

Grid-Based Strategic Air Traffic Conflict Detection

Matt R. Jardin*

NASA Ames Research Center, Moffett Field, CA 94035

M.Jardin@nasa.gov

Algorithms for strategic conflict detection are presented in this report. The algorithms are based on the use of a 4-dimensional space and time grid to represent the airspace. Aircraft trajectories are computed and stored in the appropriate grid cells while checking to see if those cells are already occupied. An occupied cell indicates a conflict, either with another aircraft, or with a weather storm cell or region of restricted airspace. First, a deterministic version of the algorithm is introduced. This is followed by the development of a stochastic conflict detection algorithm that accounts for the effects of uncertainty on trajectory and weather storm cell prediction. These grid-based conflict detection algorithms are more efficient than existing algorithms, because they eliminate the need for pairwise computation of inter-aircraft distance. The computational efficiency comes at the expense of additional required computer memory, but the required memory is within the limits of what is currently available.

I. Introduction

Conflict detection is the process of detecting conflicts among two or more aircraft, or between aircraft and some other airspace constraint such as restricted airspace or regions of bad weather. The term *strategic* means that conflict-detection is to be performed with a long look-ahead time considering the entire aircraft trajectory. This is in contrast with tactical conflict-detection which would be performed on a shorter time scale, typically just a few minutes. The boundary between tactical and strategic concepts is arbitrary but is considered here to be at about 20 minutes.

Strategic conflict-detection algorithms for air-traffic control automation have been under development since the mid-1990's.¹⁻⁵ Ground-based ATC automation was among the first applications to require large-scale conflict detection algorithms. As free-flight concepts emerged, airborne-automation tool developers also needed algorithms to efficiently detect conflicts over a large airspace domain. Conflict detection is computationally demanding, especially when considered in the context of both conflict detection and resolution, because each trial conflict-resolution maneuver initiates a new round of conflict detection. Resolving conflicts may induce one or more new conflicts along the remainder of a trajectory, which further increases the computational demands of a conflict detection algorithm.^{6,7}

This paper presents algorithms for computationally efficient strategic air traffic conflict detection. The algorithms are based on the use of a 4-dimensional (4D) grid of space and time to represent the airspace. These grid-based conflict detection algorithms are shown to be more computationally efficient than existing algorithms, because they eliminate the need for pairwise computation of inter-aircraft distance. The improved computational efficiency comes along with an increase in required computer memory, but the required increase is well within limits of currently available computer memory.

*Aerospace Engineer. MS 210-10. Senior Member AIAA.

The basic conflict detection algorithm in this paper is introduced in its deterministic form, followed by an extension to the stochastic case. By extending the basic algorithm to one that considers the inherent uncertainty on trajectory and weather prediction, it is proposed that the conflict detection algorithm behaves like a traffic congestion monitor for longer-term predictions of air traffic conflicts.

A review of prior work on the conflict detection problem is presented in the next section. This is followed by the introduction of the Conflict Grid approach to conflict resolution.

II. Conflict Detection Background

The task of conflict detection is to determine whether all aircraft within a specified spatial and time domain will maintain adequate spatial separation. Aircraft trajectories are typically discretized as a function of time so that the distances between each aircraft can be computed at each discrete instant of time and compared to the allowable separation distance. The number of point-to-point distance comparisons is expressed as

$$N_{\text{comparisons}} = N_T \cdot \sum_{i=1}^{N_{AC}} (N_{AC} - i) = \frac{N_T(N_{AC}^2 - N_{AC})}{2} \quad (1)$$

where N_T is the number of time-steps in the conflict-detection domain, and N_{AC} is the number of aircraft in the domain. The separation comparisons may be partitioned into horizontal and vertical components and expressed as

$$\begin{aligned} (\Delta x^2 + \Delta y^2) &> d_{min}^2 \\ \Delta z &> h_{min} \end{aligned} \quad (2)$$

where Δx and Δy are the distances between the two comparison aircraft in the horizontal x and y directions, Δz is the vertical distance between the aircraft, d_{min} is the required minimum horizontal separation distance, and h_{min} is the required minimum vertical separation distance. For small numbers of aircraft, the computational burden for this type of conflict search is not an issue, but as N_{AC} increases, the computational burden grows quadratically. The quadratic growth rate of the brute-force conflict search has led to a search for more efficient conflict-detection algorithms.

