ordering were enumerated and restricted to vehicle capabilities. Then, for each assignment and ordering, a straight-line approximation was used to estimate the finishing times of several paths that connected the UAV location, the corner of the obstacles zones, and targets. Then, these cost estimations were used to plan the path and the required task assignments for each UAV. Both algorithms were compared for different scenarios. The first method required intensive computations but gave optimal results, while the other was computationally efficient and suitable for real time applications but could give suboptimal solutions. In Schouwenaars et al. [20], the paper's main goal was to find optimal fuel paths, while directly avoiding collision with obstacles as well as other vehicles. In addition, the paper accounted for the UAV's dynamics by directly imposing a combination of mixed integer and linear programing constraints in the formulation. Helgason et al. [36] presented an algorithm for UAV path planning in order to avoid threats and obstacles. The problem was restricted to the UAV turn angle. The idea was to draw triangles around the obstacles that need to be avoided. Then, a heuristic algorithm was used to find the shortest route composed of few line segments from the start position to destination that coincided with the sides of triangles, which were tangent to the obstacles, and did not intersect with any interior points of any threat in order to avoid them. Zabarankin et al. [37] introduced two optimization approaches for path planning to obtain an optimal risk path that is restricted to the path length in an area of radar threats. The first approach was an analytical one that

involved using calculus of variation to reduce the optimization problem to solving a system of nonlinear differential equations with boundary conditions for a single radar. The second approach was a discrete optimization one that reformulated the problem to a weighted constrained shortest path problem on a grid and solved it using a modified labeling sitting algorithm with a preprocessing procedure. Both algorithms were tested for different examples with both one and two radars. The risk function was modeled as a risk index per unit length by assuming that the risk index is proportional to the risk factor, which depends on the radar characteristics and is inversely proportional to the distance from the UAV to the radar to the second power. McManus et al. [38] presented an online multidisciplinary intelligent mission and pilot planner that used three-dimensional graphics to support the UAV with its onboard situational awareness in order to enable a more autonomous UAV in civilian airspace. The paper brought some techniques from the robotics field such as potential field theory, which assigns a repulsive field of forces to obstacle entities that push the UAV away from them in order to avoid collision. The mission-planning problem optimized route length, flight time, and fuel burned.

2.3 Contributions

In this section, we describe our contributions based on our review of the existing literature with respect to routing a UAV in a hostile area of operation. Many papers in the literature used a single UAV and a single target in the presence of threats such as Rathbun et al. [2], Foo et al. [18], Wang et al. [19], Schouwenaars et al. [20], Zhenhua et al. [23], Ruiz et al. [24], Xinzeng et al. [25], Bortoff [26], Dogan [27], Blackmore et al. [29], Zengin and Dogan [32], Jun and D'Andrea [34], Pelosi et al. [35], Helgason et al. [36], Zabarankin et al. [37], McManus et al. [38], and Pfeiffer et al. [40]. In addition, a few papers used a single UAV and multiple targets in the presence of threats such as Dogan [28] or multiple

UAVs and a single target in the presence of threats such as Schouwenaars et al. [20], Jun and D'Andrea [34], and Carlyle et al. [39]. However, a very small number of papers used multiple UAVs and multiple targets in a one-to-one relation such as Krishna et al. [22], required an assignment and ordering such as time dependency and logical assignment Richards et al. [1], or divided targets equally among the UAVs such as Maddula et al. [31]. In our research, we consider the case with multiple UAVs that visit multiple targets with no restriction on how many targets a UAV can visit or in what order.

Many methods and approaches were used in the literature solve the UAV routing problem in the presence of threats. However, none of them considered solving the UAV routing problem in the presence of threats with branch-and-cut-and-price methodology.

Many papers in the literature introduced real-time applications such as Richards et al. [1], Rathbun et al. [2], Ruiz et al. [24], Bortoff [26], Dogan [27] and [28], Blackmore et al. [29], Jun and D'Andrea [21] and [34], and McManus et al. [38]. In this research, a pre-mission planner provides a schedule, and after the mission starts, the plan can be adjusted with the online/onboard application if it is not followed as scheduled.

Many papers in literature modeled threat zones with different shapes and considered using threat avoidance approaches to eliminate or reduce the risk of exposure to nearby threat sources on the route. If no detection is required, threat zones must not be entered. However, in our case, we modeled the threat zones as threat points, and the threat level we are trying to minimize is the threat from all threat points in the area of operation. This is because military intelligence might know the locations of the threats, but their types and shooting ranges may not be known. Therefore, the threat exposure level from all threat locations is considered. Very few papers considered using the threat level based on all threat locations in the area of operation such as Beard et al. [30] and Maddula et al. [31]. However, their approaches are different. Both constructed VORONOI diagram,

a set of edges around the threat points and based on their locations, and both used the K-best shortest paths algorithm as a searching tool. Our approach is different. We do not construct VORONOI diagram. We construct an instance graph on our sub problem but not based on the threat locations. In addition, Beard et al. [30] minimized threat cost and/or traveling distance and assigned a target to a team of UAVs in order to coordinate simultaneous arrivals for attacking, while Maddula et al. [31] minimized the total distance traveled for all UAVs, limited the threat for each UAV under a threshold, and divided targets equally among the UAVs. In our research, we consider two formulations. In the first formulation, we optimize routes that minimize the total expected fuel burn for multiple UAVs, while keeping the total threat level for all UAVs under a constant parameter. In the second formulation, we maximize the total number of visited targets, while keeping the route travel time for a UAV and the total threat level for all UAVs under constant parameters. Furthermore, we generate waypoints based on threat and target locations. Instead of traveling over high-risk edges, the UAV can visit these waypoints when traveling from a target to another in order to reduce the threat level.

Based on preliminary results we obtained after solving our first formulation problem with a small-sized problem, 10 targets, our problem is considered a Traveling Salesman Problem (TSP) with waypoint generation.

Finally, in our second formulation we find routes for multiple UAVs in the presence of threats that maximize the total number of visited targets.

In the following chapter, we show the models and approaches that we used in order to formulate and solve our problem in this research.

Chapter 3

MODELS AND APPROCHES

3.1 Model Assumptions

In this research, we make the following assumptions:

1. UAV exposure to enemy threats is the only major factor of risk.
2. Threats are modeled as stationary threat points in the area of operation.
3. The threat level is based on the UAV's exposure to a radar that is at the threat point location.
4. The positions of the threats are known, and the threat level can be estimated *a priori*.
5. The same expected wind speed and direction are used for the entire region of operation.
6. The same expected air speed is used for the entire operation.
7. UAV routes originate and terminate at the same depot.

3.2 Problem Formulation

We formulate two mixed integer linear programs to solve the Unmanned Aerial Vehicle Routing Problem (UAVRP). Throughout this dissertation, the first formulation is referred to as UAVRP1 and the second formulation is referred to as UAVRP2. We use similar notation to Visoldilokpun [41]. However, both problem formulations are based on the VRP with set covering in (2.1)-(2.3). In the first formulation (UAVRP1), we find optimal routes that minimize the total expected fuel burn for a group of UAVs in order to visit a given set of targets while keeping the total threat exposure level under a constant parameter, d. Furthermore, we generate waypoints that UAVs may visit when traveling from a target to another in order to reduce the threat level.

Let U represent a set of homogeneous UAV types located at the same depot and need to be scheduled. Let n_u be the total number of UAVs of type $u \in U$. Let F_u represents a set of all possible routes of a UAV of type u in the set U. Let the set F be the set of all possible routes for all UAVs in the set U. That is $F = \cup_{u \in U} F_u$. Let T be the set of all threats in the area of operation. Let K denotes the set of all targets that need to be visited. Let W be the set of all waypoints that may or may not be visited. Let $E[\tilde{e}_{uf}]$ denotes the expected fuel burn for a UAV of type $u \in U$ flying route $f \in F_u$. Let t_{uf} be the level of exposure to threats for a UAV $u \in U$ in a route $f \in F_u$. Let a_{kuf} be a binary constant that is equal to one when a target $k \in K$ is visited by a UAV of type $u \in U$ in a route $f \in F_u$. Similarly, the binary constant $a_{wuf} = 1$ indicates that a waypoint $w \in W$ is visited by a UAV of type $u \in U$ in a route $f \in F_u$. The binary decision variable $x_{uf} = 1$ when a UAV of type $u \in U$ services a route $f \in F_u$.

Then, the integer programming formulation of the UAVRP1 is as follows

$$min \sum_{u \in U} \sum_{f \in F_u} E[\tilde{e}_{uf}]\ x_{uf} \tag{3.1}$$

$$s.t. \sum_{u \in U} \sum_{f \in F_u} a_{kuf}\ x_{uf} \geq 1 \qquad \forall k \in K \tag{3.2}$$

$$\sum_{u \in U} \sum_{f \in F_u} a_{wuf}\ x_{uf} \geq 0 \qquad \forall w \in W \tag{3.3}$$

$$\sum_{f \in F_u} x_{uf} \leq n_u \qquad \forall u \in U \tag{3.4}$$

$$\sum_{u \in U} \sum_{f \in F_u} t_{uf}\ x_{uf} \leq d \tag{3.5}$$

$$x_{uf} \in \{0,1\} \qquad \forall\, u \in U, f \in F_u \tag{3.6}$$

The cost objective function (3.1) minimizes the total expected fuel burn for all UAVs. The set-covering constraints in (3.2) require that all targets be visited at least once,

while those in (3.3) indicate that a waypoint may be visited on the route. Observe that constraints in set (3.3) are redundant and can be eliminated from the formulation. The constraints in (3.4) imply that at most n_u routes can be assigned to each UAV of type $u \in U$. In constraint (3.5), the nonnegative constant parameter, d, limits the total threat level. Notice that it is possible to interchange the total expected fuel burn in (3.1) and the total threat level in (3.5), which can also be easily done in our implementation as well.

In our second formulation (UAVRP2), instead of minimizing the total expected fuel burn for a group of UAVs in order to visit a given set of targets, we find optimal routes that maximize the total number of visited targets. We maintain the route travel time for a UAV to a predetermined constant parameter. We still maintain the total threat level for all UAVs to a predetermined constant parameter and generate waypoints. For each target $k \in K$, let B_k be the benefit, or reward for visiting target k, and let binary variable x_k indicate whether target k is visited. Although in our computational experiments, we treat all targets as equally important, we can simply make targets have different priorities. This is easily done in our implementation as well. Notice that, for a waypoint, we can let $B_k = 0$, which indicates that a waypoint is a target but with no value, and it may or may not be visited on the route depending on the threat reduction needed. The second formulation (UAVRP2) is as follows

$$max \sum_{k \in K} B_k x_k \tag{3.7}$$

$$s.t. \sum_{u \in U} \sum_{f \in F_u} a_{kuf}\ x_{uf} \geq x_k \qquad \forall k \in K \tag{3.8}$$

$$\sum_{f \in F_u} x_{uf} \leq n_u \qquad \forall u \in U \tag{3.9}$$

$$\sum_{u \in U} \sum_{f \in F_u} t_{uf}\ x_{uf} \leq d \tag{3.10}$$

$$x_{uf} \in \{0,1\} \qquad \forall\, u \in U, f \in F_u \qquad (3.11)$$

$$x_k \in \{0,1\} \qquad \forall\, k \in K \qquad (3.12)$$

The new cost objective function in (3.7) maximizes the total number of visited targets for all UAVs. Constraints (3.9)-(3.11) are the same as constraints (3.4)-(3.6) in UAVRP1. The left-hand side of constraint set (3.8) is the same as that of constraint set (3.3) in UAVRP1. However, in UAVRP2, the right-hand side is only one when a target is visited. Constraint set (3.12) ensures that the target variables are binary. These variables can be relaxed to be continuous.

3.3 Expected Fuel Burn Modelling

In UAVRP1, the objective function is to minimize the total expected fuel burn for all UAVs. When the pre-mission planner chooses to accept a higher level of threat exposure, the UAV could travel shorter routes, which could result in less fuel burn. However, if the planner chooses to accept a lower threat exposure level, the UAV could travel a longer route, which could result in a more fuel burn.

We use the expected fuel burn model that was modeled by Visoldilokpun [41]. Let L represent the set of all links between all nodes, common base, targets, and waypoints. Then, the expected fuel burn by a UAV of type $u \in U$ for traveling a link $l \in L$ is formulated as

$$E[\,\tilde{e}_{ul}] = \frac{\tau_l \emptyset_u E[v^a]}{v_l^g} \qquad (3.13)$$

where, τ_l is the link distance between two nodes, $\emptyset_u$ denotes the fuel consumption rate for a UAV of type $u \in U$, $E[v^a]$ is the expected airspeed, and v_l^g is the ground speed and is calculated for traveling a link $l \in L$ as follows

$$v_l^g = E[v^a] + E[v^w]\cos(w_l) \qquad (3.14)$$

where $E[v^w]$ is the expected wind speed, w_l is the difference between the actual traveling angle of a UAV from node i to node j, θ_{ij}, and the wind direction, θ_w, that is $w_{ij} = |\theta_{ij} - \theta_w|$.

The expected fuel burn for a UAV of type $u \in U$ in a route $f \in F_u$ can be calculated as an additive sum of the expected fuel burn for the individual links in the route by

$$E[\tilde{e}_{uf}] = \sum_{l \in f} E[\tilde{e}_{ul}] \qquad \forall u \in U \tag{3.15}$$

3.4 Modeling the Risk of Threat

In this research, UAV exposure to enemy threats is considered a major factor of risk. In a military operation, a UAV could be exposed to enemy threat areas and be vulnerable to being shot down and disabled. This requires crucial pre-mission planning. Threat is modeled as a point threat. Each edge is assigned an associated cost that represents the risk of exposure to threat from all the threat points in the area of operation. The threat level is calculated deterministically using an approximation that was proposed by Beard et al. [30]. The threat cost for traveling an edge is calculated by assuming that the UAV Radar Cross Sectional (RCS) is uniform in all directions and is proportional to the edge length and inversely proportional to the distance from the UAV to the threat to the fourth power. Instead of calculating the integration along each edge, the approximation is used to calculate the threat cost at three points along each edge. Namely, at points along the edge that are 1/6, 1/2, and 5/6 of the edge length. Beard et al. [30] claimed that using this approximation yields errors that are less than two percent for threats that are located very close to the edge and in the order of a tenth of a percent for those that are far away. In our implementation, the level of the risk of exposure to threats is calculated from all N

sources of threats in the area of operation for each edge. The threat level, t_{ul}, for a UAV of type $u \in U$ for traveling a link $l \in L$ is calculated as follows

$$t_{ul} = \frac{\tau_l}{3} * \sum_{p=1}^{N} \left(\frac{1}{d_{1/6,p}{}^4} + \frac{1}{d_{1/2,p}{}^4} + \frac{1}{d_{5/6,p}{}^4} \right) \qquad \forall l \in L, u \in U \qquad (3.16)$$

where, τ_l is the edge length, $d_{1/6}, d_{1/2}, d_{5/6}$ are the distances from the selected three points on the edge to each threat point $p \in T$, respectively. The threat level for a UAV of type $u \in U$ for traveling in a route $f \in F_u$ can be calculated as an additive sum of the threat level for the individual links in the route as

$$t_{uf} = \sum_{l \in f} t_{ul} \qquad \forall f \in F_u, u \in U \qquad (3.17)$$

3.5 Waypoint Generation Methods

We generate waypoints that a UAV may utilize in order to reduce the level of threat exposure by avoiding traveling over high-risk links when traveling to a target. Consider a high threat level link from target v_i to target v_j. Instead of traveling the link, the UAV may travel from target v_i to a waypoint w and then proceed to target v_j in order to reduce the threat level. Since a UAV must maintain its preplanned schedule, it is up to the planner's preference to accept a certain threat exposure level. In the first formulation, when the accepted level of threat exposure decreases, the UAV will use more waypoints and travel longer routes, which increase the fuel burn and the time to complete the mission. However, as the accepted level of threat exposure increases, the UAV will use fewer waypoints and thus, travel a shorter route, which results in less fuel burn and a shorter mission time. In UAVRP2, the generated waypoints substantially affect the total number of visited targets. Therefore, several waypoint generation methods are investigated.

3.5.1 Maximin-Whole Area

In this method, we generate waypoints using the maximin distance design of experiments criterion in a two-dimensional grid representing the area of operation. The maximin distance design was first studied and introduced by Johnson et al. [46]. The experimental region is a two-dimensional grid representing the area of operation and contains the given set of targets and threat points. Based on the lowest and highest ranges of these target and threat points, and for each iteration of the maximin criterion, a huge set (typically ten-thousand) of evenly-spaced grid points is generated, representing candidate waypoints. The Euclidean distance matrix from each candidate waypoint to each target, threat point, and previously selected waypoint is calculated. The design selects the candidate waypoint that maximizes the smallest distance between the candidate waypoint and the targets, threat points, and previously selected waypoint in the two-dimensional grid. The process is repeated until the desired number of waypoints are selected. The resulting generated waypoints tend to be uniformly scattered in the experimental region based on this design. Waypoints were added to the area of operation *a priori* to optimizing the UAV routes. Figure 3-1 shows problem instances that include 5 UAVs, 10 targets, 30 threat points, and 55 waypoints generated using the Maximin-Whole Area method.

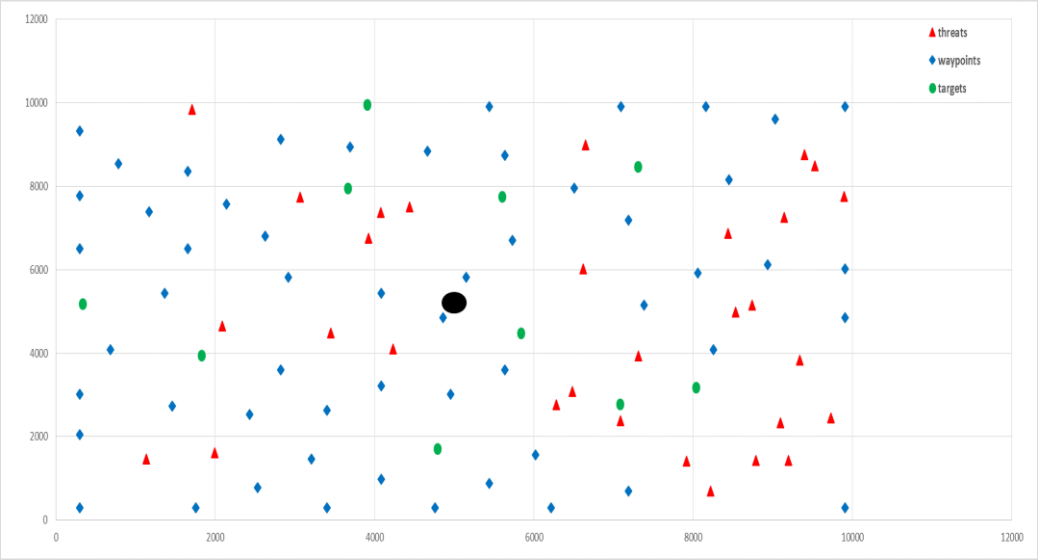


Figure 3-1 10 targets, 30 threat points, and 55 waypoints (Maximin-Whole Area)

3.5.2 Maximin-Rectangle

In this method, a rectangle is placed on any two targets in the area of operation including the source node. A set of 100 grid points is generated. Then, the distance matrix from each grid point in the set, in each rectangle, to each target and threat in the area of operation is calculated. The grid point that maximizes the smallest distance is only added to its corresponding rectangle in the area of operation. We generate one grid point per rectangle based on this method. Figure 3-2 shows problem instances that include 5 UAVs, 10 targets, 30 threat points, and 55 waypoints generated using Maximin-Rectangle method.

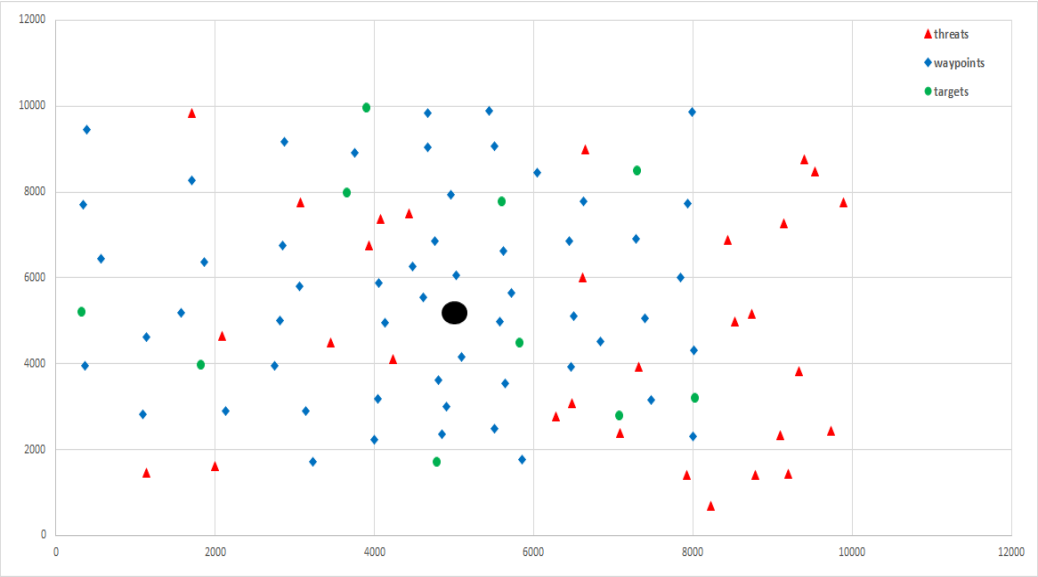


Figure 3-2 10 targets, 30 threat points, and 55 waypoints (Maximin-Rectangle)

3.5.3 Threat Reduction-Rectangle

In this method, a rectangle is placed on any two targets including the source node. A set of 100 grid points is generated in each rectangle c. Then, the threat reduction for each grid point g in a rectangle c is calculated. The threat levels, t_{ij}^{c}, t_{ig}^{c}, and t_{gj}^{c} for the links from target i to target j, from target i to grid point g, and from grid point g to target j, respectively, for each rectangle c in which target i, target j, and grid point g lie inside are calculated based on the threat level approximation described in section 3.4 (see Figure 3-3). Threat reduction for each grid point g in a rectangle c is calculated based on the following equation

$$Threat\ reduction\ (g,c) = t_{ij}^{c} - \left(t_{ig}^{c} + t_{gj}^{c}\right) \tag{3.18}$$

Only the grid point g that gives the highest threat reduction is added to its corresponding rectangle in the area of operation. We generate one grid point per rectangle based on this method. Figure 3-4 shows problem instances that include 5 UAVs, 10

targets, 30 threat points, and 55 waypoints generated using the Threat Reduction-Rectangle method.

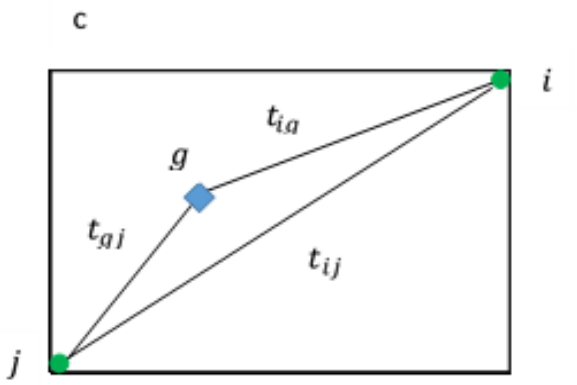


Figure 3-3 Threat Reduction for Grid Point g in a Rectangle c

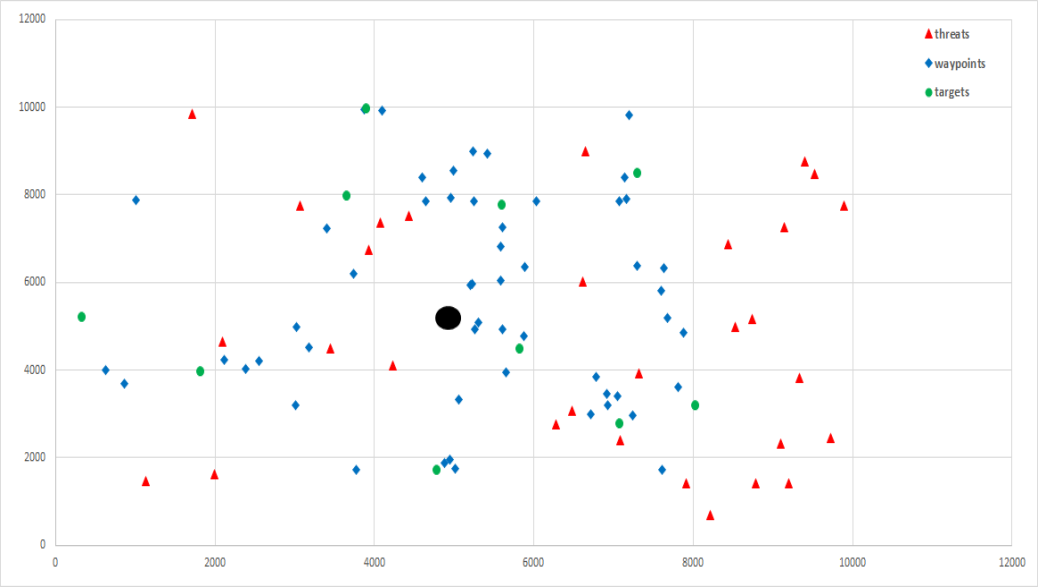


Figure 3-4 10 targets, 30 threats, and 55 waypoints (Threat Reduction-Rectangle)

3.5.4 Threat Reduction-Whole Area

In this method, based on the lowest and highest ranges of targets and threats, a set of ten-thousand grid points is randomly generated. For each grid point, the total threat reduction is calculate based on the number of rectangles that the grid point lies in. In each iteration, only the grid point with the largest total threat reduction is added to the area of operation. The process is repeated until the desired number of waypoints is generated. Figure 3-5 shows problem instances that include 5 UAVs, 10 targets, 30 threat points, and 29 waypoints generated using Threat Reduction-Whole Area.

