Unmanned Aerial Vehicle Routing in the Presence of Threats

Kamil A. Alotaibi

Department of Industrial Engineering, Taibah University, P.O.Box 344, Medina, Saudi Arabia, 41411, kotaibi@taibahu.edu.sa

Jay M. Rosenberger

Department of Industrial and Manufacturing Systems Engineering, University of Texas at Arlington, Arlington, TX 76019, jrosenbe@uta.edu

Stephen P. Mattingly

Department of Civil Engineering, University of Texas at Arlington, Arlington, TX 76019, mattingly@uta.edu

Siriwat Visoldilokpun

Kasikorn Bank, Plc. 1 Soi. Kasikornthai, Radburana Rd. Bangkok, Thailand 10140, siriwatvi@yahoo.com

We study the routing of Unmanned Aerial Vehicles (UAVs) in the presence of the risk of enemy threats. The main goal is to find optimal routes that consider targets visited, threat exposure, and travel time. We formulate a mixed integer linear program that maximizes the total number of visited targets for multiple UAVs, while limiting both the route travel time for each UAV and the total threat exposure level for all UAVs to predetermined constant parameters. The formulation considers a set covering vehicle routing problem where the risk of threat exposure and the travel time are modeled for each edge in a vehicle routing network. To reduce threat exposure, waypoints are generated within the network so routes can avoid high-risk edges. We propose several waypoint generation methods. Using the candidate waypoints, the UAV routes are optimized with branch-and-cut-and-price (BCP) methodology. Minimum dependent set constraints and a simple path heuristic are used to improve the computational efficiency of the BCP algorithm. Computational results are presented, which show that the BCP algorithm performs best when the number of waypoints generated *a priori* is about half the number of targets.

Key words: routing; UAVs with threat risk; branch and cut and price; minimum dependent set constraints; waypoint generation

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 be used for various purposes and reach places and areas where human survival is risky or impossible. 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.

The development of UAVs is evolving, and there has been an increased interest to make UAVs operate more autonomously. When UAVs operate fully autonomously, they can plan their own paths to move from one location to another and avoid obstacles or threats encountered on their routes. 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.

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.

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, which not only optimize the number of visited targets but also limit the threat level of exposure and the travel time, must be designed efficiently and effectively. An acceptable level of threat is dependent on the planner's preference or protocols. As the accepted level of exposure to threats decreases, the UAV will try to avoid flying directly over threats and thereby travel farther away from them in order to reduce the threat exposure level. Thus, the UAV will travel a longer route. However, as the accepted level of exposure to threats increases, the UAV will travel a shorter, more direct route.

1.1 Research Objective

The main goal for this research is to find optimal routes that consider the targets visited, threat exposure, and travel time. We formulate a mixed integer linear program that maximizes the total number of visited targets for multiple UAVs, while limiting both the route travel time for each UAV and the total threat level for all UAVs to predetermined constant parameters. Furthermore, we generate waypoints that a UAV can visit while traveling from one target to another in order to reduce the threat level on high-threat level edges. When the maximum allowable level of threat decreases, the UAVs visit more waypoints, and conversely, as it increases, the UAVs are 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, a set of stationary targets that need to be visited, and a set of stationary threat points. Figure 1-1 (left) shows a depot location, represented by a black circle around the center, targets, represented by green circles, and threat points, represented by red triangles. Then, the pre-mission path planner generates a set of waypoints, represented by blue diamonds as shown in Figure 1-1 (middle), 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

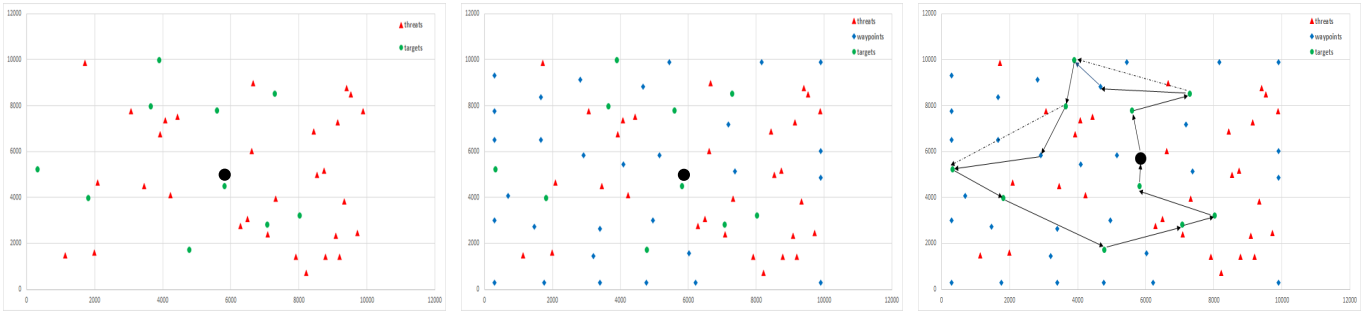

Figure 1-1 An Area of Operation Containing Targets and Threat Points (Left), Waypoints Generated and Added to the Area of Operation (Middle), and an Optimized Route (Right)

on the accepted level of threat, the pre-mission path planner obtains optimized routes. Figure 1-1 (right) shows an optimized route in which a single UAV originates and terminates at a depot location, visits a set of targets, and tries to avoid threat points along its route by visiting waypoints in order to maintain the pre-accepted level of threat.

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 widely believed to be first introduced in Dantzig and Ramser [8] in 1959. 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 at least once by at least one vehicle. In addition, vehicle capacity must not be exceeded by the total demand of customers in the path.

In general, the VRP 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. $E = \{e_{ij} = (v_i, v_j) : i \neq j, v_i, v_j \in V\}$ is a set of all 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 is a constant $n \times 1$ vector. The decision variable x represents a binary $n \times 1$ vector that indicates which routes 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 route, and an entry a_{ij} equals to 1 if customer i is serviced by route j, while it is 0 otherwise. In our model, we let the right hand side of the classical VRP's set-covering constraints in (2.2) be a binary variable and is only one when a target is visited. The VRP is considered an 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.

VRP survey papers in the literature include Desrochers et al. [3], Laporte and Nobert [4], Toth and Vigo [5], Desrochers et al. [6], and Cordeau et al. [7]. Many papers in the literature have proposed solution methods for solving the capacitated VRP, including Clarke and Wright [9], Laporte et al. [14], Balinski and Quandt [15], Gendreau et al. [16], and Osman [17]. Other papers in the literature, such as Solomon [10], Kolen et al. [11], Desrochers et al. [12],and Gambardella et al. [13], have proposed solution methods for solving the VRP in which each customer is required to be visited within a time window.

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

Many papers in the literature have studied UAV mission path planning in a hostile area of operation and tried to assess the risk associated with the routes that the UAV can take. The risk can be caused by sources of threats or obstacles such as the enemy's surface-to-air missiles, radar detection zones, no-fly zones, terrain, or other vehicles that the UAV could collide with on its route. Although threats may be stationary or moving, almost all papers have assumed stationary threats [1, 18, 19, 23-33, 35-40]. However, Jun and D'Andrea [21, 34], Rathbun et al. [2], and Schouwenaars et al. [20] allowed for moving threats. Zengin and Dogan [32] considered moving targets, but other literature has assumed targets that are stationary. In our research, we use stationary threat points and stationary targets.

Most research has used a single UAV and a single target in the presence of threats [2, 18-20, 23-27, 29, 32, 34-38, 40]. In addition, a few papers used a single UAV and multiple targets [28] or multiple UAVs and a single target [20, 34, 39]. However, a very small number of papers in the literature studied multiple UAVs and multiple targets [1, 22, 30, 31]. In our research, we consider the case with multiple UAVs that visit multiple targets.

Different numbers of threats, obstacles, and no fly zones were used in the literature. Rathbun et al. [2] considered two obstacles. Wang et al. [19] used a single threat and three obstacles, which are mountains. Xinzeng et al. [25] assumed five threat zones. Bortoff [26] designated ten radar sites as threat zones. Dogan [27] and [28] used seventeen threat zones. Zengin and Dogan [32] assumed seventeen threat areas and three no fly zones. Blackmore et al. [29] considered three obstacles. Beard et al. [30] used thirty-six threat points that represent radar locations. Maddula et al. [31] considered forty threat points that represented radars sites. Pelosi et al. [35] assumed two radar areas. Helgason et al. [36] used five obstacles. Zabarankin et al. [37] presented algorithms for a single radar and for two radars. Carlyle et al. [39] presented two different case studies. One considered fifteen surface-to-air missiles, and the other considered four of them. Pfeiffer et al. [40] assumed three threat zones of the same shape in one example, while two threat zones of different shapes were in another example. In our research, we consider two examples, one with 10 targets and 30 threat points and another example with 20 targets and 60 threat points.

