AIRCRAFT ROUTE OPTIMIZATION USING THE A-STAR ALGORITHM

THESIS

Garret D. Fett, Major, USA

AFIT-ENS-14-M-06

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-14-M-06

AIRCRAFT ROUTE OPTIMIZATION USING THE A-STAR ALGORITHM

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Garret D. Fett, BS

Major, USA

March 2014

AFIT-ENS-14-M-06

AIRCRAFT ROUTE OPTIMIZATION USING THE A-STAR ALGORITHM

Garret D. Fett, BS

Major, USA

Approved:

____//signed//_________________________ 6 Mar 2014
Raymond R. Hill, PhD (Chairman) Date

____//signed//_________________________ 7 Mar 2014
Sarah G. Nurre, PhD (Member) Date

AFIT-ENS-14-M-06

Abstract

This research develops an Aviation Distance Estimation and Route Planning Tool (ADERPT) that finds least-cost aircraft routing from a designated departure airfield to an arrival airfield for the purposes of mission cost estimation and pre-mission planning. The model network consists of 43 Army airfields and 426 airports in the Contiguous United States (CONUS) with Department of Defense contract fuel. Using the A-Star algorithm and considering aircraft fuel range, ground speed, and refueling time, we determine the refuel locations that result in the most efficient route. Considering the use of both distance and travel time, we compare our model's performance with Dijkstra's algorithm, a greedy heuristic, and existing cost-estimation techniques. The ADERPT also examines the use of a grid-based network for obstacle avoidance in route planning and provides a proof of concept for its potential use as a mission planning tool.

Acknowledgments

I would like to express my sincere gratitude to my thesis advisors, Dr. Raymond Hill and Dr. Sarah Nurre for their knowledge and guidance during my time at AFIT. Thanks also to my sponsor, specifically MAJ Jaysen "Dwight" Yochim and the FORSCOM G4 Aviation Distribution section for providing the foundation for this work and for taking the time to offer assistance and feedback.

Further thanks to all my peers who helped me throughout my studies, especially Capt. Chris Jones for his VBA expertise! Finally, a special thanks to my wife and family for enduring some sleepless nights and frustrating days. I would not have made it through the last 18 months without your unceasing encouragement and support.

Garret D. Fett

Acknowledgments

Page

List of Figures

List of Tables

Page

AIRCRAFT ROUTE OPTIMIZATION USING THE A-STAR ALGORITHM

I. Introduction

Background

Army aviation assets have been in high demand over the last ten years. Operation Iraqi Freedom, New Dawn, Enduring Freedom, and other operations have stretched the capabilities of rotary-wing aircraft to the maximum. Due to the need for increased helicopters in combat environments, deployed Combat Aviation Brigades (CABs) are often directed to leave aircraft in-theater as Stay-Behind Equipment (SBE) when they redeploy to their home station. This creates the added problem of having to replenish the redeploying unit with aircraft at home-station for training, mission support, and real-world missions.

The United States Army Forces Command (FORSCOM) is responsible for determining how this replenishment of aircraft occurs, selecting aircraft from other units to be transferred to the redeploying unit. The transferred aircraft are flown to the new duty station by either the losing or gaining unit, with FORSCOM funding the cost of the aircraft movement. In the fiscally-constrained environment of today's military, the FORSCOM G-4 Aircraft Distribution section and G3/5/7 Aviation Division are required to provide estimates for the cost of all aircraft transfers.

Current cost estimate techniques use an approximated flight time between the losing and gaining duty station. The approximated flight time is based on straight line distance and is multiplied by a "cost factor" to produce a cost estimate for the flight. The cost factor incorporates fuel cost, as well as parts and consumables costs. The

FORSCOM G-4 Aircraft Distribution section is responsible for generating aircrew TDY cost estimates for all aircraft movements. This is currently done by comparing future required aircraft movements to completed movements and their associated duration and Temporary Duty (TDY) costs.

Problem Statement

This research attempts to improve current cost estimation techniques by developing an Aviation Distance Estimation and Route Planning Tool (ADERPT) that incorporates the use of the A-Star routing algorithm to find an optimum route between the losing and gaining airfield. The algorithm considers aircraft constraints (maximum distance before refueling), aviator constraints (maximum flight hours per day), and potential obstacles to the flight path (Restricted Operating Zones). The model includes all Contiguous United States (CONUS) Army Airfields and all CONUS Defense Logistics Agency (DLA) approved contract fuel locations (DLA, 2013).

Scope and Contribution

The ADERPT provides an expedient method of producing accurate flight distances and travel times between all CONUS Army Airfields and contract fuel locations. The route optimization distance and travel time calculations can be used to estimate fuel and TDY costs. The ADERPT runs on software common to DOD computer systems (Microsoft Excel) and processing times are short (less than one second). The route optimization tool could also provide value to aircrews and air mission planners. The program quickly identifies efficient fuel stops between a departure and arrival location, which can be used to assist with cross-country flight planning.

Overview

Chapter 2 of this document will provide a review of existing literature related to routing problems and use of the A-Star algorithm. Chapter 3 outlines the proposed methodology for finding optimum routes for aircraft movement cost estimation and pre-mission planning. Chapter 4 provides analysis and results of the implementation of the algorithm, and compares it to other approaches to the routing problem. Chapter 5 provides a summary of this research, discusses the limitations of the model, and proposes recommendations for future research.

II. Literature Review

Path-finding Applications

The process of path-finding over a network develops a route from a starting node to a target node that minimizes "cost" while avoiding obstacles. How cost is defined can vary depending on the goal of the path-finder. Cost could be distance, time, fuel expended, or a combination of any number of factors that we seek to minimize by planning an efficient route.

Path-finding algorithms can also be used to find optimum or near optimum routes between multiple points while considering obstacles and constraints. The application of these algorithms is very diverse. Vehicle GPS navigation devices make use of such algorithms to provide drivers with efficient directions (Jenkins 2007). Military combat simulations such as the Close Combat Tactical Trainer use path-finding algorithms to move Soldiers and vehicles across a simulated battle space (Beeker 2004). Finally, path-finding algorithms are used for Artificial Intelligence (AI) in strategy video games, to smartly move computer-controlled elements through their environments (Stout 1997).

Data and Notation

Path-finding algorithms operate using a mathematical "graph" which is simply the set of nodes (sometimes referred to as vertices) that exist in the search space, or area in which we are examining. A graph could be represented as a grid, as shown in Figure 1, where each cell is a node and the arcs are implied to connect any node i to node j such that j is adjacent to i. Figure 2 shows another example of a graph in which cities are represented as node and the roads connecting cities are arcs which are assigned weights based on the distance, time, fuel cost, etc. between the two nodes. The weight of an arc could also be calculated using combinations multiple units of measure. The arcs in Figure 1 are unweighted and represented by the lines connecting cells (for simplicity, only the arcs surrounding the start node are shown). The arcs in Figure 2 are shown as lines connecting the cities and are weighted by distance (miles) and time.

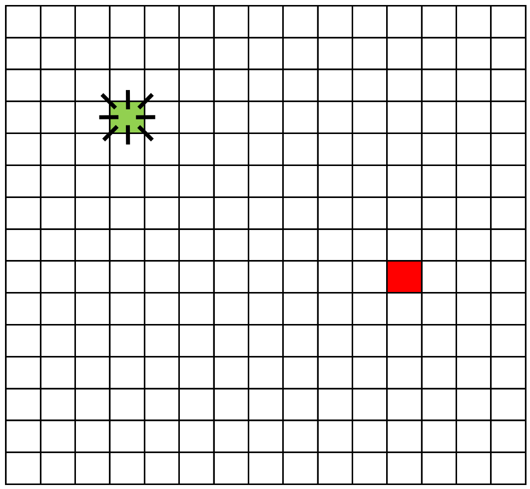


Figure 1. Example of a graph composed of grid cells. The green cell represents the start node, and red cell represents the target node.

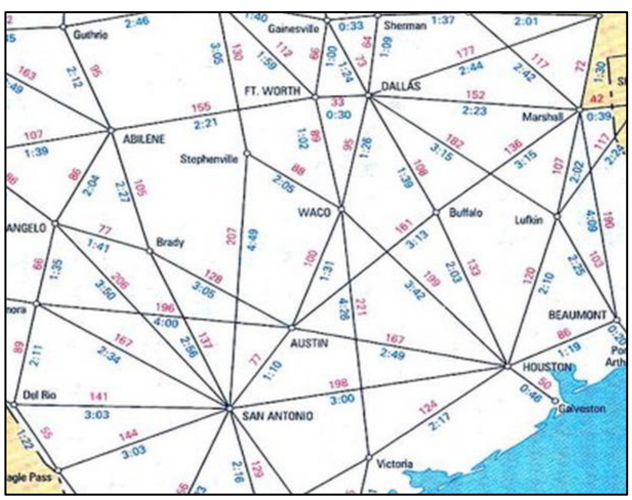


Figure 2. Network representing the transportation/road system in Southeastern Texas (taken from http://origin-ars.els-cdn.com/content/image/1-s2.0-S0360835213001459-gr5.jpg).

There are numerous algorithms used in path-finding. We first discuss, in detail, two common algorithms: Dijkstra's Shortest Path Algorithm, and the A-Star Algorithm, and then discuss several applications of these algorithms.

Dijkstra's Shortest Path Algorithm

Dijkstra's algorithm is one of the earliest algorithms for finding an optimum path from a start node to a target node. Dijkstra's algorithm works by separating nodes into two lists: those that have been visited, and those that have not been visited (Dijkstra 1959). The algorithm begins at the starting node with all nodes on the unvisited list and, iteratively, the node with the lowest cost path to it is removed from the list and placed on the visited list. The lowest cost to all nodes is initially set as infinite to indicate that the node has not been visited and to allow the first path to reach the node to become, at least temporarily, the best route to that node. The first node placed on the visited list is the starting node (usually with a cost of zero). The algorithm then examines all nodes reachable from the starting node (referred to as "neighbor nodes") and selects the lowest cost option as the current node. The current node is then moved to the visited list, it's neighbor nodes are evaluated and assigned costs. The algorithm then selects the unvisited node with the lowest cost as the current node. As the number of visited nodes expands, the forward-most edge of the explored space is referred to as the frontier.

Exploration continues until the target node is placed on the visited list, at which point the algorithm ends. Since the algorithm always examines the lowest cost path first, a more efficient route to the target node cannot exist (Beeker 2004). Dijkstra's algorithm assigns a "pointer" to each node which indicates the "parent" node that resulted in the lowest cost route to the node. Once the algorithm arrives at the target node, we can use the pointers to retrace back to the starting node along the optimal path.

We simplify Figure 2 into a $|N| = 7$ node network shown in Figure 3, where each node represents a city and the value on each arc represents miles. Using this network, we demonstrate the processing of Dijkstra's algorithm using Dallas as the start node and San Antonio as the target node.

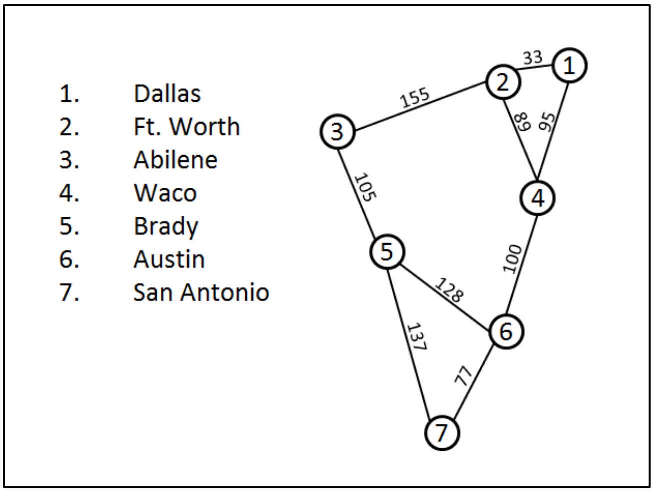


Figure 3. Simplified road network used in example of Dijkstra's algorithm (arc costs are shown in miles).

Table 1 shows the 7 iterations required to add the target node to the visited list. The algorithm begins at iteration 1 with the set of visited nodes empty and selects node 1 as the current node since it has the lowest cost (0). Iteration 2 evaluates the two neighbor nodes that can be reached from node 1 (nodes 2 and 4), selects the node with the lowest cost (node 2), and records the parent ID for the route (node 1). The algorithm iterates until iteration 7 in which the goal node is designated as the current node. We then use the parent ID "pointers" to retrace the path from the goal node to the start node and determine the least cost path to be the path travelling through nodes 7, 6, 4, and 1.

Table 1. Example of the iterations of Dijkstra's algorithm applied to the network shown in Figure 3.

					Node (distance, parent node)						
Iteration	**Unvisited**	**Visited**	**Current**	**Neighbors**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	{1,2,3,4,5,6,7}	{}			(0, -)	(∞, -)	(∞, -)	(∞, -)	(∞, -)	(∞, -)	(∞, -)
2	{2,3,4,5,6,7}	{1}	1	{2,4}		(33, 1)	(∞, -)	(95, 1)	(∞, -)	(∞, -)	(∞, -)
3	{3,4,5,6,7}	{1,2}	2	{3,4}			(188, 2)	(95, 1)	(∞, -)	(∞, -)	(∞, -)
4	{3,5,6,7}	{1,2,4}	4	{3,6}			(188, 2)		(∞, -)	(195, 4)	(∞, -)
5	{5,6,7}	{1,2,3,4}	3	{5,6}					(293, 3)	(195, 4)	(∞, -)
6	{5,7}	{1,2,3,4,6}	6	{5,7}					(293, 3)		(272, 6)
7	{5}	1,2,3,4,6,7	7								

The computational complexity of the original Dijkstra's algorithm is $O(|N|^2)$ (Cormen, Leiserson, and Rivest 1990). As the number of nodes $|N|$ increases, Dijkstra's algorithm proves to be less efficient than other algorithms. Dijkstra's algorithm is not a directed algorithm, meaning it does not give preference to nodes that move closer to the target node (Rabin 2002). Dijkstra's simply searches outward from the starting node, finding the least cost route to each node until the target node is found. Figure 4 shows that this search method explores areas of the search space that are unlikely to produce optimal solutions. With search spaces and more complex path-finding problems, this can result in long processing times.

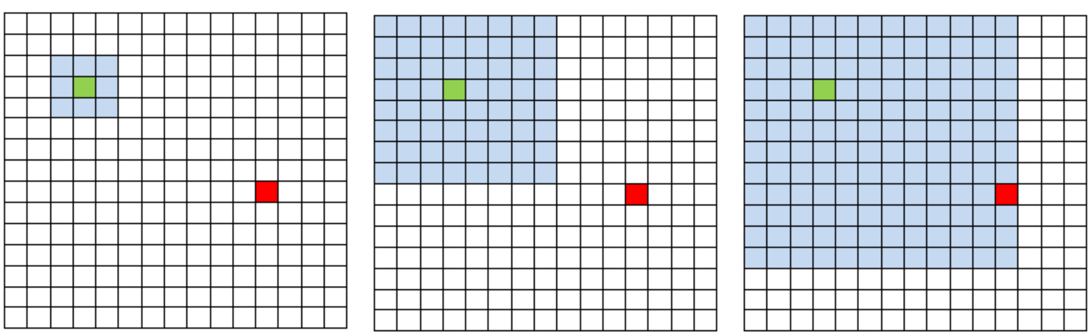


Figure 4. Three progressive stages of Dijkstra's algorithm using a grid-based graph A-Star Algorithm.

The A-Star algorithm was first presented by Hart, Nilsson, and Raphael in 1968 as the combination of a mathematical and heuristic approach to find a least cost path from a starting node to a target node (Hart, et al. 1968). A-Star builds upon the approach of Dijkstra's algorithm, but incorporates a heuristic to direct the search toward the target node.