Initial attempts at developing efficient conflict-resolution algorithms were made during the development of the Center TRACON Automation System (CTAS) at the NASA Ames Research Center.⁸ Since CTAS was one of the first practical large-scale ATC automation tools to be developed, it was the first to have a need for efficient conflict-detection algorithms.

The paper by Issacson and Erzberger¹ describes a practical approach to conflict detection for CTAS using heuristics such as altitude pruning and time-skipping to limit the conflict search space. Aircraft trajectories are first computed in time and space using 10 sec. time-steps (equivalent to 1.3 nmi at 480 knots) before applying heuristics to limit the conflict search. Altitude pruning is the process of eliminating potential conflict pairs that are never flying at the same altitude. The use of altitude pruning typically removed 60-80% of all possible trajectory pairs from the detailed conflict search. Time-skipping is used to limit the number of points that need to be checked for conflicts. Time skipping uses the fact that if two aircraft are separated by a large distance at some point in time, then there is no need to check every 10 sec. along the trajectories for a conflict because it would be physically impossible for a conflict to occur until the aircraft could travel the current separation distance. A further reduction in computation is achieved by only computing the sum of the squares of Δx and Δy when both of those terms are individually less than the required separation. The computational performance of this algorithm still grows as $O(n^2)$, but the actual number of computations is proportionally reduced through the applied heuristics.

The paper by Sridhar and Chatterji² examines aircraft conflict detection using principles from computer science search and sort algorithms. A sorting-based algorithm partitions the airspace into a Cartesian grid. Aircraft are stored in the grid cells by using a hashing function to convert the x - and y -coordinates into grid indices. The grid matrix is then unwrapped into a single vector of bin numbers. The vector of bin numbers is sorted using an efficient sorting algorithm (e.g., Heapsort, Quicksort) so that repeated bin numbers might be easily located. From the repeated bin numbers, the corresponding grid locations and the associated aircraft numbers are uniquely determined. The computational complexity of this algorithm is shown to be $O(n \log n)$.

For the Quicksort method, the average number of computations, C_q was shown to be

$$C_q = 8(M \cdot n + 1) \log_2(M \cdot n) \quad (3)$$

where M is the maximum number of bins occupied by any single aircraft trajectory and n is the number of aircraft. In Ref. 2, an example is given where the average ground speed is assumed to be 500 knots, and the conflict look-ahead time is 20 min so that the maximum number of bins per aircraft for 5nmi by 5nmi grid cells is given by

$$M = \left\lceil \frac{500 \text{ n.mi./hr} \cdot (20/60) \text{ hr}}{5 \text{ nmi}} \right\rceil = 34 \quad (4)$$

For $n = 100$ and $M = 34$, equation (3) gives 31,765 computations. Even though this is an $n \log_2(n)$ algorithm, the proportionality constant in the area of interest still makes this computationally intensive.

The next algorithm considered in Ref. 2 is accumulator-based. The mapping of aircraft location to bin number is the same as in the sorting algorithm, but in this case, each time an aircraft is mapped into a bin, the number of aircraft in that bin is incremented by one. The search for conflicts can then be limited to those bins with more than one aircraft. Once the potential conflict bins have been located, the corresponding aircraft are directly checked for conflict.

The average number of computations required by the accumulator algorithm was shown to be

$$C_a = M \cdot n + (I_{max} + 1)(J_{max} + 1) \quad (5)$$

where M and n are as previously defined, and I_{max} and J_{max} are the maximum number of grid cells in the airspace domain of interest. In Ref. 2, an 800-nmi square region is examined for a 20 min conflict look-ahead time so that $M = 34$ once again. In this case, I_{max} and J_{max} are both 160 for 5 nmi grid cells. For 100 aircraft, this requires 29,321 computations, which is only marginally better than the sorting algorithm. The benefit of the accumulator algorithm is realized for greater numbers of aircraft since the number of computations grows as $O(n)$ instead of as $O(n \log n)$.