Many research 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 a 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. Jun and D'Andrea [34] divided an area of operation, involving moving threats, into hexagonal cells. 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. In our research, we model threats as points that represent radars that are located at these threat points in two-dimensional areas of operation.

Many approaches and methods were considered to find optimized routes for UAVs in the presence of threats. The methods can be classified as graph based approaches, probabilistic approaches, and deterministic approaches. 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, which can form potential paths from a starting point to a target location [23, 26, 30, 31]. Other graph approaches can be found in De Filippis et al. [33], Jun and D'Andrea [34], Helgasonet al. [36], McManus et al. [38], and Carlyle et al. [39]. Probabilistic approaches to obtain optimal paths for UAVs in the presence of threats have also been considered in the literature. The most common probabilistic method used is building a probability map of the area of operation [21, 22, 27, 28, 32-34]. Other probabilistic approaches include Rathbun et al. [2], Xinzeng et al. [25], Blackmore et al. [29], Pelosi et al. [35], Carlyle et al. [39], and Pfeiffer et al. [40]. Other research has considered deterministic approaches to obtain optimal paths for UAVs in the presence of threats [1, 18-20, 23, 36-38]. Ruiz et al. [24], Bortoff [26], Beard et al. [30], and Maddula et al. [31] assumed that the threat risk function is proportional to the Radar Cross Section (RCS) of the UAV and reciprocal to the fourth power of the distance from the UAV to the radar.

2.3 Contributions

In this section, we describe our contributions based on our review of the existing literature with respect to routing UAVs in a hostile area of operation. A very small number of papers in the literature studied multiple UAVs and multiple targets. In Krishna et al. [22], each UAV was assigned to exactly one target. In Richards et al. [1], multiple targets were assigned to a group of UAVs with time dependency and logical assignments constraints. In Beard et al. [30], 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. In Maddula et al. [31], targets were assigned to UAVs and divided equally among them. In our research, we consider multiple UAVs to 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 to 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 a branch-and-cut-and-price methodology with a simple path heuristic.

Some papers in the literature proposed algorithms that are suitable for real-time applications [1, 2, 21, 24, 26-29, 34, 38]. In this research, a pre-mission planner provides only a schedule, which can be used to initiate the mission. However, after the mission starts, and if it is not followed as planned, the schedule can be adjusted with the available online/onboard application.

Very few papers considered using a threat level based on all threats in the area of operation such as Beard et al. [30] and Maddula et al. [31]. Similarly, in our case, the threat exposure level from all threat points in the area of operation is considered. However, their approaches are different from ours. Both constructed a VORONOI diagram, a set of edges around the threat points, and both used the K-best shortest paths algorithm as a search tool. We do not construct a VORONOI

diagram, and we optimize routes with BCP methodology using a simple path heuristic. 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 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.

The remainder of this paper is structured as follows. Section 3 describes the model and algorithmic approaches, which include new waypoint generation methods and a new simple path heuristic. Although these models and approaches are developed for UAV routing in a hostile area, they could be used for other vehicle routing problems in which the vehicles must avoid certain obstacles. Section 4 provides computational results, and we discuss conclusions and future research in Section 5.

3. MODELS AND APPROACHES

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, so the threat level can be estimated *a priori*.
5. The ground speeds that a UAV will fly between two targets or waypoints is known *a priori*.
6. UAV routes originate and terminate at the same depot (or base).

3.2 Problem Formulation

We formulate a mixed integer linear program to solve the Unmanned Aerial Vehicle Routing Problem (UAVRP). The formulation is based on the VRP with set covering in (2.1)-(2.3). In UAVRP, we find optimal routes that maximize the total number of visited targets. We maintain the route travel time for a UAV and the total threat level for all UAVs to predetermined constant parameters. Furthermore, we generate waypoints that UAVs may visit when traveling from a target to another in order to reduce the threat level *a priori*.

Let U represent a set of UAV types located at the same base 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, $F = \bigcup_{u \in U} F_u$. Let K denote the set of all targets. Let t_{uf} be the level of exposure to threats for a UAV of type $u \in U$ 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 $f \in F_u$. The binary decision variable $x_{uf} = 1$ when a UAV of type $u \in U$ services a route $f \in F_u$. 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 $k \in K$ is visited. Although in our computational experiments we treat all targets as equally important, we can simply make targets have different priorities. The UAVRP formulation is as follows

$$\max \sum_{k \in K} B_k x_k \tag{3.1}$$

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

$$\sum_{f \in F_u} x_{uf}\ \leq\ n_u \qquad \forall u \in U \tag{3.3}$$

$$\sum_{u \in U} \sum_{f \in F_u} t_{uf}\ x_{uf}\ \leq\ d \tag{3.4}$$

$$x_{uf} \in \{0,1\} \qquad \forall\, u \in U, f \in F_u \tag{3.5}$$

$$x_k\ \in \{0,1\} \qquad \forall\, k \in K \tag{3.6}$$

The cost objective function in (3.1) maximizes the total number of visited targets for all UAVs. The right hand side of the set-covering constraints in (3.2) is only one when a target is visited. The constraints in (3.3) imply that at most n_u routes can be assigned to each UAV of type $u \in U$. In constraint (3.4), the positive constant parameter, d, limits the total threat level. Although constraint set (3.6) ensures that the target variables are binary, these variables can be relaxed to be continuous.

3.3 Modeling the Risk of Threat

The threat level is calculated deterministically using an approximation proposed by Beard et al. [30]. In addition to the set of targets K, let T be the set of all threats in the area of operation, and let W be the set of all waypoints. The network for the UAV routing is given by $G(V, E)$, where the set of nodes $V = K \cup W \cup \{v_0\}$, and E is the complete set of directed edges between pairs of nodes in $V \times V$. The threat for traveling an edge $e \in E$ 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, an approximation is used to calculate the threat 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 the approximation yields errors of less than two percent for nearby threats and in the order of a tenth of a percent for far away threats. In our implementation, the level of the risk of exposure to threats is calculated from all T sources of threats in the area of operation for each edge. For each edge $e \in E$, let τ_e be the length of the edge, and let $d_{1/6,e_p}, d_{1/2,e_p}, d_{5/6,e_p}$ be the distances from the selected three points on the edge e to each threat point $p \in T$, respectively. The threat level, t_e, for a UAV of type $u \in U$ for traveling an edge $e \in E$ is calculated as follows

$$t_e = \frac{\tau_e}{3} \sum_{p \in T} \left(\frac{1}{{d_{1/6,e_p}}^4} + \frac{1}{{d_{1/2,e_p}}^4} + \frac{1}{{d_{5/6,e_p}}^4} \right) \qquad \forall e \in E \tag{3.7}$$

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 edges in the route as

$$t_{uf} = \sum_{e \in f} t_e \qquad \forall f \in F_u, u \in U \tag{3.8}$$

3.4 Waypoint Generation Methods

We generate waypoints that a UAV may use in order to reduce the level of threat exposure by avoiding traveling over high-risk edges. Consider a high-threat level edge from target v_i to target v_j. Instead of traveling the edge, 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, the planner's preference or protocol must accept a certain threat exposure level. In the formulation, when the accepted level of threat exposure decreases, the UAV will use more waypoints and travel longer routes, which increases 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 a shorter mission time. In UAVRP, the waypoints generated before optimizing UAV routes substantially affect the total number of visited targets. Therefore, several waypoint generation methods are investigated.

3.4.1 Maximin

The *Maximin* method generates 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. [45]. 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. Figure 3-1 shows problem instances that include 5 UAVs, 10 targets, 30 threat points, and 55 waypoints generated using the *Maximin* method.

