AIAA-2001-4232
Severe Weather Avoidance Using Informed Heuristic Search

S. Bokadia and J. Valasek
Texas A&M University
College Station, TX

AIAA Guidance, Navigation, and Control Conference & Exhibit
6-9 August 2001
Montreal, Canada
SEVERE WEATHER AVOIDANCE USING INFORMED HEURISTIC SEARCH
Sangeeta Bokadia* and John Valasek
Texas A&M University, College Station, TX 77843-3141

Abstract
Severe weather conditions pose a large threat to the safety of airplanes. For general aviation aircraft, the only permissible action when a thunderstorm is in the flight path is to detour around the thunderstorm. In this paper, an algorithm is developed for general aviation aircraft, which takes a radar image of the thunderstorm as the input, and determines the safest path around with minimum detour. The method used is A* search with modification. A* search is an informed search technique which makes use of an evaluation function that determines the total path cost for any given point. The evaluation function is composed of the actual path cost and a heuristic function to give the estimated cost of the remaining path. In this paper, a heuristic function is formulated which gives a measure of the detour and also addresses the constraints imposed by the desirability of the path. An algorithm for A* search using the heuristic function is developed, and used to determine the flight path in some sample cases. Test cases of stationary and moving thunderstorms show the method to be reliable and fast.

Introduction
As the domain of an airplane is the atmosphere for most of its operation, atmospheric processes assume a lot of significance in navigation and guidance. Weather has been reported as a cause or a factor in 21.75% of all the aviation accidents in the General Aviation category during the year 1990. During the same year, 25.9% of the fatal accidents in the General Aviation category were weather related.

In the wake of these figures, the need for understanding the weather phenomena and their effects does not seem to be overemphasized. And now with the advent of the concept of Free Flight, the need for the development of a reliable and fast weather avoidance algorithm has become more exigent. Free Flight is a new system of managing air traffic. In this system pilot has greater flexibility and can exercise more control over the route to be taken between two airports. Free flight rules begin after the initial departure restrictions and end at the initiation of the arrival sequencing to the destination airport’s terminal space. But an important issue in free flight is the safety of an airplane. Weather avoidance algorithms can ensure the safety of the airplane in free flight to a greater degree, in event of changing weather conditions.

In recent years, this area has gathered attention of lots of researchers. The microburst-related accident at Dallas-Fort Worth Airport in 1985 prompted the development of guidance strategies in an event of microburst encounter. Thunderstorm is another weather phenomenon that has caught lot of attention. The most popular approach to solving the problem is determining the shortest path between two points along which the regions of intense weather activity can be avoided. In one of the earliest works, the shortest path algorithm developed by Dijkstra is used to develop a simple method to guide an aircraft through weather impacted airspace. Though this method determines a safe route but it does not promise to give a desirable route. Another approach is developed in Ref. 8 that takes into account the desirability of a route. This method is based on the basic Bellman-Ford algorithm and it also addresses the constraints crucial to the route planning for an airplane, besides finding the shortest path. These two methods used optimization algorithms.

Another technique that can be used for solving such problems is A* search. This approach is used in Ref. 10 to develop mission adaptable routes for any general scenario. In this paper, A* search has been used specifically to resolve weather conflict. It has been used to determine a path that resolves weather conflict that arises due to the presence of a thunderstorm on the original route. For this purpose, a heuristic function has been developed that has two components. The first component gives the actual cost of moving from the starting point to the point under consideration and the second component is the estimated cost of moving from this point to the end point. The second component also takes in to account the effect of heavy weather activity. Then an algorithm has been developed that uses the principle of A* search to determine the best path from one point to another.

* Graduate Research Assistant, Department of Aerospace Engineering. Student Member AIAA.
Assistant Professor and Director, Flight Simulation Laboratory, Department of Aerospace Engineering.
Senior Member AIAA.
valasek@aero.tamu.edu
Copyright © 2001 by S. Bokadia and J. Valasek. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

American Institute of Aeronautics and Astronautics
point in the presence of a thunderstorm. For the purpose of showing the results, a radar image of a thunderstorm is simulated. The results show the path suggested by the algorithm to avoid the thunderstorm. Some of the sample cases that are being considered in this paper have moving thunderstorms.