The combination of the mathematical and heuristic approaches proves significant. While heuristics can generally not guarantee a lowest cost path, Dijkstra's algorithm can. And while Dijkstra's algorithm expands out from the starting node in all directions, a heuristic focuses the search and can converge on the target node much quicker. The combination results in the ability of the A-Star algorithm to guarantee a least cost path, if one exists, and finds it searching the smallest number of nodes possible (Hart, et al. 1968).

If $f(n)$ is the lowest cost path to the target node through node n:

$$f(n) = g(n) + h(\mathrm{n}) \tag{1}$$

Where

n is the current node,

$g(n)$ is the *actual* cost of the path from the starting node to the current node, and

$h(n)$ is the *actual* cost of the path from the current node to the target node.

The A-Star algorithm calculates an estimate of $f(n)$, denoted $f'(n)$ based on estimated costs for $g(n)$ and $h(n)$, using the following equation:

$$f'(n) = g'(n) + h'(n) \tag{2}$$

where:

$g'(n)$ is the *estimated* cost of the path from the starting node to the current node, and

$h'(n)$ is the *estimated* cost of the path from the current node to the target node.

The algorithm evaluates nodes within the search space to minimize $f(n)$. In this evaluation function $g(n)$ by itself is equivalent to performing Dijkstra's algorithm. We would begin our undirected search at the starting node and expand out to nodes that minimize path cost, but do not necessarily move us closer to the target node. It is the addition of the heuristic component, $h(n)$, that helps direct the search toward the target node. The heuristic serves as an estimation function, estimating the cost for reaching the target from each node that is evaluated (Beeker 2004).

The term "heuristic" is derived from the Greek word "heuriskein," which means "to discover" (Zanakis and Evans 1981). Operations Researchers have long used heuristic procedures to reduce the search space in problem-solving activities (Tonge, 1961). Heuristics effectively seek to find good solutions to difficult problems in a reasonable amount of computational time. There are many situations when the implementation of a heuristic is useful. One such situation is when a heuristic improves the performance of an optimizer by providing starting solutions or when the heuristic guides the search thereby reducing the number of candidate solutions (Zanakis and Evans 1981). Hart, Nilsson, and Raphael (1968) exploit this benefit by integrating a heuristic function into their algorithm. Figure 5 shows a comparative study done by Sathyaraj, et al. (2008) of the computational time of Dijkstra's algorithm and the A-Star algorithm as the number of nodes in a network increase.

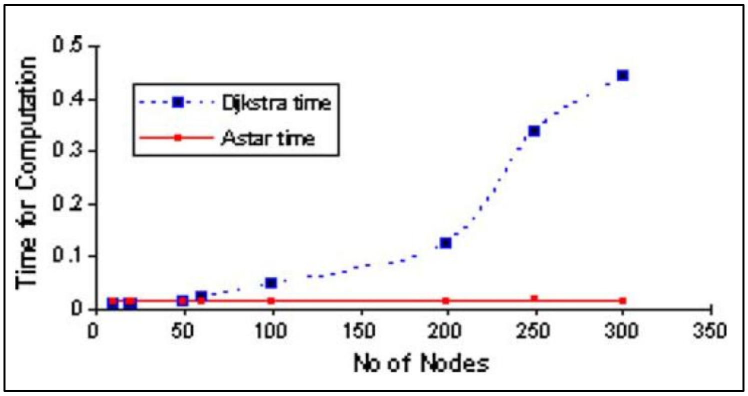


Figure 5. Computation time comparison of A-Star vs. Dijkstra's algorithm (Sathyaraj, et al. 2008).

A-Star does not dictate the type of heuristic to use in the algorithm. Instead, the heuristic can be formulated and tailored to the needs of the user. An important property of the heuristic is admissibility. A heuristic is considered "admissible" if the estimated cost of reaching the target node is always less than the actual cost, for all nodes. That is if $h'(n) \le h(n) \forall n \in N$ (Beeker 2004). An A-Star algorithm containing an admissible heuristic guarantees an optimum path, if one exists, while an inadmissible heuristic does not.

The processing time of the A-Star algorithm is significantly influenced by the type of heuristic used in the evaluation function (Soltani, et al. 2003). A gross underestimation of $h(n)$ causes the algorithm to search a broader space, resulting in longer processing times. A heuristic that overestimates $h(n)$ does not guarantee an optimal solution, but can provide a "good" solution quickly (Patel 2011).

Two commonly used heuristics for $h'(n)$, Euclidean distance and Manhattan distance, illustrate the role the heuristic plays in the search. Euclidean distance uses the Pythagorean Theorem to generate a "straight line distance" between two nodes. It can be applied to our city/road network to generate a cost estimate from node 2 to the target node as shown in Figure 6(b). Euclidean distance produces an admissible heuristic since there can be no shorter path between two nodes. Manhattan distance is commonly used in grid-based graphs and estimates the distance to the target node by counting only vertical and horizontal moves. This heuristic is inadmissible since a shorter path to the target node exists. Figure 6(a) shows a Manhattan distance heuristic applied to our grid-based graph problem.

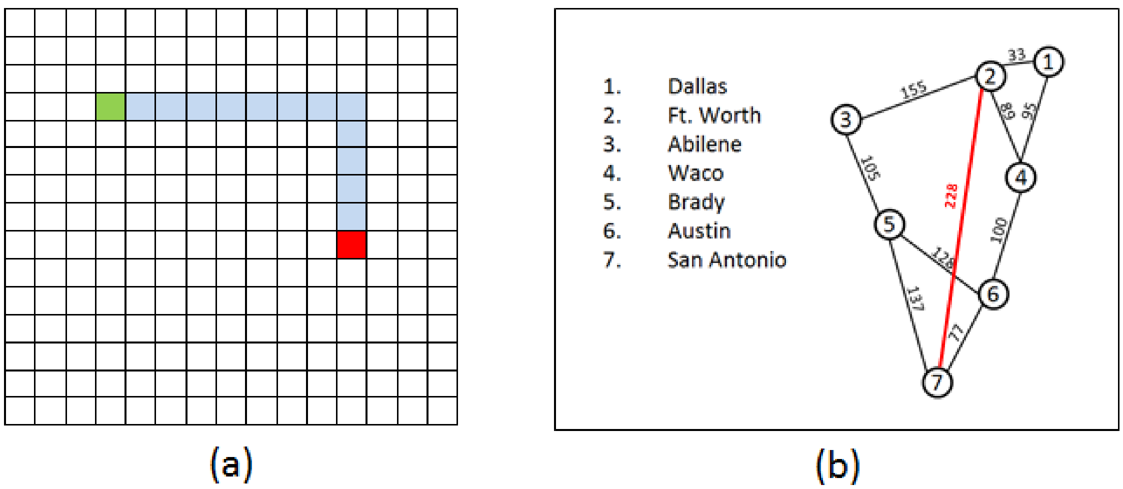


Figure 6. (a) Manhattan distance heuristic from the start node to the target node (b) Euclidean distance heuristic from node 2 to the target node.

Aside from the guiding heuristic, the A-Star algorithm operates very much like Dijkstra's algorithm, evaluating nodes and maintaining open and closed lists of visited and unvisited nodes. The algorithm also maintains pointers to track the parent of each node. The A-Star pseudocode shown in Figure 7 was originally written by James Matthews in his article Basic A-Star Pathfinding Made Simple (2002).

```
1. Let P = the start node
2. Assign f(n), g(n), and h(n) values to P
3. Add P to the Open list
4. Let B = the best node from the Open list (lowest f(n) value)
     If B is the goal node, then quit - a path has been found
     If Open list is empty, then quit - a path cannot be found
5. Let C_i = all valid nodes connected to B
     Assign f(n), g(n), and h(n) values to C_i
     Check whether C_i is on the Open or Closed list
          If so, check to see if f(n) is lower
               If so, update the path
          Else, add C_i to the Open list
6. Return to step 4
```

Figure 7. A-Star Pseudocode (Matthews 2002).

Route Optimization and Obstacle Avoidance Applications

Previous work related to path-finding and obstacle avoidance has been applied to aviation route planning. Szczerba, et al. (2000) developed a Sparse A-Star Search (SAS) route planner which seeks to minimize a cost array while meeting certain constraints. Szczerba, et al. (2000) utilize a grid-based graph and incorporate a Map Cost (MC) array which can combine "cost layers" such as the terrain, threat exposure, and weather associated with each grid cell. This Map Cost, along with a flight distance cost are used to compute each actual cost, $g(n)$, and estimated cost, $h'(n)$, as the algorithm progresses. The Map Cost array allows a search for a route that not only seeks to minimize the distance travelled, but also considers other factors that may impact the ability of an aircrew to successfully complete a flight.

The SAS route planner also incorporates constraints in the algorithm that can prevent infeasible routes. Szczerba, et al. (2000) discuss a route distance constraint which prevents routes from exceeding the fuel capacity of an aircraft, an approach angle constraint which prevents routes from approaching the destination airfield at an angle that is not aligned with the runway, and a turn angle constraint which prevents turns that would exceed the maximum angle of bank of an aircraft (Szczerba, et al. 2000).

The U.S. Army Research Laboratory (ARL) developed the Aviation Weather Routing Tool (AWRT) to efficiently plan manned and unmanned aircraft routes while avoiding hazardous weather (Jameson, Knapp, and Measure 2009). AWRT uses the A-Star algorithm and a grid-based graph which includes a weather cost for each grid cell based on the presence of adverse weather conditions at that location. The AWRT operates in four dimensional space (3-D and time) and allows the user to input a risk tolerance that effects the likelihood that the planned route will traverse through adverse weather conditions.

Conclusion

There are numerous approaches to finding an efficient route for a single entity to travel between two points. Our proposed model combines some of the techniques outlined in this chapter to conduct sequential iterations of the A-Star algorithm using network-based and grid-based graphs to find an optimal flight route between two locations while avoiding known obstacles and conforming to a set of constraints.

III. Methodology

Introduction

The Aviation Distance Estimation and Route Planning Tool (ADERPT) provides two primary functions: route optimization and obstacle avoidance. The route optimization portion of the model seeks a least cost route from a starting location to an ending location by selecting refuel locations that minimize the total route distance or travel time while considering multiple constraints. The obstacle avoidance portion uses a grid-based network and seeks an optimum route from a starting location to an ending location avoiding obstacles along the flight route. The user can choose to implement only one of the functions, or can implement them both in series.

Distance Calculations

All geographic coordinates used in the model are latitude/longitude coordinates expressed in Decimal Degrees (DD). To account for the spherical curvature of the earth, we use great-circle distance calculations as outlined in AFR 51-40, Air Navigation (Departments of the Air Force and Navy 1983).

$$d = 60 * \cos^{-1}[\sin L_1 * \sin L_2 + \cos L_1 * \cos L_2 * \cos(\lambda_2 - \lambda_1) \qquad (3)$$

Where

d is the great-circle distance between two coordinates.

L_1 and L_2 are the departure and arrival latitude, respectfully.

λ_1 and λ_2 are the departure and arrival longitude, respectfully.

Route Optimization

<u>Overview</u>

The Route Optimization portion of the model generates an optimum route between two locations by selecting refuel locations that minimize the total distance of the route. The network consists of 439 nodes representing 43 CONUS Army Airfields and 396 airports with contract fuel available. Figure 8 shows a map displaying the location of all 439 nodes.

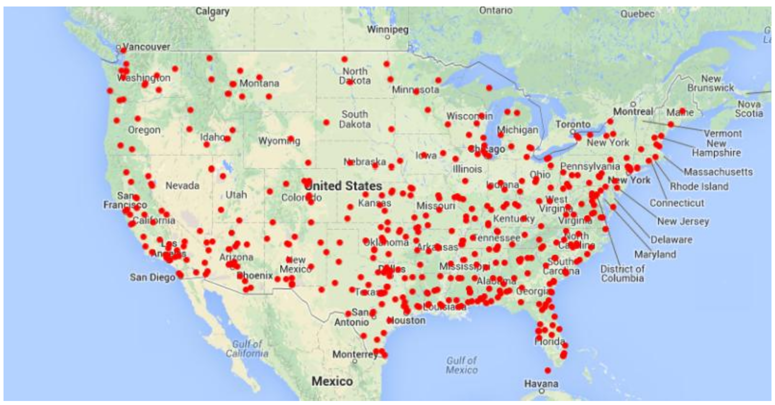


Figure 8. Map of 439-node network.

Each arc in the network represents the great-circle distance between two nodes. These arc distances are pre-processed and stored in a distance matrix to reduce processing time. Arc lengths which exceed the user-selected aircraft maximum fuel range are eliminated from consideration as the algorithm searches for an optimum route. A portion of the distance matrix is shown in Figure 9.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	229	36	191	171	98	125	78	232	19	24	43	113	248	78
2	229	0	263	98	72	140	222	223	28	245	242	192	149	41	207
3	36	263	0	219	202	128	130	84	265	29	38	72	137	280	90
4	191	98	219	0	45	94	138	155	78	210	212	149	82	84	140
5	171	72	202	45	0	74	150	153	64	189	188	130	78	79	137
6	98	140	128	94	74	0	100	85	138	117	118	56	30	153	69
7	125	222	130	138	150	100	0	47	210	141	149	98	76	220	48
8	78	223	84	155	153	85	47	0	217	94	102	59	76	230	16
9	232	28	265	78	64	138	210	217	0	249	248	193	142	17	201
10	19	245	29	210	189	117	141	94	249	0	10	62	132	265	95
11	24	242	38	212	188	118	149	102	248	10	0	65	135	264	102
12	43	192	72	149	130	56	98	59	193	62	65	0	70	208	50
13	113	149	137	82	78	30	76	76	142	132	135	70	0	154	60
14	248	41	280	84	79	153	220	230	17	265	264	208	154	0	213
15	78	207	90	140	137	69	48	16	201	95	102	50	60	213	0

Figure 9. A portion of the pre-processed distance matrix showing nodes 1-15. Row and column numbers represent node numbers.

Model Assumptions

The route optimization model assumes the aircraft travels at a constant Ground Speed (GS). It does not account for acceleration during departure or deceleration during approach. The model assumes the aircrew will be able to navigate the assigned route without deviating due to inclement weather, Air Traffic Control (ATC) instructions, or other possible reasons. The calculated route distances are based on “straight-out” departures and “straight-in” arrivals and no distance is added for any required departure or arrival procedures. We also assume fuel is always available at all airports included in the model and do not consider refuel hours of operation. Finally, the model assumes the user factors in fuel consumption during start-up and ground taxi when inputting the max fuel range.

Inputs

The user selects the starting location and ending location from a dropdown list that includes 43 CONUS Army Airfields and 396 airfields with contract fuel available.

The user also enters the maximum distance allowed before refueling, the planned ground speed, and the time required to refuel. Like Zeisler's (2000) Intra-Theater Airlift Model, these inputs allow for adaptable application across various aircraft Mission-Design Series (MDS) with different fuel ranges, cruise airspeeds, and refueling times. Additionally, it allows the user to tailor the fuel range to a specific flight profile; a Visual Flight Rules (VFR) flight profile requires aviators to plan a 20-minute fuel reserve into the flight while an Instrument Flight Rules (IFR) flight profile requires a 30-minute fuel reserve (Department of the Army 2008). Finally, the user has the option to search for a route that minimizes the total flight distance between the starting and ending location or to search for a route that minimizes the total travel time. While distance minimization requires no further explanation, the method for minimizing travel time is explained in the following section.

Model Procedure

The route optimization – Distance Minimization A-Star algorithm (DMA-Star) begins by collecting the start and target nodes from the user input form. The start node is then added to the open list. The algorithm then uses the pre-processed distance matrix to identify all feasible successor nodes (refuel locations that are closer than the user-defined maximum distance before refueling), adds them to the open list, and calculates $g'(n)$, $h'(n)$, and $f'(n)$ for each. The model selects the node with the lowest $f'(n)$ value and designates it as the current node. The algorithm then iterates, identifying all feasible successor nodes and terminates when the goal node is designated as the current node. If the goal node has not been reached and the open list contains no nodes, the model produces an error message indicating that an optimum solution could not be found.