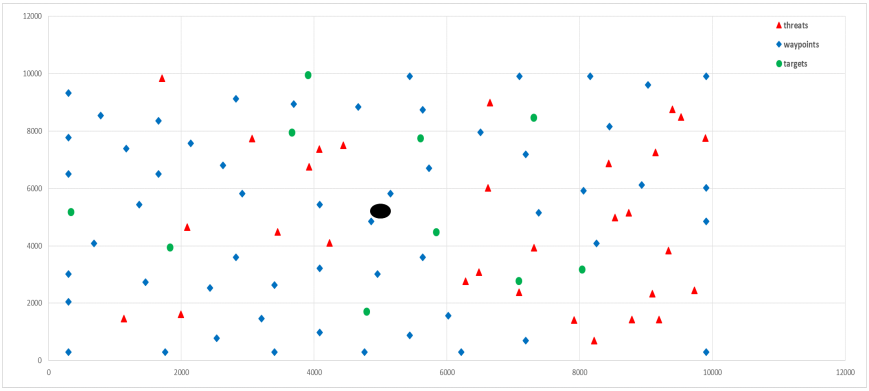

Figure 0-1 10 targets, 30 threat points, and 55 waypoints (*Maximin*)

3.4.2 Rectangular Threat Reduction

The *Rectangular Threat Reduction* method generates waypoints that reduce threat. In this method, a rectangle is placed between any two targets including the base. The total number of such rectangles is $\binom{|K|+1}{2}$. Then, a set of 100 grid

points is generated within a rectangle q formed by any pair of targets (or the base) (k_i, k_j) as the corners. Next, for target k_i, target k_j, and each grid point g in rectangle q, the threat levels, $t^q_{k_i,k_j}$, $t^q_{k_i,g}$, and t^q_{g,k_j}, (see Figure 3-2), are calculated from all the threats in the area of operation based on equation 3.7. Then, the threat reduction for each grid point g in a rectangle q is calculated as follows

$$Threat\ reduction\ (g,q) = t^q_{k_i,k_j} - \left(t^q_{k_i,g} + t^q_{g,k_j}\right) \tag{3.9}$$

For each pair (k_i, k_j), a grid point g with the highest threat reduction is added to the area of operation. Figure 3-3 shows problem instances that include 5 UAVs, 10 targets, 30 threat points, and 55 waypoints generated using the *Rectangular Threat Reduction* method.

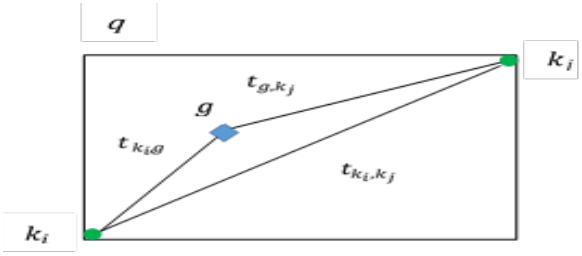

Figure 0-2 Threat Reduction for Grid Point g in a Rectangle q

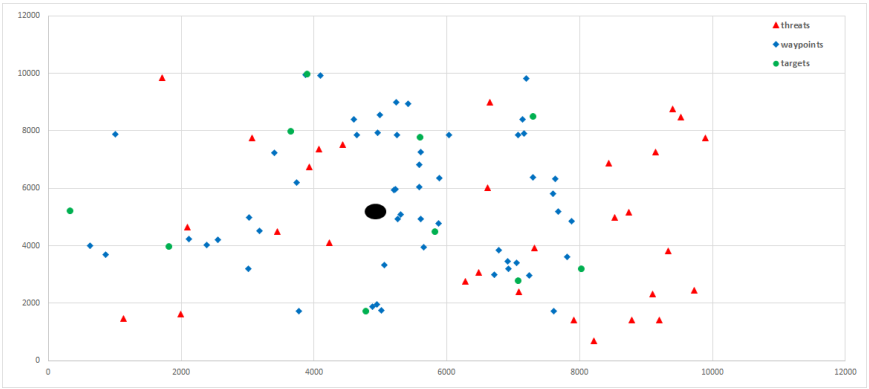

Figure 0-3 10 targets, 30 threats, and 55 waypoints (*Rectangular Threat Reduction*)

3.4.3 Maximin Reduced

The *Maximin* method, described in section 3.4.1, generates unimportant waypoints, which are unlikely to be visited by a UAV. Consequently, the *Maximin Reduced* method eliminates likely unimportant waypoints. After the *Maximin* method is done, a rectangle is placed between any two targets including the base as described in Section 3.4.2. 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 10 targets, 30 threat points, and 55 waypoints generated using *Maximin* method. After applying the *Maximin Reduced* method to this problem instance, the number of waypoints in the area of operation decreases from 55 waypoints down to only 29 waypoints as shown in Figure 3-4.

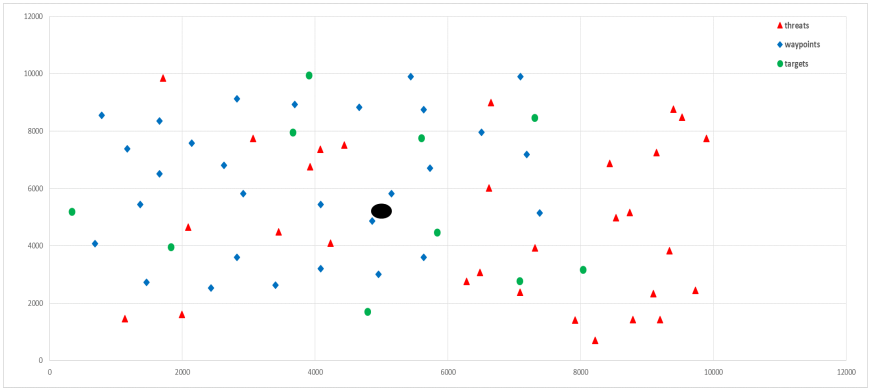

Figure 0-4 10 targets, 30 threats, and 29 waypoints (*Maximin Reduced*)

3.4.4 Maximin Threat Reduction

The *Maximin Threat Reduction* method removes waypoints from the *Maximin Reduced* method based upon their threat reduction. Specifically, after an initial set of waypoints is determined from the *Maximin Reduced* method, the total threat reduction for each waypoint is calculated based on the set of rectangles where the waypoint lies. This is done by placing a rectangle, q, between any two targets in the area of operation including the base as described in Section 3.4.2. Then, the threat reduction for each waypoint w inside a rectangle q is calculated based on equation 3.7. Next, for each of remaining waypoint w in the area of operation, the set of rectangles N_w^q in which it lies is determined. Then, the total threat reduction for each waypoint w based on the set of rectangles, N_w^q, that the waypoint lies in is calculated based on the following equation

$$Total\ Threat\ reduction\ (w) = \sum_{q \in N_w^q} Threat\ reduction\ (w, q) \tag{3.10}$$

An example of the total threat reduction for one waypoint lying in two rectangles is shown in Figure 3-5.

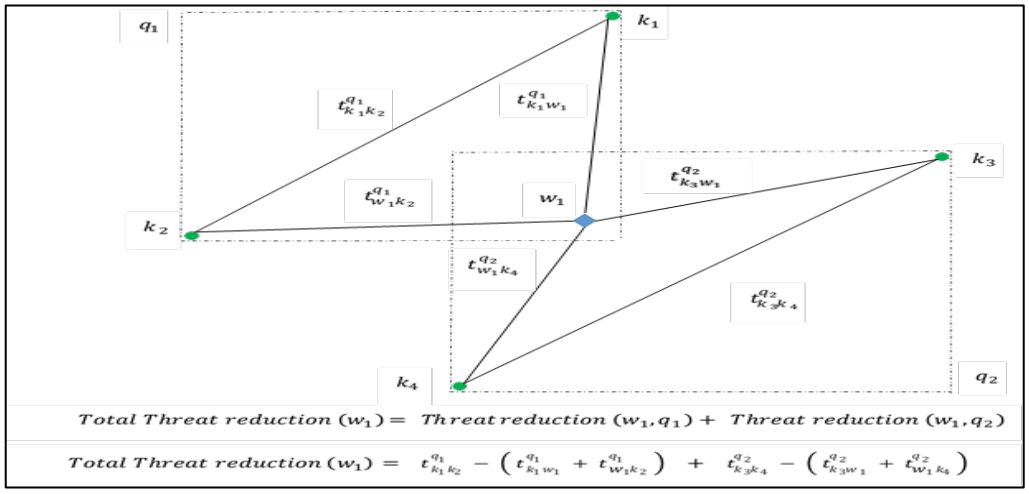

Figure 0-5 Total Threat Reduction of a Waypoint Lying in Two Rectangles

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.

3.5 Branch-and-Cut-and-Price Methodology

Even in a small-sized VRP, a large number of possible routes exist. The branch-and-cut-and price (BCP) methodology is an efficient method when dealing with problems with a large number of variables. In BCP, a restricted master problem (RMP) considers only a subset of all the routes $\bar{F} \subset F$. BCP methodology is used as a platform to solve our problem. To ensure we obtain an integer solution, our problem is solved within a branch-and-bound tree. First, the linear programing relaxation of the problem 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. 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. Afterwards, in the pricing step, new routes, variables, with negative reduced costs are generated using a 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 [43] branching logic. The BCP steps are repeated, and when there is no route needed to be added to the RMP, the steps are terminated, and an optimal integer solution x^* is found.