The approach taken in Ref. 3 is to use a geometric hashing algorithm to identify clusters of aircraft that may be in conflict with one another. Once the clusters have been identified, they are sent as inputs to the conflict-resolution algorithm, which then uses brute-force conflict-search techniques to check potential resolution maneuvers for conflicts. The geometric hashing algorithm used for identifying clusters is similar to the accumulator method of Ref. 2, but with the additional discretization of the time dimension to create a 4-D grid of the airspace. The grid cells are sized according to the minimum separation requirement (5 nmi), and presumably the time-grid size is to be set short enough that the fastest aircraft would not travel through more than the minimum separation distance during one time interval (a value of 0.1 min was used in the example given in the paper). Since conflicts might occur with aircraft in neighboring grid cells, of which there are 27 in 3-D space or 81 in 4-D space (adding the time dimension to the three spatial dimensions), the neighboring cells are also checked. In addition to looking at the immediately neighboring grid cells, certain other non-conflict aircraft are identified that are sufficiently close to the cluster that should be treated as constraints during the conflict-resolution phase. In Ref. 2 it is suggested that the cluster identification algorithm tends to take time linear in the number of aircraft, but the conflict detection in the cluster conflict-resolution phase is still $O(n^2)$, where n is the number of aircraft in a cluster.

Additional research has been conducted in the field of conflict-probability prediction, which is closely related to conflict detection.⁴ Instead of making binary decisions on whether a conflict will occur, a probability is assigned to each potential conflict so that conflict-resolution maneuvers are initiated in such a way as to minimize the cost of the maneuver.

III. Conflict Grid Method

An efficient conflict-detection algorithm called the Conflict Grid (CG) method is now introduced. The CG method is similar to the algorithm presented in Ref. 3, though a few key differences result in the CG method requiring a negligible amount of additional computations in the context of aircraft conflict detection and resolution. The deterministic version of the CG is developed first, followed by the introduction of a stochastic version of the CG that accounts for uncertainty in trajectory and storm cell prediction.

A. Deterministic Conflict Grid

The idea behind the CG method is to store aircraft trajectories in a 4-D grid space (three spatial dimensions and time) as they are computed by setting the values of the corresponding grid cells to 1 (binary “on,” or “true”). An illustration of the CG for the 3-D case (two horizontal dimensions and time) is provided in figure 1. After the

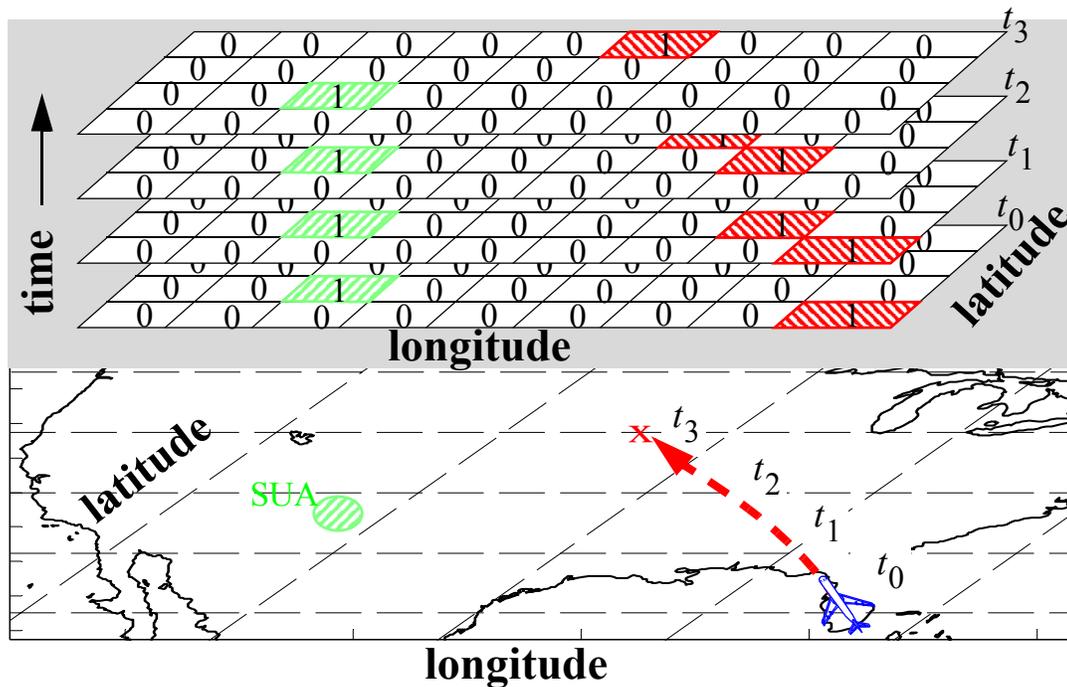


Fig. 1. The Conflict Grid method of conflict detection.