Weather Information

Any destructive weather phenomenon is known as severe weather\textsuperscript{12}. This term usually refers to localized storms. These weather conditions correspond to the localized regions of strong wind shear, violent updrafts and downdrafts and heavy downpours. All these phenomenon can cause considerable damage to the aircraft. Strong wind shear can damage the structure of the airplane. Violent updrafts or downdrafts can cause a significant change in the altitude of the airplane and can make the pilot lose control over the airplane. Hence the regions with these weather conditions should be avoided\textsuperscript{13,14}.

The data that is needed in general by the pilot includes radar reflectivity, wind speed and direction, turbulence, icing severity and temperature. But for the purpose of the present research radar data is used for detecting the intensity of thunderstorm. In order to simulate a radar image, a 2D Gaussian function is chosen to give the intensity of weather at any point \((x,y)\) due to a thunderstorm. In order to get a more realistic image, the intensity computations are carried out for a series of thunderstorms. And then region around one thunderstorm as shown in Figure 1 is chosen for the application of algorithm.

Table 1 gives the intensity (or radar reflectivity) and corresponding weather information for different colors in the radar image shown in Figure 1.

The general rule of thumb for the safe passage of an aircraft is to avoid the regions with intensity 30dBZ or greater. Hence the yellow and red regions in the radar image are the ones to be avoided. The point to be noted is that in reality there may be uncertainty in the measurement of these weather data. Hence the radar data that is used as input can have some errors or it may not display the complete weather information very faithfully. Some of the factors that reduce the accuracy of radar data in real world are precipitation attenuation, accumulation of ice on radome, ground clutter etc\textsuperscript{15}. This fact should be accounted by the algorithm.

<table>
<thead>
<tr>
<th>Color</th>
<th>(I(x,y)) (Intensity or Radar Reflectivity (dBZ))</th>
<th>Weather Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>(I(x,y) &lt; 5)</td>
<td>None</td>
</tr>
<tr>
<td>Blue</td>
<td>(5 &lt; I(x,y) &lt; 20)</td>
<td>Light</td>
</tr>
<tr>
<td>Green</td>
<td>(20 &lt; I(x,y) &lt; 30)</td>
<td>Moderate</td>
</tr>
<tr>
<td>Yellow</td>
<td>(30 &lt; I(x,y) &lt; 40)</td>
<td>Heavy</td>
</tr>
<tr>
<td>Red</td>
<td>(40 &lt; I(x,y))</td>
<td>Intense</td>
</tr>
</tbody>
</table>

\(f^*\) Search

\(f^*\) search is an informed heuristic search technique\textsuperscript{11}. It aims at minimizing the total path cost, \(f\). In this search technique, at every node in the search tree the value of the function \(f\) is evaluated. This evaluation function is made up of two parts. The first one, \(g\), gives the actual cost incurred in reaching the node \(n\) and the second one, \(h\) (also known as heuristic function) gives the estimated cost of the path from node \(n\) to the goal state. Hence,

\[
f(n) = g(n) + h(n)\tag{1}
\]

At every step in \(f^*\) search, the node with the lowest value of \(f\) is selected and expanded first. Figure 2 shows the procedure of \(f^*\) search.

The heuristic function \(h\) is said to be admissible if it never overestimates the cost of reaching the goal from node \(n\), or in other words, the actual cost of reaching the goal from node \(n\) is greater than or equal to the estimated cost \(h(n)\). If the heuristic function chosen for \(f^*\) search is admissible, it can be shown that this search technique is complete and...
optimal. This implies that if a solution to a given problem exists, then this technique will find the solution and it will find the best solution. This is the main advantage of A* search strategy. But the disadvantage of this search strategy is the amount of time and memory it requires. Since this search method keeps all the generated nodes in its memory, it quickly runs out of space. But slight modifications in this algorithm can make it generate acceptable solutions in finite time.