The route optimization –Time Minimization A-Star algorithm (TMA-Star) model is structured the same way as the DMA-Star model, with two modifications. First, the $g'(n)$ values of $h'(n)$, and $f'(n)$ are in units of time (in hours) instead of distance. To accomplish this, the algorithm divides $g'(n)$ and $h'(n)$ by the estimated ground speed of the flight.

The second deviation from the DMA-Star model is that the user-defined ground time required to refuel is incorporated into time minimization model. The resulting formula is:

$$f'(n) = g1'(n) + g2'(n) + h1'(n) \tag{4}$$

$$g1'(n) = \frac{g(n)}{GS}$$

$$g2'(n) = FS * RT$$

$$h1'(n) = \frac{h'(n)}{TAS}$$

where:

$g1'(n)$ is the estimated flight time of the route from the starting node to node n,

GS is the planned Ground Speed of the flight (in knots),

$g2'(n)$ is the estimated flight time of the route from the starting node to node n,

FS is the number of fuel stops required to arrive at node n,

RT is the total ground time required to refuel the aircraft (in hours),

and

$h1'(n)$ is the estimated flight time of the route from the node n to the target node.

Outputs

When the goal node is designated as the current node, the algorithm exits the loop and retraces the route path from the goal node to the start node using the Parent ID property of each node. The model then displays a table listing the refuel locations in sequential order, along with the distance and flight time of each flight leg, the total distance of the route, the total flight time of the route (not including ground time during refueling), and the total administrative time of the route (including ground time during refueling). Table 2 shows an example of the Route Optimization program output.

The inclusion of total flight time and total time in the output are important when considering aircrew flight time and duty day constraints. Aviation unit Standard Operating Procedures (SOPs) and Composite Risk Management (CRM) tools normally include limit aviators on the number of flight hours allowed per day, and the length of the duty day (Department of the Army, 1999). Considering these limitations while reviewing the “total flight time” and “total time” outputs of the route optimization model allows a mission planner to anticipate the location(s) in which an aircrew may need to Remain Over Night (RON). This also allows FORSCOM Aviation Distribution personnel to anticipate the number of days required to complete the flight, and forecast TDY costs accordingly.

Table 2. Example of the Route Optimization output of a flight originating from New Hanover International Airport and terminating at McClellan Airfield. The refuel time is 1 hour, as reflected in the Admin Time.

	ICAO	Location	State	LAT	LONG	Leg Dist. (NMs)	Total Dist. (NMs)	Leg Flight Time (Hrs)	Total Flight Time (Hrs)	Total Time (Hrs)
Start Point	KILM	NEW HANOVER INTL	NC	34.270603	-77.902558	0.0	0.0	0.0	0.0	0.0
Fuel Stop 1	KAVL	ASHEVILLE RGNL	NC	35.436194	-82.541806	239.0	239.0	1.7	1.7	2.7
Fuel Stop 2	KEOD	SABRE ARMY AIRFIELD	TN	36.5682	-87.4808	249.3	488.3	1.8	3.5	5.5
Fuel Stop 3	KSGF	SPRINGFIELD BRANSON NATL	MO	37.2457	-93.3886	286.5	774.8	2.0	5.5	8.5
Fuel Stop 4	KHYS	HAYS MUNICIPAL AIRPORT HAYS	KS	38.8422	-99.2732	294.2	1069.0	2.1	7.6	11.6
Fuel Stop 5	KCOS	CITY OF COLORADO SPRINGS MUNI (PETERSON FLD)	CO	38.8058	-104.7008	253.9	1322.9	1.8	9.4	14.4
Fuel Stop 6	KGJT	GRAND JUNCTION RGNL	CO	39.1224	-108.5267	179.6	1502.5	1.3	10.7	16.7
Fuel Stop 7	KENV	WENDOVER (DECKER AAF)	UT	40.7187	-114.0309	270.9	1773.4	1.9	12.7	19.7
Fuel Stop 8	KRNO	RENO/TAHOE INTL	NV	39.4991	-119.7681	273.4	2046.8	2.0	14.6	22.6
Destination	KMCC	MC CLELLAN AFLD SACRAMENTO	CA	38.6676	-121.4006	91.0	2137.8	0.6	15.3	23.3

Obstacle Avoidance

Overview

The Obstacle Avoidance portion of the model uses a grid-based node network to generate a route between two locations that avoids obstacles and considers areas that are undesirable for flight. We define an obstacle as an area through which flight is prohibited or not feasible. Examples of obstacles are Restricted Operating Zones (ROZ's), Prohibited Airspace, and Restricted Airspace. Undesirable areas create an inconvenience or increased risk to flight. Examples of undesirable areas are Class-B Airspace, Military Operations Areas (MOA's), and urban areas. The code used in this portion of the model is an adaptation of the two-dimensional path-finding program developed by Volpi (2005).

The model utilizes the latitude/longitude coordinate system to create a grid-based node system in Microsoft Excel that is a tessellation of the contiguous United States. Each column represents one-tenth of a degree of longitude, each row represents one-tenth of a degree of latitude, and each cell represents a node. The dimensions of the map are designated using the extreme points of the contiguous United States as shown in Table 3.

Table 3. Extreme points of the contiguous United States that define the corners of the tessellated grid network used in the obstacle avoidance model.

Extreme Point	Location	Latitude	Longitude
Westernmost	Cape Alava, WA	48.16	-124.73
Easternmost	W. Quoddy Head, MA	44.81	-66.95
Northernmost	Northwest Angle, MN	49.38	-95.15
Southernmost	Ballast Key, FL	24.52	-81.96

These extreme points result in a map space with dimensions 260 x 590, creating a total of 153,400 nodes.

Obstacles can be added to the map by coloring the cells corresponding with the obstacle location black. The algorithm identifies and "ignores" cells colored black, effectively eliminating these nodes from begin evaluated or added to the open list. Undesirable areas can be added to the map by entering a "map cost" into the cell(s) of the map that correspond with the geographical location of the undesirable area. A map cost assigned to a node represents the distance, in NMs, that is added to the route if it travels through that node. As the algorithm evaluates a node with a map cost assigned, it adds the map cost to the node's $f'(n)$ score, encouraging the algorithm to find a route that avoids the undesirable area. Figure 10 shows an area in the Northwestern-most area of the map space that contains nodes designated as obstacles and undesirable areas.

	-125.0	-124.9	-124.8	-124.7	-124.6	-124.5	-124.4	-124.3	-124.2	-124.1	-124.0	-123.9	-123.8	-123.7	-123.6	-123.5	-123.4	-123.3	-123.2	-123.1
24.0																				
24.1																				
24.2																				
24.3																				
24.4																				
24.5																				
24.6																				
24.7																				
24.8																				
24.9						5	5	5												
25.0						5	10	10												
25.1						5	5													
25.2							5													
25.3							5													
25.4																				
25.5																				
25.6																				
25.7																				

Figure 10. Map space in Northwest portion of contiguous U.S. with rectangular obstacle vic (24.4ºN, -124.6 ºW) and undesirable area vic (25.0 ºN, -124.4 ºW).

Model Assumptions

The obstacle avoidance model assumes that the desire to avoid a given area can be converted into a map cost (distance). We also assume that all obstacles and undesirable areas extend from the ground to an infinite altitude, and cannot be avoided vertically. Finally, we include the assumption that all turns can be executed as planned and make no limitation on turn radius in the model.

Inputs

The user selects a starting and ending location from the same dropdown list of Army Airfields and contract fuel locations as in the Route Optimization program. Obstacles and undesirable areas are inputted directly to the map space by the user.

Model Procedure

The model first collects the start and target nodes from the user input form, and adds the start node to the open list. The algorithm then identifies the feasible successor nodes (the eight adjacent nodes, ignoring those nodes designated as obstacles), adds them to the open list, and calculates $g'(n)$, $h'(n)$, and $f'(n)$ scores for each using the formula below.

$$f'(n) = g'(n) + l'(n) + h'(n) \quad (5)$$

where:

$g'(n)$ is the *estimated* cost of the path from the starting node to node n,

$l'(n)$ is the map cost assigned to node n and

$h'(n)$ is the *estimated* cost of the path from node n to the target node.

The model selects the node with the lowest value of $f'(n)$ and designates it as the current node. The algorithm then iterates and terminates when the goal node is designated as the current node. If the goal node has not been reached and the open list contains no nodes, the model produces an error message indicating that an optimum solution could not be found.

Output

The algorithm exits the search when the goal node is designated as the current node and, in the same method as the Route Optimization program, it retraces the route from the goal node to the start node using the Parent ID property of each node. As it retraces the route, the program calculates and adds the distance between node coordinates using Great Circle Distance. The program outputs the total distance of the route (in

NMs) and generates a visual depiction of the route. Figure 11 shows the output from the Obstacle Avoidance model for a route between North Platte, Nebraska and Columbia, Missouri that considers obstacles and undesirable areas.

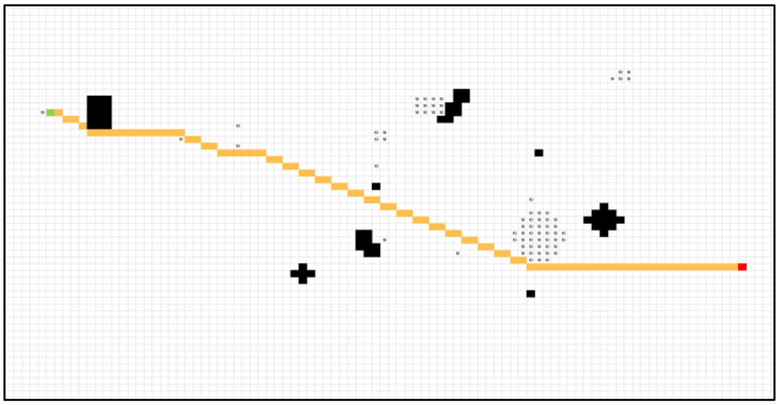


Figure 11. Obstacle avoidance program output of a route from North Platte to Columbia. Obstacles are black, undesirable areas are numbered, and the route is shown in orange.

IV. Results

Route Optimization

To evaluate the route optimization model, we run the algorithm on all 96,141 possible start and end node combinations. We use a maximum fuel range of 300 NMs and eliminate 10,874 iterations in which the start node is less than 300 NMs from the target node (a route which does not require a fuel stop). We test the remaining 85,267 iterations using the DMA-Star algorithm, TMA-Star algorithm, Dijsktra's algorithm, and a greedy heuristic and compare the results from each, focusing on route distance, route time, number of fuel stops, and processing time.

A portion of the iterative results for the three algorithms are shown in Appendix B, with aggregated results shown in Table 4. We can see from Table 4 that, since both the DMA-Star algorithm and Dijkstra's algorithm guarantee optimality, their average distance and average number of fuel stops are equivalent. Since the A-Star algorithm uses a heuristic to narrow its search space, the average processing time and number of nodes explored is reduced substantially from Dijsktra's algorithm.

Table 4. Averaged results of route optimization iterations using the Distance Minimization A-Star algorithm, Dijkstra's algorithm, and the greedy heuristic.

	Avg. Distance (NMs)	Avg. Total Time (Hrs)	Avg. Number of Fuel Stops	Avg. Processing Time (Seconds)
DMA-Star	999	10.8	3.7	.06
TMA-Star	1011	10.3	3.2	.59
Dijkstra's Algorithm	999	10.8	3.7	1.17
Greedy Heuristic	1039	10.6	3.2	.02

Table 4 also shows that while the greedy heuristic does not usually generate the shortest route, it does produce routes that average fewer fuel stops than those found using the DMA-Star algorithm and Dijkstra's algorithm. One extreme case that illustrates this disparity is the route from Albert J. Ellis Airport, North Carolina to Page Municipal, Arizona. Figure 12(a) shows the route found by the DMA-Star algorithm. While the route distance is minimized at 1,648 NMs, the route includes 9 fuel stops, resulting in a total travel time of 20.8 hours (assuming 1 hour ground time to refuel). Figure 12(b) shows the route found by the greedy heuristic. The route length is 1,730 NMs (82 NMs longer than the optimum), but only requires 5 fuel stops and a total travel time of 17.4 hours. In Figures 12(a)/(b) and all similar subsequent figures, dots shown on the maps represent refuel stops along the route.

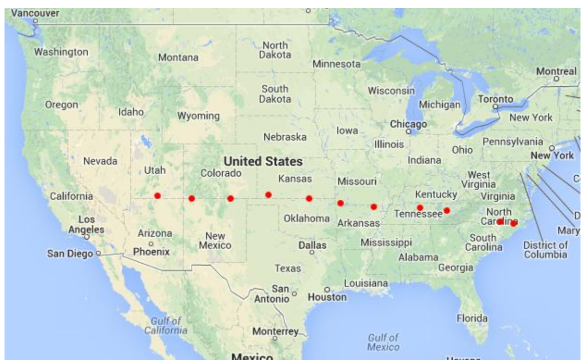


Figure 12(a). DMA-Star generated route from Albert J. Ellis Airport, North Carolina to Page Municipal, Arizona.

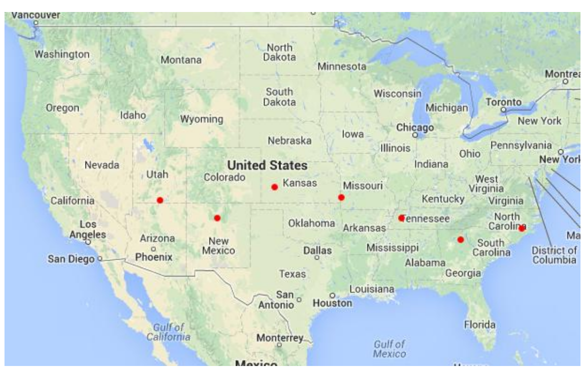


Figure 12(b). Route found using the greedy heuristic from Albert J. Ellis Airport, North Carolina to Page Municipal, Arizona.

The greedy heuristic does not always produce routes with shorter travel times, however. The route combination that results in the greatest difference in route distance between the DMA-Star and the greedy heuristic is the route between Pease Air Force Base, New Hampshire and Roberts Field, Oregon. Figure 13(a) shows the route found using the greedy heuristic, which includes 10 fuel stops and travels 2,705 NMs in 29.3 hours. Comparatively, the route found by the DMA-Star algorithm shown in Figure 13(b) is 484 miles shorter, 4.4 hours faster, with 1 less fuel stop.

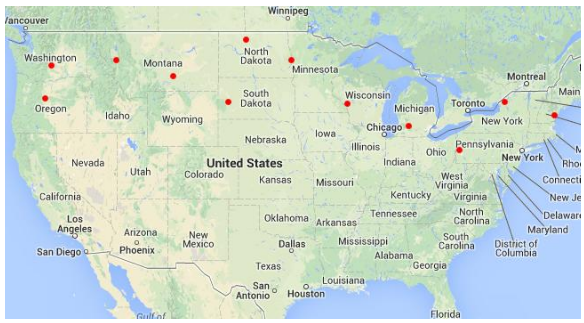


Figure 13(a). Route found by the greedy heuristic between Pease Air Force Base and Roberts Field.

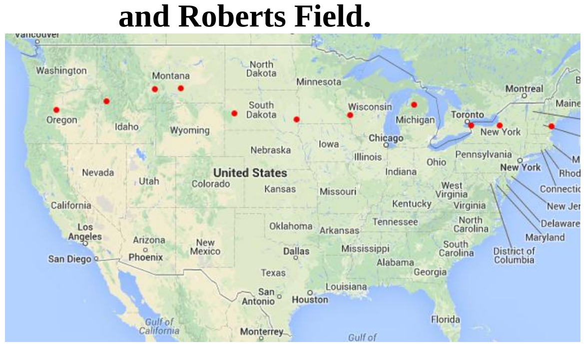


Figure 13(b). DMA-Star generated route between Pease Air Force Base and Roberts Field.