development of the deterministic CG algorithm, an extension to stochastic conflict detection is presented where grid cell values represent the probability of an active constraint in that cell and may take on any value between zero and one. The grid cell dimensions are set according to the allowable aircraft separation limits. If a grid cell is found to be already occupied, then it is immediately known that the current trajectory will be in conflict, and conflict-resolution maneuvers may be initiated. Note that regions of bad weather and special-use airspace may easily be incorporated into the conflict grid by setting the corresponding grid cell values to 1 for any of these areas of restricted airspace. The computational benefit of the CG approach is that it eliminates pairwise distance computations. The claim that the CG technique incurs a negligible amount of additional computations stems from the fact that trajectories must be stored in memory anyway. Since conflicts are detected during the storage process, no additional computations are required for conflict detection.

The airspace is first partitioned into a 4-D grid of space and time. The coordinates used here are longitude, latitude, altitude, and time, but any convenient set of independent coordinates may be used. The airspace is partitioned by creating a set of separate 3-D grids (longitude, latitude, time) for each discrete flight level. If the flight level structure of the current air transportation system is used, then each flight level extends vertically by 1,000 ft. The spatial longitude and latitude grid dimensions are set equal to one another, and this dimension is left as a variable for the purpose of conducting parametric studies of the effects of changing the grid spacing. The time grid size, Δt , is linked to the spatial grid size, Δx , such that an aircraft flying at the fastest expected ground speed would not travel more than Δx distance in Δt time. For example, if the fastest expected ground speed was 600 knots, and $\Delta x = 5$ nmi, then the time resolution would be set to $\Delta t \leq 30$ sec.

The time grid dimension is created using a rolling time window. The window is chosen to span the appropriate amount of conflict look-ahead time. The maximum flight time for an aircraft across the continental United States is less than 7 hr, so this can be used as the maximum bound on the range of the CG time-window. With these definitions, the CG may be represented by a 3-D matrix for each flight level, FL_p , as follows

$$CG_{FL_p} = CG(i, j, k) \quad \begin{cases} 0 \leq i \leq \left\lceil \frac{(\tau_{max} - \tau_{min})}{\Delta \tau} \right\rceil \\ 0 \leq j \leq \left\lceil \frac{(\lambda_{max} - \lambda_{min})}{\Delta \lambda} \right\rceil \\ 0 \leq k \leq \left\lceil \frac{(t_{max} - t_{min})}{\Delta t} \right\rceil \end{cases} \quad (6)$$

$$\Delta \tau \equiv \frac{K_x \cdot \Delta x}{\cos(\max(\lambda))}$$

$$\Delta \lambda \equiv K_x \cdot \Delta x$$

$$\Delta t \equiv \frac{(t_{max} - t_{min})}{\left(\left\lceil \frac{(t_{max} - t_{min}) \cdot V_{g_{max}}}{\Delta x} \right\rceil + 1 \right)}$$

where τ and λ are longitude and latitude, respectively, t is time, Δx is the spatial grid dimension, K_x is a unit conversion constant, $V_{g_{max}}$ is the maximum anticipated ground speed, and $\lceil a \rceil$ is defined as the nearest integer to a rounded upwards. The maximum and minimum grid dimensions are then chosen for the particular needs of the problem being solved.

As is discussed later in this paper, strategic conflict resolution may benefit from much longer look-ahead times than are typically considered for tactical conflict resolution. When considering tactical conflict resolution between two aircraft, uncertainties in trajectory prediction limit the useful look-ahead time to less than about 20 minutes. If using strategic conflict resolution as a form of traffic congestion monitoring, then it can be useful to examine conflict resolution over the entire duration of a typical flight. The Continental United States extends approximately 2,500 nmi from east to west, and 1,500 nmi from north to south. For a 7-hr conflict look-ahead time, a time grid resolution of 30 sec., and grid spacing of 5 nmi, the upper limit on the memory required for the Conflict Grid per flight level is given by

$$\mu_{CG} = \left(0.125 \frac{\text{byte}}{\text{bit}} \right) (N_x \cdot N_y \cdot N_t) = 15.8 \text{ MB} \quad (7)$$

where N_x , N_y , and N_t are the number of grid cells in the x , y , and t dimensions, respectively. Recall that for the deterministic conflict grid, only one bit is required for each grid cell. Therefore, the amount of memory required for the conflict grid at each flight level is less than 16 MB, which is not a challenge for current-day technology.

