AIAA-95-3366

Knowledge-Based Scheduling of Arrival Aircraft

K. Krzeczowski, T. Davis, and H. Erzberger
NASA Ames Research Center
Moffett Field, CA

I. Lev-Ram
Sterling Federal Systems
Palo Alto, CA 94303

C. Bergh
MIT Lincoln Laboratory
Lexington, MA 02173

AIAA Guidance, Navigation, and Control Conference

August 7-10, 1995 / Baltimore, MD

KNOWLEDGE-BASED SCHEDULING OF ARRIVAL AIRCRAFT IN THE TERMINAL AREA

K. J. Krzeczowski,* Thomas J. Davis,† and Heinz Erzberger‡
NASA Ames Research Center
Moffett Field, California 94035

Israel Lev-Ram§
Sterling Federal Systems
Palo Alto, California 94303

Christopher P. Bergh¶
M.I.T. Lincoln Laboratory
Lexington, Massachusetts 02173

Abstract

A knowledge-based method for scheduling arrival aircraft in the terminal area has been implemented and tested in real-time simulation. The scheduling system automatically sequences, assigns landing times, and assigns runways to arrival aircraft by utilizing continuous updates of aircraft radar data and controller inputs. The scheduling algorithm is driven by a knowledge base which was obtained in over two thousand hours of controller-in-the-loop real-time simulation. The knowledge base contains a series of hierarchical "rules" and decision logic that examines both performance criteria, such as delay reduction, as well as workload reduction criteria, such as conflict avoidance. The objective of the algorithms is to devise an efficient plan to land the aircraft in a manner acceptable to the air traffic controllers. This paper will describe the scheduling algorithms, give examples of their use, and present data regarding their potential benefits to the air traffic system.

Introduction

The development of an automation system for assisting terminal area air traffic controllers in efficiently managing and controlling arrival traffic has long been the objective of researchers and engineers. A fundamental building block in such a system is a planning algorithm that sequences arrival traffic and assigns runways to that traffic. The objectives are not only to reduce delays and increase capacity, but to also reduce controller workload. This report describes the algorithms used to perform the planning function for an air traffic automation tool called the Final Approach Spacing Tool (FAST).[1,2] FAST is the terminal area component of the Center/TRACON Automation System (CTAS).[3]

The planning algorithm in FAST attempts to achieve increased airport capacity in a manner that is acceptable to air traffic controllers. Previous research concentrated on calculating a solution which is optimized for delay reduction, but did not adequately address controller preferences and workload.[4] In simulation tests of this algorithm, controllers often did not follow the advised solution due to workload and safety issues. The algorithms in this paper are based on a set of heuristics that evolved out of the input from expert controllers. In this way, the planning process is able to emulate the controllers' own planning process, while retaining the advantage of accurate calculation of aircraft performance characteristics. Simulation results have shown comparable gains in delay reduction while achieving a much greater acceptance of the solution by air traffic controllers.

The main inputs into the planning algorithms come from a trajectory generation engine which integrates point mass equations of motion along a horizontal route with specified target altitudes.[5] A time range in which the aircraft could arrive at all potential runway thresholds is produced by feeding the extreme deviations from a nominal route to this engine. The planning algorithms use this time range and other trajectory information to accomplish the scheduling tasks.

This paper will begin by giving a description of the scheduling algorithms which includes both sequencing and runway allocation. Results of fast-time simulations that demonstrate the potential benefits of the algorithms will then be presented. Results of real-time simulations will be discussed to illustrate controller acceptance. These results are followed by some concluding remarks.

Knowledge-Based Sequencing Algorithm

Since one objective is to reduce delay, an obvious method of sequencing would be to optimize for delay reduction. Unfortunately, the optimization of the

* Research Scientist, Air Traffic Management Branch
† Research Scientist, Air Traffic Management Branch. Member AIAA.
‡ Chief Scientist for Air Traffic Management. Fellow AIAA.
§ Software Specialist
¶ Technical Staff, Terminal Air Traffic Control Automation Group

sequence is not achievable in real-time. Real-time simulations demonstrated that controllers have trouble executing an optimized sequence. They prefer a sequence that is similar in nature to a First Come First Serve (FCFS) ordering technique. Brinton[4] has shown that the delay benefits of an optimal sequence versus FCFS in the terminal airspace are minimal.

In order to fully understand the motivation for designing a knowledge-based sequencing (KBS) algorithm, a brief description of techniques employed by terminal area controllers to sequence arrival traffic is necessary. The KBS algorithm attempts to produce delay savings while taking into account techniques that controllers use to reduce workload. One such sequencing technique limits "overtakes" (one aircraft passing another) to those that take place a sufficient distance from the airport to allow for maneuvering space. As the aircraft approach the airport, they descend and converge on the runway making it difficult for a controller to maintain separation of passing aircraft. The algorithm is designed to sequence an overtake only when the pass is predicted to take place far enough from the airport to meet the required separation.