3.5.1 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 (3.4), is considered a binary knapsack constraint, since its coefficients are nonnegative constants. From (3.4), 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) and was reported in Nemhauser and Wolsey [41]. According to Kellerer et al. [46], the minimal dependent sets are also known as minimal cover inequalities, and defining facets for the knapsack polytope were originally studied in 1975 [47-49]. Consider the following feasible set for the binary knapsack problem

$$S = \left\{ x \in \{0,1\}^n \mid \sum_{j\in F} t_j x_j \le d \right\} \tag{3.11}$$

where, $F = \{1,2,\dots,n\}$, $t_j \in \mathbb{R}^1_+$, $\forall j \in F$, and $d \in \mathbb{R}^1_+$. Assume that the coefficients $t_j \in \mathbb{R}^1_+$, $\forall j \in F$, are ordered monotonically such that $t_1 \ge t_2 \ge \cdots \ge t_n$. Let $C \subset F$ such that $\sum_{j\in C} t_j > d$. 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 [41], the inequality $\sum_{j\in C} x_j \le |C| - 1$, where C is a dependent set, is a valid inequality for feasible set S. In our implementation, the feasible region for the binary knapsack constraint in (3.4) is

$$\hat{S} = \left\{ x \in \{0,1\}^{|F|} \mid \sum_{u\in U}\sum_{f\in F_u} t_{uf}\ x_{uf} \le d \right\} \tag{3.12}$$

Let binary constant $a_{Cuf} = 1$ if a variable $x_{uf} \in C$, and 0 otherwise. 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 $\hat{S}$

$$\sum_{u \in U} \sum_{f \in F_u} a_{Cuf}\ x_{uf}\ \leq |C| - 1 \qquad \forall C \in D \tag{3.13}$$

Consider the continuous relaxation for the feasible set of the knapsack constraint in (3.12)

$$\tilde{S} = \left\{ 0 \leq x \leq 1 \mid \sum_{u \in U} \sum_{f \in F_u} t_{uf}\ x_{uf}\ \leq\ d \right\} \tag{3.14}$$

At least one fractional solution $\tilde{x} \in \tilde{S}$ violates (3.13). The proof presented here can be found in Nemhauser and Wolsey [41]. 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 form:

$$\tilde{x}_j = 1 \qquad \text{if } j \in C \backslash \{k\}, \tag{3.15}$$

$$\tilde{x}_j = 0 \qquad \text{if } j \in F \backslash C, \tag{3.16}$$

$$\tilde{x}_j = \frac{d - \sum_{j \in C \backslash \{k\}} t_j}{t_k} \qquad \text{if } j = k. \tag{3.17}$$

From the last fractional form in (3.17), it is clear that

$$\sum_{j \in C} \tilde{x}_j\ = |C| - 1 + \frac{d - \sum_{j \in C \backslash \{k\}} t_j}{t_k}\ >\ |C| - 1 \tag{3.18}$$

Therefore, $\exists \tilde{x} \in \tilde{S}$ that does not satisfy the valid inequalities in (3.13), so the MDS constraints in (3.13) cut off the fractional solution $\tilde{x}$. Notice that the integer coefficients of the MDS constraints in (3.13) should also encourage solution integrality.

3.5.2 Cut Generation

At each node of the tree, after the linear programming relaxation of the problem is solved to obtain a solution x^*, a cut generation step is called in order to try to generate an MDS constraint. If an MDS constraint 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 similar to solving the linear programming relaxation of a binary knapsack problem, Chvátal [44]. Consider the following integer programing problem

$$\min \sum_{u \in U} \sum_{f \in F_u} (1 - x^*_{uf})\ \tilde{x}_{uf} \tag{3.19}$$

$$s.t. \sum_{u \in U} \sum_{f \in F_u} t_{uf}\ \tilde{x}_{uf}\ \geq\ d + \varepsilon \tag{3.20}$$

$$\tilde{x}_{uf} \in \{0,1\}, \qquad \forall\, u \in U, f \in F_u \tag{3.21}$$

A feasible solution to the problem in (3.19)-(3.21) 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 $|\bar{F}|$ be the current total number of variables, x_{uf}, that are already in the RMP. Let i in a variable x_{uf}^{i} represent the i^{th} order of the sorted sequence $i = 1,2, \ldots |\bar{F}|$. Let set C be a minimum dependent set. Beginning from $i = 1$, variable x_{uf}^{i} 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 $\boldsymbol{i = 1}$.

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

Check:

if $\boldsymbol{i \leq |\bar{F}|}$ then

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

    Return the MDS $\boldsymbol{C}$.

  else

    Return to Selection.

  end if

else

  Return $\emptyset$.

end if

3.5.3 Reduced Cost

The LP relaxation problem of the UAVRP after relaxing the integrality requirement and adding the MDS constraints is as follows

$$\max \sum_{k \in K} B_k x_k \tag{3.22}$$

$$s.t. \sum_{u \in U} \sum_{f \in F_u} a_{kuf}\ x_{uf} \geq x_k \qquad \forall k \in K \tag{3.23}$$

$$\sum_{f \in F_u} x_{uf} \leq n_u \qquad \forall u \in U \tag{3.24}$$

$$\sum_{u \in U} \sum_{f \in F_u} t_{uf}\ x_{uf} \leq d \tag{3.25}$$

$$\sum_{u \in U} \sum_{f \in F_u} a_{Cuf}\ x_{uf} \leq |C| - 1 \qquad \forall C \in D \tag{3.26}$$

$$0 \leq x_k \leq 1 \qquad \forall k \in K \tag{3.27}$$

$$0 \leq x_{uf} \qquad \forall u \in U, f \in F_u \tag{3.28}$$

Let the dual variable for the target constraint in (3.23) be represented by $\pi_{k(e)}$, and let $\pi_{k(e)}^{*}$ be the value of this dual variable if the tail of the edge e is a target k; otherwise $\pi_{k(e)}^{*}$ is zero. In addition, let the dual variables for the constraints

in (3.24), (3.25), and (3.26) be represented by π_u, λ, and ρ_C, respectively. Finally, let x^*, $\pi^*_{k(e)}$, π^*_u, λ^*, and ρ^*_C be an obtained optimal solution after solving the LP relaxation problem (3.22)-(3.28). Therefore, the reduced cost $\bar{c}_{uf}$ for variable x_{uf} can be calculated as follows

$$\bar{c}_{uf} = -\sum_{k\in f} a_{kuf}\,\pi^*_{k(e)} - \pi^*_u - t_{uf}\,\lambda^* - \sum_{C\in D} a_{Cuf}\,\rho^*_C,$$
$$\forall\, f \in F_u, \forall\, u \in U \tag{3.29}$$

Using the equation in (3.8), the reduced cost $\bar{c}_{uf}$ for a variable x_{uf} can be rewritten as a edge-based reduced cost as

$$\bar{c}_{uf} = -\sum_{k\in f} \pi^*_{k(e)} - \pi^*_u - \sum_{e\in f} t_e\,\lambda^* - \sum_{C\in D} a_{Cuf}\,\rho^*_C,$$
$$\forall\, f \in F_u, \forall\, u \in U \tag{3.30}$$

Based on (3.30), we can say that the term $(-t_e\,\lambda^* - \pi^*_{k(e)})$ represents the cost of edge $e \in E$ in a graph G. In addition, the term $(-\pi^*_u)$ represents the cost of using a UAV $u \in U$. Finally, the cost of existing in a minimum dependent set $C \in D$ is represented by the term (ρ^*_C), and the total cost of existing MDS costraints for the route f is $\sum_{C\in D} a_{Cuf}\,\rho^*_C$.

3.5.4 Column Generation

After the RMP is solved at a node of the tree using only a subset $\bar{F} \subseteq F$, and the algorithm is done generating MDS constraints, the dual solution is sent to the column generation subproblem in order to generate new routes so they can be added to the RMP. For UAVRP, the column generation step must find a simple path in G with negative reduced cost that originates and terminates at the base v_0 in order to add it to the RMP. The integer programming shortest path algorithm (IPSP) [41] is used as our column generation algorithm engine to find a simple path with negative reduced cost.

In addition to the network $G(V,E)$, we split the base node in a source node v_0 and a terminal node v_{n+1}. For each edge e in E, let y_e be a binary variable in which $y_e = 1$ indicates that edge e is in the generated route, and let the cost of selecting edge e be $c_e = -\pi^*_{k(e)} - t_e\,\lambda^*$. 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 edges $e \in f^*$. That is $\sum_{e\in f^*}\{-\pi^*_{k(e)} - t_e\,\lambda^*\}$ from (3.30). 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. Note that since the column generation subproblem finds new routes after generating MDS constraints, the ρ^*_C term in (3.30) is not applicable. If the path f^* has a negative reduced cost, $\bar{c}_{uf^*} < 0$, then it is added to the RMP.