The procedure for conflict detection is now described. At the beginning of the conflict-detection loop, the CG is cleared so that the value of each grid cell is set to zero. The next step is to store any weather constraints or special-use airspace constraints in the CG by setting the corresponding constrained airspace grid cell values to one. Next, each aircraft is placed sequentially in what is called the Active Aircraft List (AAL). The AAL is traversed to compute a

predicted trajectory for each aircraft. The trajectory is generally computed and stored as a set of vectors of three spatial coordinates versus time. These vectors are interpolated to the discrete time values of the conflict grid. As the values are interpolated, the corresponding values of the conflict grid are checked. If the grid cell values are zero, then that means there are no prior aircraft occupying that cell, and that the airspace of that cell is not restricted by bad weather or other constraint. In that case, the value of that grid cell is set to 1 to signify that it is occupied by the current aircraft. If any of the trajectory points for the current aircraft are found to be in conflict at any of the grid cells, then a conflict has been detected.

A few subtle features of the CG algorithm are now addressed. If just the occupied grid cells are marked as such, it may occur that aircraft in neighboring grid cells are in conflict with one another, because the basic CG algorithm does not include any space between neighboring grid cells (Fig. 2). Another situation that might occur is that the

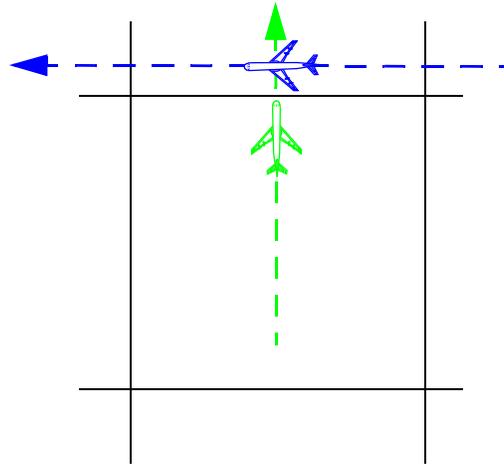


Fig. 2. Illustration of a missed conflict at a grid cell boundary.

discretized time-steps of a trajectory might overstep a grid cell so that a real conflict is not identified (Fig. 3).

One solution to these kinds of conflicts is to use a grid cell buffering technique. With grid-cell buffering, the CG cells would be made half the desired size, but then each time a trajectory point was stored in the grid, the neighboring cells would also be marked as being occupied (Fig. 4). By doing this in all 3 dimensions for the single flight-level problem (x, y, t) , the types of conflicts mentioned above would all be detected. The cost of grid cell buffering is that the amount of memory required for the CG for any given aircraft spacing would increase by a factor of 8 (2^3). For the example given in equation (7), the amount of memory required would increase from 15.8 Mbytes to 126 Mbytes, which is still well within the memory capabilities of current-day computers. There would also be some minor computational costs incurred by requiring that additional grid cells be set for each trajectory point.

Another approach is to accept that these types of conflicts will occur and then to resolve the conflicts tactically. Even though inter-aircraft separation may be predicted to be less than the minimum allowable separation, the CG method ensures that, on average, there is enough airspace available to support all of the aircraft.

Finally, a stochastic extension would generalize the basic CG method and would also eliminate many of the subtle types of missed conflict alerts addressed above without having to resort to grid cell buffering. This stochastic extension is now presented.