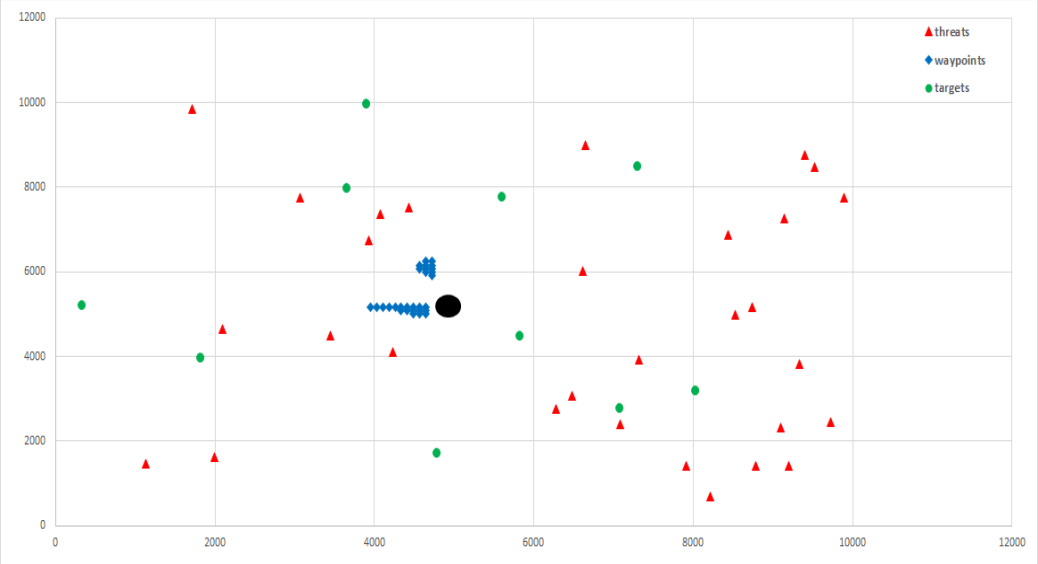


Figure 3-5 10 targets, 30 threats, and 29 waypoints (Threat Reduction-Whole Area)

3.5.5 Threat Reduction and Maximin-Rectangle

In this method, a rectangle is placed on any two targets including the source node. A set of 100 grid points is generated. For each grid point, we calculate the minimum distance to each threat and target and the threat reduction as well. Then, we find a Pareto frontier. From the Pareto frontier, first, we add the grid point that has largest threat

reduction. Then, we do maximin within the Pareto frontier and add the second grid point. We generate two grid points per rectangle based on this method. Figure 3-6 shows problem instances that include 5 UAVs, 10 targets, 30 threat points, and 110 waypoints generated using Threat Reduction and Maximin-Rectangle method.

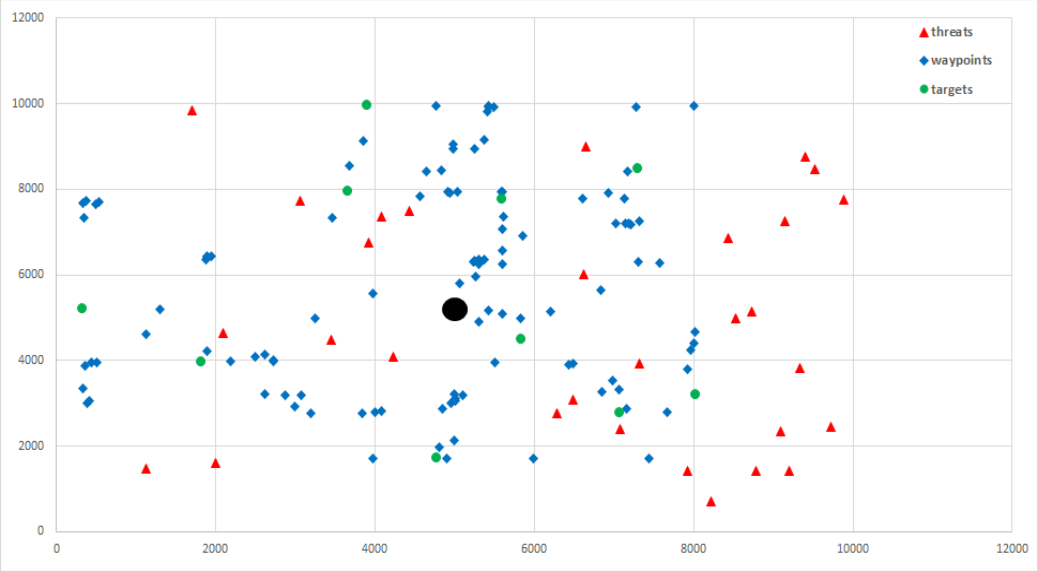


Figure 3-6 10 targets, 30 threat points, and 110 waypoints (Threat Reduction and Maximin-Rectangle)

3.5.6 Threat Reduction and Maximin-Whole Area

In this method, based on the lowest and highest ranges of targets and threats, a set of ten-thousand grid points is randomly generated. For each grid point, we calculate the minimum distance to each threat and target and the total threat reduction based on the set of rectangles in which the grid point lies. Then, we find a Pareto frontier. From the Pareto frontier, we add the grid point with the largest total threat reduction. Then, we do maximin within the Pareto frontier to add the second grid point. We generate two grid points at a time, and the process is repeated until the desired number of waypoints is generated.

3.5.7 Maximin-Whole Area-Removing

This method is based on the Maximin-Whole Area method, described in section 5.3.1. This method removes unimportant waypoints, which are unlikely to be visited by a UAV, from the area of operation. After the Maximin-Whole Area method is done, a rectangle is placed between any two targets including the source node. That total number of such rectangles is $\binom{\text{Number of Targets+Source Node}}{2}$. Then, any waypoint that does not lie in any one of the rectangles is removed. For example, recall that, Figure 3-1 shows problem instances that include 5 UAVs, 10 targets, 30 threat points, and 55 waypoints generated using the Maximin-Whole Area. After applying the Maximin-Whole Area-Removing to this problem instances, the number of waypoints in the area of operation went from 55 waypoints down to only 29 waypoints as shown in Figure 3-7.

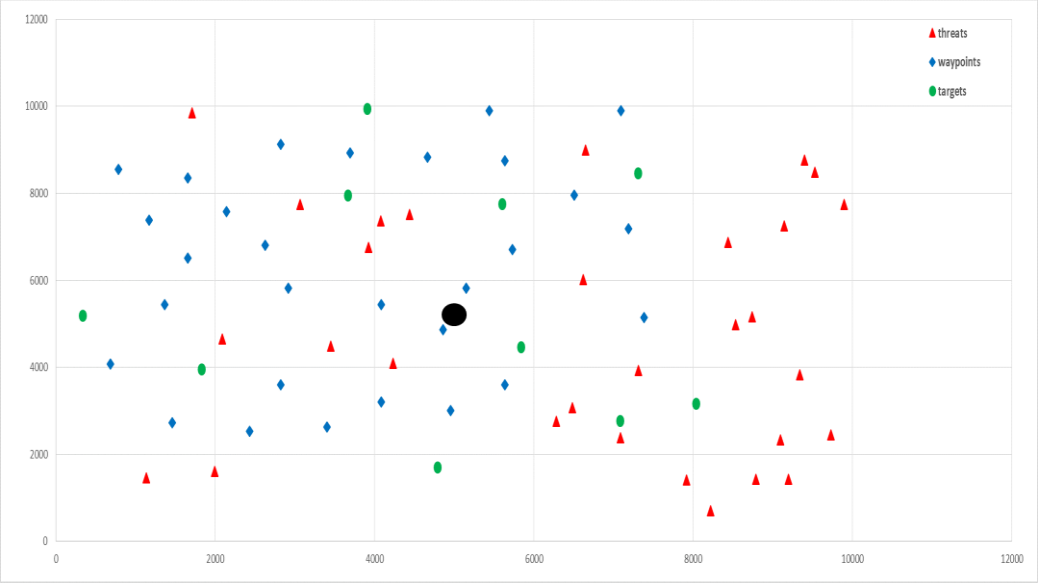


Figure 3-7 10 targets, 30 threats, and 29 waypoints (Maximin-Whole Area-Removing)

3.5.8 Maximin-Whole Area-Removing-Total Threat Reduction

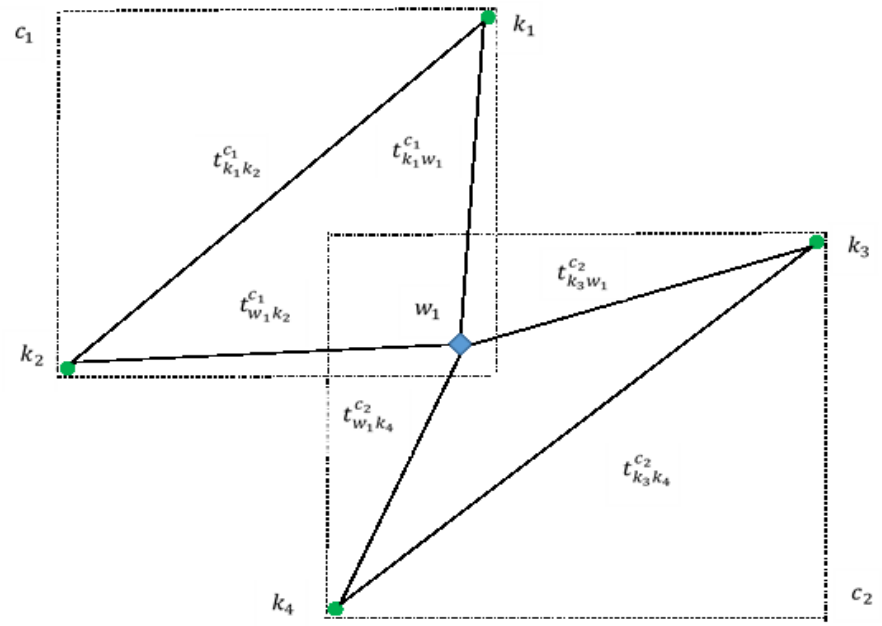
In this method, after the Maximin-Whole Area-Removing waypoint generation, the total threat reduction for each remaining waypoint in the area of operation is calculated based on the number based on the set of rectangles in which the waypoint lies. This is done by placing a rectangle, c, between any two targets in the area of operation including the source node. Then, the process takes each waypoint, w, of the remaining waypoints in the area of operation and determines the set of rectangles N_w^c in which it lies. After that, the total threat reduction for a waypoint, w, is calculated. The threat levels, t_{ij}^c, t_{iw}^c, and t_{wj}^c for the links going from target i to target j, from target i to waypoint w, and from waypoint w to target j, respectively, for each rectangle c in which target i, target j, and waypoint w lie inside, are calculated based on the threat level approximation described in section 3.4. The threat reduction for each waypoint w inside a rectangle c is calculated as follows

$$Threat\ reduction\ (w,c) = t_{ij}^c - \left(t_{iw}^c + t_{wj}^c\right) \tag{3.19}$$

Then, the total threat reduction for each waypoint w based on the set of rectangles, N_w^c, that the waypoint lies in is calculated based on the following equation

$$Total\ Threat\ reduction\ (w) = \sum_{c \in N_w^c} Threat\ reduction\ (w,c) \tag{3.20}$$

An example of the total threat reduction for one waypoint lying in two rectangles is shown in Figure 3-8. Then, the waypoints are sorted from highest to lowest based on their total threat reduction. Finally, depending on the planner's preference for the number of waypoints, the best set of waypoint in terms of highest total threat reduction are selected.



$$Total\ Threat\ reduction\ (w_1) = \ Threat\ reduction\ (w_1, c_1) + \ \ Threat\ reduction\ (w_2, c_2)$$

$$Total\ Threat\ reduction\ (w_1) = \ \ t^{c_1}_{k_1k_2} - \left(t^{c_1}_{k_1w_1} + t^{c_1}_{w_1k_2}\right) \ + \ \ t^{c_2}_{k_3k_4} - \left(t^{c_2}_{k_3w_1} + t^{c_2}_{w_1k_4}\right)$$

Figure 3-8 Total Threat Reduction of a Waypoint Lying in Two rectangles

3.6 Minimum Dependent Set (MDS) Constraints

In integer programming, the constraint $\sum_{u\in U}\sum_{f\in F_u} t_{uf}\ x_{uf} \le d$, in (3.5) and in (3.10) as well, is considered a binary knapsack constraint, since its coefficients are nonnegative constants. From (3.5), a set of valid inequality constraints that can cut off some fractional solutions and encourage integrality can be derived. The set of valid inequality constraints are called Minimum Dependent Sets (MDS) according to Visoldilokpun [41] and was reported in Nemhauser and Wolsey [42]. According to Kellerer et al. [47], the minimal dependent sets are also known as minimal cover inequalities, and defining facets for the knapsack polytope was originally studied back in 1975 by Balas [48],

Hammer et al. [49], and Wolsey [50]. Consider the following feasible set for the binary knapsack problem

$$S = \left\{ x \in \{0,1\}^n \mid \sum_{j \in N} q_j x_j \leq g \right\} \tag{3.21}$$

Where, $N = \{1,2,\dots,n\}$, $q_j \in Z_+^1 \ \forall j \in N$, and $g \in Z_+^1$ are positive integers. Assume that $q_j \leq b \ \forall j \in N$, and these coefficients are ordered monotonically such that $q_1 \geq q_2 \geq \cdots \geq q_n$. Let $C \subset N$ such that $\sum_{j \in C} q_j > b$. Then, set C is called a *dependent set* when the characteristic vector $x^C \notin S$ and its components $x_j^C = 1 \ \forall j \in C$ and $x_j^C = 0 \ \forall j \notin C$. Otherwise, $x^C \in S$ and set C is called an *independent set*. The dependent set C is called minimum when all of its contained subsets are independent. Notice that, in order to maintain feasibility, it is impossible to set all of the variables in set C to one at the same time. Therefore, according to Nemhauser and Wolsey [42], the inequality $\sum_{j \in C} x_j \leq |C| - 1$, where C is a dependent set, is a valid inequality for feasible set S.

In our implementation, the feasible region for the binary knapsack constrain in (3.5) is

$$\hat{S} = \left\{ x \in \{0,1\}^{|F|} \mid \sum_{u \in U} \sum_{f \in F_u} t_{uf} \ x_{uf} \ \leq \ d \right\} \tag{3.22}$$

By letting a finite set D that contains all minimum dependent sets for $\hat{S}$, then the following MDS constraints are valid inequalities for the feasible region $\hat{S}$

$$\sum_{u,f \in C} x_{uf} \ \leq |C| - 1 \qquad \forall C \in D \tag{3.23}$$

Consider the continuous linear programing relaxation for the feasible set of the knapsack constraint in (3.22)

$$\tilde{S} = \left\{ 0 \leq x \leq 1 \mid \sum_{u \in U} \sum_{f \in F_u} t_{uf}\ x_{uf}\ \leq\ d \right\} \tag{3.24}$$

There is at least one fractional solution $\tilde{x} \in \tilde{S}$ that violates (3.23). The proof presented here can be found in Visoldilokpun [41] and in Nemhauser and Wolsey [42].

Let $C \subset F$ be a minimum dependent set, such that $\forall\, k \in C$ and $C \backslash \{k\}$ is independent. That is, removing any variable from the dependent set C gives a set whose coefficients' sum does not exceed d. If $\sum_{j \in F} t_j > d$, then consider $\tilde{x}$ of the following forms:

$$\tilde{x}_j = 1 \qquad if\ j \in C \backslash \{k\}, \tag{3.25}$$

$$\tilde{x}_j = 0 \qquad if\ j \in F \backslash C, \tag{3.26}$$

$$\tilde{x}_k = \frac{d - \sum_{j \in C \backslash \{k\}} t_j}{t_k} \qquad if\ j = k. \tag{3.27}$$

From the last fractional form in (3.27), it is clear that

$$\sum_{j \in C} \tilde{x}_j\ = |C| - 1 + \frac{d - \sum_{j \in C \backslash \{k\}} t_j}{t_k}\ >\ |C| - 1 \tag{3.28}$$

Therefore, $\exists \tilde{x} \in \tilde{S}$ that does not satisfy the valid inequalities in (3.23). This means that the MDS constraints in (3.23) cuts off the fractional solution $\tilde{x}$.

3.7 Branch and Cut and Price Methodology

In a VRP, even with a small-sized problem, there could be a large number of possible routes. The Branch-and-Cut-and-Price (BCP) methodology is a very efficient method when dealing with problems with a large number of variables. In BCP, only a subset of variables is considered. BCP methodology is used as a platform to solve our problem. It is based on the known integer programing branch-and-bound algorithm for solving mixed integer linear programing problems. To make sure we obtain an integer solution, our problem is solved within a branch-and-bound tree. First, the continuous relaxation of the problem, which is called the Restricted Master Problem (RMP), which only includes a subset of all the routes $\bar{F} \subset F$, is solved within each node of the tree to obtain a solution x^*. Then, the cut step is performed. Traditionally, cuts are added in order to tighten the relaxation so that the feasible region of the sub problem is nearly approximated by the relaxed set. However, in this implementation, the cut step generates valid inequalities called the Minimum Dependent Set (MDS) constraints, $C \subset D,$ and adds them to the RMP in order to cut a fractional solution x^* and encourage integrality. After that, in the pricing step, new routes, variables, with negative reduced costs are generated using dynamic column generation algorithm and added to the RMP as needed. Finally, bounds are updated and branching is carried out using a variant of Ryan and Foster [44] branching logic. The diagram in Figure 3-9 shows an overview of the BCP process at each node of the tree. The BCP steps are repeated and when there is no route, variable, needed to be added to the RMP, the steps are terminated and the optimal integer solution x^* is found.

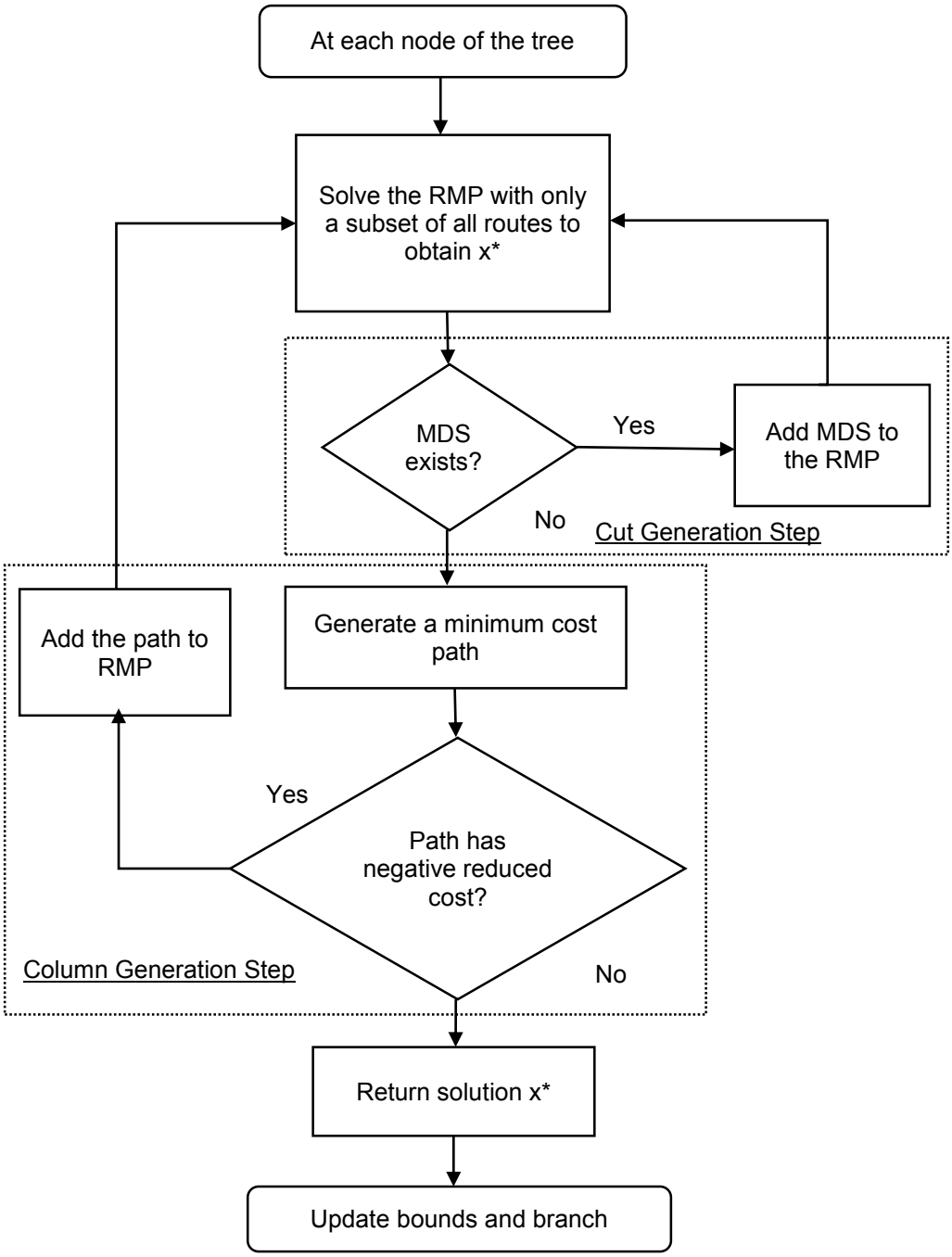


Figure 3-9 An Overview of the BCP Process at Each Node of the Tree.

3.7.1 Cut Generation

At each node of the tree, after the continuous relaxation of the problem is solved to obtain a solution x^*, a cut generation step is called in order to try to generate a Minimum Dependent Set (MDS) constraint. If an MDS exists, then, it will be added as a cut in the RMP in order to cut off a fractional solution x^* and encourage solution integrality. The MDS constraints are generated using a heuristic method used in Visoldilokpun [41], which is similar to solving the linear programming relaxation of a binary Knapsack problem Chvátal [45].

Consider the following integer programing problem

$$min \sum_{u \in U} \sum_{f \in F_u} (1 - x^*_{uf})\ \tilde{x}_{uf} \tag{3.29}$$

$$s.t. \sum_{u \in U} \sum_{f \in F_u} t_{uf}\ \tilde{x}_{uf}\ \geq\ d + \varepsilon \tag{3.30}$$

$$\tilde{x}_{uf} \in \{0,1\}, \qquad \forall\, u \in U, f \in F_u \tag{3.31}$$

A feasible solution to the problem in (3.29)-(3.31) could be one where the objective value is less than ε. This implies that there should be at least one MDS constraint that cuts off the fractional solution x^*. The heuristic begins by sorting variables, x_{uf}, in an ascending order with respect to their corresponding terms $\frac{1-x^*_{uf}}{t_{uf}}$. In addition, larger t_{uf} values are used in order to break ties. Let N be the current total number of variables, x_{uf}, that are already in the RMP. Let i in a variable x^i_{uf} represent the i^{th} order of the sorted sequence $i = 1,2, \dots N$. Let set C be a minimum dependent set. Beginning from $i = 1$, variable x^i_{uf} is greedily added into set C until the complete minimum dependent set C is found. The steps of the heuristic that generates the MDS cuts is illustrated by Algorithm 1.

Algorithm 1 MDS Heuristic

Initialization: Let a set $\boldsymbol{C} \leftarrow \emptyset$ be an MDS, and $\boldsymbol{i} = \boldsymbol{1}$.

Selection: Add a variable $\boldsymbol{x_{uf}^{i}}$ into the MDS , $\boldsymbol{C} \leftarrow \boldsymbol{C} \cup \{\boldsymbol{u}, \boldsymbol{f}\}$, and $\boldsymbol{i} \leftarrow \boldsymbol{i} + \boldsymbol{1}$.

Check:

if $\boldsymbol{i} \leq \boldsymbol{N}$ then

if $\sum_{\boldsymbol{u} \in \boldsymbol{U}} \sum_{\boldsymbol{f} \in \boldsymbol{F_u}} \boldsymbol{a_{Cuf}}\ \boldsymbol{t_{uf}}\ \boldsymbol{x_{uf}^{*}} > \boldsymbol{d}$ and $\sum_{\boldsymbol{u} \in \boldsymbol{U}} \sum_{\boldsymbol{f} \in \boldsymbol{F_u}} \boldsymbol{a_{Cuf}}\ \boldsymbol{x_{uf}^{*}} > |\boldsymbol{C}| - \boldsymbol{1}$ then

Return the MDS $\boldsymbol{C}$.

else

Return to Selection.

end if

else

Return $\emptyset$.

end if

The set of MDS constraints is as follows

$$\sum_{u \in U} \sum_{f \in F_u} a_{Cuf}\ x_{uf} \leq |C| - 1 \qquad \forall C \in D \tag{3.32}$$

where the binary constant $a_{Cuf} = 1$ if a variable $x_{uf} \in C$, and 0 otherwise. Notice that the integer coefficients of the MDS constraints in (3.32) should also encourage solution integrality.