In addition, each route must be completed within certain predetermined amount of time r, since a prolonged missions may alert an enemy. For each edge e in E, let s_e be the ground speed, and let m_e be the travel time over edge e, which is calculated as

$$m_e = \frac{\tau_e}{s_e} \tag{3.31}$$

Let $\delta^+_{v_i} = \{j: (i,j) \in E\}$ be the set of all edges that are departing from node v_i, $\delta^-_{v_i} = \{j: (j,i) \in E\}$ 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 $\bar{F} \subset F$ denote 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 UAVRP, the formulation of the subproblem is as follows

$$\min \sum_{e \in E} \{-\pi^*_{k(e)} - t_e \, \lambda^*\} \, y_e \tag{3.32}$$

$$s.t. \sum_{e \in \delta^+_{v_i}} y_e - \sum_{e \in \delta^-_{v_i}} y_e \; = \; b(v_i) \qquad \forall v_i \in V \tag{3.33}$$

$$\sum_{e \in \delta^+_{v_i}} y_e \leq 1 \qquad \forall v_i \in V \tag{3.34}$$

$$\sum_{e \in f} y_e \leq \; |\Omega(f)| \qquad \forall f \in \bar{F} \tag{3.35}$$

$$\sum_{e \in \psi} y_e \; \leq \; h(\psi) \qquad \forall \psi \in \mathcal{H} \tag{3.36}$$

$$\sum_{e \in f} m_e \, y_e \leq r \tag{3.37}$$

$$y_e \in \{0,1\} \qquad \forall e \in E \tag{3.38}$$

The objective function in (3.32) minimizes the reduced cost in UAVRP. The flow balance constraints in set (3.33) 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 (3.34) limit the outflow for each node to be at most one. Both the set in (3.33) and (3.34) can help in preventing some cycles from forming. The route elimination constraints in set (3.35) prevent previously generated routes already included in the RMP from being generated in future iterations. The left-hand side represents the edges of the route that starts from the base, visits some targets, and returns to the base. 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 (3.35) is added a *priori* to solving the subproblem since $\bar{F}$ is a finite set. The invalid walks, $\psi \in \mathcal{H}$, are eliminated by the walk or cycle elimination constraints in set (3.36), where $h(\psi) = |\Omega(f)| - 1$. The set (3.36) is added dynamically to the subproblem. In constraint (3.37), the nonnegative constant parameter, r, limits the route travel time for a UAV. The constraints in set (3.38) require the decision variables y_e to be binary, which indicates edges selected in the route f^*.

Notice that the edge 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 negative cost cycle is shown in Figure 3-6. However, the cost of the simple path could be positive or negative.

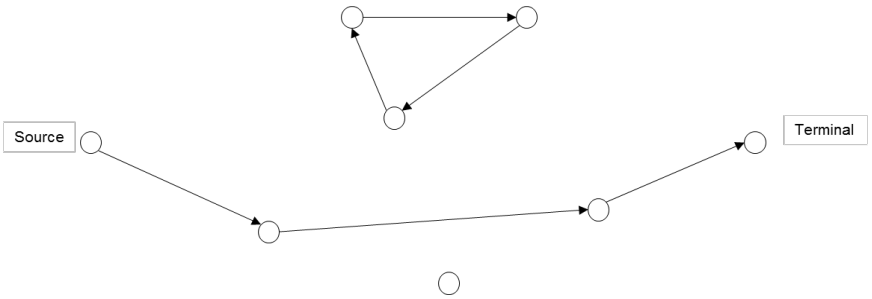

Figure 0-6 A Simple Path with Negative Cost Cycle

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. We considered the following four scenarios

1. If we find a simple path with a negative reduced cost, and do not find a cycle, the simple path will be added to 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. 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, sequentially adding cycle elimination constraints is time consuming. Therefore, a simple path heuristic is discussed next.

3.5.5 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 a cycle elimination constraint 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. 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 3-6. 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.

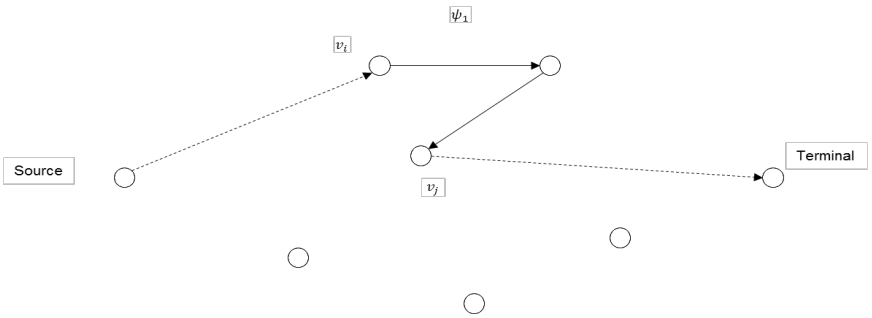

Figure 0-7 Generating Simple Path from a Cycle Using Simple Path Heuristic

In each iteration of the heuristic, an edge e 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 3-7. When the edge between v_i and v_j is deleted, two new edges 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 (3.35) or (3.37).

3.5.6 Branching Logic

We use the follow-on branching, which was developed by Vance et al. [42] and is a variant of Ryan and Foster branching [43], 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 the space, which prevents the same solution from occurring on different branches. It also requires a clear definition of a solution in the leaves of the branch-and-bound tree. 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 with either a or b 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 deleted. 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 deleted, while all other departing edges from target a and and all other entering edges to target b will be kept.

4. COMPUTATIONAL STUDY

A computational study to solve the UAVRP with BCP methodology was done. The UAVRP was implemented using the BCP framework that was coded in C++ using 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 solver interface with COIN-OR. A 2.67 GHz Intel Xeon computer was used to conduct the study.

We experimented with two different algorithms. The first one considers only the Delayed Column Generation algorithm with the simple path heuristic, referred to as the DCG. The second one considers DCG with the Minimum Dependent Set heuristic, referred to as the DCG-MDS. In other experiments, we considered algorithms without the simple path heuristic, but they require very long times to generate routes.

We created different sets of problem instances with 10 targets and 20 targets. With each case, 5 UAVs are available. The UAVs originate and terminate at the same base, which is located at the center of the area of operation. The targets and threat points are randomly generated. In this computational study, 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, allowing only lower threat levels increases the use of waypoints, hence, encourages longer routes as well. However, allowing less travel time encourages a shorter route.

4.1 Results for Ten-Target Case

At the beginning, we obtained the minimum route travel time for visiting all the 10 targets with only one UAV, which is 280.31 hours. Then, this value is used to represent the maximum value for the predetermined maximum route travel time parameter r in (3.37) in order to limit the route travel time for a UAV in our sub problem. Setting a value lower than this maximum limit for the route travel time will definitely encourage the use of more UAVs, since no one UAV can visit all 10 targets alone. In addition, 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 corresponding to the minimum route travel time for visiting all the 10 targets with only one UAV, which is 6.58E-07. We limit the total threat level for all UAVs, but this value was obtained for a single UAV. 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, based on a set of problem instances that includes 10 targets, 30 threat points, and 55 waypoints, visiting all targets and waypoints in the area of operation yields a total threat level value equal to 9.67E-04, and visiting only the closest target yields a threat level value equal to 1.60E-09. In order to cover a range of threat levels, the 6.58E-07 is only used as a reference level to obtain more levels of the predetermined constant parameter d in (3.4). The experiment includes two levels above the reference threat level value by multiplying by 10 and 10^2 and two levels below the reference threat level value by multiplying by 10^{-1} and 10^{-2}, so the four levels of d considered are 6.58E-06, 6.58E-07, 6.58E-08, and 6.58E-09.

We experimented with four waypoints generation methods, *Rectangular Threat Reduction, Maximin, Maximin Reduced,* and *Maximin Threat Reduction.* At the beginning, we experimented with only the *Rectangular Threat Reduction* method and the *Maximin* method. Since *Rectangular Threat Reduction* method is based on placing a rectangle on any two targets including the base, we have a total of 55 rectangles, $\binom{11}{2}$, and 55 waypoints. For the sake of a fair comparison between the two waypoint generation methods, we create 2 sets of problem instances. Both instances include 10 targets, 30 threat points, and 55 waypoints, shown in Figure 4-1. The 55 waypoints in the first set (left) are generated using the *Rectangular Threat Reduction* method, and the 55 waypoints in the second set (right) are generated using the *Maximin* method.