Another technique used by controllers is to attempt to keep aircraft that arrive "in- trail" in succession in the final sequence. Two aircraft are "in-trail" if they enter the terminal area over the same arrival fix, with similar speeds and near-minimum legal separation. Because these aircraft are already spaced properly, it would increase a controller's workload to split the two aircraft apart in order to fit another aircraft from a different stream between the in-trail aircraft. This is one instance where an optimal sequencing algorithm might select a sequence that puts the aircraft from a separate stream between the in-trail aircraft in order to provide some small savings in delay reduction. The KBS algorithm weighs the potential delay savings of sequencing an aircraft between in-trail aircraft, against the workload advantage of keeping the in-trail aircraft in succession.

Controllability is the aircraft's potential to absorb delay by performing acceptable maneuvers in order to fit behind other aircraft in the sequence. A controller would be reluctant to issue extreme vectors to an aircraft in order to have that aircraft fit the recommended sequence. The KBS algorithm will rarely sequence an aircraft in a position where it does not meet controllability requirements.

No matter how good a sequencing algorithm is, the recommended sequence will not be followed 100 % of the time. The algorithm must accept these deviations and adjust the sequence. The KBS will detect that the controller is not following the sequence, when the system of aircraft has deviated far enough from the original plan. At this point, the algorithm will adjust the sequence to align with the controller. The difficulty is to have the KBS algorithm react correctly in a timely manner and remain stable.

The remaining sections on the KBS algorithm give an explanation as to how these heuristics are implemented. The next section describes how the sequencing problem is broken up into local sequences. This is followed by a description of how the local sequences are determined and merged together to form the final sequence.

Constraining the Sequence

It was learned through extensive real-time simulations that to produce an acceptable sequence it is necessary to consider all merges within the airspace, (not just the merge on the final approach course). To do this, the sequencing problem is broken into a network of common trajectory segments. A trajectory segment is a portion of a trajectory that falls within a defined segment of flight. Figure 1 shows an aircraft and its trajectory broken into four trajectory segments referred to as: "LONG_LEFT", "DOWNWIND_LEFT", "BASE_LEFT", and "FINAL".

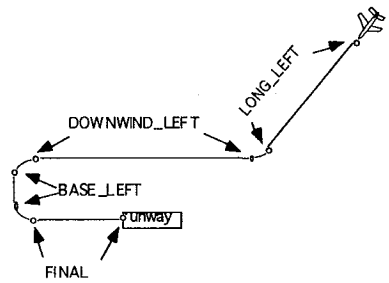


Fig. 1. Typical trajectory segments for an arriving aircraft

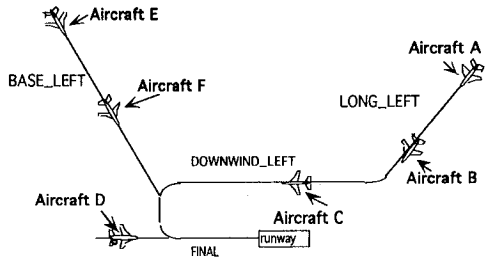


Fig. 2. Sequencing example of merging arrival aircraft

Determining a sequence for a given runway is the process of both creating a relative order of aircraft on each trajectory segment and combining the orders into a consistent sequence for that runway. Figure 2 depicts a situation where six aircraft merge to land on the same runway. The tree in Fig. 3 is a visualization of how the KBS algorithm perceives this situation. Each trajectory segment has a branch within the tree that represents a merging of aircraft on that trajectory segment. The leaves of the tree, denoted by the shadowed boxes, represent the aircraft that are currently on the segment.

The sequencing starts by creating a relative order of aircraft for each leaf in the tree. These aircraft are then merged up the tree, creating a relative order in each node of the tree. The relative order of the leaves and nodes being merged is preserved as the sequencing process

proceeds; constraining the number of possible sequences available. The final merge at the top of the tree is the resulting sequence to the runway.