3.7.2 Reduced Cost

The continuous relaxation problem of UAVRP1 after relaxing the integrality requirement and adding the MDS constraints is as follows:

$$min \sum_{u \in U} \sum_{f \in F_u} E\left[\, \tilde{e}_{uf}\right] \, x_{uf} \tag{3.33}$$

$$s.t. \sum_{u \in U} \sum_{f \in F_u} a_{kuf} \; x_{uf} \; \geq \; 1 \qquad \forall k \in \; K \tag{3.34}$$

$$\sum_{u \in U} \sum_{f \in F_u} a_{wuf} \; x_{uf} \; \geq \; 0 \qquad \forall w \in \; W \tag{3.35}$$

$$\sum_{f \in F_u} x_{uf} \quad \leq \; n_u \qquad \forall u \in \; U \tag{3.36}$$

$$\sum_{u \in U} \sum_{f \in F_u} t_{uf} \; x_{uf} \; \leq \; d \tag{3.37}$$

$$\sum_{u \in U} \sum_{f \in F_u} a_{Cuf} \; x_{uf} \; \leq |C| - 1 \qquad \forall C \in \; D \tag{3.38}$$

$$0 \; \leq x_{uf} \qquad \forall \; u \in U, f \in F_u \tag{3.39}$$

Let the dual variables for the constraints in (3.34), (3.35), (3.36), (3.37) and (3.38) be represented by π_l, π_w, π_u, ρ_t, and ρ_C, respectively. Let x^*, π_l^*, π_w^*, π_u^*, ρ_t^*, and ρ_C^* be an obtained optimal solution after solving the relaxation problem (3.33)-(3.39). Therefore, the reduced cost $\bar{c}_{uf}$ for variable x_{uf} can be calculated as follows

$$\bar{c}_{uf} = E\left[\, \tilde{e}_{uf}\right] - \sum_{k \in f} a_{kuf} \; \pi_l^* - \sum_{w \in f} a_{wuf} \; \pi_w^* - \pi_u^* - t_{uf} \; \rho_t^* - \sum_{C \in D} a_{Cuf} \; \rho_C^*,$$

$$\forall \; f \in F_u, \forall \; u \in U \tag{3.40}$$

Using the equations in (3.15) and (3.17), the reduced cost $\bar{c}_{uf}$ for a variable x_{uf} can be rewritten as a link-based reduced cost as

$$\bar{c}_{uf} = \sum_{l \in f} E[\, \tilde{e}_l] - \sum_{l \in f} t_l \; \rho_t^* - \sum_{l \in f} \pi_l^* - \sum_{w \in f} \pi_w^* - \pi_u^* - \sum_{C \in D} a_{Cuf} \; \rho_C^*,$$

$$\forall\, f \in F_u, \forall\, u \in U \tag{3.41}$$

Observe that constraints in set (3.35) can be eliminated from the formulation. Their corresponding dual value π_w^* should be zero since these constraints are redundant. Based on (3.41), we can say that the term $(E[\,\tilde{e}_l] - t_l\,\rho_t^* - \pi_l^*)$ represents the cost of link $l \in L$ in a graph G. In addition, the term $(-\pi_u^*)$ represents the cost of using a UAV $u \in U$. Finally, the term (ρ_C^*) represents the cost of adding a MDS cut, if MDS $C \in D$ existed. Observe that, the reduced cost for a variable that was already generated and added to the RMP is zero because the term $(\sum_{C\in D} a_{Cuf}\,\rho_C^*)$ is negative. Therefore, the term $(\sum_{C\in D} a_{Cuf}\,\rho_C^*)$ does not need to be subtracted when generating a new variable that has not been added to the RMP.

Similarly, the continuous relaxation problem of the UAVRP2 after relaxing the integrality requirement and adding the MDS constraints is as follows

$$max \sum_{k\in K} B_k x_k \tag{3.42}$$

$$s.t. \sum_{u\in U}\sum_{f\in F_u} a_{kuf}\; x_{uf} \;\geq\; x_k \qquad \forall k \in K \tag{3.43}$$

$$\sum_{f\in F_u} x_{uf} \;\leq\; n_u \qquad \forall u \in U \tag{3.44}$$

$$\sum_{u\in U}\sum_{f\in F_u} t_{uf}\; x_{uf} \;\leq\; d \tag{3.45}$$

$$\sum_{u\in U}\sum_{f\in F_u} a_{Cuf}\; x_{uf} \;\leq |C| - 1 \qquad \forall C \in D \tag{3.46}$$

$$0 \;\leq\; x_k \;\leq\; 1 \qquad \forall\, k \in K \tag{3.47}$$

$$0 \leq x_{uf} \qquad \forall\, u \in U, f \in F_u \tag{3.48}$$

By assigning dual variables for the problem constraints and obtaining an optimal solution after solving the relaxation problem, the new reduced cost $\bar{c}_{uf}$ for variable x_{uf} can be calculated as follows

$$\bar{c}_{uf} = -\sum_{k \in f} a_{kuf}\, \pi_l^* - \pi_u^* - t_{uf}\, \rho_t^* - \sum_{C \in D} a_{Cuf}\, \rho_C^*,$$

$$\forall\, f \in F_u, \forall\, u \in U \tag{3.49}$$

The new reduced cost $\bar{c}_{uf}$ for a variable x_{uf} can be rewritten as a link-based reduced cost as

$$\bar{c}_{uf} = -\sum_{l \in f} t_l\ \rho_t^* - \sum_{l \in f} \pi_l^* - \pi_u^* - \sum_{C \in D} a_{Cuf}\, \rho_C^*,$$

$$\forall\, f \in F_u, \forall\, u \in U \tag{3.50}$$

Based on (3.50), we can say that the term $(-t_l\, \rho_t^* - \pi_l^*)$ represents the cost of link $l \in L$ in a graph G. Observe that, the linked-based reduced cost $\bar{c}_{uf}$ for variable x_{uf} in (3.50) in UAVRP2 is the same as the one in (3.41) in UAVRP1 without the term $(\sum_{l \in f} E[\,\tilde{e}_l])$.

In the next section, we discuss generating new variables, routes, using column generation.

3.7.3 Column Generation

The column generation step is usually used as a pricing step in BCP. The column generation step generates new routes and adds them to the RMP only when they are needed. This actually helps in improving the efficiency of the BCP methodology. The column generation step must find a simple path with negative reduced cost in order to add it to the RMP. The simple path originates and terminates at the same base, visits targets, and may visit waypoints.

In UAVRP1, we create an instance graph. Let the source node v_0 and the terminal node v_{n+1} represent the depot, where n is the number of all given targets and waypoints. Each edge, y_l, of the graph is assigned an associated cost $c_{ij} = E[\,\tilde{e}_l] - t_l\,\rho_t^* - \pi_l^*$, which represents the cost for traveling from node v_i to node v_j over a link $l \in L$ from equation (3.41). Let f^* be the minimum cost path found. Then, the total cost of traveling route f^*, is just the additive sum of the cost of traveling over all links $l \in f^*$. That is $\sum_{l \in f^*}\{E[\,\tilde{e}_l] - t_l\,\rho_t^* - \pi_l^*\}$ from (3.41). In order to obtain the reduced cost $\bar{c}_{uf^*}$, the cost, $-\pi_u^*$, of using a UAV of type $u \in U$ is added. If the path f^* has a negative reduced cost, $\bar{c}_{uf^*} < 0$, then it is added to the RMP.

After the RMP problem is solved at a node of the tree using only a subset $\bar{F} \subseteq F$, and if an MDS cut does not exist, the dual solution is sent to the column generation sub problem in order to generate a new variable. The Integer Programming Shortest Path algorithm (IPSP), Nemhauser and Wolsey [42], is used as our column generation algorithm engine to find a simple path with negative reduced cost in an instance graph G. The sub problem is called a Minimum Cost Network Flow problem [41] and is solved using a mixed integer linear programing solver to obtain the minimum cost route f^*.

Let $\delta_{v_i}^+ = \{j\colon (i,j) \in E\}$ be the set of all edges that are departing from node v_i, $\delta_{v_i}^- = \{j\colon (j,i) \in E\}$ be the set of all entering edges to node v_i, and $\Omega(f)$ be the set of all visited nodes, targets and waypoints, in route f. Let $\bar{F} \subset F$ donate a finite set of all paths that were previously generated and added to the RMP. A simple path and a negative cost cycle are referred to as walks and denoted ψ. The negative cost cycles are called invalid walks, and the set of all invalid walks is denoted $\mathcal{H}$.

For the UAVRP1, the formulation of the MCNF sub problem is as follows

$$\min \sum_{l \in L} \{\, E[\,\tilde{e}_l] - \, t_l \, \rho_t^* - \pi_l^*\}\, y_l \tag{3.51}$$

$$s.t. \sum_{l \in \delta^+_{v_i}} y_l - \sum_{l \in \delta^-_{v_i}} y_l = b(v_i) \qquad \forall v_i \in V \qquad (3.52)$$

$$\sum_{l \in \delta^+_{v_i}} y_l \leq 1 \qquad \forall v_i \in V \qquad (3.53)$$

$$\sum_{l \in f} y_l \leq |\Omega(f)| \qquad \forall f \in \bar{F} \qquad (3.54)$$

$$\sum_{l \in \psi} y_l \leq h(\psi) \qquad \forall \psi \in \mathcal{H} \qquad (3.55)$$

$$y_l \in \{0,1\} \qquad \forall l \in L \qquad (3.56)$$

The objective function in (3.51) minimizes the reduced cost. The flow balance constraints in set (3.52) require paths to begin at the source node v_0 and end at the terminal node v_{n+1}, where $b(v_0) = 1$, $b(v_{n+1}) = -1$, and $b(v_i) = 0 \; \forall \; v_i \in V \backslash \{v_0, v_{n+1}\}$. The constraints in set (3.53) limit the outflow for each node to be at most one. Both the set in (3.52) and (3.53) can help in preventing some cycles from forming. The route elimination constraints in set (3.54) prevent previously generated routes and already included in the RMP from being generated in future iterations. The left-hand side represents the links of the route that starts from the depot, visits some targets, and returns to the depot. While the right-hand side represents the nodes that are visited in the route without the source/terminal node. For example, visiting two nodes means the left-hand side includes three edges while the right-hand side equals two. The set in (3.54) is added a *priori* to solving the MCNF since $\bar{F}$ is a finite set. The invalid walks, $\psi \in \mathcal{H}$, are eliminated by the walk elimination constraints in set (3.55), where $h(\psi) = |\Omega(f)| - 1$. The set (3.55) is added dynamically to the MCNF. The constraints in set (3.56) require the decision variables y_l to be binary, which indicates links selected in the route f^*.

Similarly, in UAVRP2, we create an instance graph. Each edge, y_l, of the graph is assigned an associated cost $c_{ij} = -t_l \, \rho^*_t - \pi^*_l$, which represents the cost for traveling from

node v_i to node v_j over a link $l \in L$ from equation (3.50). Let f^* be the minimum cost path found. Then, the total cost of traveling route f^*, is just the additive sum of the cost of traveling over all links $l \in f^*$. That is $\sum_{l \in f^*}\{-t_l\, \rho_t^* - \pi_l^*\}$ from (3.50). In order to obtain the reduced cost $\bar{c}_{uf^*}$, the cost, $-\pi_u^*$, of using a UAV of type $u \in U$ is added. If the path f^* has a negative reduced cost, $\bar{c}_{uf^*} < 0$, then it is added to the RMP.

After the RMP problem is solved at a node of the tree using only a subset $\bar{F} \subseteq F$, and if a MDS cut does not exist, the dual solution is sent to the column generation sub problem in order to generate a new variable. We still use the Integer Programming Shortest Path algorithm (IPSP) as our column generation algorithm engine to find a simple path with negative reduced cost in an instance graph G.

Let r be a predetermined constant parameter to limit the route travel time. Let m_l be the travel time over a link l and is calculated as

$$m_l = \frac{\tau_l}{v_l^g} \tag{3.57}$$

where, τ_l is the link distance between node i and node j and v_l^g is the ground speed, calculated as previously described in section 3.3.

For the UAVRP2, the formulation for MCNF sub problem is as follows

$$\min \sum_{l \in L}\{-t_l\ \rho_t^* - \pi_l^*\}\, y_l \tag{3.58}$$

$$s.t. \sum_{l \in \delta_{v_i}^+} y_l - \sum_{l \in \delta_{v_i}^-} y_l\ =\ b(v_i) \qquad \forall v_i \in V \tag{3.59}$$

$$\sum_{l \in \delta_{v_i}^+} y_l \leq 1 \qquad \forall v_i \in V \tag{3.60}$$

$$\sum_{l \in f} y_l \leq\ |\Omega(f)| \qquad \forall f \in \bar{F} \tag{3.61}$$

$$\sum_{l \in \psi} y_l \leq h(\psi) \qquad \forall \psi \in \mathcal{H} \tag{3.62}$$

$$\sum_{l \in f} m_l \, y_l \leq r \tag{3.63}$$

$$y_l \in \{0,1\} \qquad \forall l \in L \tag{3.64}$$

The objective function in (3.58) minimizes the reduced cost in UAVRP2. In constraint (3.63), the nonnegative constant parameter, r, limits the route travel time for a UAV.

Notice that the link cost c_{ij} and the reduced cost $\bar{c}_{uf^*}$ are not restricted in sign. As a result, a simple path generated by the column generation process usually has an embedded negative cost cycles. A simple path with a cost cycle is shown in Figure 3-10. The cost of the cycle is negative. However, the cost of the simple path could be positive or negative.

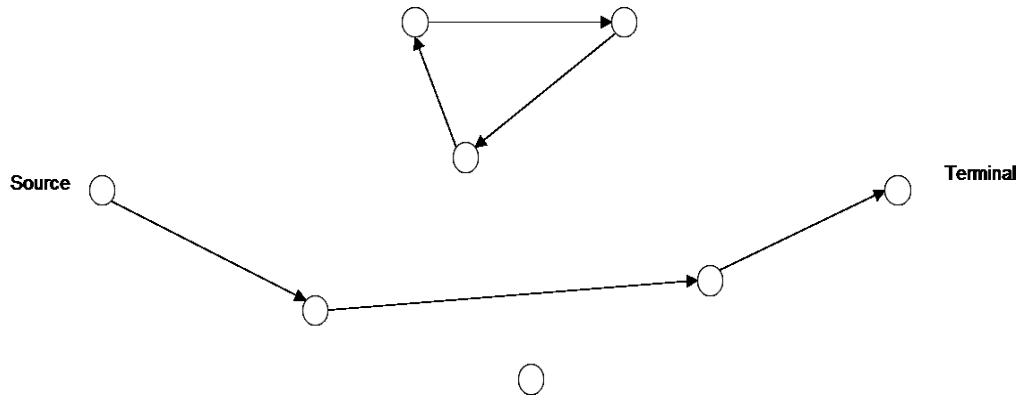


Figure 3-10 A Simple Path with Negative Cost Cycle

Therefore, the simple path and the negative cost cycle cannot be added together to the RMP. Usually, a set of cut constraints that eliminate negative cost cycles are added to the sub problem. According to Visoldilokpun [41], we have the following four scenarios

1. If we find a simple path with a negative reduced cost, and we do not find a cycle, the simple path will be added to the RMP.
2. If we find a simple path with a negative reduced cost, and we find negative cost cycles, the simple path alone will be added to the RMP and no cuts to eliminate the cycles are added to the sub problem.
3. If we find a simple path with positive reduced cost and, we do not find cycles, the simple path will be ignored, not added to the RMP. The column generation step is terminated since no simple path with a negative reduced cost is found.
4. If we find a simple path with positive reduced cost and we find negative cost cycles, the simple path will be ignored, not added to the RMP, and cuts to eliminate the cycles will be added to the sub problem. The column generation step is repeated until a simple path with a negative reduced cost is found or shows that no such path exists.

However, the negative cost cycle cuts are time consuming. Therefore, a simple path heuristic is discussed in the next section.

3.7.4 Simple Path Heuristic

If we find a simple path with positive reduced cost and we find negative cost cycles in the column generation step, we ignore the simple path and add cycle elimination constraints to eliminate the negative cost cycles. The step is then repeated until we find a path with negative reduced cost or show that no such path exists. This process can be time

consuming and impractical with large-sized problems. Therefore, in order to expedite the process and make the column generation step more efficient, we use a simple path heuristic that was introduced by Visoldilokpun [41]. The main goal of the simple path heuristic is to generate simple paths from the negative cost cycles and the one that has the least negative reduced cost is added to the RMP.

Let $f^{\bar{\psi}}$ represent a simple path. Let a set of all negative cost cycles that might accompany the simple path be denoted Θ_f. Thus, the path $f \equiv f^{\bar{\psi}} \cup \Theta_f$, where $\Theta_f \neq \emptyset$, represents a path that has negative cost cycles. Assume that the path f that was found by the column generation algorithm has a simple path $f^{\bar{\psi}}$ with positive reduced cost and has a negative cost cycle $\psi \in \Theta_f$ as shown in Figure 3-10. Instead of adding walk elimination constraints to eliminate the cost cycle, the simple path heuristic will be invoked to generate simple paths from negative cost cycles.

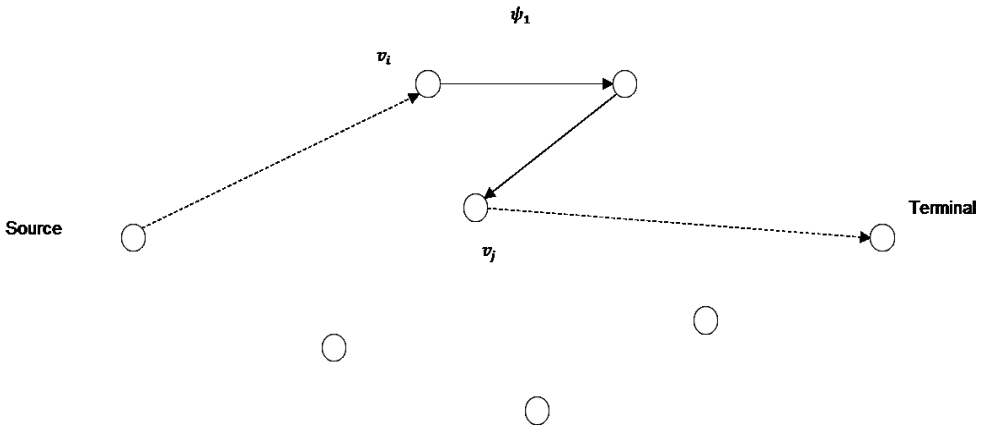


Figure 3-11 Generating Simple Path from a Cycle Using Simple Path Heuristic

In each iteration of the heuristic, a link l that is part of a cycle is deleted and two new edges are heuristically added in order to form a new simple path as shown in Figure 3-11. When the link between v_i and v_j is deleted, two new links are added; one connects the source node and node v_i and the other connects node v_j and the terminal node. The

number of paths that the heuristic can generate equals the number of edges in a cycle, and the one that will be added to the RMP is the one that has the most negative reduce cost if such path exists and does not violate other path constraints, such as (3.54), (3.61) or (3.63).

3.7.5 Branching Logic

We use the follow-on branching, which was developed by Vance et al. [43] and is a variant of Ryan and Foster branching [44], as our branching logic because it is computationally more efficient than conventional variable branching and can be applied to the column generation sub problem. Thus, at each node of the search tree, column generation can be implemented.

In general, branching requires partitioning and no overlapping. This prevents the same solution from occurring on different branches. It also requires a clear definition of a solution in the leaves. Consider two consecutive targets; a and b. In the branch-and-bound tree, the up branch requires that any route that has target a must have target b following it, any route that has target b must have target a preceding it, and all other routes are deleted. In the sub problem, the edge that is departing from target a and entering target b will be kept, while all other departing edges from target a and all other entering edges to target b will be set to 0.

On the other side of the branch-and-bound tree, the down branch requires deleting any route that has a followed by b and keeping all other routes. In the sub problem, the edge that is departing from target a and entering target b will be set to 0, while all other departing edges from target a and and all other entering edges to target b will be kept.

In the following chapter, we present computational study for the first formulation (UAVRP1) and for the second formulation (UAVRP2).

Chapter 4

COMPUTATIONAL STUDY

We did a computational study to solve the UAVRP1 and the UAVRP2 with BCP methodology. We implemented the UAVRP1 and the UAVRP2 using the BCP framework that was coded in C++ and part of the Computational Infrastructure for Operations Research (COIN-OR) project, which is an open-source mathematical software used in the operations research community. IBM ILOG CPLEX Optimization Studio 12.5 was used as a linear programing solver interface with COIN-OR. A 2.67GHz Intel Xeon computer was used to conduct our study.

We experimented with four different algorithms. The first one considers only the Delayed Column Generation algorithm, referred to as the DCG. The second one considers the Delayed Column Generation algorithm and the Minimum Dependent Set heuristic, referred to as the DCG-MDS. The third one considers the Delayed Column Generation algorithm and the simple path heuristic, referred to as the DCG-HEU. The fourth one considers the Delayed Column Generation algorithm, the simple path heuristic, and the Minimum Dependent Set heuristic, referred to as the DCG-HEU-MDS. The DCG and the DCG-MDS spent very long times to generate routes. Therefore, only the DCG-HEU algorithm and the DCG-HEU-MDS algorithm are considered.

We create different sets of problem instances. With each set, 5 UAVs are used. The UAVs originates and terminates at the same base, which is located at the center of the area of operation. The targets and threat points are randomly generated.

4.1 Computational Study for the First Formulation (UAVRP1)

At the beginning, we ignored the total threat level constraint in (3.37) and solved the problem without it. The obtained optimal solution included the total expected fuel burn

and the corresponding total threat level. We considered this case as the case where the constant parameter, d, equals 100%. The total threat level of the optimal solution found in this case was used as a reference level to obtain more levels of d. Six more levels of d were considered. They are 95%, 90%, 85%, 80%, 75%, and 75% of the threat level of the optimal solution found in the case with $d = 100\%$.

We created 2 sets of problem instances. In both sets, the waypoints generated using the Maximin-Whole Area method previously described in chapter 3. The first set of problem instances included 10 targets, 30 threat points, and 30 waypoints as shown in Figure 4-1. The second set of the problem instances included 20 targets, 60 threat points, and 60 waypoints as shown in Figure 4-2.

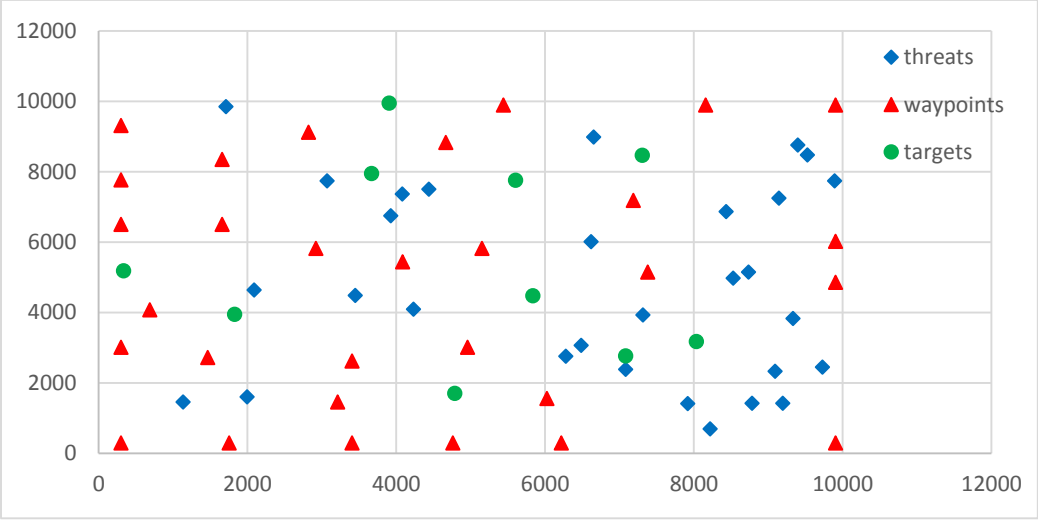


Figure 4-1 Set 1 of Problem Instances, 10 Targets, 30 Threat Points, and 30 Waypoints (Maximin-Whole Area)

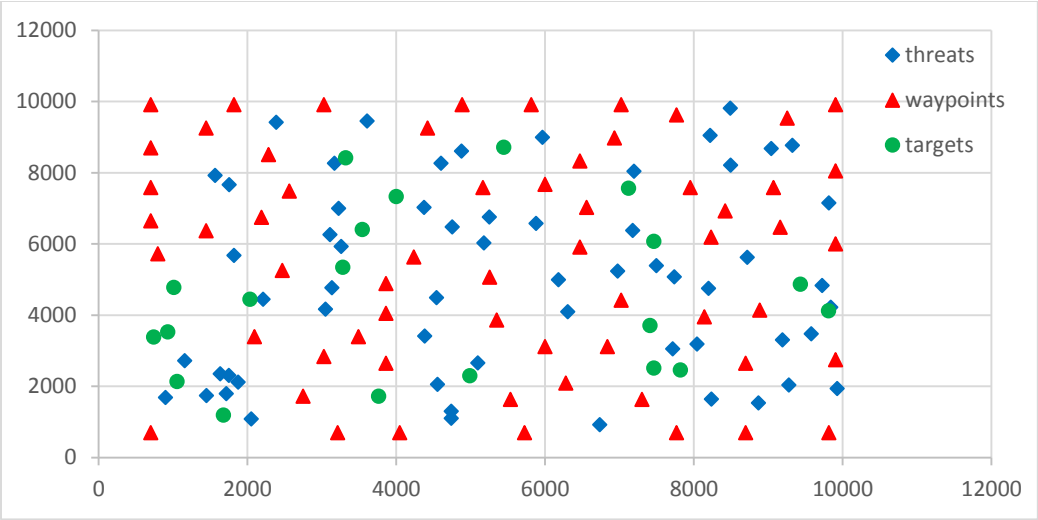


Figure 4-2 Set 2 of Problem Instances, 20 Targets, 60 Threat Points, and 60 Waypoints (Maximin-Whole Area)

4.1.1 Results for Ten-Target Case (UAVRP1)

For the ten targets case, the run time is set at 20 hours for each d level and with each algorithm type. The results for the first set that includes 5 UAVs, 10 targets, 30 waypoints, and 30 threat points, for both algorithms, DCG-HEU and DCG-HEU-MDS, are presented in Table 4-1. The columns include seven levels of d, the total threat level of the optimal solution, the optimal total expected fuel burn, the number of UAVs used, the visited route, and the number of visited waypoints.