Comparing the DMA-Star model with the TMA-Star model, we find that using time as the cost we seek to minimize essentially "weights" the cost of the route's distance and the cost of increased ground time due to fuel stops. This results in routes that are slightly longer than optimum, but fewer fuel stops, on average, resulting in lower total travel times.

Table 5. Comparison of DMA-Star and TMA-Star results.

	Avg. Distance (NMs)	Avg. Total Time (Hrs)	Avg. Number of Fuel Stops	Avg. Processing Time (Seconds)
Distance Minimization A-Star	1077.6	3.7	4.1	.06
Time Minimization A-Star	1085.5	3.2	3.5	.59

The greatest example of the disparity in total time occurs with the route between Lancaster, California and Jacksonville, North Carolina. As shown in Figures 14(a) and 14(b), the TMA-Star model finds a route that is 4 miles longer than the route found by the DMA-Star model, but includes 4 fewer fuel stops and saves 3.97 hours of total time.

Figure 14(a). Route between Lancaster and Jacksonville using the DMA-Star model.

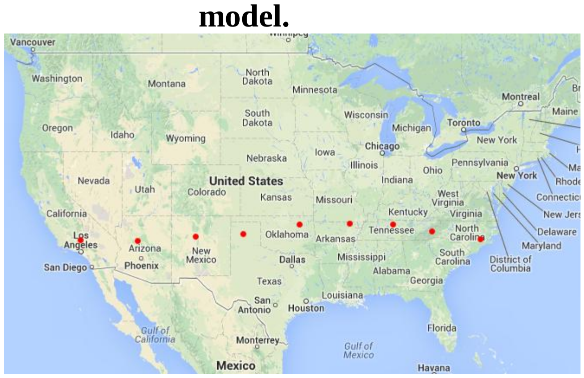


Figure 14(b). Route between Lancaster and Jacksonville using the TMA-Star model.

We now compare the optimum routes between the departure and arrival locations found using the TMA-Star route optimization model to the straight line distances used by FORSCOM for cost estimation. Table 6 shows a relatively small average difference between the two methods of distance estimation.

Table 6. Comparison of averaged results of distance estimates using the Time Minimization A-Star algorithm and straight line distance.

Estimation Method	Average Distance (NMs)
TMA-Star Route	1011.6
Straight Line Distance	996.1

While the straight line distance method generally provides acceptable distance estimates of feasible route distances, this is not always the case. The route between Key West International Airport, Florida and Brownsville South Padre International Airport, Texas provides the best example of a gross underestimation of route distance by using straight line distance. As shown in Figure 15, a direct route between the two airports is not possible (using a max fuel range of 300 NMs). Because of this, the straight line distance method underestimates the route distance by 295 NMs (35 percent) when compared to the feasible route found using the TMA-Star algorithm.

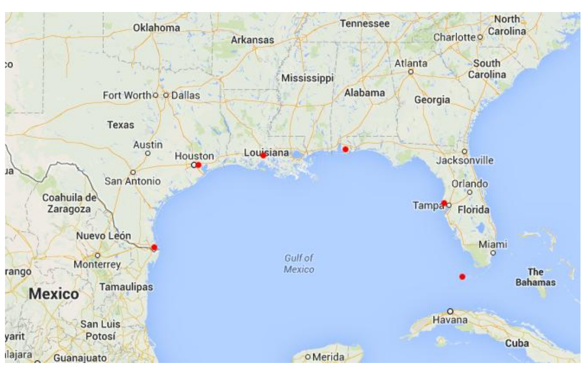


Figure 15. Route between Key West and Brownsville using the TMA-Star model.

Obstacle Avoidance

To evaluate the obstacle avoidance model, we randomly select 100 start nodes and 100 corresponding target nodes. To prevent excessive processing times we replace node pairings that result in a straight line distance longer than 300 NMs until all 100 node pairings have a straight line distance of 300 NMs or less. The map space used for testing contains no obstacles or undesirable areas. Testing using the A-Star algorithm, Dijkstra's algorithm, and a greedy heuristic produces the individual results shown in Appendix B and the aggregated results in Table 7. The A-Star algorithm and Dijsktra's algorithm both produce optimum routes, but the A-Star algorithm finds the route in a fraction of the time Dijsktra's takes and searches a much smaller space. The greedy heuristic performs well, generating routes within approximately 2 percent of optimum.

Table 7. Averaged results of 100 randomly selected obstacle avoidance iterations using the A-Star and Dijkstra's algorithms and the greedy heuristic.

	Avg. Distance (NMs)	Avg. Number of Nodes Explored	Avg. Processing Time (Seconds)
A-Star Algorithm	200.52	284.96	.17
Dijkstra's Algorithm	200.52	4195.18	28.59
Greedy Heuristic	204.01	31.93	.01

Figures 16 and 17 provide a visual comparison of the search area used by Dijkstra's algorithm and the A-Star algorithm and show the effectiveness of the heuristic in guiding A-Star's search toward the target node. While Dijkstra's algorithm expands the search in all directions, the directed A-Star search is concentrated on improving areas.

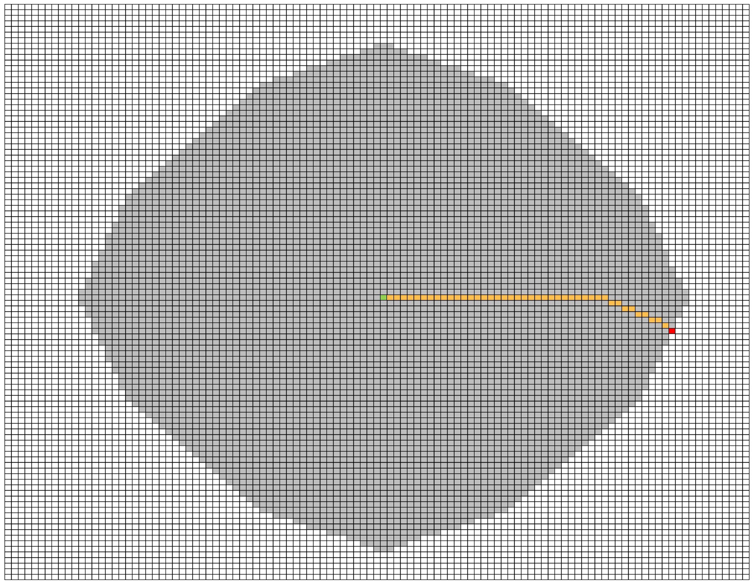


Figure 16. Obstacle avoidance route from Greer, SC to Wilmington, NC using Dijkstra's algorithm. The start node is shown in green, target node in red, route in orange, and explored nodes in grey.

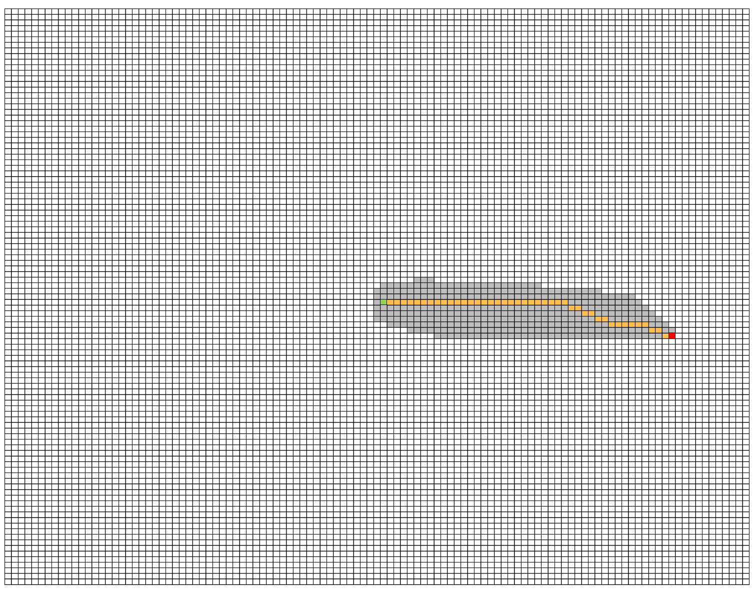


Figure 17. Obstacle avoidance route from Greer, SC to Wilmington, NC using the A-Star algorithm.

V. Conclusions and Future Research

In today's fiscally constrained military environment, accurate cost estimation and efficient use of resources are predominant concerns. The ADERPT is effective in quickly finding efficient flight routes and is a useful tool for cost estimation and air mission planning. While current distance estimation procedures employed by FORSCOM are in most cases sufficient, testing showed that the straight line distance estimation technique grossly underestimated the length of a feasible route on multiple occasions, and by as much as 78 percent. The Time Minimization A-Star model's (TMA-Star) use of time as "cost" results in routes that simultaneously minimize flight distance and fuel stops. This approach is more consistent with aircrew mission planning, and results in routes that minimize TDY costs.

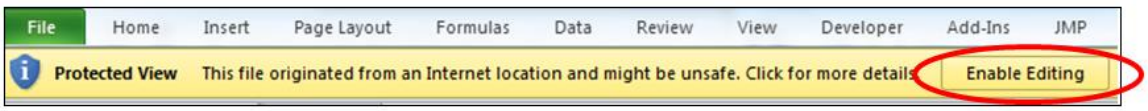
One limitation of the route optimization model is that it does not maximize the route distance traveled within the limitations of aircrew daily flight hour and duty day restrictions. Future efforts could focus on a multicriteria optimization approach to address this issue.

The proposed obstacle avoidance model provides a proof of concept for the use of a grid based network in routing aircraft around obstacles. The A-Star algorithm proved superior to the other methods tested in terms of route distance and processing time. The obstacle avoidance model concept has potential for use as both a route planning tool as well as a dynamic, in-cockpit, navigation aid. Future work should translate the model to a more applicable programming language, and improve both the shape and resolution of the tessellation.

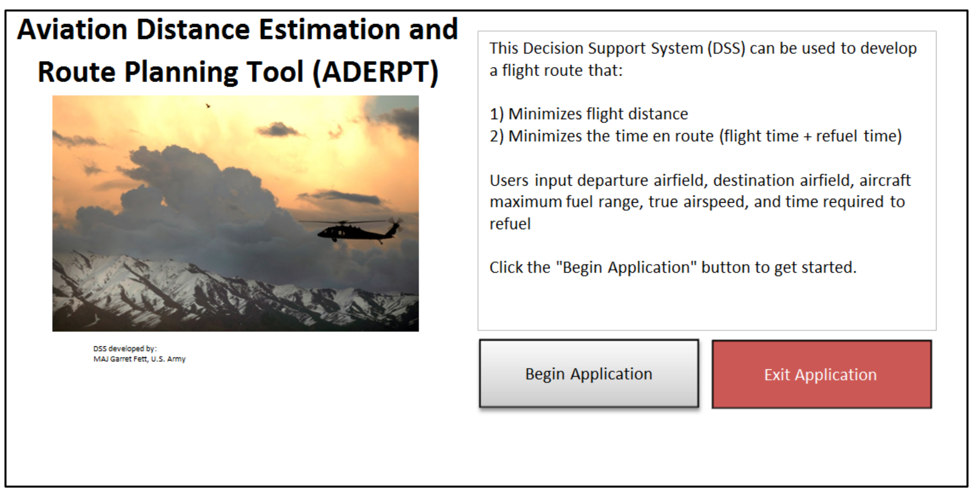
Appendix A: Model Guide

Overview

Upon opening the model, ensure you click “Enable” on the alert banners at the top of the screen.

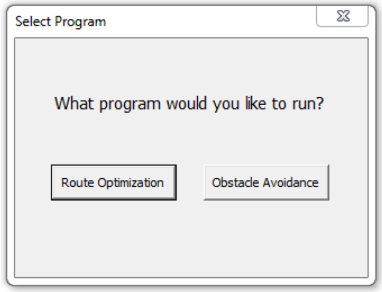


The model home screen provides a description of the model functions and user inputs. Click the “Begin Application” button to start the program.

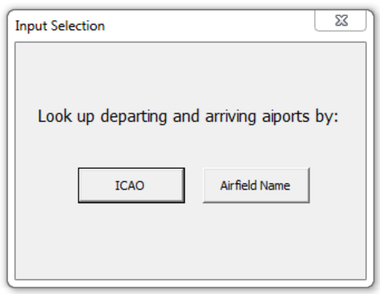


Route Optimization

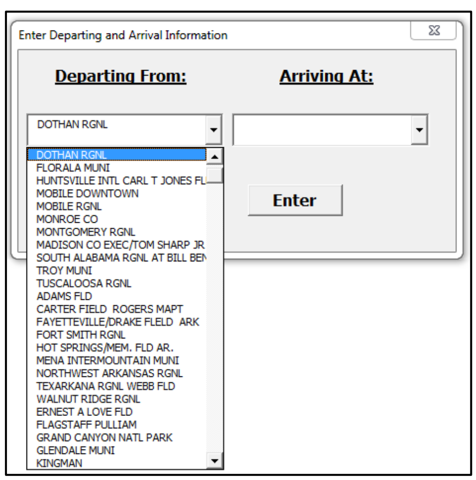
A pop-up window allows the user to select the desired program. Click the “Route Optimization” button to continue.



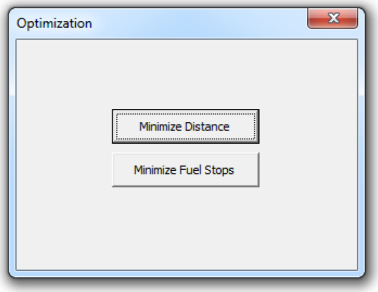
Users can input the departing and arriving airports using International Civil Aviation Organization (ICAO) airport code, or by using the airfield name. Click on the desired option.



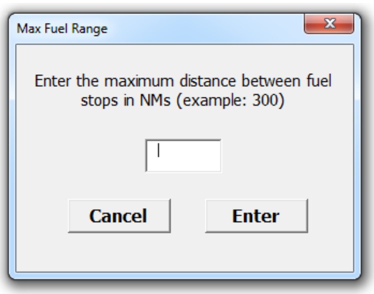
The user is then prompted to enter the departing and arriving airports using the selected method. The airports can be selected from the dropdown menus, or typed in the text box.



The next user input window provides the user with two types of route optimization to choose from. Choosing "Minimize Distance" will select fuel stops that result in the shortest possible route. The "Minimize Fuel Stops" results in a route that may not be the shortest distance, but requires the fewest number of fuel stops to reach the destination.



The final user input window asks the user to enter the maximum distance allowed between fuel stops. The user enters this distance as a number in the text box and clicks on the enter button.



After inputting the max fuel range, the model executes the appropriate algorithm and displays the route information as shown in the screen shot below. The user then has the option to return to the model home screen, exit the program, or view a map of the route.