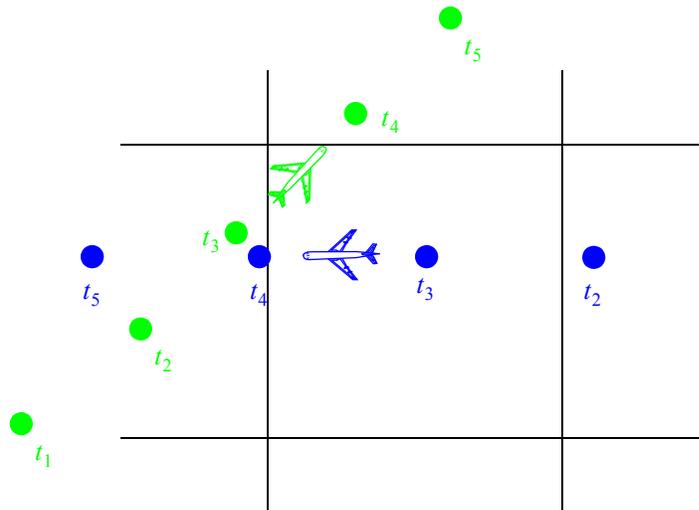


Fig. 3. Illustration of a missed conflict resulting from time discretization.

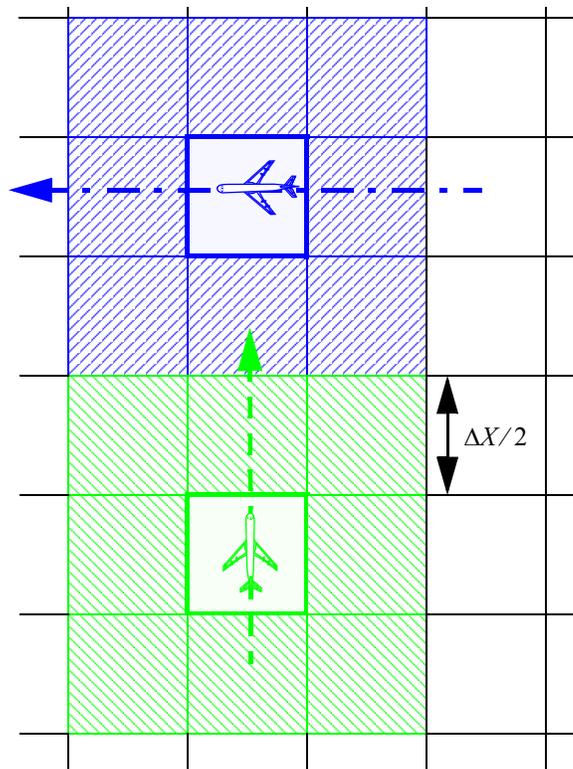


Fig. 4. Grid-cell buffering to eliminate missed conflicts.

B. Stochastic Conflict Grid

Aircraft trajectory prediction, atmospheric prediction, and many other facets of the NAS are better characterized as stochastic processes than as deterministic processes. Errors in wind models can lead to rather large errors in down-range trajectory predictions. Storm forecast errors can either completely miss actual storms that develop, or can predict storms that never materialize. This suggests the need for a stochastic approach to conflict detection.

The probabilistic nature of predicted conflicts may be considered with a simple extension to the deterministic CG algorithm. The extended algorithm is referred to as the Stochastic Conflict Grid (SCG). The basic idea is to store the probability that *at least one* active constraint exists in any given grid cell rather than using a binary *yes* or *no* value. An active constraint is any entity that requires exclusive use of the airspace, such as an aircraft, a weather storm cell, special-use airspace, or even an aircraft trailing wake vortex. This generalizes the CG technique by incorporating important conflict-probability concepts from prior research on pairwise conflict-probability estimation.^{4,5} Some minor increase in computational overhead is incurred for the CG method itself, but as with the grid cell buffering technique, the overhead is negligible when compared with the other computational costs involved in trajectory computation. Apart from the conflict-detection algorithm, computing and maintaining models of aircraft trajectory-prediction accuracy might add significant overhead. The additional computational burden may be reduced by using simple probability models for aircraft trajectory-prediction uncertainty.

The probability that a constraint, c_i , will be active in a particular grid cell may be modeled as a Bernoulli trial so that the probability is given by

$$P_i \equiv p(\text{constraint } i \text{ is active}) \quad (8)$$

and the probability that the constraint will not be active is simply

$$p(\text{constraint } i \text{ is not active}) = (1 - P_i) \quad (9)$$

For a set of n possible constraints, an easy way to compute the probability, \bar{P} , that at least one of those constraints will be active is to compute the probability that no constraints will be active and then to subtract this from 1:

$$\bar{P} \equiv p(\text{at least one active constraint}) = 1 - \prod_{i=1}^n (1 - P_i) \quad (10)$$

where Π is the serial product operator.