Table 4-1 Results for Set 1 for Both DCG-HEU and DCG-HEU-MDS Algorithms

Test	d Value	Total Threat Level	Total Expected Fuel Burn (lbs.)	Number of Used UAVs	Route	Number of Visited Waypoints
d=100%	6.58E-07	6.58E-07	28,030.81	1	1 2 5 7 10 3 8 4 6 9	0
d=95%	6.25E-07	5.05E-07	28,043.58	1	1 2 5 7 (14) 10 3 8 4 6 9	1
d=90%	5.92E-07	5.05E-07	28,043.58	1	1 2 5 7 (14) 10 3 8 4 6 9	1
d=85%	5.59E-07	5.05E-07	28,043.58	1	1 2 5 7 (14) 10 3 8 4 6 9	1
d=80%	5.26E-07	5.05E-07	28,043.58	1	1 2 5 7 (14) 10 3 8 4 6 9	1
d=75%	4.93E-07	3.88E-07	28,233.76	1	1 2 (18) 5 7 (14) 10 3 8 4 6 9	2
d=70%	4.60E-07	3.88E-07	28,233.76	1	1 2 (18) 5 7 (14) 10 3 8 4 6 9	2

The results show that as the threat level tolerance d decreases, the total expected fuel burn increases and the number of waypoints visited increases. Notice that, in the route for the case with $d = 100\%$, the link between target 7 and 10 is a high threat level link. The route in the case with $d = 95\%$ shows that as the threat level decreases, the UAV will no longer travel over this high threat level link. Instead, the UAV will try to visit a waypoint, in particular 14, in order to comply with the reduction in the threat level. A similar thing can be said about the high threat level link between targets 2 and 5 as shown in the case with $d = 75\%$. The results also show that we use only one UAV. Based on these results, the UAVRP1 for the ten targets case is considered a traveling salesman problem with waypoint generation.

In order to aid the mission path planners to pick the case that is acceptable for their risk preference, we plotted the efficient frontiers for the total expected fuel burn and the corresponding total threat level as shown in Figure 4-3.

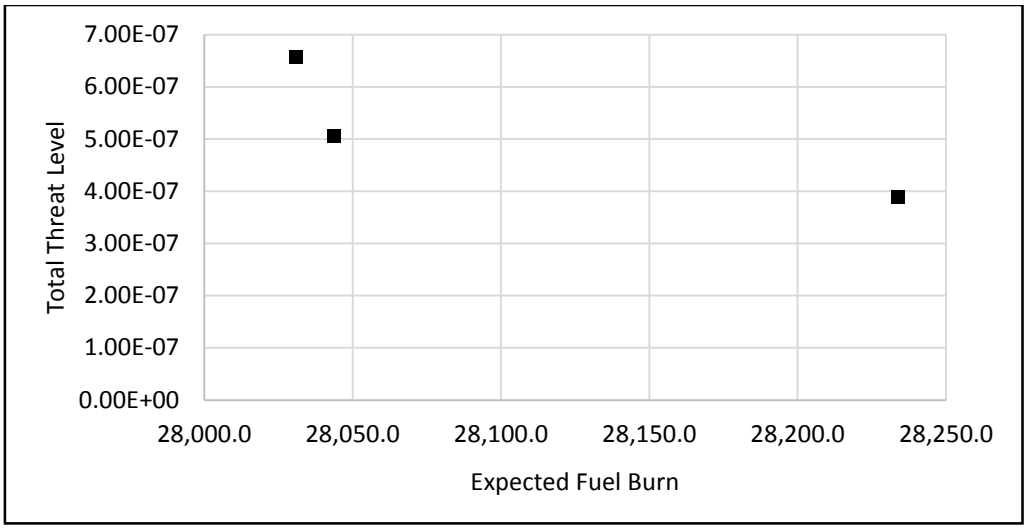


Figure 4-3 Efficient Frontier for Set 1

Table 4-2 shows the total number of variables that were generated in the column generation step and added to the RMP until finding the best solution and until the BCP algorithm run finished for the first set, 5 UAVs, 10 targets, 30 waypoints, and 30 threat points, for both algorithms, the DCG-HEU and the DCG-HEU-MDS.

Table 4-2 Generated Variables for Set 1

	DCG-HUE		DCG-HUE-MDS	
D	Number of Generated Variables until the Best Solution Found	Number of Generated Variables until the Algorithm Finished	Number of Generated Variables until the Best Solution Found	Number of Generated Variables until the Algorithm Finished
100%	45	60	45	60
95%	173	219	1210	1225
90%	55	123	106	150
85%	81	137	81	137
80%	55	117	90	124
75%	947	1247	66	547
70%	61	408	860	920

In Table 4-2, we actually could not see any clear relationship between the reduction of the total threat level and the number of generated variables for both algorithms. However, when comparing the case $d = 100\%$ with each other individual level of d alone, we can see an increase in the number of generated variables because of the reduction of the total threat level. The DCG-HEU-MDS algorithm uses the minimum dependent set constraints, which cut some fractional solutions and encourage the solution integrality. Therefore, we expect that the DCG-HEU-MDS algorithm to generate fewer variables than the DCG-HEU algorithm. However, based on the results in Table 4-2, we notice that the DCG-HEU algorithm generated fewer variables than the DCG-HEU-MDS algorithm to obtain the same optimal solution except for the case with $d = 75\%$, where we had to increase the run time above the 20 hour limit in order to obtain an optimal solution for the DCG-HEU algorithm in this particular case.

Figure 4-4 and Figure 4-5 show the CPU time in seconds until the best solution is found and until the BCP algorithm run finishes, respectively, for the first set, 5 UAVs, 10 targets, 30 waypoints, and 30 threat points, for both algorithms, the DCG-HEU and the DCG-HEU-MDS.

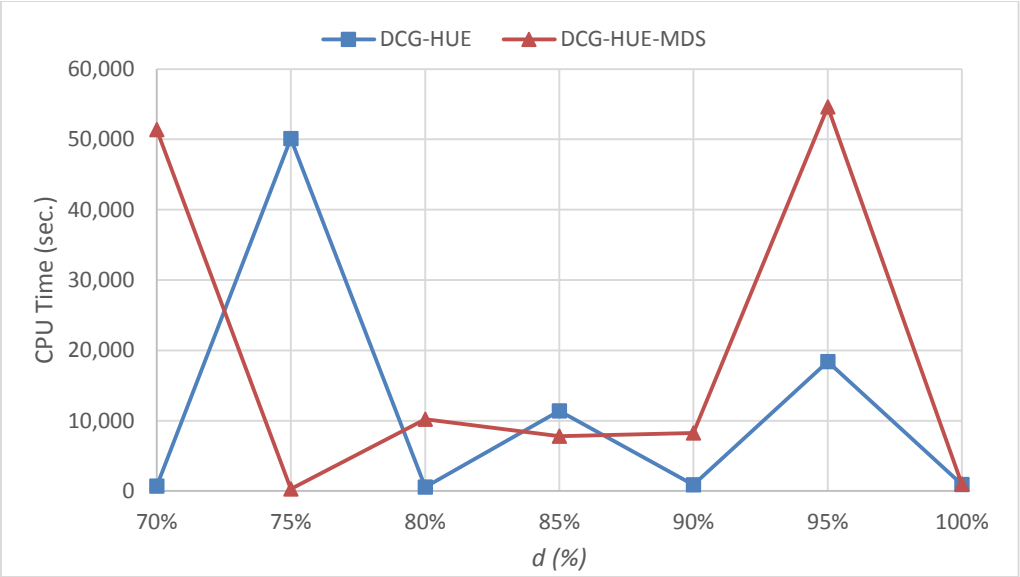


Figure 4-4 The CPU Time until Best Solution Found for Set 1

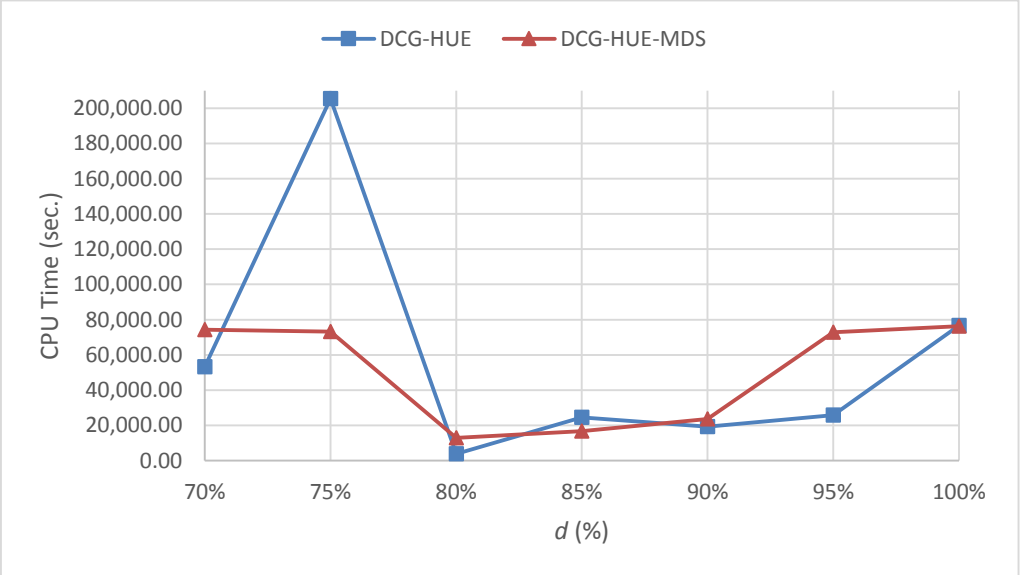


Figure 4-5 The CPU Time until The BCP Algorithm Finished for Set 1

Again, we expect the DCG-HEU-MDS algorithm to find an optimal solution faster than the DCG-HEU algorithm. However, in terms of CPU time, neither of the two algorithms, the DCG-HEU and the DCG-HEU-MDS, perform better than the other.

Table 4-3 shows the total number of MDS cuts that are generated in the cut generation step until the best solution is found and until the BCP algorithm run finishes for the first set, 5 UAVs, 10 targets, 30 waypoints, and 30 threat points, for only the DCG-HEU-MDS algorithm.

Table 4-3 MDS Cuts for Set 1 for DCG-HEU-MDS Algorithm

Test	MDS Cuts until the Best Solution Found	MDS Cuts until the Algorithm Finished
d = 100%	0	0
d = 95%	1	1
d = 90%	1	1
d = 85%	0	0
d = 80%	1	1
d = 75%	4	4
d = 70%	1	1

Notice that in the case with $d = 75\%$, we generated more MDS than other cases. This case actually shows how the DCG-HUE-MDS algorithm could improve the efficiency and CPU time for obtaining the solution. Table 4-2 shows that the DCG-HEU-MDS algorithm generates fewer variables than the DCG-HEU algorithm for the case $d = 75\%$. In addition, Figure 4-5 and Figure 4-6 indicate that the DCG-HEU-MDS algorithm finds the optimal solution faster than DCG-HEU algorithm for the case $d = 75\%$.

4.1.2 Results for Twenty-Target Case (UAVRP1)

For the twenty targets case, we decided to set the run time at 4 hours for each d level and with each algorithm type in order to test our code for the use of more than one

UAV since the results for the ten targets case, at 20 hours run time, showed that we are only using a single UAV.

The results for the second set of problem instances that includes 5 UAVs, 20 targets, 60 waypoints, and 60 threat points for the DCG-HEU algorithm and for the DCG-HEU-MDS algorithm based on the 4-hour run time limit are presented in Table 4-4 and Table 4-5, respectively. The columns include seven levels of d, the total threat level of the best-known solution, the best-known total expected fuel burn, the number of UAVs used, the visited routes, and the number of visited waypoints.

For the 20-targets case and based on the 4-hour run time limit, our goal is to test our code and both the DCG-HEU algorithm and the DCG-HEU-MDS algorithm for the use of more than one UAV.

For the results of the DCG-HEU algorithm in Table 4-4, only in the case with $d = 100\%$ the solution is optimal while in the other d cases the best-known solutions are found within the 4-hour run time limit. In the cases with $d = 95\%$, $d = 90\%$, $d = 80\%$, and $d = 75\%$, as the threat level d decreases, the number of used UAVs increases and the total expected fuel burn increases. The number of visited waypoints was the same for these d cases based on the available 4-hour run time limit. For the case with $d = 70\%$, the algorithm actually could not find a feasible solution based on the available 4-hour run time limit. Since a waypoint is used on the link between targets 11 and 5 across all d levels, we can conclude that it is a high threat level link. As the threat level decreases, the UAV no longer travels over this high threat level link. Instead, the UAV tries to visit a waypoint, in particular 72, in order to comply with the reduction in the threat level. Similarly, for the high threat level links appear between target 17 and target 9 and between target 13 and target 19.

For the results of the DCG-HEU-MDS algorithm in Table 4-5, only in the case with $d = 100\%$ the solution is optimal while in the other d cases the best-known solutions are found within the 4-hour run time limit. A clear pattern between the decrease of the threat level d and the number of used UAVs, the total expected fuel burn, and the number of visited waypoints does not appear. This is due to the limited 4-hour run time. However, a comparison of each individual d level with $d = 100\%$ indicates that the reduction of the threat level increases the number of UAVs used, the total expected fuel burn, and the number of visited waypoints. For the case with $d = 70\%$, the algorithm actually could not find a feasible solution within 4 hours.

Table 4-4 Results for Set 2 for the DCG-HEU Algorithm

Test	D	Total Threat Level	Total Expected Fuel Burn	Number of Used UAVs	Route	Number of Visited Waypoints
d=100%	1.22E-05	1.22E-05	32,118.5	1	11 5 4 1 3 10 17 9 6 14 12 20 13 19 8 18 16 7 2 15	0
d=95%	1.16E-05	4.33E-06	42,638.6	2	1st route: 4 5 Second route: 11 (72) 5 1 3 10 (39) 17 (42) 9 6 12 14 20 13 (52) 19 8 18 16 7 2 15	4
d=90%	1.10E-05	4.33E-06	42,638.6	2	1st route: 4 5 2nd route: 11 (72) 5 1 3 10 (39) 17 (42) 9 6 12 14 20 13 (52) 19 8 18 16 7 2 15	4
d=85%	1.04E-05	No feasible solution is found	–	–	–	–
d=80%	9.78E-06	4.33E-06	42,638.6	2	1st route: 4 5 2nd route: 11 (72) 5 1 3 10 (39) 17 (42) 9 6 12 14 20 13 (52) 19 8 18 16 7 2 15	4
d=75%	9.16E-06	4.41E-06	49,133.0	3	1st route: 4 5 2nd route: 15 7 3rd route: 11 (72) 5 1 3 10 (39) 17 (42) 9 6 12 14 20 13 (52) 19 8 18 16 2 15	4
d=70%	8.55E-06	4.88E-06	33,675.2	1	11 (72) 5 4 1 3 10 17 (42) 9 6 12 14 20 13 (52) 19 8 18 16 7 2 15	3

Table 4-5 Results for Set 2 for the DCG-HEU-MDS Algorithm

Test	D	Total Threat Level	Total Expected Fuel Burn	Number of Used UAVs	Route	Number of Visited Waypoints
d=100 %	1.22E-05	1.22E-05	32,118.5	1	11 5 4 1 3 10 17 9 6 14 12 20 13 19 8 18 16 7 2 15	0
d=95%	1.16E-05	1.05E-05	48,643.6	3	1st route: (69) 20 2nd route: 11 5 1 3 17 (42) 9 6 14 12 13 19 8 18 16 7 2 15 3rd route: 4 1 10 3	2
d=90%	1.10E-05	1.05E-05	48,643.6	3	1st route: (69) 20 2nd route: 11 5 1 3 17 (42) 9 6 14 12 13 19 8 18 16 7 2 15 3rd route: 4 1 10 3	2
d=85%	1.04E-05	5.13E-06	41,189.4	2	1st route: 11 4 5 2nd route: (72) 5 1 3 10 17 (42) 9 6 12 14 20 13 (52) 19 8 18 16 7 2 15	3
d=80%	9.78E-06	2.95E-06	42,711.7	2	1st route: 11 4 5 2nd route: (72) 5 1 3 10 (39) 17 (42) 9 6 12 14 20 13 (52) 19 8 18 16 7 2 15	4
d=75%	9.16E-06	8.40E-06	36,235.9	2	1st route: 11 (78) 2nd route: (72) 5 4 1 3 10 17 9 6 12 14 20 13 19 8 18 16 7 2 15	2
d=70%	8.55E-06	No feasible solution is found	–	–	–	–

Table 4-6 shows the total number of variables that were generated in the column generation step and added to the RMP until finding the best-known solution and until the BCP algorithm run finishes for the second set, 5 UAVs, 20 targets, 60 waypoints, and 60 threat points, for both algorithms, the DCG-HEU and the DCG-HEU-MDS.

Table 4-6 Generated Variables for Set 2

	DCG-HUE		DCG-HUE-MDS	
d	Number of Generated Variables until the Best Solution Found	Number of Generated Variables until Algorithm Finished	Number of Generated Variables until the Best Solution Found	Number of Generated Variables until the Algorithm Finished
100%	83	87	83	87
95%	39	69	80	157
90%	39	69	80	144
85%	–	96	62	119
80%	39	69	61	89
75%	42	114	96	138
70%	60	107	–	160

The DCG-HEU-MDS algorithm uses the minimum dependent set constraints, which cut some fractional solutions and encourage solution integrality. Therefore, we expect the DCG-HEU-MDS algorithm to generate fewer variables than the DCG-HEU algorithm; however, based on the results in Table 4-6, the DCG-HEU algorithm generates fewer variables than the DCG-HEU-MDS algorithm. This is may be due to the 4-hour run time limit and that the obtained solutions for both algorithms are not the same.

Figure 4-6 and Figure 4-7 show the CPU time in seconds until the best solution is found and until the BCP algorithm is finished, respectively, for the set of 5 UAVs, 20 targets, 60 waypoints, and 60 threat points for both algorithms, the DCG-HEU and the DCG-HEU-MDS.

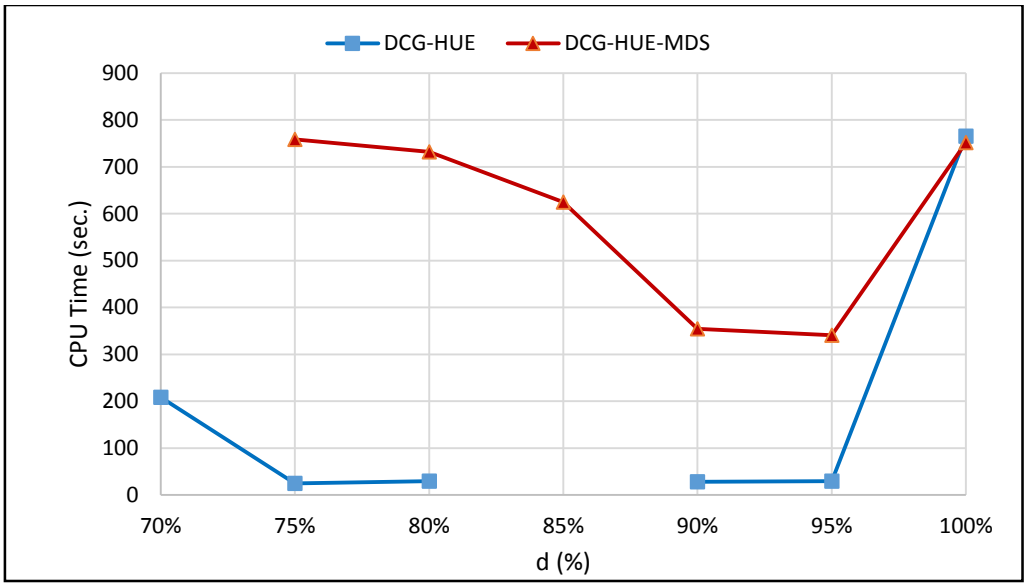


Figure 4-6 The CPU Time until Best Solution Found for Set 2

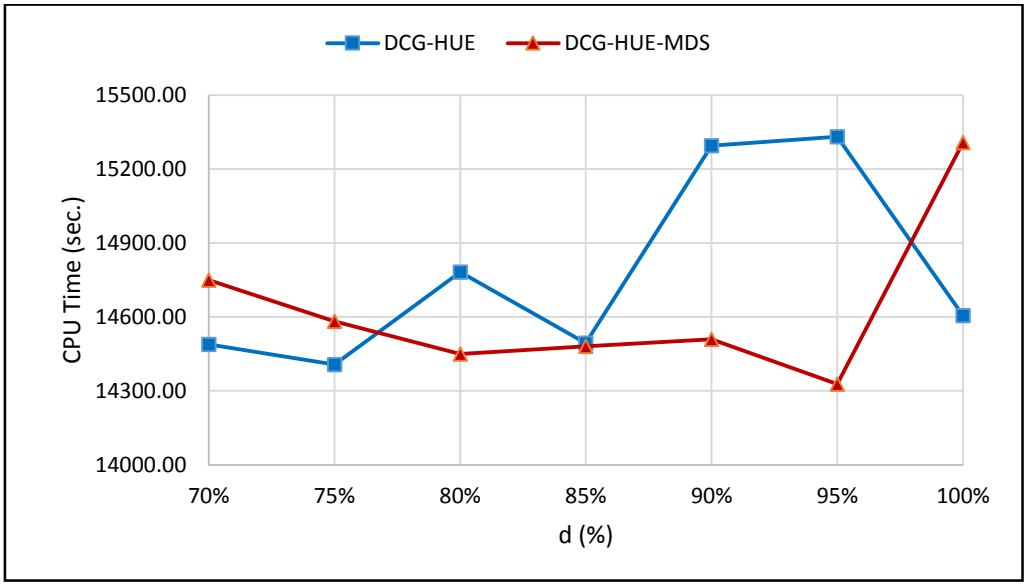


Figure 4-7 The CPU Time until The BCP Algorithm Finished for Set 2

We expect the DCG-HEU-MDS algorithm to find the best solution faster than DCG-HEU algorithm; however, based on the 4-hour run time limit, Figure 4-6 shows that the DCG-HEU finds the best-known solution faster than the DCG-HEU-MDS. This may be due to the difference between the solutions of the two algorithms. In addition, Figure 4-7 indicates that in terms of CPU time until the algorithm is finished, neither of the two algorithms, the DCG-HEU and the DCG-HEU-MDS, perform better than the other.

Table 4-7 shows the total number of MDS cuts that are generated in the cut generation step until the best solution is found and until the BCP algorithm run finishes for the second set, 5 UAVs, 20 targets, 60 waypoints, and 60 threat points, for only the DCG-HEU-MDS algorithm.

Table 4-7 MDS Cuts for Set 2 for DCG-HEU-MDS Algorithm

Test	MDS Cuts until the Best Solution Found	MDS Cuts until the Algorithm Finished
d = 100%	0	0
d = 95%	1	1
d = 90%	1	1
d = 85%	1	1
d = 80%	1	1
d = 75%	1	1
d = 70%	1	1

Notice that in all cases, except $d = 100\%,$ we generated only one MDS cut based on the 4-hour run time limit. This is a very small number of MDS cuts. This may also justify why the DCG-HUE-MDS algorithm did not improve the CPU time and did not generate fewer variables than the DCG-HEU algorithm.

In the following section, we present a computational study for the problem's second formulation (UAVRP2).

4.2 Computational Study for the Second Formulation (UAVRP2)

In this section, we analyze how the total number of visited targets responds to varying the levels of the total allowable threat for all UAVs and varying the levels of the allowable route travel time for each UAV. Intuitively, visiting more targets encourages a longer route. In addition, less total allowable threat level increases the use of more waypoints, hence, encourages longer routes as well. However, less allowable travel time encourages a shorter route.

4.2.1 Results for Ten-Target Case (UAVRP2)

At the beginning, we obtained the route travel time that corresponds to an optimal solution found for the 10-target case in Table 4-1, in $d = 100\%$ with only one UAV. That is 280.31 hours. Then, this value is used to represent the maximum value for the predetermined constant parameter r in order to limit the route travel time for a UAV in our sub problem. This is actually a reasonable maximum limit value because it is obtained for a single UAV, and we limit the route travel time for a single UAV as well. Setting a value lower than this maximum limit for the route travel time will definitely encourage the use of more UAVs. No one UAV can visit all targets alone. In order to visit all the 10 targets, at least two UAVs are needed. As a result, four more evenly spaced levels of r are considered. They are 224.25, 168.18, 112.12, and 56.06 hours.