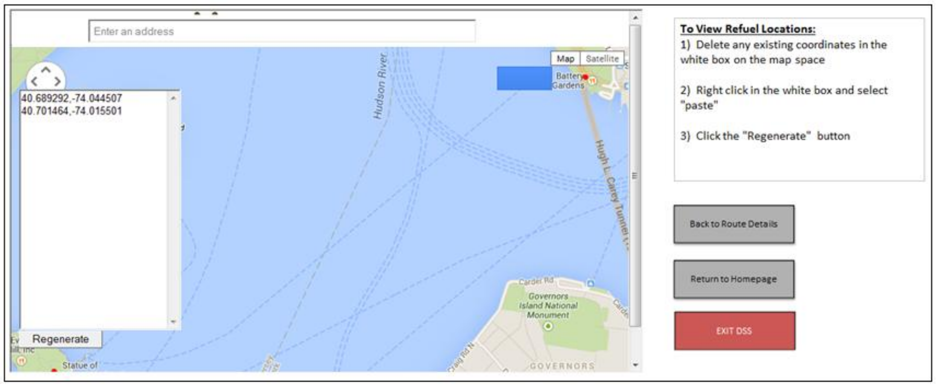
	ICAO	Location	State	LAT	LONG	Leg Dist. (NMs)	Total Dist. (NMs)	Leg Flight Time (Hrs)	Total Flight Time (Hrs)	Total Time (Hrs)
Start Point	KILM	NEW HANOVER INTL	NC	34.270603	-77.902558	0.0	0.0	0.0	0.0	0.0
Fuel Stop 1	KAVL	ASHEVILLE RGNL	NC	35.436194	-82.541806	239.0	239.0	1.7	1.7	2.7
Fuel Stop 2	KEOD	SABRE ARMY AIRFIELD	TN	36.5682	-87.4808	249.3	488.3	1.8	3.5	5.5
Fuel Stop 3	KSGF	SPRINGFIELD BRANSON NATL	MO	37.2457	-93.3886	286.5	774.8	2.0	5.5	8.5
Fuel Stop 4	KHYS	HAYS MUNICIPAL AIRPORT HAYS	KS	38.8422	-99.2732	294.2	1069.0	2.1	7.6	11.6
Fuel Stop 5	KCOS	CITY OF COLORADO SPRINGS MUNI (PETERSON FLD)	CO	38.8058	-104.7008	253.9	1322.9	1.8	9.4	14.4
Fuel Stop 6	KGJT	GRAND JUNCTION RGNL	CO	39.1224	-108.5267	179.6	1502.5	1.3	10.7	16.7
Fuel Stop 7	KENV	WENDOVER (DECKER AAF)	UT	40.7187	-114.0309	270.9	1773.4	1.9	12.7	19.7
Fuel Stop 8	KRNO	RENO/TAHOE INTL	NV	39.4991	-119.7681	273.4	2046.8	2.0	14.6	22.6
Destination	KMCC	MC CLELLAN AFLD SACRAMENTO	CA	38.6676	-121.4006	91.0	2137.8	0.6	15.3	23.3

Return to Homepage

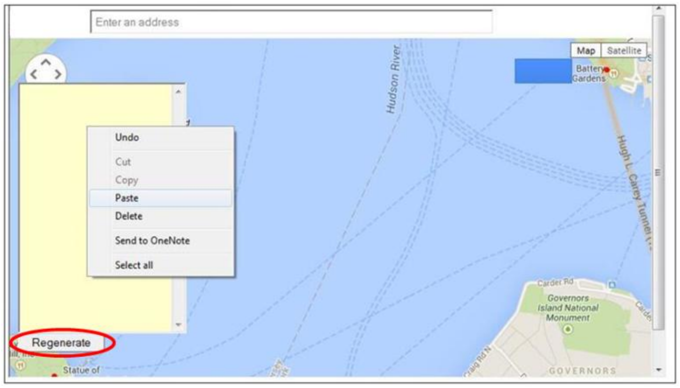
Map Refuel Locations

EXIT DSS

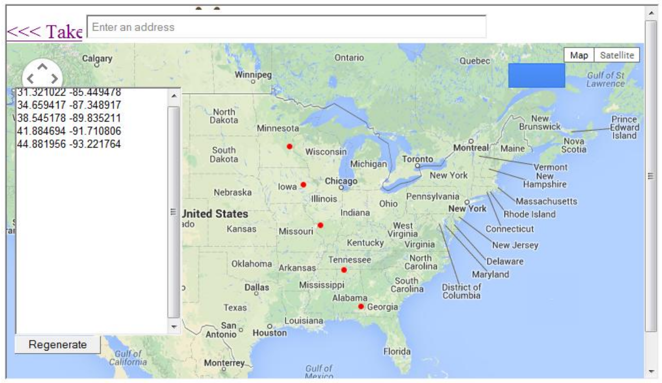
Clicking on “Map Refuel Locations” will display the map shown in the figure below. To use this function, the user must have internet access. To display the route, first delete any coordinates contained in the white box on the left side of the map.



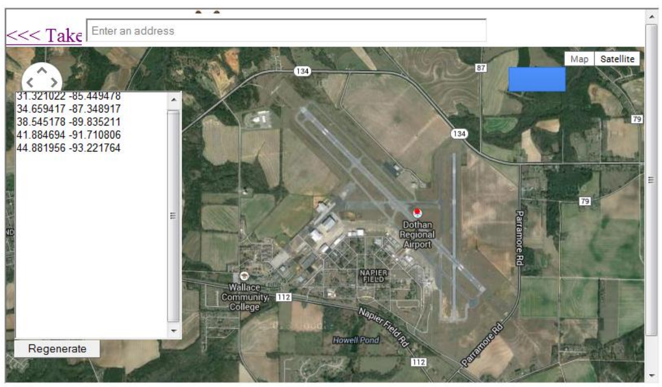
Then right click in the white box and select “paste.” Finally, click the “Regenerate” button located below the white box.



The map then displays the route with red dots identifying the starting airport, all refuel locations, and the destination airport as shown in the figure below.

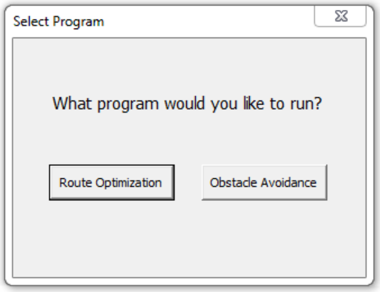


The user can click and drag on the map to move around the map space. Additionally, adjusting the mouse scroll wheel allows the user to zoom in on desired areas as shown in the figure below.

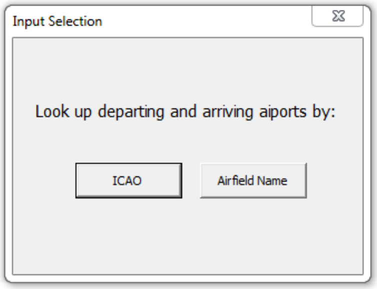


Obstacle Avoidance

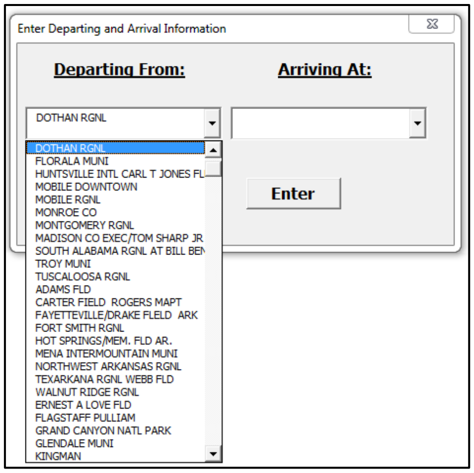
A pop-up window allows the user to select the desired program. Click the "Obstacle Avoidance" button to continue.



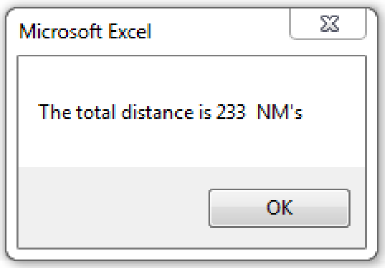
Users can input the departing and arriving airports using International Civil Aviation Organization (ICAO) airport code, or by using the airfield name. Click on the desired option.



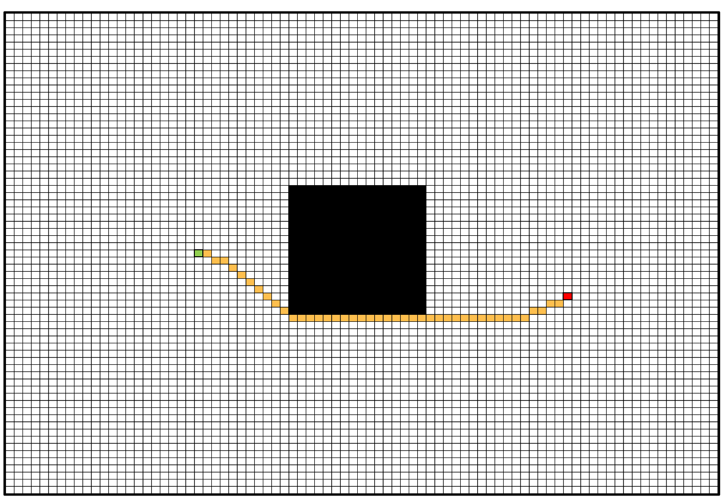
The user is then prompted to enter the departing and arriving airports using the selected method. The airports can be selected from the dropdown menus, or typed in the text box. Click "Enter" when finished.



Upon clicking the “Enter” button, the obstacle avoidance algorithm will find the optimum route between the departing and arriving location. A message box will be displayed with the route length and a visual depiction of the route will be shown.



The departing location is shown in green, arriving location in red, and route in orange.



To return to the home page, click the “Intro” tab at the bottom of the screen.

Appendix B: Iterative Model Results

Table 8. Individual results of 100 randomly selected Distance Minimization Route Optimization iterations using the A-Star and Dijkstra's algorithms and the greedy heuristic (1 of 2).

Iteration	A-Star				Greedy				Dijkstra's				Straight Line Distance
	Distance	Time	Nodes Explored	Fuel Stops	Distance	Time (Sec)	Nodes Explored	Fuel Stops	Distance	Time	Nodes Explored	Fuel Stops	Distance
1	1859.8	0.07	13	7	1934.1	0.05	9	7	1859.8	1.89	404	7	1829.4
2	1017.6	0.06	12	4	1044.4	0.02	5	3	1017.6	1.57	330	4	1010.9
3	1244.7	0.05	11	5	1407.7	0.03	7	5	1244.7	1.98	429	5	1214.1
4	964.8	0.07	12	4	981.0	0.02	5	3	964.8	1.24	255	4	958.5
5	1611.4	0.14	32	7	1657.8	0.03	7	5	1611.4	1.37	301	7	1596.9
6	1617.2	0.12	25	7	1695.8	0.03	7	5	1617.2	2.02	435	7	1611.6
7	1153.8	0.07	13	5	1157.6	0.02	5	3	1153.8	1.23	258	5	1152.8
8	788.8	0.03	6	2	796.3	0.02	4	2	788.8	1.01	206	2	787.7
9	325.7	0.02	3	1	365.3	0.01	3	1	325.7	0.28	58	1	325.2
10	1714.9	0.08	17	8	1777.9	0.04	8	6	1714.9	1.36	296	8	1701.5
11	2108.4	0.15	33	9	2197.6	0.05	9	7	2108.4	1.89	409	9	2090.5
12	710.1	0.03	5	2	713.5	0.02	4	2	710.1	1.33	279	2	708.7
13	636.3	0.02	4	2	702.3	0.02	4	2	636.3	0.80	171	2	636.2
14	844.8	0.03	5	3	854.8	0.02	4	2	844.8	1.33	282	3	844.4
15	931.2	0.06	12	3	1017.2	0.02	5	3	931.2	1.43	299	3	929.5
16	392.4	0.02	3	1	400.0	0.02	3	1	392.4	0.60	123	1	392.0
17	915.3	0.06	11	3	957.7	0.02	5	3	915.3	1.62	345	3	912.6
18	571.2	0.02	5	2	587.2	0.02	4	2	571.2	1.01	206	2	569.5
19	474.0	0.02	4	1	478.6	0.01	3	1	474.0	0.88	178	1	473.5
20	792.8	0.03	6	3	794.2	0.02	4	2	792.8	0.64	147	3	776.7
21	587.9	0.03	5	2	588.9	0.01	3	1	587.9	1.06	218	2	585.0
22	648.4	0.05	10	3	715.9	0.02	4	2	648.4	0.88	184	3	639.0
23	325.1	0.02	3	1	325.4	0.01	3	1	325.1	0.46	92	1	325.0
24	1112.5	0.09	19	4	1178.8	0.03	6	4	1112.5	1.79	380	4	1086.1
25	1493.5	0.10	20	5	1535.3	0.03	7	5	1493.5	1.65	348	5	1492.5
26	961.7	0.03	6	3	1004.5	0.02	5	3	961.7	1.60	336	3	961.2
27	1158.2	0.04	9	4	1206.0	0.03	6	4	1158.2	1.39	290	4	1157.8
28	796.6	0.04	7	3	809.2	0.02	4	2	796.6	1.46	304	3	796.1
29	756.0	0.02	4	2	776.8	0.02	4	2	756.0	1.20	250	2	755.9
30	428.9	0.02	4	1	443.6	0.01	3	1	428.9	0.23	54	1	427.4
31	1406.6	0.09	18	6	1445.8	0.03	6	4	1406.6	1.39	293	6	1405.6
32	804.5	0.05	10	2	824.2	0.02	4	2	804.5	1.49	311	2	802.9
33	1554.2	0.07	16	6	1669.7	0.04	8	6	1554.2	1.88	406	6	1486.0
34	784.5	0.02	4	2	784.5	0.02	4	2	784.5	0.98	201	2	782.4
35	793.6	0.04	7	3	890.7	0.02	5	3	793.6	0.57	130	3	745.2
36	1189.3	0.04	8	4	1234.2	0.03	6	4	1189.3	1.75	374	4	1184.5
37	1769.9	0.08	17	8	1817.1	0.04	8	6	1769.9	1.42	310	8	1744.8
38	1025.0	0.03	6	4	1045.9	0.02	5	3	1025.0	1.61	340	4	1022.0
39	667.9	0.03	6	2	723.3	0.02	4	2	667.9	1.27	264	2	667.3
40	521.4	0.02	4	2	528.0	0.01	3	1	521.4	0.54	115	2	521.2
41	1371.2	0.05	10	5	1380.2	0.03	6	4	1371.2	1.34	282	5	1367.9
42	409.1	0.02	3	1	441.8	0.01	3	1	409.1	0.27	61	1	409.1
43	364.4	0.02	3	1	364.4	0.01	3	1	364.4	0.36	74	1	364.1
44	381.6	0.02	3	1	383.7	0.01	3	1	381.6	0.57	118	1	380.5
45	437.7	0.02	3	1	437.9	0.01	3	1	437.7	0.62	128	1	437.7
46	1079.9	0.04	9	3	1106.9	0.02	5	3	1079.9	0.77	173	3	1077.1
47	1670.4	0.12	24	6	1675.0	0.03	7	5	1670.4	1.79	389	6	1659.8
48	1951.1	0.17	33	9	2015.7	0.05	9	7	1951.1	1.92	414	9	1944.9
49	999.1	0.05	9	4	1002.0	0.02	5	3	999.1	1.51	317	4	998.3
50	557.0	0.05	9	1	562.5	0.01	3	1	557.0	0.86	176	1	553.8

Table 9. Individual results of 100 randomly selected Distance Minimization Route Optimization iterations using the A-Star and Dijkstra's algorithms and the greedy heuristic (2 of 2).