Rather than storing the individual probabilities for all of the potential constraints, it may be desirable to maintain a running total probability that at least one constraint will be active. This is easily shown to be given by

$$\bar{P}_i = 1 - (1 - P_i)(1 - \bar{P}_{i-1}) \quad (11)$$

where \bar{P}_i is the running total probability that any constraint up to and including constraint c_i will be active, and \bar{P}_{i-1} is the running total probability that any constraint will be active before considering the new constraint, c_i .

If trajectory-prediction error distributions are modeled as Gaussian, then the case of a two-aircraft conflict using the SCG would be identical to the method developed by Erzberger et al.^{4,5} for pairwise conflict-probability estimation. Following this method, the choice of whether or not to make a conflict-resolution maneuver would be based on the minimization of the expected cost of resolution. Low-probability conflicts are ignored because the expected cost of resolving a conflict that might not occur is too high. Over time, the probability increases to the point that delaying a conflict-resolution maneuver would result in a higher expected cost, because short-term conflict-resolution maneuvers are less efficient than strategic resolution maneuvers.

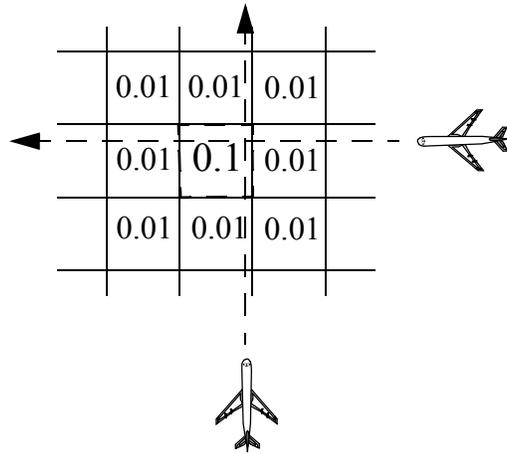


Fig. 5. Example of low predicted conflict probability for two aircraft.

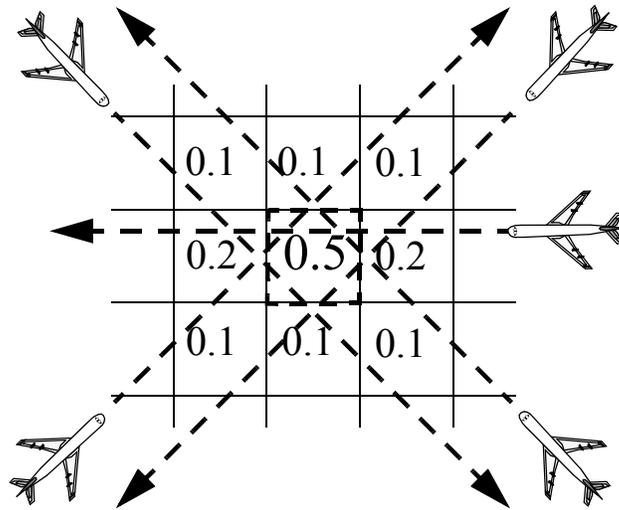


Fig. 6. Example of higher conflict probability with multiple aircraft.

By the use of the SCG, this same conflict-probability method is generalized to the case of multiple aircraft and to the case of conflicts with any other type of constraint. Imagine a case where two aircraft are predicted to converge at some instant an hour in the future (Fig. 5). The probability that either aircraft will be in that particular grid cell (at that location at that instant in time) is low because of the growth rate of trajectory-prediction error. In this case, it would be more efficient to wait to see how the potential conflict develops. Now imagine that many aircraft are predicted to converge at the same point (Fig. 6). The probability that *any one aircraft* will be in that grid cell now increases, potentially to the point that the next aircraft predicted to be in that grid cell should be required to make a conflict-resolution maneuver. In this way, the SCG method behaves like a traffic congestion monitoring algorithm in addition to being a conflict detection algorithm.