Similarly, we obtained the total threat level for the optimal solution found for the 10-target case in Table 4-1, in $d = 100\%$ with only one UAV. That is 6.58E-07. We did not use this value to represent the maximum value for the predetermined constant parameter d to limit the total threat level for all UAVs in our master problem similar to what we did with the route travel time limit above. The reason is because this value was obtained for a single UAV, but we limit the total threat level for all UAVs. In addition, setting a value lower or higher only encourages the use of more or fewer waypoints, but does not encourage the

use of more UAVs. Furthermore, visiting all targets and waypoints in the area of operation yielded a threat level value equals 9.67E-04 and visiting only the closest target yielded a threat level value equal to 1.60E-09 based on a set of problem instances that includes 10 targets, 30 threat points, and 55 waypoints, generated using Maximin-Whole Area method. As a result and in order to cover a wide range of threat levels, the 6.58E-07 was only used as a reference level to obtain more levels of d. We decided to experiment with two levels above the reference threat level value by multiplying by 10 and 10^2, respectively, and two levels below the reference threat level value by multiplying by 10^{-1} and 10^{-2}, respectively. Then, in a similar way to establishing a maximum value for the route travel time above, the value 6.58E-05 was used to represent the maximum value for the predetermined constant parameter d to limit the total threat level for all UAVs in our master problem. As a result, the considered four levels of d are 6.58E-06, 6.58E-07, 6.58E-08, and 6.58E-09.

At the beginning, we tested six of the waypoint generation methods, summarized in Table 4-8 and previously described in Chapter 3. Notice that the Threat Reduction-Whole Area method and the Threat Reduction and Maximin-Whole Area method generated waypoints in nearby clusters and not throughout the area of operation. In addition, the Maximin-Rectangle method and Threat Reduction and Maximin-Rectangle method did not give good results. Therefore, only the results of the Threat Reduction-Rectangle method and the Maximin-Whole Area method are discussed.

Table 4-8 Waypoint Generation Methods

Method	Rectangle	Whole Area
Threat Reduction	√	×
Maximin	–	√
Threat Reduction and Maximin	–	×

First, we experimented with the Threat Reduction-Rectangle method and the Maximin-Whole Area method. Since Threat Reduction-Rectangle method is based on placing a rectangle on any two targets including the source node and generating one grid point per a rectangle, we have a total of 55 rectangles, $\binom{11}{2}$. This generates a total of 55 waypoints. For the sake of fair comparison between the two waypoint generation methods, we create 2 sets of problem instances. Both include 10 targets, 30 threat points, and 55 waypoints, shown in Figure 4-8 and Figure 4-9. The 55 waypoints in the first set are generated using the Threat Reduction-Rectangle method, and the 55 waypoints in the second set are generated using the Maximin-Whole Area method.

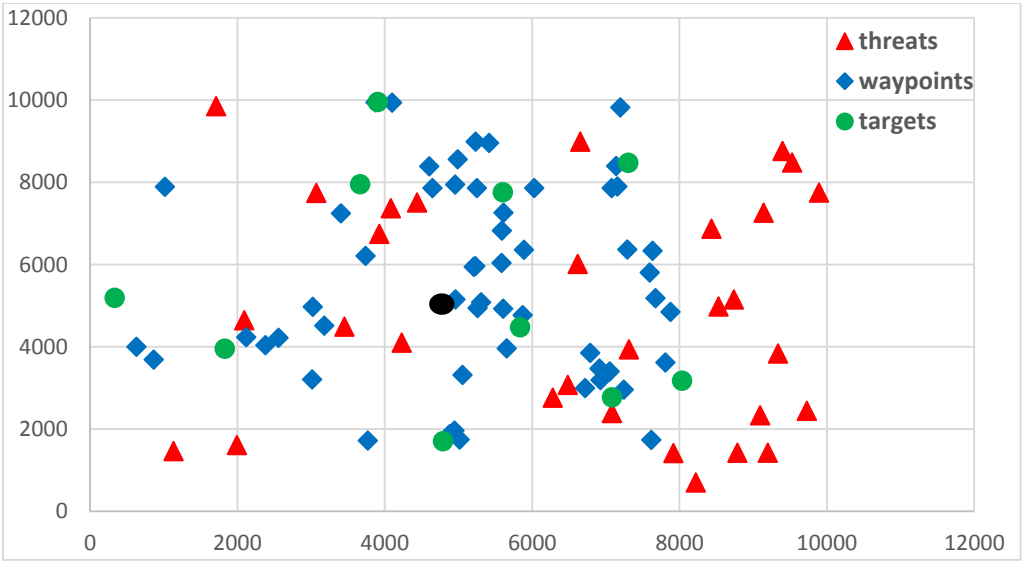


Figure 4-8 10 Targets, 30 Threats, and 55 Waypoints (Threat Reduction-Rectangle)

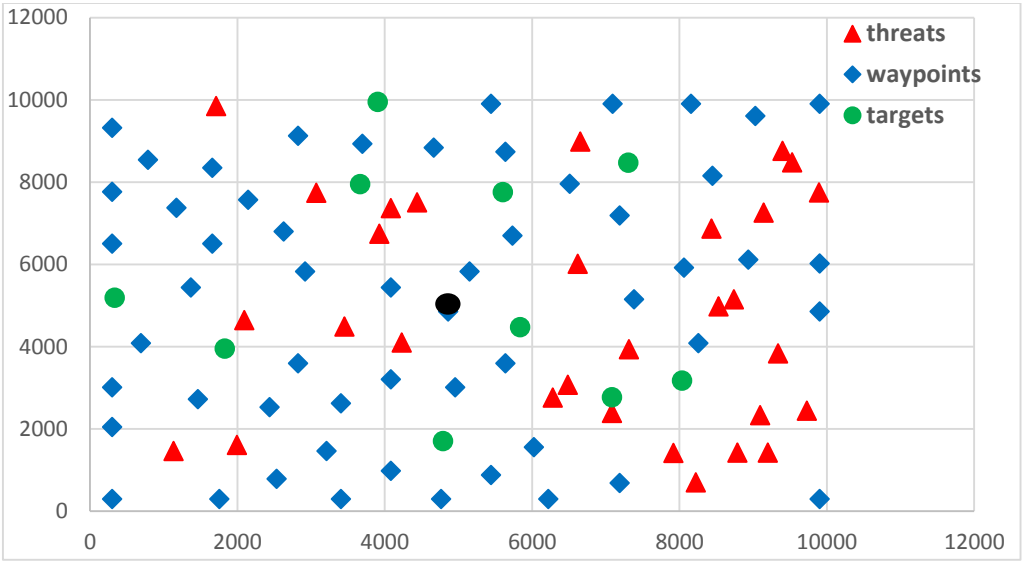


Figure 4-9 10 Targets, 30 Threat Points, and 55 Waypoints (Maximin-Whole Area)

We considered only the DCG-HEU algorithm and the DCG-HEU-MDS algorithm to solve each set of the problem instances. The run time was set at 4 hours for each set of problem instances with each route travel time level r, each threat level d, and each algorithm type. Our goal now is to see how the total number of visited targets for multiple UAVs respond to varying the levels of r and d within the available 4-hour run time for the two waypoint generation methods.

The results for the two waypoint generation methods for the DCG-HEU algorithm are presented in Table 4-9 and Table 4-10, respectively. Each table includes four varying levels for the constant parameter r, which represents the limit for the route travel time, and four varying levels for the constant parameter d, which represents the limit for the total threat level for all UAVs. This is actually a total of 16 runs for each table. For each run, the solution we obtained includes the maximum total number of visited targets, our objective function, the number of UAVs used, the time, in seconds, until best-known solution found, the time, in seconds, until the algorithm finished, the number of generated variables until

best-known solution is found, and the number of generated variables until the algorithm finished. For example, for the run at $r_4 = 224.25$ and $d_4 = 6.58E-06$ in Table 4-10, the maximum total number of visited targets is 10, the number of UAV used is 5, the time until best-known solution found was 15.41 seconds, the time until the algorithm finished was 16.56 seconds, the number of generated variables until best-known solution found was 17, and finally, the number of generated variables until the algorithm finished was 18.

Table 4-9 10 Results for DCG-HEU Algorithm for 10 Targets, 30 Threat Points, and 55 Waypoints (Threat Reduction-Rectangle)

Total Number of Visited Targets							
		r, Route Travel Time (Sub problem)					
		MIN	1	2	3	4	MAX
d, Total Threat level (Master)	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	0	1	0	
	2	6.58E-08	1	0	4	0	
	3	6.58E-07	1	3	4	6	
	4	6.58E-06	1	3	4	6	
	MAX	6.58E-05					
Number of UAVs Used							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	0	1	0	
	2	6.58E-08	1	0	3	0	
	3	6.58E-07	1	2	2	2	
	4	6.58E-06	1	2	2	2	
	MAX	6.58E-05					
Time until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	4.55	0.48	1.80	0.44	
	2	6.58E-08	4.57	0.51	8.84	0.44	
	3	6.58E-07	4.79	6.06	5.38	10.55	
	4	6.58E-06	4.89	3.89	4.51	3.07	
	MAX	6.58E-05					

Table 4-9 Continued

Time until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	2	6.58E-08	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	3	6.58E-07	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	4	6.58E-06	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	MAX	6.58E-05					
Number of Generated Variables until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	1	3	1	
	2	6.58E-08	2	1	9	1	
	3	6.58E-07	2	5	6	9	
	4	6.58E-06	2	5	6	5	
	MAX	6.58E-05					
Number of Generated Variables until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	15	18	16	
	2	6.58E-08	2	21	9	30	
	3	6.58E-07	2	8	6	11	
	4	6.58E-06	2	8	6	22	
	MAX	6.58E-05					

Table 4-10 Results for DCG-HEU Algorithm for 10 Targets, 30 Threat Points, and 55 Waypoints (Maximin-Whole Area)

Total Number of Visited Targets							
		r, Route Travel Time (Sub problem)					
		MIN	1	2	3	4	MAX
d, Total Threat Level (master)	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	0	0	1	
	2	6.58E-08	1	0	0	1	
	3	6.58E-07	1	2	6	10	
	4	6.58E-06	1	10	10	10	
	MAX	6.58E-05					
Number of UAVs Used							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	0	0	1	
	2	6.58E-08	1	0	0	1	
	3	6.58E-07	1	1	3	5	
	4	6.58E-06	1	5	5	5	
	MAX	6.58E-05					
Time until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2.86	0.45	0.61	2.87	
	2	6.58E-08	3.31	0.49	0.56	1.58	
	3	6.58E-07	2.74	4.35	43.89	423.04	
	4	6.58E-06	2.25	1878.81	12047.39	15.41	
	MAX	6.58E-05					
Time until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2236.42	Time Ran Out	Time Ran Out	Time Ran Out	
	2	6.58E-08	2289.94	Time Ran Out	Time Ran Out	Time Ran Out	
	3	6.58E-07	2273.88	Time Ran Out	TimeRan Out	428.25	
	4	6.58E-06	2212.12	Time Ran Out	12064.45	16.56	
	MAX	6.58E-05					

Table 4-10 Continued

Number of Generated Variables until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	1	1	3	
	2	6.58E-08	2	1	1	2	
	3	6.58E-07	2	3	8	28	
	4	6.58E-06	2	28	15	17	
	MAX	6.58E-05					
Number of Generated Variables until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	9	7	11	
	2	6.58E-08	2	28	28	24	
	3	6.58E-07	2	28	9	29	
	4	6.58E-06	2	30	17	18	
	MAX	6.58E-05					

For DCG-HEU and for both waypoint generation methods, the Threat Reduction-Rectangle and the Maximin-Whole Area, the maximum total number of visited targets varies in response to decreasing or increasing the route travel time r, the total threat level d, or both. Generally, as both the route travel time r and the total threat level d decrease, the maximum total number of visited targets also decreases. Similarly, as both the route travel time r and the total threat level d increase, the maximum total number of visited targets also increases.

For the DCG_HEU algorithm, the waypoint generation using the Maximin-Whole Area method performs better than the Threat Reduction-Rectangle method. Overall, a total of 54 targets were visited for all of the 16 runs using the Maximin-Whole Area compared to a total of 35 targets for the waypoint generation using the Threat Reduction-Rectangle method. In terms of achieving the mission, with the Maximin-Whole Area, all the available 10 targets are visited in 4 runs, which are at $(r = 224.25,\ d = 6.58E-06)$, $(r = 168.18,$

$d = 6.58E-06)$, $(r = 112.12,\ d = 6.58E-06)$, and $(r = 224.25,\ d = 6.58E-07)$, while, none of the runs with the Threat Reduction-Rectangle method were able to visit all available targets. Notice that with the Maximin-Whole Area, the algorithm terminated before completing the 4-hour run time limit during 7 runs, while, none of the runs with the Threat Reduction-Rectangle were able to finish before reaching the 4-hour run time limit. Since the number of visited targets are more with the waypoint generation using the Maximin-Whole Area than with the Threat Reduction-Rectangle method, the results also show that the number of UAVs used and the number of generated variables, until the best-known solution found and until the algorithm finished, are more, overall, when using the Maximin-Whole Area than when using the Threat Reduction-Rectangle method.

The results for the two-waypoint generation methods for the DCG-HEU-MDS algorithm are presented in Table 4-11 and Table 4-12, respectively.

Table 4-11 Results for DCG-HEU-MDS Algorithm for 10 Targets, 30 Threat Points, and 55 Waypoints (Threat Reduction-Rectangle)

Number of Visited Targets							
		r, Route Travel Time (Sub problem)					
		MIN	1	2	3	4	MAX
d, Total Threat Level (Master)	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	0	1	0	
	2	6.58E-08	1	0	4	0	
	3	6.58E-07	1	3	4	4	
	4	6.58E-06	1	3	4	9	
	MAX	6.58E-05					
Number of UAVs Used							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	0	1	0	
	2	6.58E-08	1	0	3	0	
	3	6.58E-07	1	2	2	2	
	4	6.58E-06	1	2	2	3	
	MAX	6.58E-05					

Table 4-11 Continued

Time until the Best Known Solution Found

		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	4.52	0.48	1.88	0.44	
	2	6.58E-08	4.54	0.47	6.95	0.53	
	3	6.58E-07	4.66	5.93	6.22	5.01	
	4	6.58E-06	4.56	5.55	3.77	127.42	
	MAX	6.58E-05					

Time until the Algorithm Terminated

		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	2	6.58E-08	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	3	6.58E-07	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	4	6.58E-06	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	MAX	6.58E-05					

Number of Generated Variables until the Best Known Solution Found

		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	1	3	1	
	2	6.58E-08	2	1	9	1	
	3	6.58E-07	2	5	6	7	
	4	6.58E-06	2	5	6	12	
	MAX	6.58E-05					

Number of Generated Variables until the Algorithm Terminated

		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	15	17	15	
	2	6.58E-08	2	20	9	30	
	3	6.58E-07	2	7	6	11	
	4	6.58E-06	2	7	6	13	
	MAX	6.58E-05					

Table 4-11 Continued

Number of MDS Cuts until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	0	0	0	0	
	2	6.58E-08	0	0	0	0	
	3	6.58E-07	0	0	0	1	
	4	6.58E-06	0	0	0	2	
	MAX	6.58E-05					

Number of MDS Cuts until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	0	0	0	0	
	2	6.58E-08	0	0	0	0	
	3	6.58E-07	0	0	0	1	
	4	6.58E-06	0	0	0	2	
	MAX	6.58E-05					

Table 4-12 Results for DCG-HEU-MDS Algorithm for 10 Targets, 30 Threat Points, and 55 Waypoints (Maximin-Whole Area)

Total Number of Visited Targets							
		r, Route Travel Time (Sub problem)					
		MIN	1	2	3	4	MAX
d, Total Threat Level (Master)	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	0	0	1	
	2	6.58E-08	1	0	4	1	
	3	6.58E-07	1	2	10	10	
	4	6.58E-06	1	10	8	10	
	MAX	6.58E-05					

Number of UAVs Used							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	0	0	1	
	2	6.58E-08	1	0	2	1	
	3	6.58E-07	1	1	5	5	
	4	6.58E-06	1	5	3	5	
	MAX	6.58E-05					

Table 4-12 Continued

Time until the Best Known Solution Found

		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2.74	0.45	0.53	2.46	
	2	6.58E-08	2.41	0.56	12.37	1.59	
	3	6.58E-07	2.62	4.07	266.59	426.43	
	4	6.58E-06	2.40	1857.62	48.18	15.61	
	MAX	6.58E-05					

Time until the Algorithm Terminated

		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2237.07	Time Ran Out	Time Ran Out	Time Ran Out	
	2	6.58E-08	2294.64	Time Ran Out	Time Ran Out	Time Ran Out	
	3	6.58E-07	2266.33	Time Ran Out	268.354	431.07	
	4	6.58E-06	2228.67	Time Ran Out	Time Ran Out	16.80	
	MAX	6.58E-05					

Number of Generated Variables until Best Known Solution Found

		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	1	1	3	
	2	6.58E-08	2	1	6	2	
	3	6.58E-07	2	3	25	28	
	4	6.58E-06	2	28	17	17	
	MAX	6.58E-05					

Number of Generated Variables until the Algorithm Terminated

		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	9	8	12	
	2	6.58E-08	2	27	35	23	
	3	6.58E-07	2	28	25	29	
	4	6.58E-06	2	30	20	18	
	MAX	6.58E-05					

Table 4-12 Continued

Number of MDS Cuts until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	0	0	0	0	
	2	6.58E-08	0	0	1	0	
	3	6.58E-07	0	0	16	0	
	4	6.58E-06	0	0	7	0	
	MAX	6.58E-05					
Number of MDS Cuts until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	0	0	1	4	
	2	6.58E-08	0	0	2	0	
	3	6.58E-07	0	0	16	0	
	4	6.58E-06	0	0	7	0	
	MAX	6.58E-05					

For the DCG-HEU-MDS algorithm and for both waypoint generation methods, the waypoint generation using the Maximin-Whole Area obviously performed better than the Threat Reduction-Rectangle method. Overall, a total of 60 targets were visited for all of the 16 runs using the Maximin-Whole Area compared to a total of 36 targets for the waypoint generation using the Threat Reduction-Rectangle method. In terms of achieving the mission, with the Maximin-Whole Area, all the available 10 targets are visited in 4 runs, while none of the runs with the Threat Reduction-Rectangle method were able to visit all available targets. With the Maximin-Whole Area, the algorithm terminated before completing the 4-hour run time limit in 7 runs, while, none of the runs with the Threat Reduction-Rectangle method were able to finish before reaching the 4-hour run time limit. Since the number of visited targets is more with the waypoint generation using the Maximin-Whole Area than with the Threat Reduction-Rectangle method, the results also show that the number of UAVs used, the number of generated variables, until the best-known solution found and until the algorithm finished, the number of MDS cuts, until

the best known solution found and until the algorithm finished, are more, overall, when using the Maximin-Whole Area than when using the Threat Reduction-Rectangle method.

Based on the previous discussion, we can conclude that the waypoint generation using the Maximin-Whole Area method gives better results than the waypoint generation using the Threat Reduction-Rectangle method for both DCG-HEU and DCG-HEU-MDS algorithms. As a result, we decided to ignore the rectangle method and just focus on improving the results that were obtained using the Maximin-Whole Area.

The problem we optimize is a computationally hard one. We generated 55 waypoints, using the Maximin-Whole Area, and added them all together *a priori* to an area of operation that already contained 10 existing targets. The waypoints increase the size of the problem even though they are not required to be visited. Based on our implementation, a set of problem instances that includes 10 targets and 55 waypoints is equivalent to solving a problem with 65 nodes. If we consider a smaller number of waypoints, the computational complexity of the problem will exponentially decrease as the total number of targets and waypoints decreases. The time for optimizing the sub problem, detecting cycles in the solution, and generating new routes will be reduced. As a result, we expect to obtain even better results with fewer waypoints than the 55 waypoints that we are currently using.

In order to improve the results for both the DCG-HEU and the DCG-HEU-MDS algorithms that were previously obtained for the 55 waypoints generated using the Maximin-Whole Area, we decided to use the method Maximin-Whole Area-Removing method in order to remove unnecessary waypoints, which are unlikely to be visited by a UAV. Initially, the process removes any waypoint that does not lie in any one of the 55 rectangles. Consequently, the number of waypoints in the area of operation decreased from 55 waypoints to only 29 waypoints. Now, the area of operation, shown in Figure 4-10,

includes 5 UAVs located at the base, 10 targets, 30 threat points, and the remaining 29 waypoints.

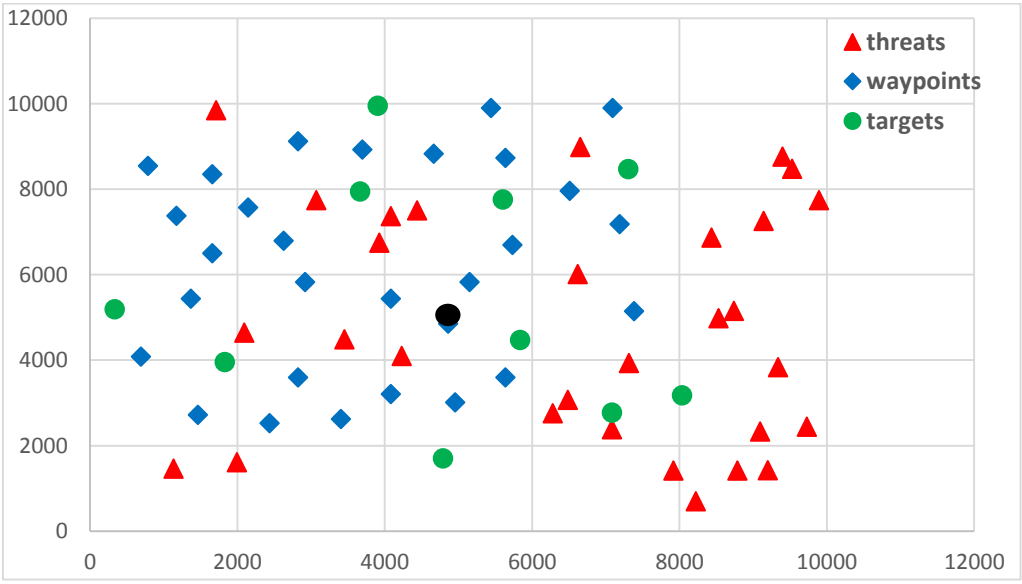


Figure 4-10 10 Targets, 30 Threat points, and 29 Waypoints (Maximin-Whole Area-Removing)

We tested the retained 29 waypoints in the area of operation for both DCG-HEU and DCG-HEU-MDS, and the results are presented in Table 4-13 and Table 4-14, respectively.

Table 4-13 Results for DCG-HEU Algorithm for 10 Targets, 30 Threat Points, and 29 Waypoints (Maximin-Whole Area-Removing)

Total Number of Visited Targets							
		r, Route Travel Time (Sub problem)					
		MIN	1	2	3	4	MAX
d, Total Threat Level (master)	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	0	0	0	
	2	6.58E-08	1	0	5	3	
	3	6.58E-07	1	2	10	10	
	4	6.58E-06	1	10	10	10	
	MAX	6.58E-05					

Number of UAVs Used							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	0	0	0	
	2	6.58E-08	1	0	3	2	
	3	6.58E-07	1	1	5	5	
	4	6.58E-06	1	5	5	5	
	MAX	6.58E-05					

Time until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2.23	0.13	0.14	0.14	
	2	6.58E-08	2.22	0.13	29.40	4.77	
	3	6.58E-07	2.24	2.82	3451.42	10.53	
	4	6.58E-06	2.23	161.84	36.77	2.71	
	MAX	6.58E-05					

Time until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	609.31	Time Ran Out	Time Ran Out	Time Ran Out	
	2	6.58E-08	595.88	Time Ran Out	Time Ran Out	Time Ran Out	
	3	6.58E-07	606.24	Time Ran Out	3451.927	12.09	
	4	6.58E-06	593.76	193.716	37.405	4.88	
	MAX	6.58E-05					

Table 4-13 Continued

Number of Generated Variables until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	1	1	1	
	2	6.58E-08	2	1	9	7	
	3	6.58E-07	2	3	18	23	
	4	6.58E-06	2	20	16	18	
	MAX	6.58E-05					
Number of Generated Variables until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	10	9	14	
	2	6.58E-08	2	18	17	46	
	3	6.58E-07	2	20	18	26	
	4	6.58E-06	2	23	17	24	
	MAX	6.58E-05					

Table 4-14 Results for DCG-HEU-MDS Algorithm for 10 Targets, 30 Threat Points, and 29 Waypoints (Maximin-Whole Area-Removing)

Total Number of Visited Targets							
		r, Route Travel Time (Sub problem)					
		MIN	1	2	3	4	MAX
d, Total Threat Level (Master)	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	0	0	0	
	2	6.58E-08	1	0	5	3	
	3	6.58E-07	1	2	10	9	
	4	6.58E-06	1	10	10	9	
	MAX	6.58E-05					
Number of UAVs Used							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	0	0	0	
	2	6.58E-08	1	0	3	2	
	3	6.58E-07	1	1	5	3	
	4	6.58E-06	1	5	5	2	
	MAX	6.58E-05					

Table 4-14 Continued

Time until the Best Known Solution Found

		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2.53	0.13	0.13	0.13	
	2	6.58E-08	2.22	0.13	25.67	4.78	
	3	6.58E-07	2.24	1.31	3145.42	6391.81	
	4	6.58E-06	2.23	149.10	37.19	1131.98	
	MAX	6.58E-05					

Time until the Algorithm Terminated

		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	634.92	Time Ran Out	Time Ran Out	Time Ran Out	
	2	6.58E-08	613.70	Time Ran Out	Time Ran Out	Time Ran Out	
	3	6.58E-07	611.41	Time Ran Out	3145.915	Time Ran Out	
	4	6.58E-06	617.76	181.519	37.844	Time Ran Out	
	MAX	6.58E-05					

Number of Generated Variables until the Best Known Solution Found

		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	1	1	1	
	2	6.58E-08	2	1	9	7	
	3	6.58E-07	2	3	20	19	
	4	6.58E-06	2	20	16	15	
	MAX	6.58E-05					

Number of Generated Variables until the Algorithm Terminated

		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	10	9	14	
	2	6.58E-08	2	18	17	42	
	3	6.58E-07	2	20	20	19	
	4	6.58E-06	2	23	17	17	
	MAX	6.58E-05					

Table 4-14 Continued

Number of MDS Cuts until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	0	0	0	0	
	2	6.58E-08	0	0	0	0	
	3	6.58E-07	0	0	4	4	
	4	6.58E-06	0	0	0	6	
	MAX	6.58E-05					
Number of MDS Cuts until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	0	0	0	0	
	2	6.58E-08	0	0	0	2	
	3	6.58E-07	0	0	4	4	
	4	6.58E-06	0	0	0	6	
	MAX	6.58E-05					

As expected, when comparing the DCG-HEU algorithm for the previous 55 waypoints generated using Maximin-Whole Area method and the remaining 29 waypoints generated using the Maximin-Whole Area-Removing method, the algorithm performs better with the 29 waypoints than with the 55 waypoints. Overall, a total of 64 targets are visited for all of the 16 runs using the 29 waypoints compared to a total of 54 targets for the 55 waypoints. In terms of achieving the mission, with the 29 waypoints, all of the 10 available targets are visited in 5 runs compared to 4 runs with the 55 waypoints. Notice that, with the 29 waypoints, the algorithm terminated before completing the 4-hour run time limit in 9 runs compared to 7 runs that were able to finish before reaching the 4-hour run time limit with the 55 waypoints. Since the number of visited targets is more with the 29 waypoints than with 55 waypoints, the results also show that the number of UAVs used and the number of generated variables, until the best-known solution is found and until the algorithm is finished, are more, overall, when using the 29 waypoints than when using the 55 waypoints.