Iteration	A-Star				Greedy				Dijkstra's				Straight Line Distance
	Distance	Time	Nodes Explored	Fuel Stops	Distance	Time (Sec)	Nodes Explored	Fuel Stops	Distance	Time	Nodes Explored	Fuel Stops	Distance
51	1103.0	0.07	15	4	1103.8	0.03	6	4	1103.0	0.70	159	4	1089.4
52	628.2	0.02	4	2	656.7	0.02	4	2	628.2	0.42	87	2	628.0
53	619.1	0.02	4	2	649.9	0.02	4	2	619.1	0.96	203	2	618.8
54	1797.5	0.10	23	7	1942.2	0.04	9	7	1797.5	1.35	296	7	1763.7
55	1105.1	0.11	22	3	1131.7	0.03	6	4	1105.1	1.69	358	3	1093.9
56	1035.6	0.05	10	3	1162.7	0.03	6	4	1035.6	1.27	268	3	1028.4
57	965.8	0.04	6	3	1007.4	0.02	5	3	965.8	1.61	342	3	965.1
58	1834.1	0.09	17	8	1862.7	0.04	8	6	1834.1	1.94	415	8	1825.7
59	329.7	0.02	3	1	335.9	0.01	3	1	329.7	0.45	91	1	329.7
60	566.8	0.03	5	2	576.7	0.02	4	2	566.8	0.65	138	2	566.6
61	1335.5	0.07	14	6	1557.0	0.03	7	5	1335.5	1.91	414	6	1291.1
62	626.7	0.03	4	2	635.8	0.02	4	2	626.7	1.03	212	2	626.2
63	774.0	0.03	5	3	816.7	0.02	4	2	774.0	1.08	226	3	773.1
64	967.8	0.06	11	3	1034.6	0.02	5	3	967.8	1.32	277	3	966.3
65	1941.9	0.20	38	8	2050.2	0.04	9	7	1941.9	2.00	433	8	1929.2
66	2198.0	0.30	65	11	2440.6	0.05	10	8	2198.0	2.00	430	11	2149.8
67	847.5	0.02	5	3	872.4	0.02	5	3	847.5	0.68	148	3	847.0
68	500.1	0.02	3	1	500.1	0.01	3	1	500.1	0.47	103	1	497.9
69	1711.5	0.18	39	7	1871.1	0.04	8	6	1711.5	2.04	439	7	1684.7
70	876.4	0.05	7	3	888.8	0.02	4	2	876.4	1.54	325	3	874.9
71	993.3	0.04	8	4	1015.2	0.02	5	3	993.3	1.13	237	4	978.0
72	1348.7	0.04	8	5	1475.0	0.03	7	5	1348.7	1.38	301	5	1321.4
73	768.1	0.03	5	3	782.6	0.02	4	2	768.1	0.50	115	3	766.0
74	661.5	0.02	4	2	675.1	0.02	4	2	661.5	0.75	157	2	661.3
75	542.5	0.02	4	2	543.7	0.01	3	1	542.5	0.27	58	2	542.4
76	918.9	0.03	7	3	935.9	0.02	5	3	918.9	0.59	130	3	909.8
77	2258.8	0.16	34	10	2292.8	0.04	9	7	2258.8	1.96	422	10	2243.4
78	706.7	0.03	5	3	724.5	0.02	4	2	706.7	0.54	123	3	703.6
79	1136.7	0.05	10	4	1190.6	0.03	6	4	1136.7	1.41	298	4	1116.9
80	587.9	0.07	13	2	607.9	0.02	4	2	587.9	1.06	218	2	571.4
81	788.3	0.05	9	3	797.0	0.02	4	2	788.3	1.25	261	3	787.1
82	1196.2	0.04	8	4	1208.8	0.02	6	4	1196.2	0.89	201	4	1195.6
83	1294.8	0.08	17	5	1359.6	0.03	7	5	1294.8	1.70	363	5	1273.5
84	685.8	0.02	5	2	685.8	0.02	4	2	685.8	0.32	73	2	547.4
85	876.5	0.06	13	3	881.3	0.02	5	3	876.5	1.52	318	3	872.3
86	879.7	0.04	8	3	1006.9	0.02	5	3	879.7	0.57	132	3	873.3
87	1258.3	0.07	13	5	1293.1	0.03	6	4	1258.3	1.50	315	5	1256.3
88	1298.4	0.10	20	6	1346.2	0.03	6	4	1298.4	1.93	411	6	1290.3
89	1454.0	0.05	10	5	1468.7	0.03	7	5	1454.0	1.43	313	5	1398.3
90	437.0	0.02	3	1	438.8	0.01	3	1	437.0	0.25	59	1	419.6
91	1153.1	0.05	10	4	1176.2	0.02	5	3	1153.1	1.34	282	4	1151.9
92	1121.1	0.04	7	4	1133.6	0.02	5	3	1121.1	1.47	318	4	1120.4
93	982.8	0.05	10	5	1000.7	0.02	5	3	982.8	1.41	296	5	981.5
94	498.2	0.03	5	1	512.7	0.01	3	1	498.2	0.78	157	1	497.8
95	675.8	0.05	9	2	700.5	0.02	4	2	675.8	0.82	172	2	673.6
96	1120.9	0.06	12	4	1167.9	0.03	6	4	1120.9	1.48	309	4	1118.9
97	916.5	0.05	11	3	1057.7	0.02	5	3	916.5	0.45	105	3	903.4
98	1376.5	0.04	8	5	1461.4	0.03	7	5	1376.5	1.75	373	5	1371.6
99	1322.9	0.06	12	6	1388.6	0.03	7	5	1322.9	0.83	187	6	1294.1
100	1003.5	0.03	6	3	1012.2	0.02	5	3	1003.5	1.01	221	3	1001.1

Table 10. Comparison of DMA-Star route distances and straight line distance estimations (1 of 2).

Iteration	A-Star		Straight Line Distance	Difference	
	Distance (NMs)	Fuel Stops	Distance (NMs)	NMs	%
1	1859.8	7	1829.4	30.47	1.67%
2	1017.6	4	1010.9	6.75	0.67%
3	1244.7	5	1214.1	30.60	2.52%
4	964.8	4	958.5	6.35	0.66%
5	1611.4	7	1596.9	14.45	0.90%
6	1617.2	7	1611.6	5.57	0.35%
7	1153.8	5	1152.8	0.97	0.08%
8	788.8	2	787.7	1.05	0.13%
9	325.7	1	325.2	0.50	0.15%
10	1714.9	8	1701.5	13.38	0.79%
11	2108.4	9	2090.5	17.91	0.86%
12	710.1	2	708.7	1.39	0.20%
13	636.3	2	636.2	0.05	0.01%
14	844.8	3	844.4	0.38	0.04%
15	931.2	3	929.5	1.73	0.19%
16	392.4	1	392.0	0.35	0.09%
17	915.3	3	912.6	2.71	0.30%
18	571.2	2	569.5	1.74	0.31%
19	474.0	1	473.5	0.42	0.09%
20	792.8	3	776.7	16.15	2.08%
21	587.9	2	585.0	2.97	0.51%
22	648.4	3	639.0	9.33	1.46%
23	325.1	1	325.0	0.16	0.05%
24	1112.5	4	1086.1	26.35	2.43%
25	1493.5	5	1492.5	1.03	0.07%
26	961.7	3	961.2	0.44	0.05%
27	1158.2	4	1157.8	0.45	0.04%
28	796.6	3	796.1	0.49	0.06%
29	756.0	2	755.9	0.06	0.01%
30	428.9	1	427.4	1.49	0.35%
31	1406.6	6	1405.6	1.00	0.07%
32	804.5	2	802.9	1.61	0.20%
33	1554.2	6	1486.0	68.24	4.59%
34	784.5	2	782.4	2.08	0.27%
35	793.6	3	745.2	48.33	6.49%
36	1189.3	4	1184.5	4.73	0.40%
37	1769.9	8	1744.8	25.11	1.44%
38	1025.0	4	1022.0	2.99	0.29%
39	667.9	2	667.3	0.63	0.09%
40	521.4	2	521.2	0.20	0.04%
41	1371.2	5	1367.9	3.26	0.24%
42	409.1	1	409.1	0.00	0.00%
43	364.4	1	364.1	0.27	0.07%
44	381.6	1	380.5	1.13	0.30%
45	437.7	1	437.7	0.00	0.00%
46	1079.9	3	1077.1	2.81	0.26%
47	1670.4	6	1659.8	10.58	0.64%
48	1951.1	9	1944.9	6.20	0.32%
49	999.1	4	998.3	0.76	0.08%
50	557.0	1	553.8	3.18	0.57%

Table 11. Comparison of DMA-Star route distances and straight line distance estimations (2 of 2).

Iteration	A-Star		Straight Line Distance	Difference	
	Distance (NMs)	Fuel Stops	Distance (NMs)	NMs	%
51	1103.0	4	1089.4	13.57	1.25%
52	628.2	2	628.0	0.23	0.04%
53	619.1	2	618.8	0.36	0.06%
54	1797.5	7	1763.7	33.76	1.91%
55	1105.1	3	1093.9	11.23	1.03%
56	1035.6	3	1028.4	7.24	0.70%
57	965.8	3	965.1	0.70	0.07%
58	1834.1	8	1825.7	8.42	0.46%
59	329.7	1	329.7	0.01	0.00%
60	566.8	2	566.6	0.20	0.04%
61	1335.5	6	1291.1	44.44	3.44%
62	626.7	2	626.2	0.55	0.09%
63	774.0	3	773.1	0.91	0.12%
64	967.8	3	966.3	1.58	0.16%
65	1941.9	8	1929.2	12.70	0.66%
66	2198.0	11	2149.8	48.17	2.24%
67	847.5	3	847.0	0.53	0.06%
68	500.1	1	497.9	2.18	0.44%
69	1711.5	7	1684.7	26.79	1.59%
70	876.4	3	874.9	1.51	0.17%
71	993.3	4	978.0	15.24	1.56%
72	1348.7	5	1321.4	27.24	2.06%
73	768.1	3	766.0	2.08	0.27%
74	661.5	2	661.3	0.15	0.02%
75	542.5	2	542.4	0.09	0.02%
76	918.9	3	909.8	9.03	0.99%
77	2258.8	10	2243.4	15.40	0.69%
78	706.7	3	703.6	3.12	0.44%
79	1136.7	4	1116.9	19.81	1.77%
80	587.9	2	571.4	16.57	2.90%
81	788.3	3	787.1	1.21	0.15%
82	1196.2	4	1195.6	0.52	0.04%
83	1294.8	5	1273.5	21.32	1.67%
84	685.8	2	547.4	138.34	25.27%
85	876.5	3	872.3	4.22	0.48%
86	879.7	3	873.3	6.38	0.73%
87	1258.3	5	1256.3	1.95	0.16%
88	1298.4	6	1290.3	8.04	0.62%
89	1454.0	5	1398.3	55.66	3.98%
90	437.0	1	419.6	17.43	4.15%
91	1153.1	4	1151.9	1.19	0.10%
92	1121.1	4	1120.4	0.67	0.06%
93	982.8	5	981.5	1.31	0.13%
94	498.2	1	497.8	0.48	0.10%
95	675.8	2	673.6	2.13	0.32%
96	1120.9	4	1118.9	2.07	0.19%
97	916.5	3	903.4	13.13	1.45%
98	1376.5	5	1371.6	4.89	0.36%
99	1322.9	6	1294.1	28.78	2.22%
100	1003.5	3	1001.1	2.40	0.24%

Table 12. Results of 100 obstacle avoidance iterations using A-Star and Dijkstra's algorithms and the greedy heuristic (1 of 2).

Iteration	A-Star			Greedy			Dijkstra's		
	Distance (NMs)	Time (Sec)	Nodes Explored	Distance (NMs)	Time (Sec)	Nodes Explored	Distance (NMs)	Time (Sec)	Nodes Explored
1	233	0.11	277	245	0.02	43	233	38.94	5218
2	162	0.03	128	166	0.00	25	162	3.91	1834
3	200	0.10	268	200	0.01	34	200	14.95	3621
4	264	0.31	472	264	0.01	39	264	40.84	6226
5	68	0.00	13	68	0.00	13	68	0.21	380
6	61	0.00	19	61	0.00	11	61	0.12	287
7	110	0.01	61	110	0.00	16	110	0.82	806
8	203	0.07	213	209	0.00	31	203	14.80	3486
9	274	0.31	503	274	0.01	47	274	53.51	6984
10	269	0.59	719	269	0.01	34	269	57.70	7341
11	146	0.00	24	154	0.00	24	146	2.32	1418
12	281	0.22	403	281	0.01	52	281	64.88	7684
13	56	0.00	13	56	0.00	8	56	0.05	182
14	211	0.18	383	211	0.00	29	211	19.38	4163
15	63	0.00	23	63	0.00	9	63	0.13	295
16	285	0.39	566	291	0.01	42	285	40.10	6127
17	280	0.27	471	280	0.01	44	280	52.37	6961
18	252	0.44	616	252	0.01	32	252	38.46	6044
19	24	0.00	3	24	0.00	3	24	0.00	28
20	248	0.01	49	266	0.01	49	248	41.57	6187
21	251	0.50	663	253	0.01	37	251	44.97	6445
22	260	0.24	430	260	0.01	46	260	43.67	6391
23	288	0.63	754	294	0.01	39	288	69.84	8017
24	276	0.20	386	280	0.01	39	276	36.79	5817
25	46	0.00	13	46	0.00	8	46	0.04	155
26	159	0.04	138	165	0.01	24	159	5.20	2153
27	233	0.32	511	245	0.01	50	233	32.66	5636
28	146	0.03	137	152	0.00	25	146	4.18	1863
29	228	0.17	370	232	0.01	31	228	20.36	4323
30	74	0.00	30	74	0.00	13	74	0.25	434
31	196	0.16	357	196	0.01	25	196	14.60	3674
32	214	0.09	250	218	0.01	32	214	12.14	3305
33	222	0.09	254	222	0.00	34	222	15.36	3554
34	242	0.10	261	242	0.01	38	242	19.00	4212
35	234	0.08	228	242	0.01	37	234	16.75	3924
36	136	0.03	122	136	0.01	20	136	2.94	1585
37	240	0.18	375	240	0.01	42	240	31.68	5379
38	258	0.01	51	270	0.01	51	258	49.16	6755
39	196	0.02	78	204	0.00	32	196	8.21	2696
40	288	0.05	170	287	0.01	47	288	39.45	5924
41	267	0.47	649	273	0.01	36	267	51.70	6915
42	168	0.05	182	172	0.00	23	168	5.68	2245
43	262	0.22	414	262	0.01	47	262	46.38	6544
44	240	0.17	354	240	0.01	36	240	27.56	5103
45	266	0.62	695	266	0.01	34	266	56.10	7120
46	245	0.48	635	251	0.01	41	245	61.40	7543
47	124	0.01	76	124	0.00	19	124	1.30	1038
48	14	0.00	2	14	0.00	2	14	0.00	2
49	34	0.00	7	34	0.00	5	34	0.01	63
50	328	0.49	650	328	0.02	59	328	122.13	10398

Table 13. Results of 100 obstacle avoidance iterations using A-Star and Dijkstra's algorithms and the greedy heuristic (2 of 2).