![Figure 2: Stages in A* search](image)

**Developing the Weather Avoidance Algorithm**

**Constraints**

The flight path chosen by the weather avoidance algorithm, besides being the shortest one that avoids severe weather, should also lie within some constraints imposed by the threshold comfort level, pilot workload and the aircraft’s physical limitations. The constraints as identified in Ref. 8 and Ref. 10 are as follows:

- **Minimum Segment Length:** The distance between any two vertices in the flight path should be greater than or equal to a predetermined minimum distance. This is required to reduce the workload of the pilot and also the aircraft should be able to complete one turn before starting another one. This is taken care of in our algorithm by fixing the distance between two nodes.
- **Maximum Turning Angle:** Typical turns in flight path have heading changes less than 60°, but in any case the heading changes should not exceed 90°. This constraint is imposed in our algorithm by limiting the range of nodes that can be generated from a given node.
- **Minimum Number of Turns:** The flight path generated should be piecewise linear and the path should have as few turns as possible. This is required to reduce the workload of the pilot.

The weather avoidance algorithm should be able to address all of these constraints, either in the development of the heuristic function, or by limiting the nodes that can be generated. The nodes here refer to the points in the grid for which the algorithm calculates the value of the evaluation function.

**Thunderstorm Avoidance Algorithm**

To begin with, the map is divided into a number of grids. The size of these grids can be either bigger than or the same as the grid size of the radar image. In the case when the grid size of the map is bigger than that of the radar image, the intensity of each grid in the map is set to the maximum intensity found in the area of the grid. Also, the region with the intensity greater than 25dBZ in the map is set as inaccessible. These steps are taken to account for the uncertainties and inaccuracies in the radar image. Hence in the search tree for the shortest path between the given two points, no node can be generated in this area. Before beginning the search, the length, l, of the inaccessible region along the path is determined. Depending upon the chosen minimum segment length and the number of turns that are allowed in the flight path, the segment l is divided into k segments. Along each segment a line is drawn perpendicular to the original path and the nodes are chosen along this length. All the nodes that lie on one line segment have the same depth, and the depth of the node is determined by the line segment on which it falls. The distance between two consecutive nodes is determined by the grid size. The two limits within which the nodes can be chosen are set by the extent of the inaccessible region and the chosen maximum turning angle.

Figure 3 depicts a simple case where a circular thunderstorm lies on the flight path. As mentioned above, the lines along which the nodes will be chosen

American Institute of Aeronautics and Astronautics
are shown. If a node at depth 1 lies between points 1 and 2, then the path through this node will never give the shortest path. Hence, the nodes that lie between points 1 and 2 should be eliminated to reduce the number of computations. Thus, the number of nodes that can be generated at every stage are limited. If the starting point and the end point are not in the thunderstorm, then the root node in the search tree is the starting point and the terminal node is the end point. This algorithm assumes that the thunderstorm lies ahead of the airplane in the flight path and that neither of the two points, the starting point and the end point, lies in the inaccessible region.

![Diagram](image)

**Figure 3: Simplified Example**

Next a heuristic function is developed. The actual cost of the path is the actual distance traveled to reach the node. If there is a thunderstorm between a node and it’s parent node, then the cost of the path is set to a very high value. The heuristic function is the straight-line distance from the point to the end point. If a thunderstorm lies on this path, then a slight deviation is also added to the straight-line distance. Such a heuristic function gives a better estimate of the distance between the given point and the end point than just a straight-line distance.

In order to take care of the memory requirements, modifications are made to the A* search algorithm. These changes are made along the lines of the Simplified Memory- Bounded A*(SMA*) algorithm\(^\text{11}\). In SMA*, a specific amount of space is allocated to the storage of nodes. Whenever the need arises to generate a successor node and there is no memory left, then the node with the highest \(f^{*}\) cost is dropped from the memory. In our algorithm for severe weather avoidance, a similar principle is used. In addition, the algorithm also drops from memory the nodes for which the value of evaluation function is greater than a threshold value. In an SMA* search, the memory of the best forgotten path is also retained to avoid regeneration of the same nodes. But in our case, because of the way the problem is modeled, the search tree can never have the same node at two different depths. Hence, the possibility of regeneration of the same nodes at two different depths is eliminated. Besides, once a node has been reached in the search tree, the best path from this node to the terminal node is independent of the path followed in reaching this node. Unlike the SMA* search, we do not need to retain the best-forgotten path in the memory. Furthermore, a separate A* search is carried out for the path that lies to the left of thunderstorm and the path that lies to the right of thunderstorm. The shortest path among these two is the one that is finally selected by the algorithm.