When comparing the DCG-HEU-MDS algorithm for the previous 55 waypoints generated using the Maximin-Whole Area method and the remaining 29 waypoints generated using the Maximin-Whole Area-Removing method, the algorithm performs slightly better with the 29 waypoints than with the 55 waypoints. Overall, a total of 62 targets are visited for all of the 16 runs using the 29 waypoints compared to a total of 60 targets for the 55 waypoints. However, with the 29 waypoints, all the available 10 targets are visited in 3 runs compared to 5 runs with the 55 waypoints. The algorithm terminated before completing the 4-hour run time limit in 7 runs with both 29 and 55 waypoints. Although the number of visited targets are more with the 29 waypoints than with 55 waypoints, the results show that the number of UAVs used, the number of generated variables, until the best-known solution is found and until the algorithm is finished, and the number of MDS cuts, until the best known solution is found and until the algorithm is finished, are more, overall, when using the 55 waypoints than when using the 29 waypoints.

Comparing both algorithms, DCG-HEU and DCG-HEU-MDS, for only the 29 waypoints generated using the Maximin-Whole Area-Removing method, the DCG-HEU algorithm performs slightly better with the 29 waypoints than the DCG-HEU-MDS algorithm does with the same number of waypoints. Overall, a total of 64 targets are visited for all of the 16 runs using the DCG-HEU algorithm compared to a total of 62 targets using the DCG-HEU-MDS algorithm. In terms of achieving the mission, with the DCG-HEU algorithm, all the available 10 targets are visited in 5 runs compared to 3 runs with the DCG-HEU-MDS algorithm. The DCG-HEU algorithm terminates before completing the 4-hour run time limit in 9 runs compared to 7 runs that are able to finish before reaching the 4-hour run time limit with the DCG-HEU-MDS algorithm. Since the number of visited targets is more with the DCG-HUE algorithm than the DCG-HUE-MDS algorithm, the

results also show that the number of UAVs used is more, overall, when using the DCG-HUE algorithm than when using DCG-HUE-MDS algorithm.

The DCG-HEU-MDS algorithm uses the Minimum Dependent Set constraints, which cut some fractional solutions and encourage the solution integrality, while, the DCG-HUE algorithm does not. We expect the DCG-HEU-MDS algorithm to generate fewer variables than the DCG-HEU algorithm does for the same solution. Nevertheless, the overall number of generated variables until the best-known solution is found and until the algorithm is finished using the DCG-HEU-MDS algorithm is fewer than those generated using the DCG-HEU algorithm.

In the quest to obtain even better results, we decided to use the method Maximin-Whole Area-Removing-Total Threat Reduction in order to calculate the total threat reduction for each waypoint of the 29 waypoints based on the number of rectangles that a waypoint lies inside. This is done by placing a rectangle between any two targets in the area of operation including the source node. We have 10 targets plus one source node. That is a total of 55 rectangles, $\binom{11}{2}$. Then, the process takes each of the 29 waypoints and determines the number of rectangles that it lies inside. After that, the total threat reduction for a waypoint is calculated. Sorting the 29 waypoints based on their total threat reduction from highest to lowest, we decided to test our two algorithms, the DCG-HEU algorithm and the DCG-HEU-MDS algorithm, for the best 15 waypoints and best 8 waypoints, shown in Figure 4-11 and Figure 4-12, respectively, in terms of highest total threat reduction.

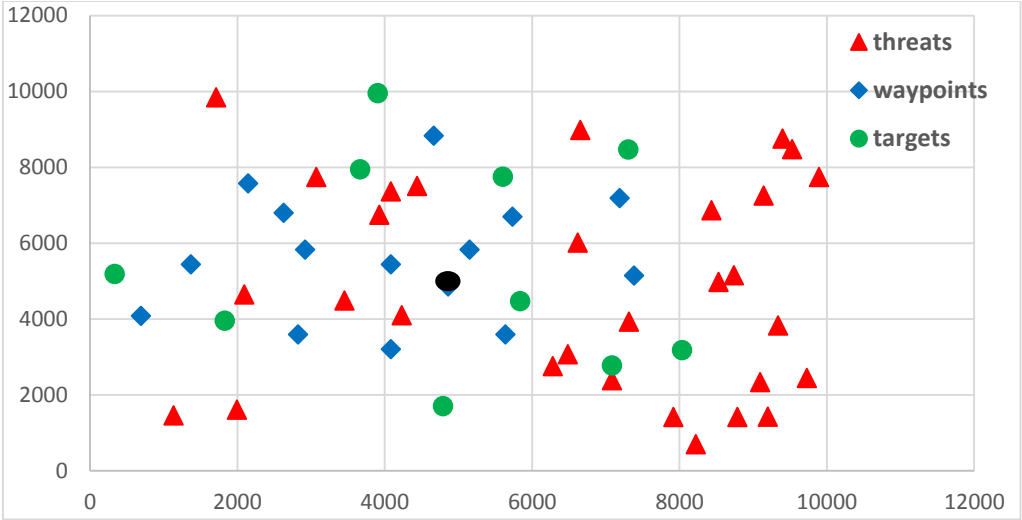


Figure 4-11 10 Targets, 30 Threat Points, and Best 15 Waypoints (Maximin-Whole Area-Removing-Total Threat Reduction)

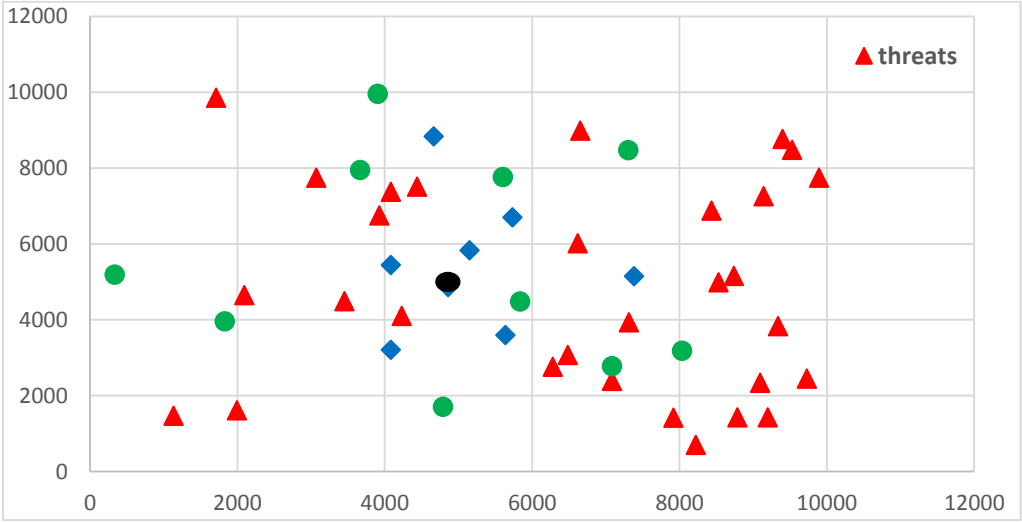


Figure 4-12 10 Targets, 30 Threat Points, and Best 8 Waypoints (Maximin-Whole Area-Removing-Total Threat Reduction)

Starting with the best 15 waypoints case, the results for both algorithms, the DCG-HEU algorithm and the DCG-HEU-MDS algorithm, are presented in Table 4-15 and Table 4-16, respectively.

Table 4-15 Results for DCG-HEU Algorithm for 10 Targets, 30 Threat Points, and Best 15 Waypoints (Maximin-Whole Area-Removing-Total Threat Reduction)

Total Number of Visited Targets							
		r, Route Travel Time (Sub problem)					
		MIN	1	2	3	4	MAX
d, Total Threat Level (master)	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	2	1	1	
	2	6.58E-08	1	6	8	8	
	3	6.58E-07	1	10	10	10	
	4	6.58E-06	1	10	10	10	
	MAX	6.58E-05					

Number of UAVs Used							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	1	1	1	
	2	6.58E-08	1	5	2	2	
	3	6.58E-07	1	5	4	4	
	4	6.58E-06	1	5	4	4	
	MAX	6.58E-05					

Time until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	6.35	4646.17	1759.91	2806.58	
	2	6.58E-08	7.53	12373.96	8132.09	5166.00	
	3	6.58E-07	5.44	1231.85	508.84	15.49	
	4	6.58E-06	6.45	52.40	42.32	11.49	
	MAX	6.58E-05					

Time until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	71.13	4871.769	Time Ran Out	Time Ran Out	
	2	6.58E-08	78.83	Time Ran Out	Time Ran Out	Time Ran Out	
	3	6.58E-07	72.38	1442.666	509.026	28.96	
	4	6.58E-06	70.70	52.872	42.567	60.36	
	MAX	6.58E-05					

Table 4-15 Continued

Number of Generated Variables until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	168	206	489	
	2	6.58E-08	2	514	64	159	
	3	6.58E-07	2	41	22	16	
	4	6.58E-06	2	26	18	17	
	MAX	6.58E-05					
Number of Generated Variables until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	201	4614	3531	
	2	6.58E-08	2	875	738	1939	
	3	6.58E-07	2	47	22	18	
	4	6.58E-06	2	26	19	20	
	MAX	6.58E-05					

Table 4-16 Results for DCG-HEU-MDS Algorithm for 10 Targets, 30 Threat Points, and Best 15 Waypoints (Maximin-Whole Area-Removing-Total Threat Reduction)

Total Number of Visited Targets							
		r, Route Travel Time (Sub problem)					
		MIN	1	2	3	4	MAX
d, Total Threat Level (Master)	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	0	1	1	
	2	6.58E-08	1	6	8	2	
	3	6.58E-07	1	10	10	10	
	4	6.58E-06	1	10	10	10	
	MAX	6.58E-05					
Number of UAVs Used							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	0	1	1	
	2	6.58E-08	1	4	2	1	
	3	6.58E-07	1	5	3	3	
	4	6.58E-06	1	5	3	3	
	MAX	6.58E-05					

Table 4-16 Continued

Time until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	4.53	0.05	966.42	751.46	
	2	6.58E-08	5.93	10908.87	7569.49	0.25	
	3	6.58E-07	5.74	1240.46	22.21	386.67	
	4	6.58E-06	5.18	46.52	135.98	2097.39	
	MAX	6.58E-05					
Time until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	66.89	Time Ran Out	Time Ran Out	Time Ran Out	
	2	6.58E-08	64.05	Time Ran Out	Time Ran Out	Time Ran Out	
	3	6.58E-07	69.09	1461.346	199.271	386.998	
	4	6.58E-06	77.56	46.986	573.216	2203.899	
	MAX	6.58E-05					
Number of Generated Variables until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	1	41	103	
	2	6.58E-08	2	696	68	3	
	3	6.58E-07	2	41	17	19	
	4	6.58E-06	2	26	14	18	
	MAX	6.58E-05					
Number of Generated Variables until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	1271	2554	3739	
	2	6.58E-08	2	1080	1470	153	
	3	6.58E-07	2	47	22	19	
	4	6.58E-06	2	26	20	22	
	MAX	6.58E-05					

Table 4-16 Continued

Number of MDS Cuts until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	0	0	1	1	
	2	6.58E-08	0	187	4	0	
	3	6.58E-07	0	0	4	0	
	4	6.58E-06	0	0	6	4	
	MAX	6.58E-05					
Number of MDS Cuts until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	0	1	249	206	
	2	6.58E-08	0	206	13	0	
	3	6.58E-07	0	0	4	0	
	4	6.58E-06	0	0	6	4	
	MAX	6.58E-05					

When comparing the DCG-HEU algorithm for the previous 29 waypoints generated using the Maximin-Whole Area-Removing method and the best 15 waypoints generated using the Maximin-Whole Area-Removing-Total Threat Reduction method, the algorithm clearly performs better with the best 15 waypoints than with the previous 29 waypoints. Overall, a total of 90 targets are visited for all of the 16 runs using the 15 waypoints compared to a total of 64 targets for the 29 waypoints. In terms of achieving the mission, with the 15 waypoints, all the available 10 targets are visited in 6 runs compared to 5 runs with the 29 waypoints. With the 15 waypoints, the algorithm terminated before completing the 4-hour run time limit in 11 runs compared to 9 runs that were able to finish before reaching the 4-hour run time limit with the 29 waypoints. Since the number of visited targets are greater with the 15 waypoints than with the 29 waypoints, the results show that the number of UAVs used, and the number of generated variables until the best-known solution is found and until the algorithm is finished, are also more, overall, when using the 15 waypoints than the 29 waypoints.

When comparing the DCG-HEU-MDS algorithm for the previous 29 waypoints generated using the Maximin-Whole Area-Removing method and the 15 waypoints generated using the Maximin-Whole Area-Removing-Total Threat Reduction method, the algorithm performs better with the 15 waypoints than with the 29 waypoints. Overall, a total of 82 targets are visited for all of the 16 runs using the 15 waypoints compared to a total of 62 targets for the 29 waypoints. In terms of achieving the mission, with the 15 waypoints, all the 10 targets are visited in 6 runs compared to 3 runs with the 29 waypoints. The algorithm terminated before completing the 4-hour run time limit in 10 runs with the 15 waypoints compared to 7 runs with the 29 waypoints. Since the number of visited targets are more with the 15 waypoints than with the 29 waypoints, the results show that the number of UAVs used, the number of generated variables, until the best-known solution is found and until the algorithm is finished, and the number of MDS cuts, until the best known solution is found and until the algorithm is finished, are also more, overall, when using the 15 waypoints than the 29 waypoints.

Comparing both algorithms, the DCG-HEU and the DCG-HEU-MDS, for only the 15 waypoints generated using the Maximin-Whole Area-Removing-Total Threat Reduction method, the DCG-HEU algorithm performs better with the 15 waypoints than the DCG-HEU-MDS algorithm does with the same number of waypoints. Overall, a total of 90 targets are visited for all of the 16 runs using the DCG-HEU algorithm compared to a total of 82 targets using the DCG-HEU-MDS algorithm. In both algorithms, all the available 10 targets are visited in 6 runs. The DCG-HEU algorithm terminates before completing the 4-hour run time limit in 11 runs compared to 10 runs that are able to finish before reaching the 4-hour run time limit with the DCG-HEU-MDS algorithm. Since the number of visited targets is more with the DCG-HUE algorithm than with the DCG-HUE-MDS algorithm, the

results show that the number of UAVs used are also more, overall, when using the DCG-HUE algorithm than when using DCG-HUE-MDS algorithm.

The DCG-HEU-MDS algorithm uses the Minimum Dependent Set constraints, which cut some fractional solutions and encourage the solution integrality, while, the DCG-HUE algorithm does not. We expect the DCG-HEU-MDS algorithm to generate fewer variables than the DCG-HEU algorithm for same solution. Nevertheless, by looking at the tables, we can see that the number of generated variables, until the best-known solution is found and until the algorithm is finished, using the DCG-HEU-MDS algorithm are fewer than those generated using DCG-HEU algorithm.

Finally, for the 10-targets case, we decided to test our two algorithms, the DCG-HEU algorithm and the DCG-HEU-MDS algorithm, for the best 8 waypoints in terms of highest total threat reduction. The results for both algorithms for the 8 waypoints are presented in Table 4-17 and Table 4-18, respectively.

Table 4-17 Results for DCG-HEU Algorithm for 10 Targets, 30 Threat Points, and best 8 Waypoints (Maximin-Whole Area-Removing-Total Threat Reduction)

Total Number of Visited Targets							
		r, Route Travel Time (Sub problem)					
		MIN	1	2	3	4	MAX
d, Total Threat Level (master)	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	2	1	1	
	2	6.58E-08	1	6	8	8	
	3	6.58E-07	1	10	10	10	
	4	6.58E-06	1	10	10	10	
	MAX	6.58E-05					

Number of UAVs Used							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	1	1	1	
	2	6.58E-08	1	4	3	3	
	3	6.58E-07	1	5	4	3	
	4	6.58E-06	1	5	4	4	
	MAX	6.58E-05					

Time until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	0.36	665.36	139.26	205.02	
	2	6.58E-08	0.30	1606.82	27.71	14.89	
	3	6.58E-07	0.36	1063.56	15.84	17.38	
	4	6.58E-06	0.45	93.57	5.01	1.36	
	MAX	6.58E-05					

Time until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	7.08	940.791	Time Ran Out	Time Ran Out	
	2	6.58E-08	6.57	Time Ran Out	Time Ran Out	Time Ran Out	
	3	6.58E-07	7.91	1160.835	15.948	17.57	
	4	6.58E-06	7.63	94.432	5.229	1.55	
	MAX	6.58E-05					

Table 4-17 Continued

Number of Generated Variables until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	135	63	66	
	2	6.58E-08	2	318	22	15	
	3	6.58E-07	2	56	20	19	
	4	6.58E-06	2	18	13	16	
	MAX	6.58E-05					
Number of Generated Variables until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	206	4019	4767	
	2	6.58E-08	2	3346	4702	4651	
	3	6.58E-07	2	74	21	20	
	4	6.58E-06	2	19	14	18	
	MAX	6.58E-05					

Table 4-18 Results for DCG-HEU-MDS Algorithm for 10 Targets, 30 Threat Points, and best 8 Waypoints (Maximin-Whole Area-Removing-Total Threat Reduction)

Total Number of Visited Targets							
		r, Route Travel Time (Sub problem)					
		MIN	1	2	3	4	MAX
d, Total Threat Level (Master)	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	1	1	1	
	2	6.58E-08	1	6	8	8	
	3	6.58E-07	1	10	10	10	
	4	6.58E-06	1	10	10	10	
	MAX	6.58E-05					
Number of UAVs Used							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	1	1	1	1	
	2	6.58E-08	1	4	3	2	
	3	6.58E-07	1	5	4	2	
	4	6.58E-06	1	5	5	5	
	MAX	6.58E-05					

Table 4-18 Continued

Time until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	0.66	2258.91	59.24	136.16	
	2	6.58E-08	0.58	934.05	16.19	179.79	
	3	6.58E-07	0.38	610.43	14.06	62.13	
	4	6.58E-06	0.62	57.81	52.94	32.36	
	MAX	6.58E-05					

Time until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	7.42	Time Ran Out	Time Ran Out	Time Ran Out	
	2	6.58E-08	7.75	Time Ran Out	Time Ran Out	Time Ran Out	
	3	6.58E-07	7.84	697.734	14.279	62.46	
	4	6.58E-06	8.48	58.49	53.181	33.18	
	MAX	6.58E-05					

Number of Generated Variables until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	864	16	46	
	2	6.58E-08	2	313	17	67	
	3	6.58E-07	2	62	17	21	
	4	6.58E-06	2	18	19	21	
	MAX	6.58E-05					

Number of Generated Variables until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	2	3701	4628	4771	
	2	6.58E-08	2	3975	4725	4673	
	3	6.58E-07	2	87	18	22	
	4	6.58E-06	2	19	20	22	
	MAX	6.58E-05					

Table 4-18 Continued

Number of MDS Cuts until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	0	3	3	1	
	2	6.58E-08	0	3	4	12	
	3	6.58E-07	0	1	3	7	
	4	6.58E-06	0	0	5	4	
	MAX	6.58E-05					
Number of MDS Cuts until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	56.06	112.12	168.18	224.25	280.31
	1	6.58E-09	0	57	470	489	
	2	6.58E-08	0	512	6	43	
	3	6.58E-07	0	1	3	7	
	4	6.58E-06	0	0	5	4	
	MAX	6.58E-05					

When comparing the DCG-HEU algorithm for the previous best 15 waypoints and the best 8 waypoints, both generated using Maximin-Whole Area-Removing-Total Threat Reduction method, overall, a total of 90 targets are visited for all of the 16 runs using both the best 8 waypoints and using the previous best 15 waypoints as well. Obtaining the same number of visited targets with fewer waypoints is more preferable. In addition, by looking at the time until the best-known solution is found, the DCG-HEU algorithm actually is faster when using 8 waypoints than 15 waypoints. Therefore, we consider the DCG-HEU algorithm's performance better with 8 waypoints than 15 waypoints. Overall, the number of UAVs used is the same for both the 8 waypoints and the 15 waypoints. All the available 10 targets are visited in 6 runs with both 8 and 15 waypoints. The algorithm terminates before completing the 4-hour run time limit in 11 runs with both the 8 and 15 waypoints. The number of generated variables until the best-known solution is found is fewer, overall, when using 8 waypoints than 15 waypoints. The number of generated variables until the algorithm is finished is more, overall, when using the 8 waypoints than the 15 waypoints.

When comparing the DCG-HEU-MDS algorithm for the 15 waypoints and the 8 waypoints, both generated using Maximin-Whole Area-Removing-Total Threat Reduction method, the algorithm performs better with 8 waypoints than 15 waypoints. Overall, a total of 89 targets are visited for all of the 16 runs using the 8 waypoints compared to a total of 82 targets for the 15 waypoints. For both, the 8 waypoints and the 15 waypoints, all the available 10 targets are visited in 6 runs. The algorithm terminates before completing the 4-hour run time limit in 10 runs with both the 8 waypoints and the 15 waypoints. Since the number of visited targets are more with 8 waypoints than with 15 waypoints, the results show that the number of UAVs used, the number of generated variables until the best-known solution is found and until the algorithm is finished, are also more, overall, when using the 8 waypoints than the 15 waypoints. However, the number of MDS cuts until the best-known solution is found are more, overall, when using the 15 waypoints than the 8 waypoints, while, the number of MDS cuts until the algorithm is finished, appear greater, overall, when using the 8 waypoints than the 15 waypoints.

Comparing both algorithms, the DCG-HEU and the DCG-HEU-MDS, for only the 8 waypoints, generated using the Maximin-Whole Area-Removing-Total Threat Reduction method, the DCG-HEU algorithm performs slightly better with the 8 waypoints than the DCG-HEU-MDS algorithm does with same number of waypoints. Overall, a total of 90 targets were visited for all of the 16 runs using the DCG-HEU algorithm compared to a total of 89 targets using the DCG-HEU-MDS algorithm. In both algorithms, all the available 10 targets are visited in 6 runs. Both algorithms terminate before completing the 4-hour run time limit in 10 runs and use the same number of UAVs.

When looking at the number of generated variables, until the best-known solution is found and until the algorithm is finished, the DCG-HEU-MDS algorithm uses the Minimum Dependent Set constraints, which cut some fractional solutions and encourage

solution integrality, while, the DCG-HUE algorithm does not. We expect the DCG-HEU-MDS algorithm to generate fewer variables than the DCG-HEU algorithm does for the same solution. However, the results show that the number of generated variables, until the best-known solution is found and until the algorithm is finished, using the DCG-HEU-MDS algorithm are actually more than those generated using DCG-HEU algorithm. Overall results for all the 16 runs for the 29 waypoints, the best 15 waypoints, and the best 8 waypoints for both algorithms are presented in Table 4-19.

Table 4-19 Overall Results for the 29 Waypoints, the Best 15 Waypoints, and the Best 8 Waypoints

Overall (16 Runs)	29 Waypoints (Maximin-Whole Area-Removing)		Best 15 Waypoints (Maximin-Whole Area-Removing-Total Threat Reduction)		Best 8 Waypoints (Maximin-Whole Area-Removing-Total Threat Reduction)	
Item	DCG-HEU	DCG-HEU-MDS	DCG-HEU	DCG-HEU-MDS	DCG-HEU	DCG-HEU-MDS
Total Number of Visited Targets	64	62	90	82	90	89
Achieving Mission (Visiting all 10 Targets) (# Runs)	5	3	6	6	6	6
Number of UAVs Used	35	30	42	35	42	42
Terminating before Completing the 4-Hour Run Time (# Runs)	9	7	11	10	11	10
Number of Generated Variables until the Best Known Solution Found	126	121	1,748	1,055	769	1,489
Number of Generated Variables until the Algorithm Terminated	250	234	12,058	10,431	21,865	26,669
Number of MDS Cuts until the Best Known Solution Found	–	14	–	207	–	46
Number of MDS Cuts until the Algorithm Terminated	–	16	–	689	–	1,597

Based on the previous results and discussions, we can conclude that, for the 10-targets case, the best 8 waypoints gives better results than any other number of waypoints we have tested for both the DCG-HEU algorithm and DCG-HEU-MDS algorithm based on the 4-hour run time limit. In addition, for the 10-targets case, for the same number of waypoints, the DCG-HEU algorithm performs better than the DCG-HEU-MDS algorithm.