Iteration	A-Star			Greedy			Dijkstra's		
	Distance (NMs)	Time (Sec)	Nodes Explored	Distance (NMs)	Time (Sec)	Nodes Explored	Distance (NMs)	Time (Sec)	Nodes Explored
51	165	0.09	262	165	0.00	21	165	7.95	2647
52	304	0.27	466	304	0.01	47	304	48.27	6758
53	158	0.05	191	158	0.00	21	158	4.98	2118
54	181	0.06	205	181	0.01	32	181	10.88	3002
55	153	0.05	191	153	0.00	21	153	4.42	1931
56	266	0.44	618	278	0.01	50	266	79.40	8450
57	300	0.34	523	304	0.01	42	300	54.47	7150
58	230	0.15	339	242	0.01	37	230	23.80	4653
59	108	0.02	89	108	0.00	15	108	1.21	999
60	271	0.37	559	271	0.01	41	271	48.10	6548
61	202	0.13	321	206	0.01	27	202	13.68	3506
62	288	0.02	57	306	0.01	57	288	78.79	8432
63	320	0.56	690	320	0.02	55	320	99.23	9612
64	184	0.03	120	196	0.01	35	184	11.64	3273
65	328	0.43	601	328	0.02	50	328	96.56	9557
66	166	0.04	161	166	0.01	23	166	4.94	2111
67	158	0.08	231	158	0.00	20	158	6.52	2388
68	251	0.25	425	251	0.01	43	251	35.32	5797
69	222	0.14	322	234	0.01	36	222	21.00	4392
70	172	0.04	157	172	0.00	26	172	5.03	2089
71	68	0.00	13	74	0.00	13	68	0.21	379
72	210	0.20	402	210	0.01	27	210	19.17	4172
73	156	0.01	71	156	0.01	30	156	6.04	2339
74	33	0.00	6	33	0.00	6	33	0.01	71
75	273	0.57	707	273	0.01	36	273	58.16	7350
76	68	0.00	11	68	0.00	11	68	0.21	273
77	140	0.00	23	144	0.00	23	140	2.04	1309
78	173	0.05	179	185	0.01	31	173	8.55	2754
79	74	0.01	35	74	0.00	10	74	0.22	405
80	296	0.31	470	296	0.01	45	296	43.43	6402
81	180	0.09	257	184	0.00	24	180	8.88	2800
82	124	0.02	95	124	0.00	17	124	1.65	1168
83	198	0.05	179	198	0.00	31	198	8.59	2751
84	202	0.05	164	210	0.01	32	202	8.89	2859
85	84	0.00	30	88	0.00	13	84	0.25	431
86	54	0.00	9	58	0.00	9	54	0.05	169
87	125	0.02	96	131	0.00	22	125	2.30	1389
88	128	0.00	25	140	0.00	25	128	2.75	1535
89	278	0.18	359	278	0.01	52	278	62.70	7645
90	344	0.65	751	362	0.02	61	344	132.66	11311
91	222	0.37	565	222	0.01	36	222	43.17	6277
92	300	0.27	471	300	0.01	46	300	47.39	6591
93	288	0.06	170	296	0.01	47	288	38.48	5919
94	236	0.09	251	244	0.01	37	236	17.60	3957
95	190	0.02	74	190	0.00	31	190	7.00	2489
96	308	0.66	780	308	0.01	41	308	82.91	8496
97	312	0.32	515	316	0.01	44	312	62.02	7612
98	326	0.52	669	338	0.01	58	326	110.12	10194
99	308	0.01	61	326	0.02	61	308	101.78	9683
100	102	0.01	64	108	0.00	18	102	1.03	890

Appendix C: VBA Code

Route Optimization

```
'The Route Optimization program uses the A-Star algorithm to find fuel stops between a departure and
arrival airfield that minimize the route cost (distance or time – depending on user selection).

Option Explicit
Public Arriving_Airfield As String
Public Departure_Airfield As String
Public StartNum As Integer
Public EndNum As Integer
Public Cancel1 As Boolean
Public MaxRange As Long
Public True_AS As Double
Public RefuelTime As Double
Public Type Node
Num As Integer
ParID As Integer
ScoreF As Double
ScoreG1 As Double
ScoreG2 As Double
ScoreH As Double
Open As Boolean
Closed As Boolean

End Type

Sub RouteOptimizationAdmin()

Call Clear_RouteOptimization  'Clears existing route information
Worksheets("Intro").Select
Input_Selection.Show  'Shows Departure/Arrival Point data entry form

If Cancel1 = True Then  'Returns to the homepage if user clicks "cancel"
Call Return_to_Homepage
Exit Sub
End If

End Sub

Sub Clear_RouteOptimization()
'This subroutine clears existing route information from the "Route" sheet

Sheets("Route").Select
Range("A2:L100").Select
Selection.ClearContents
Range("A1").Select

End Sub

Sub A_StarDistOptimization()
'This sub serves as the main framework for the Route Optimization A-Star algorithm

Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim NumNodes As Integer  'Number of Nodes in distance matrix

Dim CurNode As Node  'Current Node being evaluated from
Dim TestNode As Node
Dim BestNode As Node
Dim StartNode As Node
Dim GoalNode As Node
Dim Openlist(1 To 439) As Node

Application.ScreenUpdating = False
Max_Fuel_Range.Show
Airspeed.Show
Refuel_Delay.Show

StartNode.Num = StartNum
StartNode.ParID = 0
StartNode.ScoreF = 0
```

```
StartNode.ScoreG1 = 0
StartNode.Closed = False
StartNode.Open = True

GoalNode.Num = EndNum
CurNode = StartNode
NumNodes = 439

' Add the start node to the open list
Openlist(StartNode.Num) = StartNode
Openlist(StartNode.Num).Open = True

While CurNode.Num <> GoalNode.Num
Call Check_Open_Set(i, Openlist, NumNodes)  'Check to make sure the open list is not empty
Call Get_the_Best_Node(i, NumNodes, CurNode, BestNode, Openlist)  'Find the node with the best F-Score
If CurNode.Num = GoalNode.Num Then 'If the current node is the goal node, exit the loop
GoalNode = CurNode
Call Build_the_Route(StartNode, GoalNode, Openlist, True_AS)
End If

'Remove current node from open list and add to the closed list
Openlist(CurNode.Num).Open = False
Openlist(CurNode.Num).Closed = True

'Calculate F/G scores for all "neighbors"
For j = 1 To NumNodes
If Worksheets("Distance Matrix").Cells(CurNode.Num, j) <= MaxRange And CurNode.Num <> j Then
TestNode.Num = j
TestNode.ParID = CurNode.Num
TestNode.ScoreG1 = CurNode.ScoreG1 + Worksheets("Distance Matrix").Cells(CurNode.Num, TestNode.Num)
TestNode.ScoreH = Worksheets("Distance Matrix").Cells(TestNode.Num, GoalNode.Num)
TestNode.ScoreF = TestNode.ScoreG1 + TestNode.ScoreH
TestNode.Open = True
TestNode.Closed = False

'If the neighbor has not been evaluated, add it to the open list
If Openlist(TestNode.Num).Open = False And Openlist(TestNode.Num).Closed = False Then
Openlist(TestNode.Num) = TestNode
Openlist(TestNode.Num).Open = True
End If

'If the neighbor is on the open list, but this is a better path through it, update the parameters
If Openlist(TestNode.Num).Open = True Then
If TestNode.ScoreF < Openlist(TestNode.Num).ScoreF Then
Openlist(TestNode.Num) = TestNode       'Updated Node(j) with the best route and parent ID to reach it
End If
End If

'If the neighbor is on the closed list, but this is a better path through it, update the parameters
and put it back on the open list
If Openlist(TestNode.Num).Closed = True Then
If TestNode.ScoreF < Openlist(TestNode.Num).ScoreF Then
Openlist(TestNode.Num) = TestNode       'Updates Node(j) with the best route and parent ID to reach it
Openlist(TestNode.Num).Closed = False 'Removes the node from the closed list
Openlist(TestNode.Num).Open = True 'Places the node back on the open List
End If
End If
End If
Next j
Wend

Sheets("Route").Select

End Sub

Sub Get_the_Best_Node(i, Num As Integer, Cur As Node, Best As Node, Openlist() As Node)
'This subroutine designates the node with the best F-Score as the current node

Dim NumNodes As Integer
Dim BestNode As Node
Dim CurNode As Node

NumNodes = 439

'Set the BestNode.ScoreF = Big M
Best.ScoreF = 10000

'Cycles through all nodes to find the node with the lowest F-Score
For i = 1 To NumNodes
```

```
If Openlist(i).Open = True Then
If Openlist(i).ScoreF < Best.ScoreF Then
Best.ScoreF = Openlist(i).ScoreF
Cur = Openlist(i)
End If
End If
Next i

End Sub

Sub Check_Open_Set(i, Openlist() As Node, NumNodes)
'This subroutine returns an error message if there are no nodes in the Open List

Dim Test As Integer

'Cycles through all nodes, exits loop after finding a node on the open list
For i = 1 To NumNodes
If Openlist(i).Open = True Then
Test = 1
Exit For
End If
Next i

'Generates error code if there are no nodes on the open list
If Test = 0 Then
Exit Sub
MsgBox "Error"
End If

End Sub

Sub Build_the_Route(Start As Node, Goal As Node, Openlist() As Node, True_AS)
'This subroutine retraces the optimum route from Goal Node to Start Node

Dim k As Integer
Dim Route(1 To 439)

'Assigns the goal node number to the first entry in the "Route" array
k = 1
Route(k) = Goal.Num

'Continues entering route node numbers into "Route" array until reaching the start node
k = 2
While Openlist(Route(k - 1)).Num <> Start.Num
Route(k) = Openlist(Route(k - 1)).ParID
k = k + 1
Wend

Call Output_Route(Route(), k, True_AS)

End Sub

Sub Output_Route(Rte(), k As Integer, True_AS)
'This subroutine enters the route information into the "Route" output sheet

Dim i As Integer
Dim FuelStops As Integer

i = 2
k = k - 1

'Enters data for departure location into "Route" sheet
While i = 2
Worksheets("Route").Cells(i, 1).Value = "Start Point"
Worksheets("Route").Cells(i, 2).Value = Rte(k)
Worksheets("Route").Cells(i, 3).Value = Worksheets("Refuel Locations").Cells((Rte(k) + 1), 2)
Worksheets("Route").Cells(i, 4).Value = Worksheets("Refuel Locations").Cells((Rte(k) + 1), 3)
Worksheets("Route").Cells(i, 5).Value = Worksheets("Refuel Locations").Cells((Rte(k) + 1), 4)
Worksheets("Route").Cells(i, 6).Value = Worksheets("Refuel Locations").Cells((Rte(k) + 1), 5)
Worksheets("Route").Cells(i, 7).Value = Worksheets("Refuel Locations").Cells((Rte(k) + 1), 6)
Worksheets("Route").Cells(i, 8).Value = 0
Worksheets("Route").Cells(i, 9).Value = 0
Worksheets("Route").Cells(i, 10).Value = 0
Worksheets("Route").Cells(i, 11).Value = 0
Worksheets("Route").Cells(i, 12).Value = 0
k = k - 1
i = i + 1
Wend
```

```
'Enters data for all fuel stops into "Route" sheet
While k > 1
Worksheets("Route").Cells(i, 1).Value = "Fuel Stop " & i - 2
Worksheets("Route").Cells(i, 2).Value = Rte(k)
Worksheets("Route").Cells(i, 3).Value = Worksheets("Refuel Locations").Cells((Rte(k) + 1), 2)
Worksheets("Route").Cells(i, 4).Value = Worksheets("Refuel Locations").Cells((Rte(k) + 1), 3)
Worksheets("Route").Cells(i, 5).Value = Worksheets("Refuel Locations").Cells((Rte(k) + 1), 4)
Worksheets("Route").Cells(i, 6).Value = Worksheets("Refuel Locations").Cells((Rte(k) + 1), 5)
Worksheets("Route").Cells(i, 7).Value = Worksheets("Refuel Locations").Cells((Rte(k) + 1), 6)
Worksheets("Route").Cells(i, 8).Value = Worksheets("Distance
Matrix").Cells(Worksheets("Route").Cells(i, 2).Value, Worksheets("Route").Cells(i - 1, 2).Value)
Worksheets("Route").Cells(i, 9).Value = Worksheets("Route").Cells(i - 1, 9).Value +
Worksheets("Route").Cells(i, 8).Value
Worksheets("Route").Cells(i, 10).Value = (Worksheets("Route").Cells(i, 8).Value) / True_AS
Worksheets("Route").Cells(i, 11).Value = Worksheets("Route").Cells(i - 1, 11).Value +
Worksheets("Route").Cells(i, 10)
Worksheets("Route").Cells(i, 12).Value = Worksheets("Route").Cells(i - 1, 12).Value +
Worksheets("Route").Cells(i, 10).Value + RefuelTime
k = k - 1
i = i + 1
Wend

'Enters data for arrival location into "Route" sheet
Worksheets("Route").Cells(i, 1).Value = "Destination"
Worksheets("Route").Cells(i, 2).Value = Rte(k)
Worksheets("Route").Cells(i, 3).Value = Worksheets("Refuel Locations").Cells((Rte(k) + 1), 2)
Worksheets("Route").Cells(i, 4).Value = Worksheets("Refuel Locations").Cells((Rte(k) + 1), 3)
Worksheets("Route").Cells(i, 5).Value = Worksheets("Refuel Locations").Cells((Rte(k) + 1), 4)
Worksheets("Route").Cells(i, 6).Value = Worksheets("Refuel Locations").Cells((Rte(k) + 1), 5)
Worksheets("Route").Cells(i, 7).Value = Worksheets("Refuel Locations").Cells((Rte(k) + 1), 6)
Worksheets("Route").Cells(i, 8).Value = Worksheets("Distance
Matrix").Cells(Worksheets("Route").Cells(i, 2).Value, Worksheets("Route").Cells(i - 1, 2).Value)
Worksheets("Route").Cells(i, 9).Value = Worksheets("Route").Cells(i - 1, 9).Value +
Worksheets("Route").Cells(i, 8).Value
Worksheets("Route").Cells(i, 10).Value = (Worksheets("Route").Cells(i, 8).Value) / True_AS
Worksheets("Route").Cells(i, 11).Value = Worksheets("Route").Cells(i - 1, 11).Value +
Worksheets("Route").Cells(i, 10)
Worksheets("Route").Cells(i, 12).Value = Worksheets("Route").Cells(i - 1, 12).Value +
Worksheets("Route").Cells(i, 10).Value

End Sub

Sub A_StarTimeOptimization()
'This sub serves as the main framework for the Route Optimization A-Star algorithm minimizing time

Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim NumNodes As Integer  'Number of Nodes in distance matrix

Dim CurNode As Node  'Current Node being evaluated from
Dim TestNode As Node
Dim BestNode As Node
Dim StartNode As Node
Dim GoalNode As Node
Dim Openlist(1 To 439) As Node

Application.ScreenUpdating = False
Max_Fuel_Range.Show
Airspeed.Show
Refuel_Delay.Show

StartNode.Num = StartNum
StartNode.ParID = 0
StartNode.ScoreF = 0
StartNode.ScoreG1 = 0
StartNode.ScoreG2 = 0
StartNode.Closed = False
StartNode.Open = True

GoalNode.Num = EndNum
CurNode = StartNode
NumNodes = 439

' Add the start node to the open list
Openlist(StartNode.Num) = StartNode
Openlist(StartNode.Num).Open = True

While CurNode.Num <> GoalNode.Num
```

```
Call Check_Open_Set(i, Openlist, NumNodes)  'Check to make sure the open list is not empty
Call Get_the_Best_Node(i, NumNodes, CurNode, BestNode, Openlist)  'Find the node with the best F-Score
If CurNode.Num = GoalNode.Num Then 'If the current node is the goal node, exit the loop
GoalNode = CurNode
Call Build_the_Route(StartNode, GoalNode, Openlist, True_AS)
End If

'Remove current node from open list and add to the closed list
Openlist(CurNode.Num).Open = False
Openlist(CurNode.Num).Closed = True

'Calculate F/G scores for all "neighbors"
For j = 1 To NumNodes
If Worksheets("Distance Matrix").Cells(CurNode.Num, j) <= MaxRange And CurNode.Num <> j Then
TestNode.Num = j
TestNode.ParID = CurNode.Num
TestNode.ScoreG1 = CurNode.ScoreG1 + (Worksheets("Distance Matrix").Cells(CurNode.Num, TestNode.Num) /
True_AS)
TestNode.ScoreG2 = CurNode.ScoreG2 + RefuelTime
TestNode.ScoreH = (Worksheets("Distance Matrix").Cells(TestNode.Num, GoalNode.Num)) / True_AS
TestNode.ScoreF = TestNode.ScoreG1 + TestNode.ScoreG2 + TestNode.ScoreH
TestNode.Open = True
TestNode.Closed = False

'If the neighbor has not been evaluated, add it to the open list
If Openlist(TestNode.Num).Open = False And Openlist(TestNode.Num).Closed = False Then
Openlist(TestNode.Num) = TestNode
Openlist(TestNode.Num).Open = True
End If

'If the neighbor is on the open list, but this is a better path through it, update the parameters
If Openlist(TestNode.Num).Open = True Then
If TestNode.ScoreF < Openlist(TestNode.Num).ScoreF Then
Openlist(TestNode.Num) = TestNode       'Updated Node(j) with the best route and parent ID to reach it
End If
End If

'If the neighbor is on the closed list, but this is a better path through it, update the parameters
and put it back on the open list
If Openlist(TestNode.Num).Closed = True Then
If TestNode.ScoreF < Openlist(TestNode.Num).ScoreF Then
Openlist(TestNode.Num) = TestNode       'Updates Node(j) with the best route and parent ID to reach it
Openlist(TestNode.Num).Closed = False 'Removes the node from the closed list
Openlist(TestNode.Num).Open = True 'Places the node back on the open List
End If
End If
End If
Next j
Wend

Sheets("Route").Select

End Sub
```