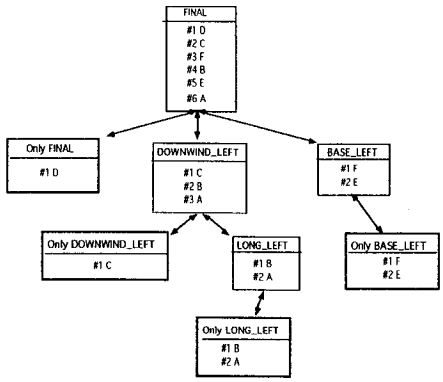


Fig. 3. Trajectory Segment tree

One advantage to this method is that it allows flexibility in creating the relative orders. A separate ordering algorithm can be designed for each trajectory segment which takes into account any peculiarities about that segment. This is how the overtake heuristic is implemented. The algorithm for long and base trajectory segments is designed to account for acceptable overtake conditions.

Another advantage of sequencing in this way is that it sets relative orders between aircraft based on trajectory segments in close spatial proximity. The first set of relative orders are comprised of aircraft currently located on a common trajectory segment. These orders are determined from current aircraft positions, not predicted data. The next set of relative orders are a merging of the first sets, for the next trajectory segment that these aircraft are approaching. This set is ordered based on a prediction of the near future. As the sequencing moves up the tree, it is forced to use predictions that are farther into the future. But each relative order produced from predicted data is constrained by a series of relative orders that were determined using data closer to the current situation.

This advantage becomes clear when compared to a different method that relies only on arrival times at the runway. Arrival times can be predicted accurately using modern methods, but in some cases do not contain enough information to produce controller-acceptable sequence. More specifically, they only convey information about the future and none about the current situation. By breaking the problem up as described, the sequencing algorithm can take advantage of current information, intermediate predictions, and longer-term arrival predictions.

Fuzzy Logic in the Sequencing Algorithm

The sequencing algorithms for specific trajectory segments make use of a general sorting function that accepts, as input, an unordered list and an ordering function which determines the relative order of two members of that list. This section is concerned with the operation of the ordering functions. Two general ordering functions are presented that are representative of all those used to determine the order of the trajectory segments. The first is a general ordering scheme used to sequence the leaves of the tree and the merge at each node up to the final approach course. The second ordering function sequences the final approach course merge.

Both functions are based on fuzzy logic reasoning techniques. Due to the complex nature of the relationship between the input and the output of the sequencing problem, it was impractical to use classical crisp logic. Fuzzy logic permits the developer to mimic the reasoning of the expert controller through linguistic rules. There have been a number of successful applications that use fuzzy set theory. Among them are the guidance control of the subway system in the city of Sendai[6] and the fuzzy logic automatic carrier landing system for the F/A-18 .[7]

Using crisp logic, a knowledge-base will consists of a set of rules:

$$\text{IF } [X \text{ is } A_k], \text{ THEN } [Z \text{ is } B_n] \qquad (1)$$

where X represents the input, A_k represents the number of input states checked, and B_n the number of possible output states. The condition "X is A_k" is limited to two possibilities: true or false. This two valued logic has no potential for a degree of belonging. For any rule to fire the condition must be an exact match, which limits the capability of this type of system to represent a knowledge base. This limitation forces the designer to incorporate a large number of rules to get the granularity necessary for real systems .[8]

Fuzzy logic extends crisp logic to include a range of membership from 0 to 1.

$$\mu_{A_k}(X) \in \{0, 1\} \qquad (2)$$

The knowledge-base representation becomes

$$\text{IF } [\mu_{A_k}(X)], \text{ THEN } [Z_k = \Omega_{B_n}(\mu_{A_k}(X))] \qquad (3)$$

Where $\Omega_{B_n}(\mu_{A_k}(X))$ converts the membership value into a firing strength for the rule. There is no longer a need to have an exact match of the left hand side of the logical expression to have a rule fire. This makes it possible for more than one rule to fire at a time. A defuzzification technique is used to combine the rule strengths into a crisp output.[9]

Figure 4 is a graphical representation of a two rule system. The left hand side of the figure represents the membership functions $\mu_{A_k}(X)$, the right hand side represents the output functions $\Omega_{B_n}(\mu_{A_k}(X))$. The outputs of $\Omega_{B_n}(\mu_{A_k}(X))$ are the areas formed by cutting off the top of the triangles at the value of $\mu_{A_k}(X)$. These areas are then combined to form a crisp

output using a center of gravity (COG) defuzzification technique depicted in Fig. 5. COG defuzzification uses the independent axis coordinate location of the center of gravity of the combined areas as the resulting crisp output.