4.2.2 Results for Twenty-Target Case (UAVRP2)

At the beginning, we obtained the route travel time that corresponded to the optimal solution found for the 20 targets in Table 4-4, in $d = 100\%$ with only one UAV. That is 321.2 hours. Then, this value was used to represent the maximum value for the predetermined constant parameter r in order to limit the route travel time in our sub problem. Four more evenly spaced levels of r (256.95, 192.71, 128.47, and 64.24 hours) were considered. Similarly, we obtained the total threat level of the optimal solution found for the 20 targets in Table 4-4, in $d = 100\%$ with one UAV. That is 1.22E-05. Then, this value was used as a reference level to obtain more levels of d. Going two levels above the reference threat level value by multiplying by 10 and 10^2, respectively, and two levels below the reference threat level value by multiplying by 10^{-1} and 10^{-2}, respectively, then, the value 1.22E-03 was used to represent the maximum value for the predetermined constant parameter d in order to limit the total threat level in our master problem. Therefore, the considered four levels of d were 1.22E-04, 1.22E-05, 1.22E-06, and 1.22E-07.

Only the methods that are based on the Maximin-Whole Area method, described in Chapter 3, are used to generate waypoints for the 20-target case. We followed the same procedure that we did for the 10-target case. First, we generated 210 waypoints using the Maximin-Whole Area. Figure 4-13 shows a set of problem instances that includes 5 UAVs

located at the base, 20 targets, 60 threat points, and the 210 waypoint. We generated 210 waypoints because we assumed that a waypoint would lie in a rectangle. We assumed that there is a rectangle between any two targets in the area of operation including the source node. We have 20 targets plus one source node. That is a total of 210 rectangles, $\binom{21}{2}$.

Then, the process follows by using the Maximin-Whole Area-Removing method, described in Chapter 3, in order to remove the waypoints that are unlikely to be visited by a UAV. This is done by removing any waypoint that does not lie in any one of the 210 rectangles. We did remove the unnecessary waypoints from the area of operation. The number of waypoints in the area of operation reduced from 210 waypoints to only 148 waypoints. Now the area of operation, shown in Figure 4-14, includes 5 UAVs located at the base, 20 targets, 60 threat points, and the 148 waypoints.

After that, we decided to use the Maximin-Whole Area-Removing-Total Threat Reduction method in order to calculate the total threat reduction for each waypoint of the 148 waypoints based on the number of rectangles that a waypoint lies in and. After that, we sorted the 148 waypoints based on their total threat reduction from highest to lowest. Then, we selected the best 40 waypoints and the best 20 waypoints in terms of highest total threat reduction in order to test our two algorithms, the DCG-HEU algorithm and the DCG-HEU-MDS algorithm.

We expect to obtain better results with fewer waypoints, 20, rather than more waypoints, 40. The waypoints increase the size of the problem even though they are not required to be visited. A set of problem instances that includes 20 targets and 40 waypoints is equivalent to solving a problem with 60 nodes, while, a set of problem instances that includes 20 targets and 20 waypoints is equivalent to solving a problem with 40 nodes. As the size of the problem, total number of targets and waypoints, decreases, the computational complexity of the problem exponentially decreases. The time for optimizing

the sub problem, detecting cycles in the solution, and generating new routes will be reduced. Therefore, we expect to obtain better results with the 20 waypoints rather than with the 40 waypoints.

Figure 4-15 shows a set of problem instances that includes 20 targets, 60 threat points, and the best 40 waypoints, generated using the Maximin-Whole Area-Removing-Total Threat Reduction method, and Figure 4-16 shows a set of problem instances that includes 20 targets, 60 threat points, and the best 20 waypoints, generated using Maximin-Whole Area-Removing-Total Threat Reduction method as well.

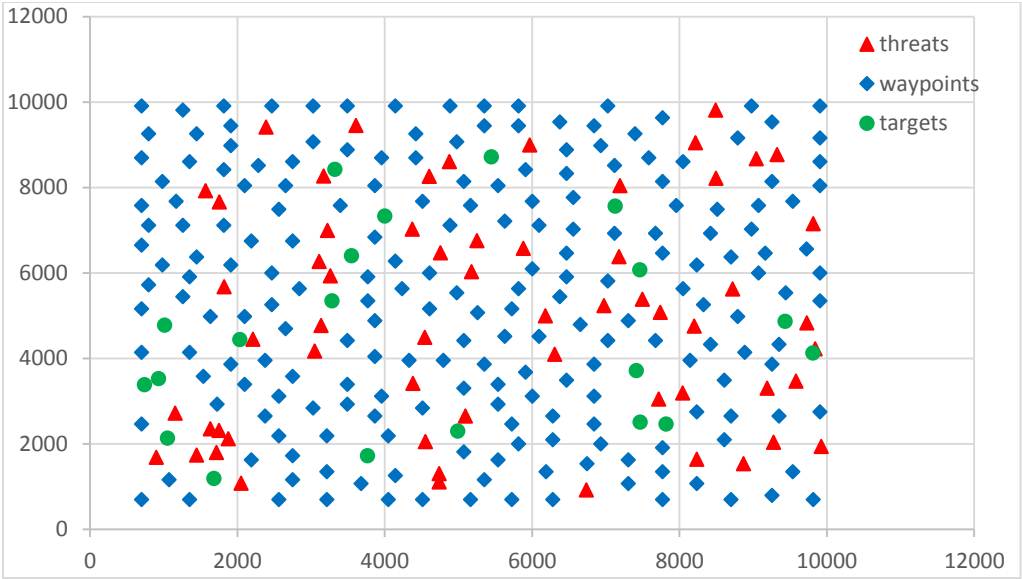


Figure 4-13 20 Targets, 60 Threat Points, and 210 Waypoints (Maximin-Whole Area)

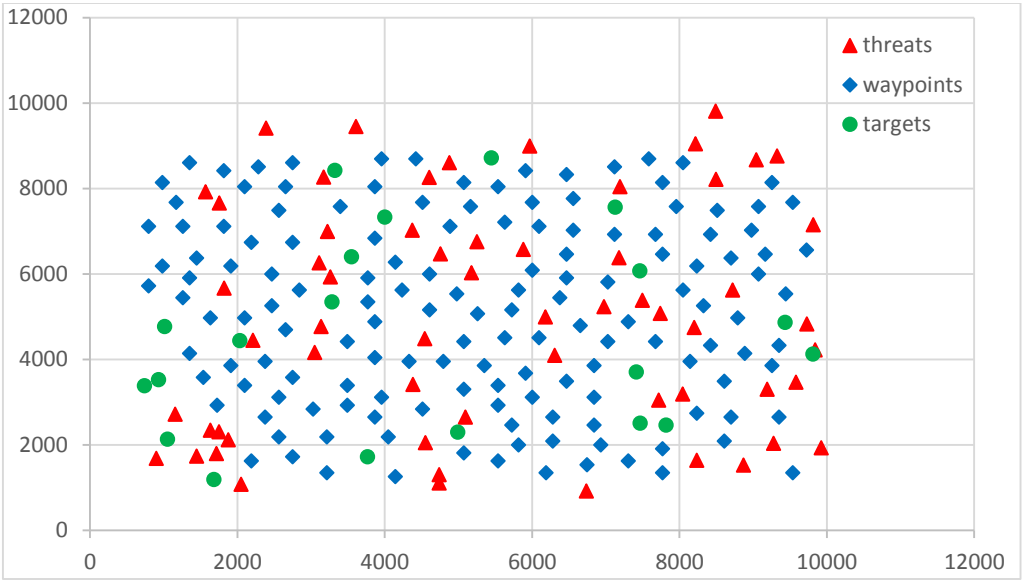


Figure 4-14 20 Targets, 60 Threat Points, and 148 Waypoints (Maximin-Whole Area-Removing)

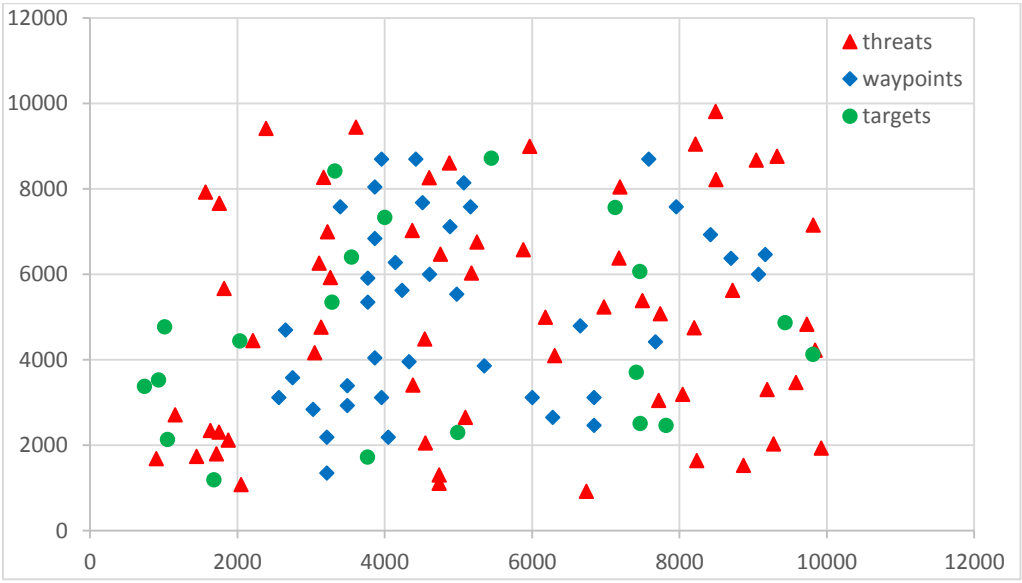


Figure 4-15 20 Targets, 60 Threat Points, and Best 40 Waypoints (Maximin-Whole Area-Removing-Total Threat Reduction)

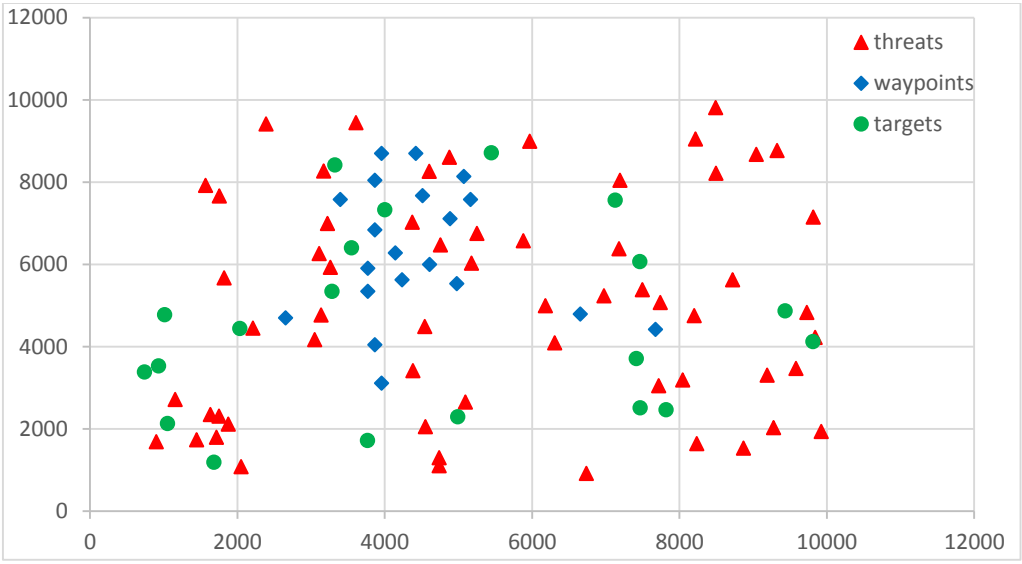


Figure 4-16 20 Targets, 60 Threat Points, and Best 20 Waypoints (Maximin-Whole Area-Removing-Total Threat Reduction)

The run time was set at 4 hours for each set of problem instances, each route travel time level r, each threat level d, and each algorithm type. Our goal now is to see how the total number of visited targets for multiple UAVs responds to varying the levels of r and d within the available 4-hour run time limit for the best 40 waypoints and the best 20 waypoints with the two algorithms, DCG-HEU and DCG-HEU-MDS.

The results for the best 40 waypoints and the best 20 waypoints for the DCG-HEU algorithm are presented in Table 4-20 and Table 4-21, respectively. Each table includes 4 varying levels of the constant parameter, r, which represents the limit for the route travel time, and four varying levels of the constant parameter, d, which represents the limit for the total threat level for all UAVs. This is actually a total of 16 runs for each table. For each run, the solution we obtained includes the maximum total number of visited targets, our objective function, the number of UAV used, the time, in seconds, until best-known solution found, the time, in seconds, until the algorithm finished, the number of generated variables until best-known solution found, and the number of generated variables until the algorithm

finished. For example, for the run at $r_4 = 256.95$ and $d_4 = 1.22E-04$ in Table 4-21, the maximum total number of visited targets is 20, the number of UAV used is 4, the time until the best-known solution is found was 0.21 seconds, the time until the algorithm finished was 12,398 seconds, the number of generated variables until the best-known solution found was 15, and finally the number of generated variables until the algorithm finished was 17.

Table 4-20 Results for DCG-HEU Algorithm for 20 Targets, 60 Threat Points, and best 40 Waypoints (Maximin-Whole Area-Removing-Total Threat Reduction)

Total Number of Visited Targets							
		r, Route Travel Time (Sub problem)					
		MIN	1	2	3	4	MAX
d, Total Threat Level (Master)	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	1	0	0	0	
	2	1.22E-06	3	1	16	1	
	3	1.22E-05	3	12	6	19	
	4	1.22E-04	3	16	6	19	
	MAX	1.22E-03					
Number of UAVs Used							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	1	0	0	0	
	2	1.22E-06	2	1	5	1	
	3	1.22E-05	2	5	1	4	
	4	1.22E-04	2	5	1	3	
	MAX	1.22E-03					
Time until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	7.58	0.67	0.69	0.59	
	2	1.22E-06	2787.98	1.95	13076.50	1.30	
	3	1.22E-05	2787.73	132.22	5.82	2279.12	
	4	1.22E-04	2795.58	584.92	10.99	2188.41	
	MAX	1.22E-03					

Table 4-20 Continued

Time until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	2	1.22E-06	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	3	1.22E-05	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	4	1.22E-04	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	MAX	1.22E-03					

Number of Generated Variables until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	3	1	1	1	
	2	1.22E-06	5	2	42	2	
	3	1.22E-05	5	16	6	23	
	4	1.22E-04	5	24	6	13	
	MAX	1.22E-03					

Number of Generated Variables until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	8	15	23	21	
	2	1.22E-06	8	35	44	54	
	3	1.22E-05	8	28	8	28	
	4	1.22E-04	8	38	8	21	
	MAX	1.22E-03					

Table 4-21 Results for DCG-HEU Algorithm for 20 Targets, 60 Threat Points, and best 20 Waypoints (Maximin-Whole Area-Removing-Total Threat Reduction)

Total Number of Visited Targets							
		r, Route Travel Time (subproblem)					
		MIN	1	2	3	4	MAX
d, Total Threat Level (master)	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	1	0	0	0	
	2	1.22E-06	4	1	1	1	
	3	1.22E-05	5	13	13	20	
	4	1.22E-04	6	16	20	20	
	MAX	1.22E-03					

Number of UAVs Used							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	1	0	0	0	
	2	1.22E-06	3	1	1	1	
	3	1.22E-05	4	4	3	3	
	4	1.22E-04	5	5	4	4	
	MAX	1.22E-03					

Time until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	5
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	1.75	0.21	0.28	0.21	
	2	1.22E-06	7865.60	0.51	0.54	0.43	
	3	1.22E-05	12013.56	573.78	26.61	831.32	
	4	1.22E-04	13387.09	181.24	609.25	12391.06	
	5	1.22E-03					

Time until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	2	1.22E-06	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	3	1.22E-05	Time Ran Out	Time Ran Out	Time Ran Out	846.64	
	4	1.22E-04	Time Ran Out	Time Ran Out	Time Ran Out	12398.48	
	MAX	1.22E-03					

Table 4-21 Continued

Number of Generated Variables until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	3	1	1	1	
	2	1.22E-06	9	2	2	2	
	3	1.22E-05	11	15	11	46	
	4	1.22E-04	16	18	19	15	
	MAX	1.22E-03					
Number of Generated Variables until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	63	29	25	75	
	2	1.22E-06	12	34	44	57	
	3	1.22E-05	16	42	46	57	
	4	1.22E-04	17	20	23	17	
	MAX	1.22E-03					

When testing DCG-HEU with the best 40 waypoints and the best 20 waypoints, both generated using the Maximin-Whole Area-Removing-Total Threat Reduction method, for the 20-targets case, the maximum total number of visited targets varies in response to decreasing or increasing the route travel time r, the total threat level d, or both. Generally, we can see that as both the route travel time r and the total threat level d decrease, the maximum total number of visited targets also decreases. Similarly, as both the route travel time r and the total threat level d increase, the maximum total number of visited targets also increases.

The DCG_HEU algorithm with the best 20 waypoints performs better than with the best 40 waypoints. Overall, a total of 121 targets are visited for all of the 16 runs using the 20 waypoints compared to a total of 106 targets using the best 40 waypoints. In terms of achieving the mission, with the best 20 waypoints, all the available 20 targets are visited in 3 runs, at $(r = 256.95,\ d = 1.22E-04)$, $(r = 256.95,\ d = 1.22E-05)$, $a\text{nd}\ (r = 192.71,$

$d = 1.22E - 04$), while none of the runs with the best 40 waypoints are able to visit all available targets. With the best 20 waypoints, the algorithm terminates before completing the 4-hour run time limit in 2 runs, while none of the runs with the best 40 waypoints are able to finish before reaching the 4-hour run time limit. Since the number of visited targets are more with best 20 waypoints than with the best 40 waypoints, the results show that the number of UAVs used and the number of generated variables until the best-known solution is found and until the algorithm is finished are also more, overall, when using the best 20 waypoints than with the best 40 waypoints.

The results for the best 40 waypoints and the best 20 waypoints for the DCG-HEU-MDS algorithm are presented in Table 4-22 and Table 4-23, respectively.

Table 4-22 Results for DCG-HEU-MDS Algorithm for 20 Targets, 60 Threat Points, and best 40 Waypoints (Maximin-Whole Area-Removing-Total Threat Reduction)

Total Number of Visited Targets							
		r, Route Travel Time (Sub problem)					
		MIN	1	2	3	4	MAX
d, Total Threat Level (Master)	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	1	0	0	0	
	2	1.22E-06	3	1	16	1	
	3	1.22E-05	3	15	15	14	
	4	1.22E-04	3	17	15	19	
	MAX	1.22E-03					
Number of UAVs Used							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	1	0	0	0	
	2	1.22E-06	2	1	5	1	
	3	1.22E-05	2	5	4	2	
	4	1.22E-04	2	5	4	3	
	MAX	1.22E-03					

Table 4-22 Continued

Time until the Best Known Solution Found

		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	7.81	0.52	0.52	0.52	
	2	1.22E-06	2780.46	1.25	13108.20	1.30	
	3	1.22E-05	2778.48	349.47	493.76	11.60	
	4	1.22E-04	2783.26	1273.82	228.21	2158.22	
	MAX	1.22E-03					

Time until the Algorithm Terminated

		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	2	1.22E-06	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	3	1.22E-05	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	4	1.22E-04	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	MAX	1.22E-03					

Number of Generated Variables until the Best Known Solution Found

		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	3	1	1	1	
	2	1.22E-06	5	2	42	2	
	3	1.22E-05	5	22	15	7	
	4	1.22E-04	5	36	16	13	
	MAX	1.22E-03					

Number of Generated Variables until the Algorithm Terminated

		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	8	15	23	21	
	2	1.22E-06	8	35	44	54	
	3	1.22E-05	8	29	17	10	
	4	1.22E-04	8	37	16	21	
	MAX	1.22E-03					

Table 4-22 Continued

Number of MDS Cuts until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	0	0	0	0	
	2	1.22E-06	0	0	0	0	
	3	1.22E-05	0	10	1	0	
	4	1.22E-04	0	4	1	0	
	MAX	1.22E-03					
Number of MDS Cuts until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	0	0	0	0	
	2	1.22E-06	0	0	0	0	
	3	1.22E-05	0	10	1	3	
	4	1.22E-04	0	4	1	0	
	MAX	1.22E-03					

Table 4-23 Results for DCG-HEU-MDS Algorithm for 20 Targets, 60 Threat Points, and best 20 Waypoints (Maximin-Whole Area-Removing-Total Threat Reduction)

Total Number of Visited Targets							
		r, Route Travel Time (Sub problem)					
		MIN	1	2	3	4	MAX
d, Total Threat Level (Master)	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	1	0	0	0	
	2	1.22E-06	4	1	1	1	
	3	1.22E-05	5	9	20	20	
	4	1.22E-04	6	19	20	20	
	MAX	1.22E-03					
Number of UAVs Used							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	1	0	0	0	
	2	1.22E-06	3	1	1	1	
	3	1.22E-05	4	2	5	4	
	4	1.22E-04	5	5	4	4	
	MAX	1.22E-03					

Table 4-23 Continued

Time until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	2.06	0.21	0.21	0.21	
	2	1.22E-06	7806.28	0.52	0.44	0.42	
	3	1.22E-05	12039.77	4.35	3063.84	6454.40	
	4	1.22E-04	13369.09	710.06	441.09	12377.81	
	MAX	1.22E-03					

Time until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	2	1.22E-06	Time Ran Out	Time Ran Out	Time Ran Out	Time Ran Out	
	3	1.22E-05	Time Ran Out	Time Ran Out	Time Ran Out	6465.607	
	4	1.22E-04	Time Ran Out	Time Ran Out	Time Ran Out	12385.189	
	MAX	1.22E-03					

Number of Generated Variables until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	3	1	1	1	
	2	1.22E-06	9	2	2	2	
	3	1.22E-05	11	7	53	43	
	4	1.22E-04	16	31	19	15	
	MAX	1.22E-03					

Number of Generated Variables until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	63	29	25	90	
	2	1.22E-06	12	34	44	57	
	3	1.22E-05	17	26	57	58	
	4	1.22E-04	17	39	23	17	
	MAX	1.22E-03					

Table 4-23 Continued

Number of MDS Cuts until the Best Known Solution Found							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	0	0	0	0	
	2	1.22E-06	0	0	0	0	
	3	1.22E-05	0	0	5	3	
	4	1.22E-04	0	8	0	0	
	MAX	1.22E-03					
Number of MDS Cuts until the Algorithm Terminated							
		Route Travel Time					
		MIN	1	2	3	4	MAX
Total Threat Level	MIN	0	64.24	128.47	192.71	256.95	321.2
	1	1.22E-07	0	0	0	3	
	2	1.22E-06	0	0	0	0	
	3	1.22E-05	4	3	5	3	
	4	1.22E-04	0	8	0	0	
	MAX	1.22E-03					

The DCG-HEU-MDS algorithm performs better with the best 20 waypoints than with the best 40 waypoints. Overall, a total of 127 targets are visited for all of the 16 runs using the best 20 waypoints compared to a total of 123 targets when using the best 40 waypoints. In terms of achieving the mission, with the best 20 waypoints, all the available 20 targets are visited in 4 runs, while none of the runs with the best 40 waypoints are able to visit all available targets. Notice that, with the best 20 waypoints, the algorithm terminates before completing the 4-hour run time limit in 2 runs, while, none of the runs with the best 40 waypoints are able to finish before reaching the 4-hour run time limit. Since the number of visited targets are more with the best 20 waypoints than with the best 40 waypoints, the results show that the number of UAVs used, the number of generated variables, until the best-known solution is found and until the algorithm is finished, and the number of MDS cuts, until the best-known solution is found and until the algorithm is finished, are also more, overall, when using the best 20 waypoints than the best 40 waypoints.

Since the best 20 waypoints gives better results than the best 40 waypoints for both the DCG-HEU algorithm and DCG-HEU-MDS algorithm, we now evaluate the performance of both algorithms for only the best 20 waypoints generated using the Maximin-Whole Area-Removing-Total Threat Reduction method.

Comparing both algorithms, the DCG-HEU and the DCG-HEU-MDS, for only the best 20 waypoints, the DCG-HEU-MDS algorithm performs better with the best 20 waypoints than the DCG-HEU algorithm does with the same number of waypoints. Overall, a total of 127 targets are visited for all of the 16 runs using the DCG-HEU-MDS algorithm compared to a total of 121 targets using the DCG-HEU algorithm. In terms of achieving the mission, with the DCG-HEU-MDS algorithm, all the available 20 targets are visited in 4 runs, compared to 3 runs with the DCG-HEU algorithm. Both algorithms terminate before completing the 4-hour run time limit in 2 runs. Since the number of visited targets are more with the DCG-HEU-MDS algorithm than with the DCG-HEU algorithm, the results show that the number of UAVs used are also more, overall, when using the DCG-HUE-MDS algorithm than the DCG-HUE algorithm. By examining the tables, we can see that the number of generated variables, until the best-known solution is found and until the algorithm is finished, using the DCG-HEU-MDS algorithm are actually greater than those generated using the DCG-HEU algorithm.

Overall results for all the 16 runs for the best 40 waypoints and the best 20 waypoints for both algorithms are presented in Table 4-24.