Obstacle Avoidance

```
'The Obstacle Avoidance program uses a grid-based network and uses the A-Star algorithm to find an
optimal path while avoiding obstacles and considering undesirable areas.  The code used in this
portion of the model is an adaptation of the two-dimensional path-finding program developed by
Leonardo Volpi (2005).

Public StartRow As Integer
Public StartCol As Integer
Public EndRow As Integer
Public EndCol As Integer
Public Departure_Airfield As String
Public Arriving_Airfield As String
Public Cancel As Boolean
Public Cancel2 As Boolean

Sub ObstacleAvoidanceAdmin()
Dim myRange As Range
Dim WallColor
Dim i As Long, j As Long, k As Long, N As Long, M As Long, NM As Long
Dim myMap(), PathStart(), PathEnd(), Path(), ErrMsg, Score, Stat

Application.ScreenUpdating = False
```

```
'Show Departure/Arrival Point data entry form
Input_Selection.Show

If Cancel2 = True Then
Call Return_to_Homepage
Exit Sub
End If

Set myRange = Range("B2:VT262")
WallColor = 1  'black for unwalkable ground"

N = 262
M = 592

'Load obstacle and "undesirable area" information into myMap
ReDim myMap(1 To N + 1, 1 To M + 1)
With myRange
For i = 1 To N
For j = 1 To M
If .Cells(i, j).Interior.ColorIndex = WallColor Then
myMap(i + 1, j + 1) = -1
Else
myMap(i + 1, j + 1) = .Cells(i, j)
End If
Next j
Next i
End With

i = 3

StartLat = Application.WorksheetFunction.VLookup(StartNum, Sheets("Refuel
Locations").Range("A2:F440"), 5, False)
StartLon = Application.WorksheetFunction.VLookup(StartNum, Sheets("Refuel
Locations").Range("A2:F440"), 6, False)
EndLat = Application.WorksheetFunction.VLookup(EndNum, Sheets("Refuel Locations").Range("A2:F440"), 5,
False)
EndLon = Application.WorksheetFunction.VLookup(EndNum, Sheets("Refuel Locations").Range("A2:F440"), 6,
False)

StartRow = ((50 - StartLat) * 10) + 2
StartCol = ((StartLon + 125) * 10) + 2
EndRow = ((50 - EndLat) * 10) + 2
EndCol = ((EndLon + 125) * 10) + 2

Worksheets("Map").Activate
Cells(StartRow, StartCol).Select
With Selection.Interior
.Pattern = xlSolid
.PatternColorIndex = xlAutomatic
.Color = 5296274
.TintAndShade = 0
.PatternTintAndShade = 0
End With

Cells(EndRow, EndCol).Select
With Selection.Interior
.Pattern = xlSolid
.PatternColorIndex = xlAutomatic
.Color = 255
.TintAndShade = 0
.PatternTintAndShade = 0
End With

ReDim PathStart(1 To 2), PathEnd(1 To 2)
PathStart(1) = StartRow
PathStart(2) = StartCol
PathEnd(1) = EndRow
PathEnd(2) = EndCol

'Start A-Star Algorithm
Call Pathfinder_A_star(myMap, PathStart, PathEnd, Path, ErrMsg, Stat)

If ErrMsg <> "" Then
MsgBox ErrMsg, vbCritical
Exit Sub
End If

End Sub
```

```
Option Explicit

Public Type Node
Row As Integer      'row of the actual node
Col As Integer      'column of the actual node
ParID As Integer    'parent node
ScoreF As Integer   'Score F (total cost)
ScoreG As Integer   'Score G (Cost of the path done)
ScoreH As Integer   'Score H (Estimated cost of the path to do)
Closed As Boolean   'indicates if the node is in the closed list
End Type
Public Dist As Long
Public NodesExplored As Long
Dim Openlist() As Node
Dim TargetNode As Node

Sub Pathfinder_A_star(Map(), PathStart(), PathEnd(), Path(), ErrMsg, Optional Stat)
Dim i As Long, c As String, j As Long, k As Long, N As Long, M As Long, NM As Long
Dim Msg As String, k_best As Long, k1 As Long, k2 As Long, Nrow As Long, Ncol As Long
Dim Goal As Boolean, ris As Boolean
Dim CurrNode As Node
Application.ScreenUpdating = False
On Error GoTo Error_handler
N = UBound(Map, 1)
M = UBound(Map, 2)
NM = N * M
ReDim Openlist(NM)
'load starting point
Openlist(1).Row = PathStart(1)
Openlist(1).Col = PathStart(2)
'load ending point
TargetNode.Row = PathEnd(1)
TargetNode.Col = PathEnd(2)

'A-star algorithm begins
ErrMsg = ""
k1 = 1
Call Compute_Score(Openlist(k1), Map)
Do
Call PickUp_TheBest_Node(k_best)
If k_best = 0 Then
ErrMsg = "Sorry, unable to find the path"
Exit Sub
End If
'switch the best node to the close list
k2 = k2 + 1
Openlist(k_best).Closed = True
Nrow = Openlist(k_best).Row 'Update the current node (Nrow/Ncol) to the best node that was selected
(k_best)
Ncol = Openlist(k_best).Col
NodesExplored = NodesExplored + 1
'searches for each adjacent node
For i = Nrow - 1 To Nrow + 1
For j = Ncol - 1 To Ncol + 1
If i > 0 And i <= N And j > 0 And j <= M Then
'check if the node is walkable
If Map(i, j) >= 0 And (i <> Nrow Or j <> Ncol) Then
ris = False
If Not ris Then
'check if it is still open
k = getNode(i, j)
If k > 0 Then
If Not Openlist(k).Closed Then
'verify if the new score is better
CurrNode.Row = i
CurrNode.Col = j
CurrNode.ParID = k_best
Call Compute_Score(CurrNode, Map)
If CurrNode.ScoreF < Openlist(k).ScoreF Then
Openlist(k) = CurrNode
End If
End If
Else
'New node. Add it to the open list
CurrNode.Row = i
CurrNode.Col = j
CurrNode.ParID = k_best
c = Worksheets("Map").Cells(CurrNode.Row, CurrNode.Col).Address(False, False)
```

```
Range(c).Select

Call Compute_Score(CurrNode, Map)
k1 = k1 + 1
Openlist(k1) = CurrNode
'check if it is the target node
If i = TargetNode.Row And j = TargetNode.Col Then
Goal = True
k2 = k2 + 1
Openlist(k1).Closed = True
Exit Do
End If
End If
End If
End If
End If
Next j, i
Loop  'main loop

Call Highlight_Path(k1)

End Sub

Private Function getNode(Nrow, Ncol)
Dim k As Long
getNode = 0
Do
k = k + 1
If Openlist(k).Col = 0 Then Exit Do
If Openlist(k).Col = Ncol And Openlist(k).Row = Nrow Then
getNode = k
End If
Loop
End Function

Private Sub PickUp_TheBest_Node(k_best As Long)
'Look for the lowest F cost square on the open list.
Dim ScoreMin As Long, k As Long, k_min As Long

Do
k = k + 1
If Openlist(k).Col = 0 Then Exit Do
If Not Openlist(k).Closed Then
If k_min = 0 Or ScoreMin >= Openlist(k).ScoreF Then
ScoreMin = Openlist(k).ScoreF
k_min = k
End If
End If
Loop
k_best = k_min
End Sub

Private Sub Compute_Score(P As Node, Map)
'computes the score of the p-th node
Dim L As Long, di As Long, dj As Long

If P.ParID > 0 Then
'take the score G of its parent
L = Map(P.Row, P.Col)
If L < 0 Then L = 100000
P.ScoreG = Openlist(P.ParID).ScoreG
If Openlist(P.ParID).Row = P.Row Or Openlist(P.ParID).Col = P.Col Then
P.ScoreG = P.ScoreG + 5 + L
Else
P.ScoreG = P.ScoreG + 7.5 + L
End If
End If

'Straight Line Distance Heuristic
di = ((P.Row - TargetNode.Row) * 5) ^ 2
dj = ((P.Col - TargetNode.Col) * 5) ^ 2
P.ScoreH = Sqr(di + dj)

'global score
P.ScoreF = P.ScoreG + P.ScoreH

End Sub

Sub Highlight_Path(k1)
```

```
Dim i As Integer
Dim k As Integer
Dim c As String

'count the path-length
i = k1: k = 0
Do
k = k + 1
i = Openlist(i).ParID
Loop Until i = 0

'build the path
ReDim Path(1 To k, 1 To 2)
Dim Lt1 As Double
Dim Lt2 As Double
Dim Ln1 As Double
Dim Ln2 As Double
Dim rngZoom As Range

i = k1
k = 0

Do
k = k + 1
Path(k, 1) = Openlist(i).Row
Path(k, 2) = Openlist(i).Col
If Openlist(i).Row <> StartRow Or Openlist(i).Col <> StartCol Then
If Openlist(i).Row <> EndRow Or Openlist(i).Col <> EndCol Then
c = Worksheets("Map").Cells(Openlist(i).Row, Openlist(i).Col).Address(False, False)
Range(c).Select
With Selection.Interior
.Pattern = xlSolid
.PatternColorIndex = xlAutomatic
.Color = RGB(254, 191, 78)
.TintAndShade = 0
.PatternTintAndShade = 0
End With
End If
End If
'Calculate the distance traveled by the path

If k > 1 Then
Lt1 = Cells(Path(k, 1), 1)
Lt2 = Cells(Path((k - 1), 1), 1)
Ln1 = Cells(1, Path(k, 2))
Ln2 = Cells(1, Path((k - 1), 2))
End If

Dist = Dist + Application.WorksheetFunction.Acos(Cos(Application.WorksheetFunction.Radians(90 - Lt1))
* Cos(Application.WorksheetFunction.Radians(90 - Lt2)) + Sin(Application.WorksheetFunction.Radians(90
- Lt1)) * Sin(Application.WorksheetFunction.Radians(90 - Lt2)) *
Cos(Application.WorksheetFunction.Radians(Ln1 - Ln2))) * 3440.065

i = Openlist(i).ParID

Loop Until i = 0

MsgBox "The total distance is " & Dist & "  NMs"
Range("A1").Select

End Sub
```

Bibliography

Beeker, Emmet. "Potential error in the reuse of Nilsson's a algorithm for path-finding in military simulations," The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology 1, no. 2 (2004): 91-97.

Defense Logistics Agency Website. "Into Plane Contract Information System (IPCIS)," https://ports.energy.dla.mil/ip_cis/ipcis.htm. 20 August 2013

Departments of the Air Force and Navy. Flying Training, Air Navigation. Air Force Regulation (AFR) 51-40. Washington: HQ USAF, 15 March 1983.

Department of the Army. Army Aviation Accident Prevention Program. DA Pamphlet (DA PAM) 385-90. Washington: HQ USA, 28 August 2007.

Department of the Army. Aviation Flight Regulations. Army Regulation (AR) 95-1. Washington: HQ USA, 15 November 2008.

Dijkstra, Edsger W. "A note on two problems in connexion with graphs," Numerische mathematik 1, no. 1 (1959): 269-271.

Hart, Peter E., Nils J. Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths," Systems Science and Cybernetics, IEEE Transactions on 4, no. 2 (1968): 100-107.

Jameson, Terry C., David I. Knapp, and Ed Measure. "A Weather Routing Tool for Unmanned and Manned Aircraft Systems," ARMY RESEARCH LAB WHITE SANDS MISSILE RANGE NM, 2009.

Jenkins, Marcus. "Introduction to Route Calculation," NAVTEQ Network for Developers, (2007).

Leiserson, Charles E., Ronald L. Rivest, and Clifford Stein. "Introduction to algorithms," Edited by Thomas H. Cormen. The MIT press, (2001).

Matthews-Generations, James. "Basic A* pathfinding made simple," AI Game Programming Wisdom (2002): 105.

Sathyaraj, B. Moses, Lakhmi C. Jain, Anthony Finn, and S. Drake. "Multiple UAVs path planning algorithms: a comparative study." Fuzzy Optimization and Decision Making 7, no. 3 (2008): 257-267.

Soltani, Amir R., Hissam Tawfik, John Yannis Goulermas, and Terrence Fernando. "Path planning in construction sites: performance evaluation of the Dijkstra, A*, and GA search algorithms," Advanced Engineering Informatics 16, no. 4 (2002): 291-303.

Stout, Bryan. "Smart moves: Intelligent pathfinding," Game developer magazine10 (1996): 28-35.

Szczerba, Robert J., Peggy Galkowski, I. S. Glicktein, and Noah Ternullo. "Robust algorithm for real-time route planning." Aerospace and Electronic Systems, IEEE Transactions on 36, no. 3 (2000): 869-878.

Volpi, Leonardo. "A-Star Pathfinder in Visual Basic." Excerpt from unpublished article. http://digilander.libero.it/foxes/Plot/Pathfinder_%20Astar.pdf. 24 August 2013.

Zanakis, Stelios H., and James R. Evans. "Heuristic "optimization": Why, when, and how to use it," Interfaces 11, no. 5 (1981): 84-91.

Zeisler, Nicholas J. "A Greedy Multiple-Knapsack Heuristic for Solving Air Mobility Command's Intratheater Airlift Problem," MS thesis, AFIT/GOR/ENS/00M-21. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 2000.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE (DD–MM–YYYY)	2. REPORT TYPE	3. DATES COVERED (*From — To*)
27-03-2014	Master's Thesis	Sep 2012-Mar 2014

4. TITLE AND SUBTITLE
Aircraft Route Optimization using the A-Star Algorithm

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Fett, Garret D., MAJ

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Graduate School of
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT-ENS-14-M-06

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
POC: MAJ Jaysen Yochim, jaysen.yochim@us.army.mil
US Army Forces Command G4 Aviation Distribution
4700 Knox St.
Fort Bragg, NC 28310
(910) 570-6468

10. SPONSOR/MONITOR'S ACRONYM(S)
FORSCOM

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution statement A:
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection from the United States.

14. ABSTRACT
This research develops an Aviation Distance Estimation and Route Planning Tool (ADERPT) that finds least-cost aircraft routing from a designated departure airfield to an arrival airfield for the purposes of mission cost estimation and pre-mission planning. The model network consists of 43 Army airfields and 426 airports in the Contiguous United States (CONUS) with Department of Defense contract fuel. Using the A-Star algorithm and considering aircraft fuel range, ground speed, and refueling time, we determine the refuel locations that result in the most efficient route. Considering the use of both distance and travel time, we compare our model's performance with Dijkstra's algorithm, a greedy heuristic, and existing cost-estimation techniques. The ADERPT also examines the use of a grid-based network for obstacle avoidance in route planning and provides a proof of concept for its potential use as a mission planning tool.

15. SUBJECT TERMS
Optimization, Routing Algorithms, Vehicle Routing, Heuristics, VBA, A-Star Algorithm, Dijkstra's Algorithm, Greedy Heuristic, Army, Aircraft

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
					Dr. Raymond R. Hill, AFIT/ENS
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (Include Area Code)
U	U	U	UU	68	(937) 785-3636 ext 7469

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18