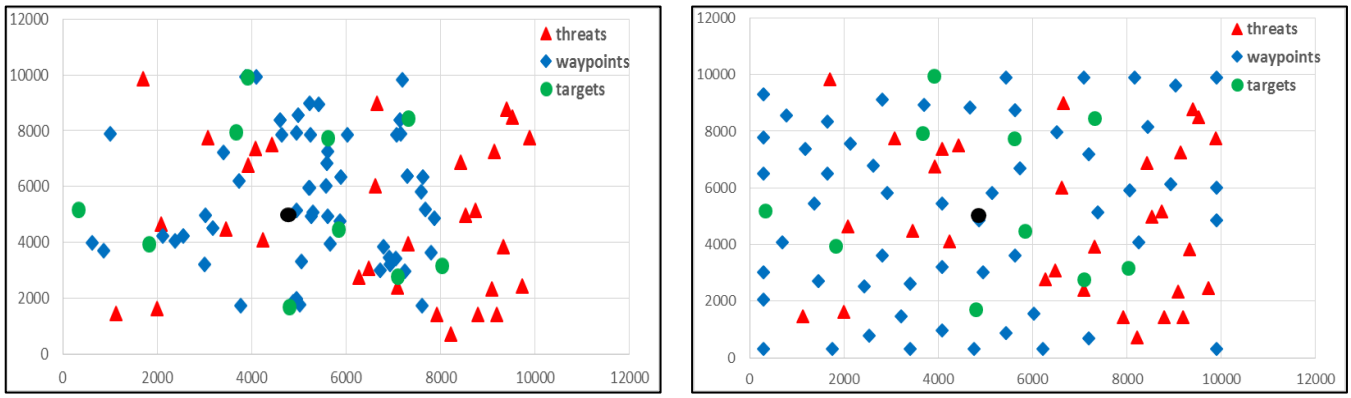

Figure 0-1 10 Targets, 30 Threats, and 55 Waypoints, using *Rectangular Threat Reduction* (Left) and *Maximin* (Right)

We considered the DCG algorithm and the DCG-MDS algorithm to solve each set of the problem instances. In an attempt to find solution results in a reasonable time, 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 responds 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 and DCG-MDS algorithms are presented in Table 4-1 and Table 4-2, respectively. Each table includes 16 executions for each of the four aforementioned levels for the constant parameters r and d, which represent the limits for the travel time for each UAV and the total threat level for all UAVs. For each run, the results we obtained include the maximum total number of visited targets, our objective function, the number of UAVs used, the CPU time (in seconds) until the best-known solution is found, the CPU time (in seconds) until the algorithm finished, the number of generated variables until the best-known solution is found, and the number of generated variables until the algorithm finished. However, what is shown here in Table 4-1 and 4-2, are only the maximum total number of visited targets and the CPU time until the algorithm finished. For example, for the run at $r_4 = 224.25$ and $d_4 = 6.58E-06$ in Table 4-2, the maximum total number of visited targets is 10, and the CPU time until the algorithm finished is 16.56 seconds. We can see short or high solution times for some cases and no solution times for others in response to decreasing or increasing the route travel time r, the total threat level d, or both. Generally, when both r and d are restricted, the algorithm becomes inefficient in finding a solution within the 4-hour run time limit. However, as both r and d increase, the algorithm becomes more efficient because it can easily determine a path that meets the constraints and reaches all ten targets.

Table 0-1 Results for DCG Algorithm for 10 Targets, 30 Threat Points, and 55 Waypoints (*Rectangular Threat Reduction*)

Total Number of Visited Targets							
		r, Maximum Route Travel Time					
		MIN	1	2	3	4	MAX
Maximum Total Threat	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					

CPU Time until the Algorithm Terminated

		r, Maximum Route Travel Time					
		MIN	1	2	3	4	MAX
d, Maximum 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					

Table 0-2 Results for DCG Algorithm for 10 Targets, 30 Threat Points, and 55 Waypoints (*Maximin*)

Total Number of Visited Targets							
		r, Maximum Route Travel Time					
		MIN	1	2	3	4	MAX
d, Maximum 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	2	6	10	
	4	6.58E-06	1	10	10	10	
	MAX	6.58E-05					
CPU Time until the Algorithm Terminated							
		r, Maximum Route Travel Time					
		MIN	1	2	3	4	MAX
d, Maximum 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	Time Ran Out	428.25	
	4	6.58E-06	2212.12	Time Ran Out	12064.45	16.56	
	MAX	6.58E-05					

For DCG and for both waypoint generation methods, *Rectangular Threat Reduction* and *Maximin*, 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 r and d decrease, the total number of visited targets also decreases. Similarly, as both r and d increase, the total number of visited targets also increases.

For the DCG algorithm, the waypoint generation using the *Maximin* method performs better than the *Rectangular Threat Reduction* method. Overall, a total of 54 targets were visited for all of the 16 runs using the *Maximin* compared to a total of 35 targets for the waypoint generation using the *Rectangular Threat Reduction* method. In terms of achieving the mission, with the *Maximin*, all the available 10 targets are visited in 4 runs, which are at $(r_4 = 224.25,\ d_4 = 6.58E-06)$, $(r_3 = 168.18,\ d_4 = 6.58E-06)$, $(r_2 = 112.12,\ d_4 = 6.58E-06)$, and $(r_4 = 224.25,\ d_3 = 6.58E-07)$, while none of the runs with the *Rectangular Threat Reduction* method were able to visit all available targets. Notice that with the *Maximin*, the algorithm terminated before completing the 4-hour run time limit during 7 runs, while, none of the runs with the *Rectangular Threat Reduction* were able to finish before reaching the 4-hour run time limit. This is an indication of how the waypoint generation method used before optimizing UAV routes can substantially affect the algorithm performance in finding a solution within the 4-hour run time limit when both the route travel time r and the total threat level d are restricted. The results for the two-waypoint generation methods for the DCG-MDS algorithm are presented in Table 4-3 and Table 4-4, respectively.

Table 0-3 Results for DCG-MDS Algorithm for 10 Targets, 30 Threat Points, and 55 Waypoints (*Rectangular Threat Reduction)*

Number of Visited Targets							
		r, Maximum Route Travel Time					
		MIN	1	2	3	4	MAX
d, Maximum 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	4	0	
	3	6.58E-07	1	3	4	4	
	4	6.58E-06	1	3	4	9	
	MAX	6.58E-05					
CPU Time until the Algorithm Terminated							
		r, Maximum Route Travel Time					
		MIN	1	2	3	4	MAX
d, Maximum 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					

Table 0-4 Results for DCG-MDS Algorithm for 10 Targets, 30 Threat Points, and 55 Waypoints (*Maximin*)

Total Number of Visited Targets							
		r, Maximum Route Travel Time					
		MIN	1	2	3	4	MAX
d, Maximum 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	4	1	
	3	6.58E-07	1	2	10	10	
	4	6.58E-06	1	10	8	10	
	MAX	6.58E-05					
CPU Time until the Algorithm Terminated							
		r, Maximum Route Travel Time					
		MIN	1	2	3	4	MAX
d, Maximum 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					

For DCG-MDS and for both waypoint generation methods, the waypoint generation using *Maximin* obviously performs better than the *Rectangular Threat Reduction* method. Overall, a total of 60 targets are visited for all of the 16 runs using *Maximin* compared to a total of 36 targets for the waypoint generation using the *Rectangular Threat Reduction* method. In terms of achieving the mission, with *Maximin*, all the available 10 targets are visited in 4 runs, while none of the runs with

the *Rectangular Threat Reduction* method are able to visit all available targets. With the *Maximin*, the algorithm terminated before completing the 4-hour run time limit in 7 runs, while none of the runs with the *Rectangular Threat Reduction* method were able to finish before reaching the 4-hour run time limit.

Based on the previous discussion, we can conclude that the waypoint generation using the *Maximin* method gives better results than the waypoint generation using the *Rectangular Threat Reduction* method for both DCG and DCG-MDS algorithms. As a result, *Maximin* is investigated further. Comparing both algorithms, DCG and DCG-MDS, for only the 55 waypoints generated using the *Maximin* method, DCG-MDS performs better, visiting a total of 60 targets, than DCG does with the same number of waypoints, visiting only 54 targets.

The problem we optimize is a computationally hard one. We generated 55 waypoints, using *Maximin*, 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 fewer waypoints, the computational complexity of the problem exponentially decreases, and the CPU time spent for optimizing the sub problem, detecting cycles in the solution, and generating new routes reduces. As a result, we expect to obtain even better results with fewer waypoints than 55 waypoints.