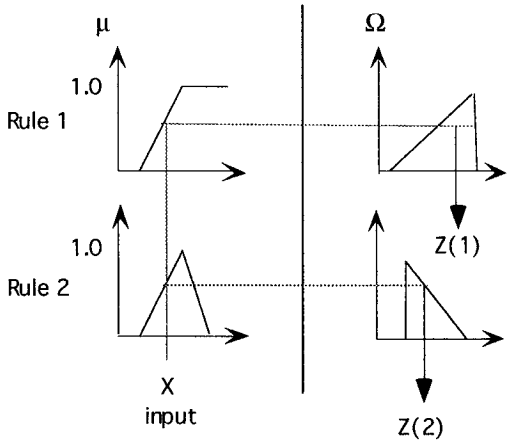


Fig. 4 Fuzzification of two rule system

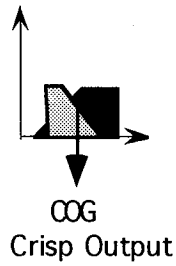


Fig. 5 Center of Gravity defuzzification method

The architecture of the fuzzy logic sequencer is shown in Fig. 6. The fuzzification process maps the system inputs into membership values for linguistic labels such as *faster, slower* . The decision making process utilizes a set of heuristic rules derived from discussions with expert controllers to operate on the membership values. The decoder uses a center of gravity scheme to defuzzify the strength of each heuristic rule into the relative order of two aircraft.

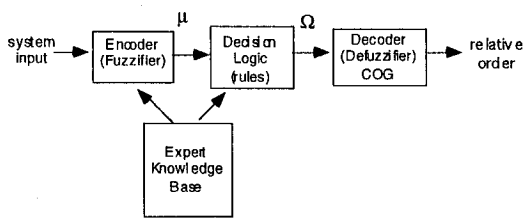


Fig. 6 Architecture of Fuzzy Sequencer

The sequencing is done on a cycle that corresponds to the update rate from the radar system (approx. once every 4.7 sec) in order for the KBS to acquire new information about the traffic situation. During each cycle the algorithm reevaluates the previous cycle's calculated sequence, and orders any new aircraft that have entered the system.

Non-Final Sequencing

The primary input to the non-final ordering function is a measure of how far ahead/behind one aircraft is to another. Trajectory segments are made up of a set of time steps at defined intervals. Each time step contains a predicted x, y, altitude, speed, and heading of an aircraft at a future time. The ordering function searches the list of time steps, associated with the trajectory segment being ordered to find the earliest instant within the segment that the two aircraft both have time steps. These two time steps are called the First Common Time Steps (FCTS). A distance is calculated from the FCTS to the end point of the trajectory segment being ordered for each aircraft. The differences in the distances, divided by the required separation for any two aircraft gives a Normalized Separation Distance (NSD).

$$NSD = \frac{(\text{distance B - distance A})}{\text{Required Separation}} \qquad (4)$$

In Eqn. 4, if the NSD is positive, aircraft A would be ahead of aircraft B; a negative value would indicate the reverse. The exact value measures how much ahead/behind A is relative to B. The required separation is defined by the aircraft weight classes and is shown in Table 1.

Table1. Required Separation (in Nautical Miles)

Leading Aircraft Type	Trailing Aircraft Type		
	Heavy	Large	Small
Heavy	4	5	6
Large	3	3	4
Small	3	3	3

The remaining inputs to the logic are: the Distance from each aircraft's current location to the specific Trajectory Segment being ordered (DTS), the speed difference between the aircraft at the FCTS, and the last calculated or previous relative order. Each input is mapped to a linguistic by a membership function μ(). The membership values are operated on by a set of heuristic rules with a firing strength of Ω(). The outputs of the firing strengths are combined using a COG defuzzification method to determine the crisp output. The sign of this output determines the relative order of the aircraft. Appendix A defines the membership functions, the firing strengths and the rules for the first ordering function.

A graph of the resulting crisp logic following the defuzzifier for the non-final ordering is shown in Fig 7. These curves were calculated based on the previous

labeled "Odd Aircraft Type." This criterion examines the aircraft together with all aircraft meeting the previous criteria (runway pair and feeder gate), and determines if the aircraft currently traversing the decision tree is an odd type (e.g. the only turbo prop in a stream of jet traffic). If this is true, then the system examines a system-wide or global delay reduction criterion. Because the aircraft in this example is an odd engine type in its stream, the delay reduction criterion is small (0 minutes). If we examined the branch on the "No" answer for "Odd Aircraft Type," we would find that the global delay reduction criterion would require a larger value (typically 2-4 minutes). The reason for the difference in delay reduction requirements on these two branches is to force the KBRA algorithm to favor pulling a dissimilar engine or weight class aircraft out of the traffic stream. This serves to reduce workload for the controller.