The one disadvantage of an SMA* search is that if the allocated memory is less than the one required for the optimal path to the target node, then this search strategy would either give a sub-optimal solution, or in the worst case, would not give a solution at all. But in this case, any path to the endpoint will have the same number of nodes. Besides, the number of nodes in the final flight path can be controlled by appropriately choosing the minimum segment length. Therefore, by making a proper choice of the number of nodes in the final path and the amount of memory allocated for the storage of nodes, the solution, if it exists, is ensured.

**Results**

The algorithm developed above is applied to some sample cases. In all the cases, the grid size is 1 mile by 1 mile.

In the first case, the radar image of a stationary thunderstorm is as shown in Figure 4. The starting point and the end point can be seen and the thunderstorm lies in the flight path. Using the definitions above, the inaccessible region is determined. The shaded portion in Figure 5 shows the inaccessible portion corresponding to the radar image of Figure 4. The starting point is point 1 and the end point is point 2. After tracing the inaccessible region using the above algorithm, the shortest flight path between point 1 and 2 that avoids the thunderstorm is obtained.

Figure 6 shows the flight path with respect to the inaccessible regions and Figure 7 shows the flight path with the radar image. The minimum segment length in this case is 8 units (i.e., 8 miles) on the radar image.
Figure 4: Case I – Radar Image

Figure 6: Case I – Flight Path obtained by the Algorithm

Figure 5: Case I – Corresponding Inaccessible Region

Figure 7: Case I – Flight Path with the Radar Image

Run on a Pentium – II PC, using MATLAB, the algorithm takes 11.4 seconds to determine the safest and shortest flight path in this case.
For the second case, the stationary thunderstorm that lies on the flight path is shown in Figure 8. The original flight path from point 1 to point 2 goes through a thunderstorm as shown in the figure. The detour shown in the figure is the flight path directed by the weather avoidance algorithm. The minimum segment length chosen for Case II is 10 units (i.e., 10 miles) on the radar image.

The algorithm takes 11.1 seconds to compute the safest path when run on a Pentium – II PC.

In Case III, a line of three stationary thunderstorms lies in the original flight path as seen in Figure 9.

As in the previous cases, point 1 is the starting point and point 2 is the terminal point. The figure also shows the flight path directed by the thunderstorm avoidance algorithm and as can be seen, it avoids the regions of intense thunderstorm activity. The minimum segment length in this case is 8 units (i.e. 8 miles) on the radar image. The algorithm takes 15.25 seconds to compute the safest path.

In the remaining two cases, the thunderstorms are moving thunderstorms. The underlying assumption in these cases is that both the airplane and the thunderstorm are moving at constant but different speeds. From time to time, the thunderstorm avoidance algorithm computes a new heading for the airplane, depending upon the current radar image. The airplane then changes its heading as directed by the thunderstorm avoidance algorithm. The time interval between two consecutive heading changes is the smaller of a) the time taken by the algorithm to compute the new heading, or b) 10 seconds. For the time interval between two consecutive heading changes, the airplane maintains a constant velocity.

Figures 10 and 11 show the results of simulation for Cases IV and V respectively. For the sake of clarity of the figures, the radar image is represented as a contour plot and the grids are not shown. The area enclosed by the red contour has intensity greater than 30 dBZ, and that enclosed by green contour has intensity greater than 20 dBZ, and the one enclosed by blue contour has intensity greater than 5 dBZ. The dotted contours show the intensity of thunderstorms when the airplane is at the starting point, 1, and the solid contours represent the intensity of thunderstorm when the airplane has reached the terminal point 2. So, the thunderstorm has moved from the position represented by the dotted contours to the position shown by solid contours in the meantime. The black line shows the path followed by the airplane with regular inputs from the thunderstorm avoidance algorithm.