In order to improve the results for both DCG and the DCG-MDS that were previously obtained for the 55 waypoints generated using *Maximin*, we used the *Maximin Reduced* method in order to remove unnecessary waypoints, which are unlikely to be visited by a UAV. Initially, the process removes any waypoint that lies outside of the 55 rectangles. Consequently, the number of waypoints in the area of operation decreases from 55 waypoints to only 29 waypoints. Now, the area of operation, shown in Figure 4-2 (upper left), includes 5 UAVs located at the base, 10 targets, 30 threat points, and the remaining 29 waypoints.

In the quest to obtain even better results, we used the *Maximin Threat Reduction* method 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 base. That is a total of 55 rectangles, $\binom{11}{2}$. Then, the process determines total threat reduction for each waypoint for all rectangles it lies inside. Sorting the 29 waypoints based on their total threat reduction from highest to lowest, we half the number of *a priori* waypoints. Specifically, we begin with the 29 waypoints shown in Figure 4-2 (upper left). Then, the following three sets of waypoints are generated based upon the highest total threat reduction: 15 waypoints, (Figure 4-2 upper right), 8 waypoints (Figure 4-2 lower left), and 4 waypoints (Figure 4-2 lower right).

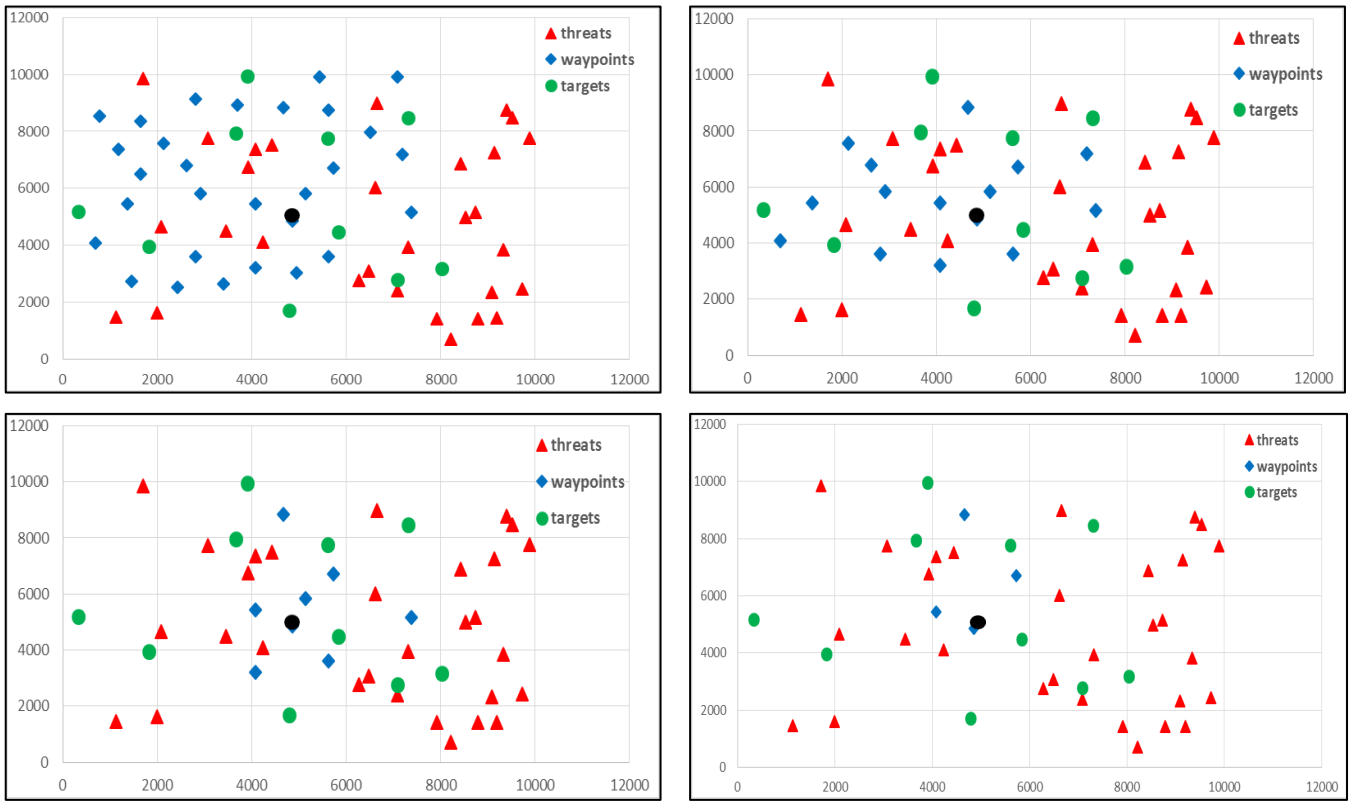

Figure 0-2 10 Targets, 30 Threat points, and 29 Waypoints using *Maximin Reduced* (Upper Left), 15 Waypoints using *Maximin Threat Reduction* (Upper Right), 8 Waypoints using *Maximin Threat Reduction* (Lower Left), and 4 Waypoints using *Maximin Threat Reduction* (Lower Right).

The overall results for all 16 runs for 55, 29, 15, 8, and 4 waypoints for both algorithms, DCG and DCG-MDS, are presented in Table 4-5.

Table 0-5 Overall Results for 10-Target Case

Overall (16 Runs)	55 Waypoints (Maximin)		29 Waypoints (Maximin Reduced)		15 Waypoints (Maximin Threat Reduction)		8 Waypoints (Maximin Threat Reduction)		4 Waypoints (Maximin Threat Reduction)	
Item	DCG	DCG-MDS	DCG	DCG-MDS	DCG	DCG-MDS	DCG	DCG-MDS	DCG	DCG-MDS
Total Number of Visited Targets	54	60	64	62	88	82	89	89	89	89
Achieving Mission (Visiting all 10 Targets) (# Runs)	4	4	5	3	6	6	6	6	6	6
Number of UAVs Used	30	32	35	30	41	35	42	42	40	40
Terminating before Completing the 4-Hour Run Time (# Runs)	7	7	9	7	10	10	10	10	12	12
Time until the Best Solution is Found	14,430	2,646	3,709	10,897	32,126	24,147	6,813	4,416	4,436	2,720
Number of Generated Variables until Best Known Solution Found	116	140	126	121	1,581	1,055	2,028	1,489	1,925	968
Number of Generated Variables until Algorithm Terminated	246	272	250	234	13,510	10,431	25,419	26,669	22,974	22,715
Number of MDS Cuts until Best Known Solution Found	–	24	–	14	–	207	–	46	–	70
Number of MDS Cuts until Algorithm Terminated	–	30	–	16	–	689	–	1,597	–	2,522

Comparing the results using the five sets of waypoints, the 55 from *Maximin*, the 29 from *Maximin Reduced*, and the 15, 8, and 4 from *Maximin Threat Reduction*, both DCG and DCG-MDS perform better with fewer waypoints. The total number of visited targets increased as the number of waypoints decreases in all but the 8-waypoint and 4-waypoint cases, in which the number of visited targets are the same. However, both algorithms perform slightly better with 4 waypoints than with 8 waypoints because they terminate before completing the 4-hour run time in 12 runs with 4 waypoints compared to only 10 runs with 8 waypoints.

Comparing DCG with DCG-MDS yields more mixed results. As noted previously, with 55 waypoints, solutions found using DCG-MDS visits more targets than those of DCG. However with 15 and 29 waypoints, DCG finds solutions visiting more targets than those of DCG-MDS. For only 4 and 8 waypoints, no clear winner algorithm emerges, as both obtained the same solution that visits 89 targets overall. In addition, both algorithms terminate before completing the 4-hour run time in the same number of runs. However, DCG-MDS reaches the solution with a fewer number of generated variables than the DCG algorithm does with the same number of waypoints. In general, DCG-MDS uses the Minimum Dependent Set constraints, which cut some fractional solutions and encourage the solution integrality, while, DCG does not. If the problem is highly fractional, DCG-MDS spends most of the CPU time in the master problem and less time generating routes in the sub problem. As a result, DCG-MDS is expected to generate fewer variables than DCG does to find an optimal solution. Consequently, DCG-MDS with 4 and 8 waypoints finds an optimal solution faster than the DCG algorithm does with the

same number of waypoints. Nonetheless, for the 10-targets case and for the same number of waypoints, neither DCG nor DCG-MDS is guaranteed to perform better than the other.

In addition, we tested both algorithms with fewer than 4 waypoints and found that the number of visited targets descreases. Therefore, based on the previous results and discussions, we can conclude that using *Maximin Threat Recution* with 4 waypoints is considered the right number of *a priori* waypoints for the 10-target case, yielding the best results witihin a 4-hour run time limit.