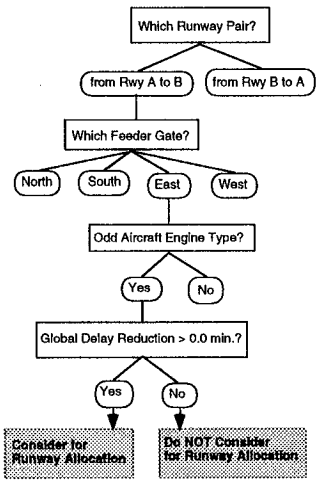


Fig. 10. Example of decision tree for selecting aircraft for runway allocation

After all eligible aircraft have passed through this decision tree and thus narrowing the list of all eligible aircraft to a smaller set, the KBRA algorithm then selects a single aircraft which appears to have the greatest delay benefits to the overall arrival system. In some cases, there may not be any aircraft which pass these criteria and in this case, the KBRA algorithm will not consider any aircraft for that update cycle. Once an aircraft is selected, it is then placed in an alternate runway KBS cycle. The entire arrival airspace sequencing problem is solved with this aircraft placed on its alternate runway. This allows the KBRA algorithm to evaluate all aspects of the particular runway allocation. Full trajectory solutions are obtained for each aircraft which in turn give accurate sequences, expected delay, and conflict detection for the entire airspace. At this point, a new and more detailed set of criteria are applied. These criteria examine trajectory based issues such as potential conflict resolution problems and exhaustion of critical degree of freedom limits. They are applied to the alternate solution set in order to make the final determination as to whether or not to change the aircraft to the alternate runway.

Once an aircraft has been switched away from a given runway, that runway is blocked off from further consideration for that aircraft. A more optimal solution would be to allow allocation of this aircraft back to its original runway if a situation warrants, but this was found to be unacceptable to controllers. Finally, once an aircraft's ETA falls below a runway's freeze time horizon, that runway will be blocked off from further consideration. After all but one runway has been blocked off, the runway assignment advisory is frozen for the remainder of the flight. In nearly all cases, the aircraft has a frozen runway assignment before twelve minutes of flight time from the runway. This twelve minute location is typically within 5-10 n.mi. inside the feeder gate.

Results

A fast-time simulation was developed in order to predict the potential benefits from the knowledge-based sequencing and runway allocation algorithms. The simulation examined a generic terminal area with two independent parallel runways. The traffic was modeled with initial positions just outside of the feeder gate and arriving with a uniform distribution over a 90 minute period. The traffic was distributed to arrive in equal proportions from each of the four feeder gates. A wide range of arrival rates, 50 aircraft per hour up to 150 aircraft per hour were considered. The simulation was structured such that over one thousand traffic samples could be tested in order to gather statistical data on the performance.

The simulation modeled a baseline scenario, the KBS and KBRA algorithms in the following manner. The baseline scenario was modeled as a pure FCFS sequence. The KBS was modeled as a FCFS sequencer with constrained position shifting (CPS). The CPS allows a shift of one sequence position between two streams of traffic if some delay savings can be achieved. Note that from the previous discussion of the KBS algorithm that this model is deficient in handling some high workload situations, such as keeping "in trail" aircraft in succession, but for statistical purposes, they are very similar.

The KBRA algorithm in the fast-time simulation was modeled as follows: aircraft were initially assigned a default runway which was the runway closest to their arrival feeder gate. As aircraft entered the terminal airspace, their predicted sequence, schedule, and delay for their default runway was compared with a predicted sequence, schedule, and delay for the alternate runway. If the alternate runway solution produced a lower overall delay for the arrival system, then the runway was changed for that aircraft to an alternate runway. Note that the equal distribution of arrival aircraft for each feeder gate will produce the most conservative predictions for potential benefits because fewer opportunities exist for runway balancing. Also note that, similar to the KBS model in the fast-time simulation, this model is deficient in modeling the workload reducing portion of KBRA.

Despite the workload model deficiencies in the fast-time KBS and KBRA models, the statistical data gathered in the fast-time simulation depict a trend which is similar to those observed in real-time controller-in-the-loop simulations using the KBS and KBRA algorithms. The fast-time simulation results are shown in Figure 11. The graph depicts delay (seconds) versus arrival rate (aircraft per hour). There are three curves in the plot: the solid line represents a baseline scenario in which pure FCFS sequencing and default runway assignment was used, the other two curves show the impact of constrained sequence shifting (similar to KBS), and runway balancing (KBRA). Note that as the arrival rate increases, the benefits from KBS and KBRA increase in terms of both absolute delay savings and percentage of delay savings. For an arrival rate of 72 aircraft per hour, the result is a delay savings of 25%. Also note that the majority of delay savings comes from the KBRA algorithm. This result is consistent with previously published results[4]. It should be noted that the KBRA algorithm depends on the results of the KBS algorithm in order to compare potential delay savings, therefore making KBS essential as a foundation for KBRA.