Table 4-24 Overall Results for the Best 40 Waypoints and the Best 20 Waypoints

Overall (16 Runs)	Best 40 Waypoints (Maximin-Whole Area-Removing-Total Threat Reduction)		Best 20 Waypoints (Maximin-Whole Area-Removing-Total Threat Reduction)	
Item	DCG-HEU	DCG-HEU-MDS	DCG-HEU	DCG-HEU-MDS
Total Number of Visited Targets	106	123	121	127
Achieving Mission (Visiting all 20 Targets) (# Runs)	0	0	3	4
Number of UAVs Used	33	37	39	40
Terminating before Completing the 4-Hour Run Time (# Runs)	0	0	2	2
Number of Generated Variables until the Best Known Solution Found	155	176	172	216
Number of Generated Variables until Algorithm Terminated	355	354	577	608
Number of MDS Cuts until the Best Known Solution Found	–	16	–	16
Number of MDS Cuts until the Algorithm Terminated	–	19	–	26

Based on the previous results and discussions, we can conclude that for the 20-targets case, the best 20 waypoints gives better results than the best 40 waypoints for both the DCG-HEU algorithm and DCG-HEU-MDS algorithm based on the 4-hour run time limit. In addition, for the 20-targets case, for the same number of waypoints, the DCG-HEU-MDS algorithm performs better than the DCG-HEU algorithm.

Chapter 5

CONCLUSIONS AND FUTURE RESEARCH

In this research, we formulated two mixed integer linear programs for routing the UAVs in the presence of threats. We did a computational study for the first formulation (UAVRP1) and presented results for the small-sized problem, 10 targets, and the large-sized problem, 20 targets. In the small-sized problem, the results show that as the total threat level decreases, both the total expected fuel burn and the number of waypoints visited increase for both algorithms, DCG-HEU and DCG-HEU-MDS. In addition, the results show that we use only one UAV for both algorithms. Based on these results, the UAVRP1 is consider a traveling salesman problem with waypoint generation. Furthermore, CPU time was a major concern. The run time for the small-sized problem was set at 20 hours. The UAVRP1 was solved to optimality for all threat levels except for only one level where we had to increase the run time above the 20-hour limit in order to obtain the optimal solution. For the large-sized problem, 20 targets, the run time was limited to 4 hours, and the results show that our code works for the use of more than one UAV for both algorithms, the DCG-HEU algorithm and the DCG-HEU-MDS algorithm.

We did a computational study for the second formulation (UAVRP2) and presented results for both the small-sized problem and the large-sized problem. Results for both the small-sized problem and the large-sized problem show that fewer waypoints gives better results for both algorithms, DCG-HEU and DCG-HEU-MDS, based on the 4-hour run time limit. This is because the waypoints are considered targets and increase the size of the problem even though they are not required to be visited. As the size of the problem, the total number of targets and waypoints, decreases, the computational complexity of the problem exponentially decreases. Therefore, we expect to obtain better results with fewer waypoints. In addition, for the small-sized problem, the DCG-HEU algorithm performs

better than the DCG-HEU-MDS algorithm when using the same number of waypoints. Furthermore, for the large-sized problem, the DCG-HEU-MDS algorithm performs better than the DCG-HEU algorithm when using same number of waypoints.

We now describe our future steps for this research. The problem we optimize in this research requires a very long run time to obtain the solution results. One of our future plans includes improving the CPU time. In this research we generate waypoints and add them all together *a priori*. Waypoints are considered targets that do not need to be visited. However, based on our implementation, a problem instances of 10 targets and 30 waypoints, for example, is equivalent to solving a problem with 40 nodes. If we consider larger problem instances, the computational complexity of the problem will exponentially increase as the number of targets and waypoints increases. Therefore, we plan not to add all waypoints together *a priori*. Instead, we consider adding waypoints dynamically and only when they are needed.

In addition, in our current implementation, we add only one simple path, at a time, with negative reduced cost to RMP. Instead, we plan to add multiple simple paths with negative reduced costs to RMP to save time in finding the solution. Furthermore, in our current implementation, we add any simple path, to the RMP, as long as it has a negative reduced cost regardless if its threat level is greater than the predetermined constant parameter d or not. Since it is not going to be part of the solution any way, we plan not to add it to the RMP to save some computational time.

Reducing the size of the problem will improve the CPU time. One way to do so, is by putting a threshold on the threat level of the links, and then delete the ones that have threat levels greater than that threshold.

In order to encourage the use of more UAVs, we could impose a constraint to limit the expected fuel burn of the route, the range of the distance that a UAV can travel, or the number of targets to be visited in the route.

In our second formulation (UAVRP2), we treated all target with equal importance. We could test the UAVRP2 with targets that have different priorities by varying their benefit values. Finally, in terms of total threat reduction, determining the best number of waypoints to use need to be investigated.

Appendix A

An Attempt to Model The Probability of Running Out of Fuel for UAVs

We study an unmanned aerial vehicle routing problem with limited risk in which the considered risk is the probability of running out of fuel for a UAV. The problem determines optimal routes that minimize the total probability of running out of fuel for UAVs in order to visit all targets while maintaining the total expected fuel burn to a constant parameter.

We use the same fuel burn model that was introduced by Visoldilokpun [41]. Let the random variable $\tilde{e}_{uf}$ represent the total fuel burn for a UAV u for traveling in route f. $\tilde{e}_{uf} = \alpha_{uf} - \beta_{uf}\tilde{v}^{w}$, where, α_{uf} is the total expectd fuel burn, the random variable $\tilde{v}^{w}$ represents the deviation of the wind speed, and β_{uf} represents the wind speed deviation effect on fuel burn. Let Cap_u represent the fuel burn capacity for a UAV u. The problem is formulated as an integer programing set-partitioning as follows

$$min \sum_{u \in U} \sum_{f \in F_u} P\left[\tilde{e}_{uf} > Cap_u\right] x_{uf} \tag{A.1}$$

$$s.t. \sum_{u \in U} \sum_{f \in F_u} a_{kuf}\ x_{uf} = 1 \qquad \forall k \in K \tag{A.2}$$

$$\sum_{f \in F_u} x_{uf} \leq n_u \qquad \forall u \in U \tag{A.3}$$

$$\sum_{u \in U} \sum_{f \in F_u} \alpha_{uf}\ x_{uf} \leq d \tag{A.4}$$

$$x_{uf} \in \{0,1\} \qquad \forall\, u \in U, f \in F_u \tag{A.5}$$

We use the same branch and cut and price methodology. At each node of the tree the continues relaxation of the problem is solved. In the cut step, MDS constraints are generated and added to the RMP. In the pricing, new routes are generated using the column generation algorithm and added to the RMP. Then, bounds are updated and branching is carried out.

The integer programing sub problem (MCNF) for the column generation step is formulated as follows

$$\min\ P\left[\alpha_{uf} - \beta_{uf}\tilde{v}^{w} > Cap_u\right] - \sum_{l\in L}\{\alpha_l\,\rho^* + \pi_k^*\}y_l \tag{A.6}$$

$$s.t.\ \sum_{l\in\delta_{v_i}^{+}} y_l - \sum_{l\in\delta_{v_i}^{-}} y_l\ =\ b(v_i) \qquad \forall v_i \in V \tag{A.7}$$

$$\sum_{l\in\delta_{v_i}^{+}} y_l \le 1 \qquad \forall v_i \in V \tag{A.8}$$

$$\sum_{l\in f} y_l \le\ |\Omega(f)| \qquad \forall f\ \in\ \bar{F} \tag{A.9}$$

$$\sum_{l\in\psi} y_l\ \le\ h(\psi) \qquad \forall\psi\ \in\ \mathcal{H} \tag{A.10}$$

$$y_l \in \{0,1\} \qquad \forall l \in L \tag{A.11}$$

$$\alpha_{uf} = \sum_{l\in f}\alpha_{ul}y_l \tag{A.12}$$

$$\beta_{uf} = \sum_{l\in f}\beta_{ul}y_l \tag{A.13}$$

$$\alpha_{uf} \le Cap_u \tag{A.14}$$

We assume that the random variable $\tilde{e}_{uf}$, follows the normal distribution with $\mu = \alpha_{uf}$ and $\sigma = -\beta_{uf}\sqrt{var(\tilde{v}^{w})}$. Therefore, the probability of running out of fuel for a UAV u in a route f is as follows

$$P\left[\tilde{e}_{uf} > Cap_u\right] = 1 - \left(\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{Cap_u}\frac{1}{\sigma}e^{\frac{-(x-\mu)^2}{2\sigma^2}}dx\right) \tag{A.15}$$

The Hessian matrix, H, derived from the normal CDF is

$$\frac{1}{\sqrt{2\pi}}\begin{bmatrix} \int_{-\infty}^{Cap_u} e^{\frac{-(x-\mu)^2}{2\sigma^2}}\left(\frac{(x-\mu)^4}{\sigma^7} - \frac{5(x-\mu)^2}{\sigma^5} + \frac{2}{\sigma^2}\right)dx & \int_{-\infty}^{Cap_u} e^{\frac{-(x-\mu)^2}{2\sigma^2}}\left(\frac{(x-\mu)^3}{\sigma^6} - \frac{3(x-\mu)}{\sigma^4}\right)dx \\ \int_{-\infty}^{Cap_u} e^{\frac{-(x-\mu)^2}{2\sigma^2}}\left(\frac{(x-\mu)^3}{\sigma^6} - \frac{3(x-\mu)}{\sigma^4}\right)dx & \int_{-\infty}^{Cap_u} e^{\frac{-(x-\mu)^2}{2\sigma^2}}\left(\frac{(x-\mu)^2}{\sigma^5} - \frac{1}{\sigma^3}\right)dx \end{bmatrix}$$

We generated random values for μ and σ in order to calculate the hessian matrix numerically. The hessian matrix is a positive semi-definite. Therefore, the probability of running out of fuel, $P[\tilde{e}_{uf} > Cap_u]$, is a convex function. Plotting the CDF function along with the random values of μ and σ shows that the probability function is convex for the values of μ and σ that are less than the UAV's fuel burn capacity as shown in Figure A-1.

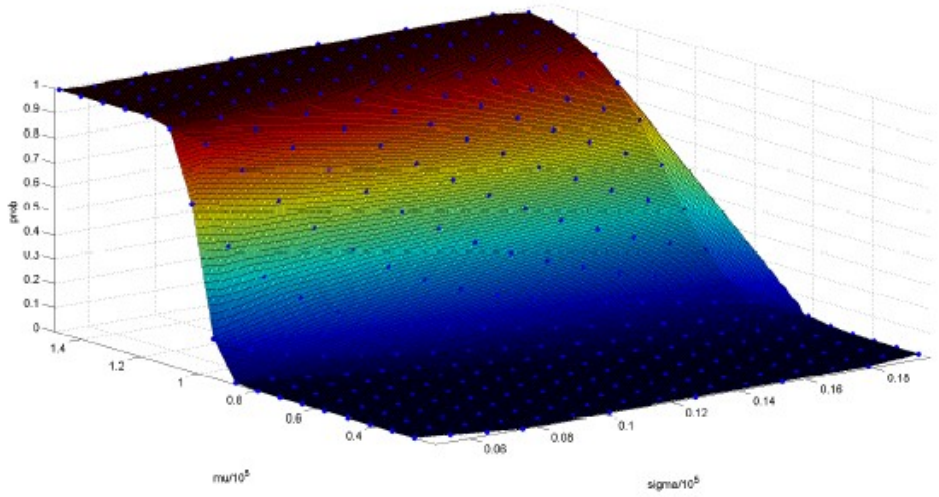


Figure A-1 3D Plot of the Probability Function, μ and σ

Since the probability of running out of fuel is a convex function, then the MCNF sub problem is a convex integer programming problem. Therefore, a cut generation step for the MCNF sub problem is needed in order to maintain its objective function convexity.

By using Kelley's cutting plane method [51], We create the following new sub problem as a cut generation step for the MCNF sub problem

$$\min \ \eta - \sum_{l \in L} \{\alpha_l \, \rho^* + \pi_k^*\} y_l \qquad (A.16)$$

$$s.t. \sum_{l \in \delta_{v_i}^+} y_l - \sum_{l \in \delta_{v_i}^-} y_l \ = \ b(v_i) \qquad \forall v_i \in V \qquad (A.17)$$

$$\sum_{l \in \delta^+_{v_i}} y_l \leq 1 \qquad \forall v_i \in V \qquad (A.18)$$

$$\sum_{l \in f} y_l \leq |\Omega(f)| \qquad \forall f \in \bar{F} \qquad (A.19)$$

$$\sum_{l \in \psi} y_l \leq h(\psi) \qquad \forall \psi \in \mathcal{H} \qquad (A.20)$$

$$y_l \in \{0,1\} \qquad \forall l \in L \qquad (A.21)$$

$$\alpha_{uf} = \sum_{l \in f} \alpha_{ul} y_l \qquad (A.22)$$

$$\beta_{uf} = \sum_{l \in f} \beta_{ul} y_l \qquad (A.23)$$

$$\alpha_{uf} \leq Cap_u \qquad (A.24)$$

$$\eta \geq F(\bar{\alpha}, -\bar{\beta}) + \nabla F(\bar{\alpha}, -\bar{\beta}) \begin{bmatrix} \alpha - \bar{\alpha} \\ \beta + \bar{\beta} \end{bmatrix} \qquad \forall \bar{\alpha}, -\bar{\beta} \in R_+ \qquad (A.25)$$

The MCNF sub problem is solved with some constraints to get the solution and new cuts are added dynamically when they are needed using the new sub problem.

We tested our model and obtained some preliminary results. At the beginning, we ignored the total expected fuel burn constraint in (A.4) and solved our problem without it. This is the case with the constant parameter $= 100\%$. The optimal solution obtained includes the total probability of running out of fuel and the corresponding total expected fuel burn. Six more levels of d were considered, which are 95%, 90%, 85%, 80%, 75%, and 75% of the expected fuel burn of the optimal solution found in the case with $d = 100\%$.

We created 3 sets of problem instances that include 10 targets, 15 targets, and 20 targets. We used 5 UAVs with each set. We tested our problem for all the three sets for only the DCG-HEU algorithm.

The preliminary results are shown in Table A-1. The results show that there exists so many routes with zero probability. This suggests that the UAV's probability of running out of fuel is not an issue based on the scope of the problem, the methods, and the approaches that were used to formulate and model the probability function.

Table A-1 Results for the Probability of Running out of Fuel for UAV

	Test	*d* Value	Probability	UAVs Used	Total Expected Fuel Burn
10 Targets	*d* = 100%	59,858.5	0.0	1 UAV	59,858.5
	d = 95%	56,865.6	0.0	3 UAV'S	52,717.0
	d = 90%	53,872.7	0.0	2 UAV'S	49,915.4
	d = 85%	50,879.7	0.0	2 UAV'S	31,524.7
	d = 80%	47,886.8	0.0	2 UAV'S	31,524.7
	d = 75%	44,893.9	0.0	2 UAV'S	31,524.7
	d = 70%	41,901.0	0.0	2 UAV'S	31,524.7
15 Targets	*d* = 100%	63,725.6	0.0	1 UAV	63,725.6
	d = 95%	60,539.3	0.0	4 UAV'S	42,366.4
	d = 90%	57,353.0	0.0	2 UAV'S	42,272.3
	d = 85%	54,166.8	0.0	3 UAV'S	36,239.2
	d = 80%	50,980.5	0.0	3 UAV'S	34,660.1
	d = 75%	47,794.2	0.0	2 UAV'S	30,856.0
	d = 70%	44,607.9	0.0	2 UAV'S	31,141.0
20 Targets	*d* = 100%	90,949.7	0.0	1 UAV	90,949.7
	d = 95%	86,402.2	0.0	1 UAV	79,108.9
	d = 90%	81,854.7	0.0	5 UAV'S	61,884.3
	d = 85%	77,307.2	0.0	2 UAV'S	73,702.3
	d = 80%	72,759.8	0.0	5 UAV'S	63,156.3
	d = 75%	68,212.3	0.0	4 UAV'S	64,669.2
	d = 70%	63,664.8	0.0	5 UAV'S	54,897.2

References

[1] Richards, A., Bellingham, J., Tillerson, M., & How, J. (2002, August). Coordination and control of multiple UAVs. In *AIAA guidance, navigation, and control conference, Monterey, CA*.

[2] Rathbun, D., Kragelund, S., Pongpunwattana, A., & Capozzi, B. (2002). An evolution based path planning algorithm for autonomous motion of a UAV through uncertain environments. In *Digital Avionics Systems Conference, 2002. Proceedings. The 21st* (Vol. 2, pp. 8D2-1). IEEE.

[3] Desrochers, M., Lenstra, J. K., & Savelsbergh, M. W. (1990). A classification scheme for vehicle routing and scheduling problems. *European Journal of Operational Research*, *46*(3), 322-332.

[4] Laporte, G., & Nobert, Y. (1987). Exact algorithms for the vehicle routing problem. North-Holland Mathematics Studies, 132, 147-184.

[5] Toth, P., & Vigo, D. (Eds.). (2001). The vehicle routing problem. Siam.

[6] Desrochers, M., Lenstra, J. K., Savelsbergh, M. W., & Soumis, F. (1988). Vehicle routing with time windows: Optimization and approximation. Vehicle routing: Methods and studies, 16, 65-84.

[7] Cordeau, J. F., Laporte, G., Savelsbergh, M. W., & Vigo, D. (2006). Vehicle routing. Transportation, handbooks in operations research and management science, 14, 367-428.

[8] Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. Management science, 6(1), 80-91.

[9] Clarke, G. U., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. Operations research, 12(4), 568-581.

[10] Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations research, 35(2), 254-265.

[11] Kolen, A. W., Rinnooy Kan, A. H. G., & Trienekens, H. W. J. M. (1987). Vehicle routing with time windows. Operations Research, 35(2), 266-273.

[12] Desrochers, M., Desrosiers, J., & Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. Operations research, 40(2), 342-354.

[13] Gambardella, L. M., Taillard, É., & Agazzi, G. (1999). Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In New ideas in optimization.

[14] Laporte, G., Nobert, Y., & Desrochers, M. (1985). Optimal routing under capacity and distance restrictions. Operations research, 33(5), 1050-1073.

[15] Balinski, M. L., & Quandt, R. E. (1964). On an integer program for a delivery problem. Operations Research, 12(2), 300-304.

[16] Gendreau, M., Hertz, A., & Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. Management science, 40(10), 1276-1290.

[17] Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Annals of operations research, 41(4), 421-451.

[18] Foo, J., Knutzon, J., Oliver, J., & Winer, E. (2006). Three-dimensional path planning of unmanned aerial vehicles using particle swarm optimization. In*11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, Virginia*.

[19] Wang, J., Liu, L., Long, T., & Wang, Z. Three-Dimensional Constrained UAV Path Planning using Modified Particle Swarm Optimization with Digital Pheromones

[20] Schouwenaars, T., De Moor, B., Feron, E., & How, J. (2001, September). Mixed integer programming for multi-vehicle path planning. In *European control conference* (Vol. 1, pp. 2603-2608).

[21] Jun, M., & D'Andrea, R. (2003, June). Probability map building of uncertain dynamic environments with indistinguishable obstacles. In *American Control Conference, 2003. Proceedings of the 2003* (Vol. 4, pp. 3417-3422). IEEE.

[22] Krishna, K. M., Hexmoor, H., Pasupuleti, S., & Llinas, J. (2005, April). Parametric control of multiple unmanned air vehicles over an unknown hostile territory. In *Proceedings of International Conference on Integration of Knowledge Intensive Multi-Agent Systems* (pp. 117-121).

[23] Zhenhua, W., Weiguo, Z., Jingping, S., & Ying, H. (2008). UAV route planning using multiobjective ant Colony system. In *2008 IEEE Conference on Cybernetics and Intelligent Systems* (pp. 797-800).

[24] Ruiz, J. J., Arévalo, O., de la Cruz, J. M., & Pajares, G. (2006, September). Using MILP for UAVs trajectory optimization under radar detection risk. In*Emerging Technologies and Factory Automation, 2006. ETFA'06. IEEE Conference on* (pp. 957-960). IEEE.

[25] Xinzeng, W., Linlin, C., Junshan, L., & Ning, Y. (2010, August). Route planning for unmanned aerial vehicle based on threat probability and mission time restriction. In *Geoscience and Remote Sensing (IITA-GRS), 2010 Second IITA International Conference on* (Vol. 1, pp. 27-30). IEEE.

[26] Bortoff, S. A. (2000, September). Path planning for UAVs. In *American Control Conference, 2000. Proceedings of the 2000* (Vol. 1, No. 6, pp. 364-368). IEEE.

[27] Dogan, A. (2003, October). Probabilistic approach in path planning for UAVs. In Intelligent Control. 2003 IEEE International Symposium on (pp. 608-613). IEEE.

[28] Dogan, A. (2003, September). Probabilistic path planning for UAVs. In proceeding of 2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations - Aerospace, Land, and Sea Conference and Workshop & Exhibition, San Diego, California, September 15-18.

[29] Blackmore, L., Li, H., & Williams, B. (2006, June). A probabilistic approach to optimal robust path planning with obstacles. In *American Control Conference, 2006* (pp. 7-pp). IEEE.

[30] Beard, R. W., McLain, T. W., Goodrich, M. A., & Anderson, E. P. (2002). Coordinated target assignment and intercept for unmanned air vehicles.*Robotics and Automation, IEEE Transactions on*, *18*(6), 911-922.

[31] Maddula, T., Minai, A. A., & Polycarpou, M. M. (2004). Multi-Target assignment and path planning for groups of UAVs. In *Recent Developments in Cooperative Control and Optimization* (pp. 261-272). Springer US.

[32] Zengin, U., & Dogan, A. (2004, September). Dynamic target pursuit by UAVs in probabilistic threat exposure map. In *Proceedings of AIAA 3rd" Unmanned Unlimited" Technical Conference, Workshop and Exhibit*.

[33] De Filippis, L., Guglieri, G., & Quagliotti, F. (2011). A minimum risk approach for path planning of UAVs. *Journal of Intelligent & Robotic Systems*, *61*(1-4), 203-219.

[34] Jun, M., & D'Andrea, R. (2003). Path planning for unmanned aerial vehicles in uncertain and adversarial environments. In *Cooperative Control: Models, Applications and Algorithms* (pp. 95-110). Springer US.

[35] Pelosi, M., Kopp, C., & Brown, M. (2012). Range-limited UAV trajectory using terrain masking under radar detection risk. *Applied Artificial Intelligence*,*26*(8), 743-759.

[36] Helgason, R. V., Kennington, J. L., & Lewis, K. R. (2001). Cruise missile mission planning: a heuristic algorithm for automatic path generation.*Journal of Heuristics*, *7*(5), 473-494.

[37] Zabarankin, M., Uryasev, S., & Pardalos, P. (2002). *Optimal risk path algorithms* (pp. 273-298). Springer US.

[38] McManus, I. A., Clothier, R. A., & Walker, R. A. (2005). Highly autonomous UAV mission planning and piloting for civilian airspace operations.

[39] Carlyle, W. M., Royset, J. O., & Wood, R. K. (2007). *Routing military aircraft with a constrained shortest-path algorithm*. NAVAL POSTGRADUATE SCHOOL MONTEREY CA DEPT OF OPERATIONS RESEARCH.

[40] Pfeiffer, B., Batta, R., Klamroth, K., & Nagi, R. (2005). Path planning for UAVs in the presence of threat zones using probabilistic modeling. *Institute of Applied Mathematics, University of Erlangen, Erlangen*.

[41] Visoldilokpun, S. (2008). *Unmanned Aerial Vehicle Routing Problem with Limited Risk*. ProQuest.

[42] Nemhauser, G. L., & Wolsey, L. A. (1988). Integer and combinatorial optimization (Vol. 18). New York: Wiley.

[43] Vance, P. H., Atamturk, A., Barnhart, C., Gelman, E., Johnson, E. L., Krishna, A., ... & Rebello, R. (1997). A heuristic branch-and-price approach for the airline crew pairing problem. preprint.

[44] Ryan, D. M., & Foster, B. A. (1981). An integer programming approach to scheduling. Computer scheduling of public transport urban passenger vehicle and crew scheduling, 269-280.

[45] Chvátal, V. (1983). Linear programming. New York: W.H. Freeman

[46] Johnson, M. E., Moore, L. M., & Ylvisaker, D. (1990). Minimax and maximin distance designs. Journal of statistical planning and inference, 26(2), 131-148.

[47] Kellerer, H., Pferschy, U., & Pisinger, D. (2004). Knapsack problems. Springer.

[48] Balas, E. (1975). Facets of the knapsack polytope. Mathematical Programming, 8(1), 146-164.

[49] Hammer, P. L., Johnson, E. L., & Peled, U. N. (1975). Facet of regular 0–1 polytopes. Mathematical Programming, 8(1), 179-206.

[50] Wolsey, L. A. (1975). Faces for a linear inequality in 0–1 variables. Mathematical Programming, 8(1), 165-178.

[51] Kelley, Jr, J. E. (1960). The cutting-plane method for solving convex programs. Journal of the Society for Industrial & Applied Mathematics, 8(4), 703-712.

Biographical Information

Kamil A. Alotaibi was born in Jeddah, Saudi Arabia. He received his B.S. degree in Electrical Engineering from King Abdul-Aziz University, in Jeddah, Saudi Arabia, in 2001. Then, he worked for one semester as a teacher at The College of Communication And Information Technology, in Riyadh, Saudi Arabia. He worked also for six years as a Senior Design Engineer at The Saudi Telecom Company in Riyadh, Saudi Arabia. He received his M.S. degree in Engineering Systems Management from St. Mary's University in San Antonio, TX, in 2009. He received his Ph.D. in Industrial Engineering under the supervision of Dr. Jay Rosenberger from The University of Texas at Arlington, Arlington, TX, in 2014. Currently, he is working at Taibah University in Saudi Arabia. His research interests are in the areas of Operations Research and Statistics.