4.2 Results for Twenty-Target Case

In the 20-target case, the minimum route travel time for visiting all 20 targets with only one UAV is 321.2 hours, and the corresponding total threat level is 1.22E-05. Consequently, the considered four levels of r are 256.95, 192.71, 128.47, and 64.24 hours, and the four levels of d are 1.22E-04, 1.22E-05, 1.22E-06, and 1.22E-07. Only the methods that are based on the *Maximin* method are used to generate waypoints for the 20-target case, since the *Rectangular Threat Reduction* method performed so poorly for the 10-taget case. First, we generated 210, $\binom{21}{2}$, waypoints using *Maximin*. Figure 4-3 (upper left) shows a set of problem instances that includes 5 UAVs located at the base, 20 targets, 60 threat points, and the 210 waypoints. *Maximin Reduced* decreased the number of waypoints in the area of operation from 210 waypoints to only 148 waypoints (Figure 4-3 upper right). *Maximin Threat Reduction* subsequently selected 40 waypoints, in terms of total threat reduction (Figure 4-3 middle left), 20 waypoints (Figure 4-3 middle right), and 9 waypoints (Figure 4-3 bottom). Overall results for all 16 runs with 40, 20, and 9 waypoints for both algorithms are presented in Table 4-6.

Like the 10-target case, the number of visited targets increases as the number of waypoints decreases. In other experiments, we tested both algorithms with fewer than 9 waypoints and found that the maximum number of visited targets decreases. Consequently, we can conclude that the 9 waypoints is considered the right number of *a priori* waypoints for the 20-target case. Observe that the ratios of waypoints-to-targets in the best solutions for both problem instances are very similar, at 4-to-10 and 9-to-20. Comparing DCG versus DCG-MDS, DCG-MDS performs better with 40 waypoints and 20 waypoints, but DCG performs better with 9 waypoints. Like the 10-target case, neither DCG nor DCG-MDS superior to the other for the 20-targets case. However, in both cases, DCG-MDS outperforms DCG with more waypoints, suggesting that it may be superior with largers problems.

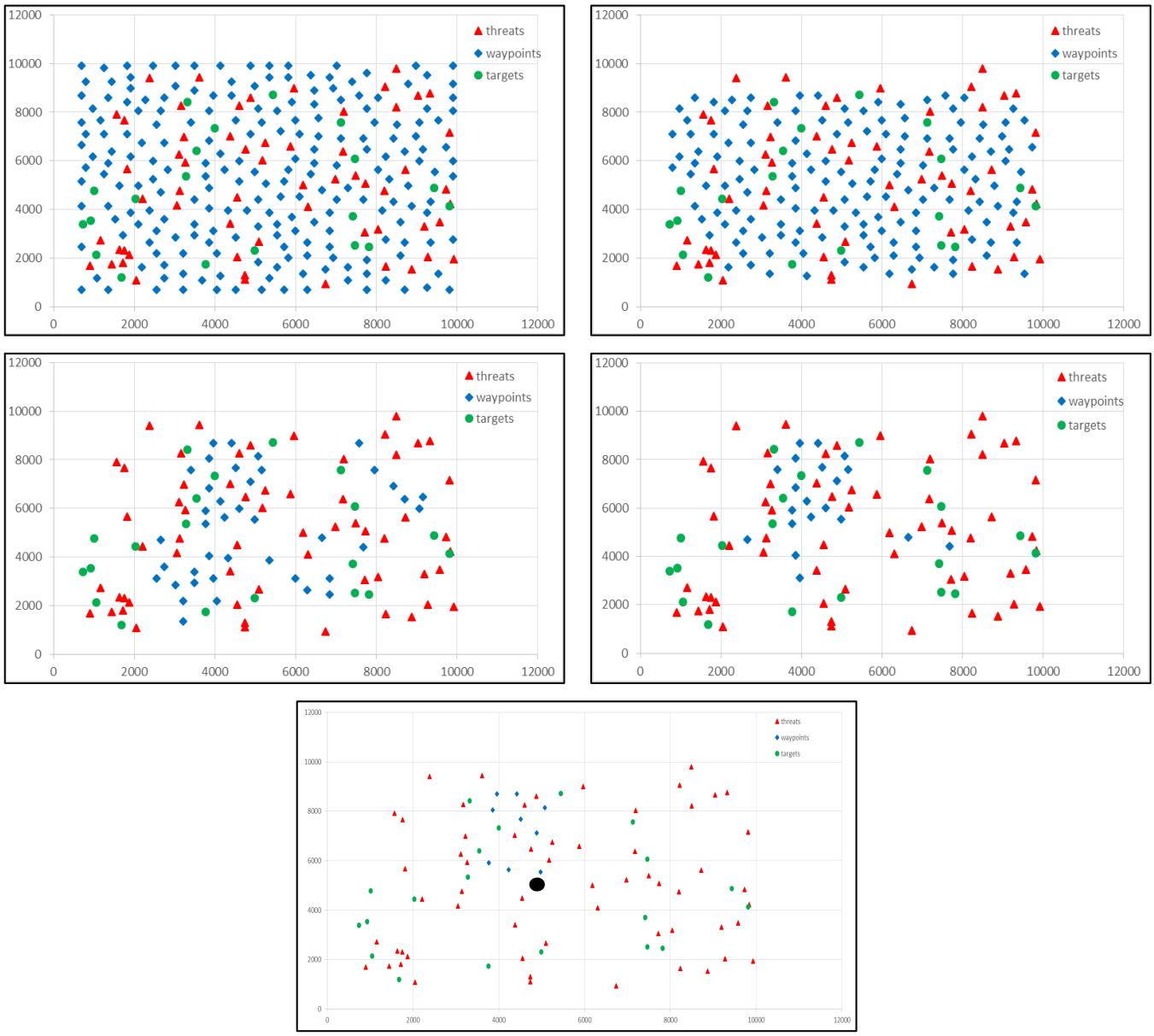

Figure 0-3 20 Targets, 60 Threat Points, and 210 Waypoints Generated using *Maximin* (Upper Left), 148 Waypoints using *Maximin Reduced* (Upper Right), 40 Waypoints using *Maximin Threat Reduction* (Middle Left), 20 Waypoints using *Maximin Threat Reduction* (Middle Right), and 9 Waypoints using *Maximin Threat Reduction* (Bottom)

Table 0-6 Overall Results for 20-Target Case

Overall (16 Runs)	40 Waypoints (Maximin Threat Reduction)		20 Waypoints (Maximin Threat Reduction)		9 Waypoints (Maximin Threat Reduction)	
Item	DCG	DCG-MDS	DCG	DCG-MDS	DCG	DCG-MDS
Total Number of Visited Targets	106	123	121	127	156	148
Achieving Mission (Visiting all 20 Targets) (# Runs)	0	0	3	4	4	3
Number of UAVs Used	33	37	39	40	50	48
Terminating before Completing the 4-Hour Run Time (# Runs)	0	0	2	2	8	7
Time until the Best Solution is Found	26,662	25,977	47,883	56,271	18,276	29,762
Number of Generated Variables until Best Known Solution Found	155	176	172	216	360	318
Number of Generated Variables until Algorithm Terminated	355	354	577	608	796	850
Number of MDS Cuts until Best Known Solution Found	–	16	–	16	–	19
Number of MDS Cuts until Algorithm Terminated	–	19	–	26	–	90

5. CONCLUSIONS AND FUTURE RESEARCH

In this research, we formulated a mixed integer linear program for routing the UAVs in the presence of threats. We did a computational study for the UAVRP and presented results for both the small-sized problem, 10 targets, and the medium-sized problem, 20 targets. Results for both the small-sized problem and the medium-sized problem show that fewer waypoints gives better results for both algorithms, DCG and DCG-MDS, based on the 4-hour run time limit. The right number of *a priori* waypoints for the small-sized problem, 4 waypoints for 10 targets and 9 waypoints for 20 targets. Both of these sets of waypoints were found using *Maximum Threat Reduction*. In addition, for both the small-sized and medium-sized problems and for using the same number of waypoints, neither DCG nor DCG-MDS performs better than the other. However, DCG-MDS performs better with more waypoints, suggesting that it may perform better with larger problems.

The problem we optimize in this research requires a very long run time to obtain solutions. Consequently, one topic of future research includes improving the CPU time. In this research we generate waypoints and add them all together *a priori*. Instead, we will 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. In addition, one way to reduce problem size is by putting a threshold on the threat level of the edges, 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 this research, we treated all targets with equal importance. We will test the UAVRP with targets that have different priorities by varying their benefit values.

Acknowledgements

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

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

[42] 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.

[43] 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.

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

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

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

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

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

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