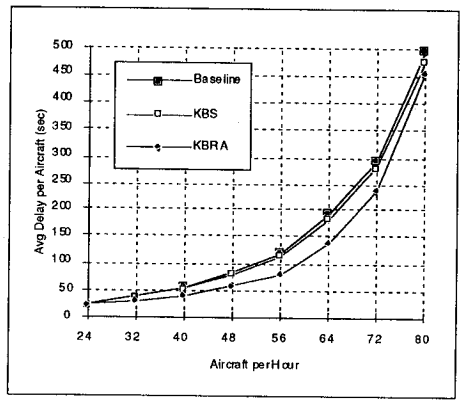


Fig. 11. Potential benefits of knowledge-based sequencing and runway allocation.

The KBS and KBRA algorithms have also been tested in over two thousand hours of real-time controller-in-the-loop simulations. These simulations were for the Dallas/Fort Worth terminal area and scenarios were built based on live traffic samples of peak traffic periods. The results of the simulations show that controllers felt that their workload was reduced while controlling up to 20% more traffic in Instrument Flight Rules (IFR) conditions. Typical arrival rates for Dallas/Fort Worth terminal airspace for a three runway configuration in IFR weather are near 100 aircraft per hour. Real-time simulations have been successfully run with arrival rates of over 120 aircraft per hour using the KBS and KBRA algorithms to advise the controllers on sequence and runway assignment.

Finally, the KBS and KBRA algorithms have been evaluated by controllers in a "shadow-mode" operating with live traffic data at Dallas/Fort Worth. These evaluations produced favorable results as well and have led the controllers to recommend proceeding to further evaluations of the system in the Dallas/Fort Worth training room.

Conclusions

Knowledge-based algorithms for sequencing and runway allocation of arrival air traffic in the terminal area have been developed and tested. The algorithms were developed by combining advanced engineering and computational methods with empirical data gathered from expert air traffic controllers. The knowledge-base sequencing (KBS) algorithm utilizes results from a trajectory-based analysis of the arrival traffic situation. The KBS algorithm examines potential merge points in the arrival traffic flow to determine relative sequences between streams of traffic. KBS then merges the streams together one-by-one, resolving conflicts along the way, until a final sequence is established on the final approach course.

The knowledge-based runway allocation (KBRA) algorithm utilizes the results from the KBS algorithm to test and analyze potential benefits from various runway assignments for arrival aircraft. The KBRA attempts to minimize overall system delay while reducing controller workload. Controller workload is minimized through a series of empirically derived heuristics which direct the KBRA algorithm.

Results from testing which included fast-time and real-time simulation, as well as "shadow" testing with live air traffic data, shows that significant benefits can be achieved by using to the results of these algorithms to advise terminal area air traffic controllers on sequencing and runway assignment. These benefits analyses show that delay reductions of 25% and airport capacity increases of up to 20% are achievable with such an advisory system. In addition, air traffic controllers that have worked with the system in real-time simulation and observed its operation in "shadow" testing have reported a perceived reduction in workload for high traffic scenarios. As a result of the controller evaluations, further testing of the system in an operational environment is planned at the Dallas/Fort Worth terminal facility in the next year.

Appendix A - Fuzzy Logic Parameters for Non - Final Approach Ordering

Inputs

Normalized Separation Distance NSD

$$NSD = \frac{\text{(distance B - distance A)}}{\text{Required Separation}}$$

Speed Difference knots (SD)

$$SD = \text{speed A - speed B}$$

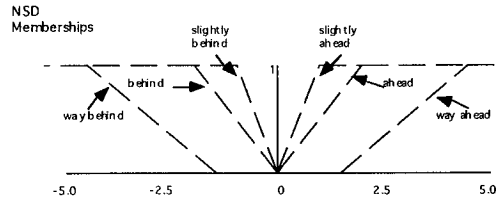
Distance to Trajectory Segment (DTS)

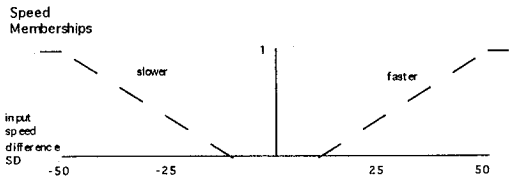
DTS = distance from trajectory segment to current location of aircraft previously ordered behind

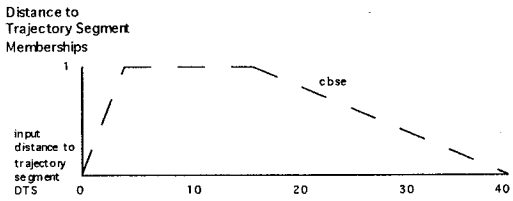
Previous Relative Order (PRO)