Weather storm cells are notoriously difficult to predict with any accuracy until they have actually developed into storms, and even then their prediction beyond 30 minutes is unreliable. Storm predictions are usually made such that a region of airspace can be identified as having some heightened potential for storm development. This heightened potential could be stored in the CG cells in a region (Fig. 7). This would result in fewer aircraft being permitted to pass

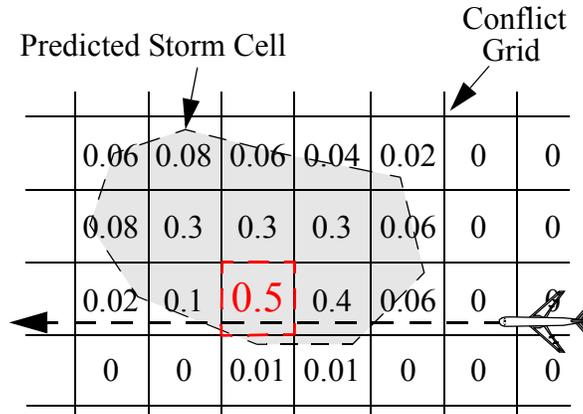


Fig. 7. Conflict probabilities between storm cells and aircraft.

through that region because of the elevated probability of storm conflict. As the time of the storm prediction neared and the prediction became more accurate, the conflict probabilities in the corresponding grid cells would increase to the point where no aircraft would be permitted in those cells. This would be a gradual process so that sudden tactical maneuvers around unpredictable storm cells would be greatly reduced. The resolution thresholds could be modified empirically via simulation with real weather data to obtain good performance with the desired level of safety.

The case of special-use airspace is even easier to consider. Once special-use airspace is activated, it is known with certainty that a constraint is active in that region so the probabilities of the corresponding grid cells would be set to 1. This would block those cells from being occupied by any aircraft. Since the conflict grid is considered in both space and time, special use airspace that was only active during certain times of the day could easily be accommodated. The same principle applies to any regions of blocked airspace, whether it be for security, noise abatement, or other purpose.

IV. Conclusions

A new method for computationally efficient strategic conflict-detection was presented. The Conflict Grid (CG) technique was shown to require a negligible amount of additional computation within the context of a system which computes and stores trajectories in memory. The Stochastic Conflict Grid (SCG) was introduced as an extension to the basic CG technique to generalize the concept of conflict detection to constraints with uncertainty. Through qualitative examples, it was proposed that the SCG technique generalizes the concept of conflict detection to that of traffic congestion monitoring.

Future studies should be conducted to identify potential challenges in implementing conflict grid techniques. Potential applications of conflict grid techniques should also be explored. For instance, it would be instructive to determine the utility of the SCG as a predictive traffic congestion monitoring tool by using it to generate 4-dimensional traffic/weather congestion maps.

V. References

- [1] Isaacson, D. R., and Erzberger, H., "Design of a Conflict Detection Algorithm for the Center/TRACON Automation System," 16th Digital Avionics Systems Conference, Irvine, CA, Oct. 26-30, 1997.

- [2] Sridhar, B., and Chatterji, G. B., "Computationally Efficient Conflict Detection Methods for Air Traffic Management," 1997 American Control Conference, Albuquerque, NM, June 4-6, 1997.
- [3] Chiang, Y.-J., Klosowski, J. T., et al, "Geometric Algorithms for Conflict Detection/Resolution in Air Traffic Management," Proc. of the 36th IEEE Conference on Decision and Control, 1997.
- [4] Paielli, R. A., and Erzberger, H., "Conflict Probability Estimation for Free Flight," NASA TM-10411, Oct. 1996.
- [5] Erzberger, H., Paielli, R., et al, "Conflict Detection and Resolution in the Presence of Prediction Error," 1st USA/Europe Air Traffic Management R&D Seminar, Saclay, France, 17-20 June, 1997.
- [6] Krozel, J., Peters, M., et al, "System Performance Characteristics of Centralized and Decentralized Air Traffic Separation Strategies," *Air Traffic Control Quarterly*, Vol. 9, No. 4, 2001.
- [7] Jardin, M., "Analytical Relationships Between Conflict Counts and Air Traffic Density," *Journal of Guidance, Control, and Dynamics*, [in press].
- [8] Erzberger, H., Tobias, L., "A Time-Based Concept for Terminal-Area Traffic Management," AGARD-CP-410, Brussels, Belgium, 1986., pp. (52-1)-(52-14).