In Case IV, a line of three thunderstorms is moving in the positive x direction at a speed of 40 miles per hour. The airplane is moving from point 1 to point 2 at a constant speed of 150 knots. Figure 10 shows the safe path followed by the airplane in this case.

In Case V, a line of thunderstorms (similar to a squall line) is heading in the negative x direction at a speed of 50 miles per hour. In Figure 11, 1 is the starting point and 2 the end point. The actual safe path followed by the airplane can also be seen in the figure.

In both the cases, the airplane safely avoids the regions bounded by the red counters, or the regions of intense weather activity.
Figure 10: Case IV – A line of three thunderstorms moving in the positive x direction

Figure 11: Case V – A line of thunderstorms moving in the negative x direction
As can be seen in Cases I, II and III, the algorithm computes the safest and shortest path between two points successfully, and the amount of time it takes is reasonable. Also, the individual segment lengths in these cases are less than the minimum segment length specified for each of the cases. Results of Cases IV and V demonstrate the effectiveness of the algorithm in event of moving thunderstorms.

**Algorithm Convergence and Stability**

The algorithm was run on a number of other sample cases too. The algorithm does fail to give a solution in an assortment of cases. The algorithm fails to compute the safe detours if the starting point of the path lies inside the region of intense thunderstorm activity. The algorithm always tries to find the solution in the forward direction, so in event of being flanked by the thunderstorms along all of the forward heading directions, the algorithm cannot retrace the path to get away from the thunderstorm. The algorithm does not permit heading changes greater than the maximum turning angle. Hence, it does not give satisfactory results if the starting point is too close to the thunderstorm and if the thunderstorm extends widely on both sides.

At some points in the above cases, the suggested path comes very close to the areas with high intensity thunderstorms. This can be avoided by reducing the lower intensity limit of the inaccessible region. In the algorithm regions with intensity greater than 25 dBZ are set as inaccessible. Safer routes can be obtained by changing this limit to say 20 dBZ, though that would mean larger deviations from the original flight path. Another way to avoid coming very close to yellow and red regions on the radar image is by specifying the minimum distance that should separate the inaccessible region and the node closest to the boundary of the inaccessible region.

The algorithm works well if given a small area on the map, but its functionality in a large sample area still needs to be established. Given the minimum segment length and the minimum number of turns, the algorithm succeeds in finding the solution. However, it should also have the ability to change the minimum segment length depending upon the size of the thunderstorm. With this capability, it will give better results and provide greater flexibility in choosing the turn points in the flight path. As seen in cases of moving thunderstorms, the algorithm generates sharp and sometimes too frequently turns, which is undesirable. Hence, the algorithm should have the capability to generate smooth and feasible turns. Also it should take into account the direction in which the thunderstorm is moving. That can reduce the number of turns in the final flight path.

Finer grid sizes will give shorter routes but it can make the algorithm computationally too expensive.

**Summary and Conclusions**

In this paper, a severe weather avoidance algorithm has been developed for General Aviation. This algorithm uses the principle of A* search, and suitable modifications are made along the lines of the SMA* search technique to reduce the time and space requirements. The modeling of the problem ensures that if the underlying assumptions are true, the algorithm will succeed in generating a safe flight path in a short interval of time. Next, the algorithm is applied to five cases of thunderstorm, and the simulation results are presented for the moving thunderstorms.

The results show that the algorithm successfully computes the safest and shortest path in a small area on the map. Also, the modifications made in the A* search technique ensure that the solution is reached in a short interval of time. Although these modifications can endanger the optimality of the solution, when specifically applied to the problem in hand, it is not a serious issue. The search tree has finite depth and the depth can be determined from the number of turns allowed in the flight path. The simulation results are encouraging as well, though better results can be ensured by further tuning of the algorithm.

**Acknowledgement**

This research is supported by the State of Texas Advanced Research Technology Program, under grant number 000512-0301-1999. The authors gratefully appreciate this support.

**References**