IF A previously ahead of B
PRO = 1
IF B previously ahead of A
PRO = -1
IF No previous order
PRO = 0

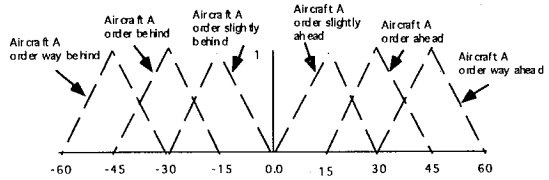
Membership Functions - μ

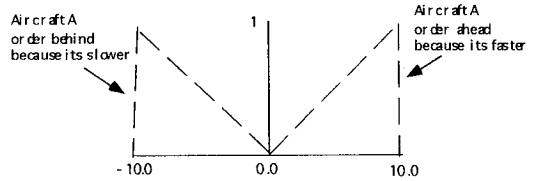


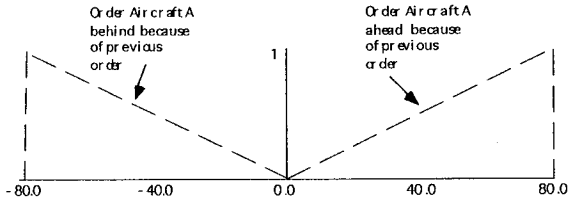




Firing Strengths Ω







Rules

1) IF **slightly ahead** THEN **Aircraft A order slightly ahead**
2) IF **slightly behind** THEN **Aircraft A order slightly behind**
3) IF **ahead** THEN **Aircraft A order ahead**
4) IF **behind** THEN **Aircraft A order behind**
5) IF **way ahead** THEN **Aircraft A order way ahead**
6) IF **way behind** THEN **Aircraft A order way behind**
7) IF **faster** THEN **Aircraft A order ahead because it's faster**
8) IF **slower** THEN **Aircraft A order behind because it's slower**
9) IF Aircraft A was sequenced ahead previously THEN
   IF **close** THEN **Order Aircraft A ahead because of previous order**
10) IF Aircraft A was sequenced behind previously THEN
    IF **close** THEN **Order Aircraft A behind because of previous order**

Appendix B - Fuzzy Logic Parameters for Final Approach Ordering

Inputs

Excessive Delay (D)

$$D_A = STA_{(aircraft\ A)} - Nominal\ Time_{(aircraft\ A)}$$

$$D_B = STA_{(aircraft\ B)} - Nominal\ Time_{(aircraft\ B)}$$

Total Delay Difference (TDD) (positive TDD indicates the order A 1st, B 2nd, reduces delay)

$$TDD = \begin{array}{l}\text{Total Delay incurred by both aircraft} \\ (B\ 1^{st}, A\ 2^{nd})\end{array} - \begin{array}{l}\text{Total Delay incurred by both aircraft} \\ (A\ 1^{st}, B\ 2^{nd})\end{array}$$

Controllability (T)

$$T_A = \frac{Slow\ Time_{(A)} - STA_{(B)}}{Required\ Separation}$$

$$T_B = \frac{Slow\ Time_{(B)} - STA_{(A)}}{Required\ Separation}$$

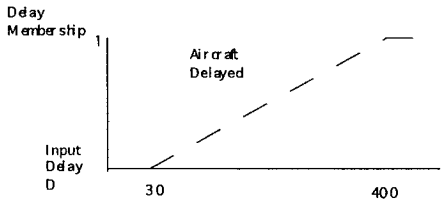
Distance to Trajectory Segment (DTS)

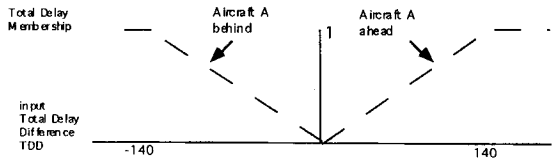
DTS = distance from trajectory segment to current location of aircraft previously ordered behind

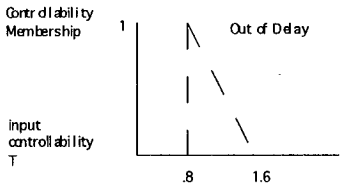
Previous Relative Order (PRO)

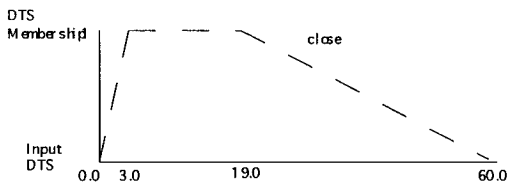
IF A previously ahead of B
PRO = 1
IF B previously ahead of A
PRO = -1
IF No previous order
PRO = 0

Membership Functions - μ

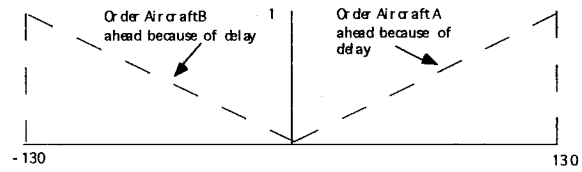


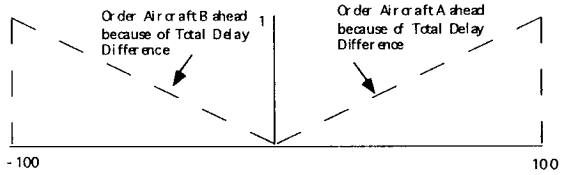


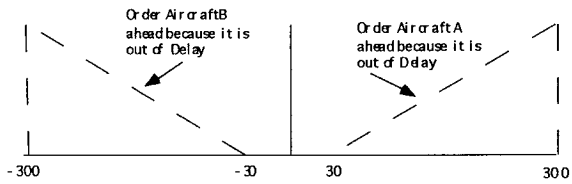


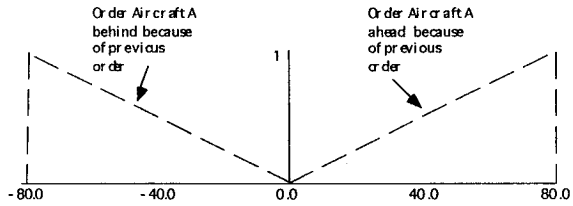


Firing Strengths Ω









Rules

1) IF **Aircraft A ahead** THEN **Order Aircraft A ahead because of Total Delay Difference**
2) IF **Aircraft A behind** THEN **Order Aircraft B ahead because of Total Delay Difference**
3) IF **Aircraft A out of delay** THEN **Order Aircraft A ahead because it is out of delay**
4) IF **Aircraft B out of delay** THEN **Order Aircraft B ahead because it is out of delay**
5) IF **Aircraft A delayed** THEN **Order Aircraft A ahead because of delay**
6) IF **Aircraft B delayed** THEN **Order Aircraft B ahead because of delay**
7) IF Aircraft A was sequenced ahead previously THEN
   IF **close** THEN **Order Aircraft A ahead because of previous order**
8) IF Aircraft A was sequenced behind previously THEN
   IF **close** THEN **Order Aircraft A behind because of previous order**

References

1. Davis, T. J.; Krzeczowski, K. J.; Bergh, C.: "The Final Approach Spacing Tool", 13th IFAC Symposium on Automatic Control in Aerospace, Palo Alto, CA, Sept. 1994.

2. Davis, T. J., Erzberger, H., Green, S. M., and Nedell, W.: "Design and Evaluation of an Air Traffic Control Final Approach Spacing Tool", Journal of Guidance, Control, and Dynamics, Vol. 14, No. 4, July-August 1991, pp. 848-854.

3. Erzberger, H., Davis, T. J., and Green, S. M., "Design of Center-TRACON Automation System," Proceedings of the AGARD Guidance and Control Panel 56th Symposium on Machine Intelligence in Air Traffic Management, Berlin, Germany, 1993, pp. 11-2-11-12.

4. Brinton, C.: "An Implicit Enumeration Algorithm for Arrival Aircraft Scheduling", 11th Digital Avionics Conference, Seattle, WA, October 1992

5. Erzberger, H.; and Tobias, L.: "A Time-Based Concept for Terminal-Area Traffic Management," Proceedings of the 1986 AGARD Conference No. 410, Efficient Conduct of Individual Flights and Air Traffic, pp. 52-1 - 52-14.

6. Yasunobu, S., and Miyamoto S., Automatic Train Operation by Predictive Fuzzy Control, Industrial Application of Fuzzy Control (M. Sugeno, Ed), North Holland, Amsterdam, 1985, pp. 1-18

7. Steinberg M..; Development and Simulation of an F/A - 18 Fuzzy Logic Automatic Carrier Landing System, IEEE Conference on Fuzzy Systems, 1993 Vol. 2, pp. 797-802

8. Turksen, I. B., ; Fuzzy expert systems for operations research and management science, Application of Fuzzy Logic Technology : vol. 2061, pp 39-45; Sept. 1993

9. Berenji, H. R., ; Learning and Tuning Fuzzy Logic Controllers Through Reinforcement, IEEE Transaction on Neural Networks: vol. 3, Number 5 pp 725-740; Sept. 1992