D.A. Spencer

Applying Artificial Intelligence Techniques to Air Traffic Control Automation

We have developed a computer program that automates rudimentary air traffic control (ATC) planning and decision-making functions. The ability to plan, make decisions, and act on them makes this experimental program qualitatively different from the more clerical ATC software currently in use. Encouraging results were obtained from tests involving simple scenarios used to train air traffic controllers.

The primary responsibility of air traffic control (ATC) is the prevention of aircraft collisions. An important secondary responsibility is to expedite traffic. These and other duties are all currently performed by human air traffic controllers. Increasing numbers of flights, however, are straining the system—a trend that is expected to continue. Consequently, the Federal Aviation Administration (FAA) has turned to computer-based aids to assist controllers in performing certain ATC functions.

The current generation of such aids came online in the early 1970s. The aids can be characterized as clerical in nature; i.e., they help human controllers to envision current and future traffic situations, and they provide information needed for directing aircraft. With a few very limited exceptions, however, the current automation system does not recommend actions to controllers.

Planned hardware and software updates [1] should improve these aids and lead to further increases in controller productivity. Nonetheless, within the foreseeable future the possibilities for improving the automation of purely clerical functions will be exhausted. Thus further productivity increases will require aids for automating the planning and decision-making processes. To that end, the FAA has embarked on the Automated En Route ATC (AERA) and Terminal ATC Automation (TATCA) projects [2–4]. The FAA's goal is to develop planning and decision-making aids for directing aircraft that are flying both en route and within an airport terminal's airspace. Under the third phase of AERA, the FAA has also considered the possibility of totally automating certain ATC functions or providing totally automated ATC in some sectors of airspace. Unfortunately, very little previous research is applicable to the design of ATC planning and decision aids, particularly to those aids which must operate autonomously.

At Lincoln Laboratory, we have developed an artificial intelligence (AI) program that automates rudimentary ATC planning and decision-making functions. We tested this experimental program by applying it to two training scenarios that were used for the on-the-job training of air traffic controllers at an en route ATC facility. Without manual assistance, the AI program handled both scenarios successfully, and was able to keep up with the first (and easier) scenario in real time. It is important to note that our intent was to develop a program with a novice level of proficiency sufficient to handle the two training problems. The available resources did not allow the development of a robust expert controller. However, the AI system was structured so that, given further development, it could evolve into a true expert controller.

Our research differs from most previous work in that we considered an en route controller's total job, not just particular subfunctions. (See the box "A Brief History of Artificial Intelligence Approaches to ATC.") The AI program that we developed makes plans and acts on them. This capability distinguishes the program from the current generation of operational ATC automation tools, which, as stated earlier, simply perform clerical tasks that aid the

A Brief History of Artificial Intelligence Approaches to ATC

R.B. Wesson [1] was the first to apply artificial intelligence (AI) techniques to ATC. His work influenced much later research, including our work at Lincoln Laboratory. Wesson attempted to automate the complete controller's function rather than focusing on one aspect of the controller's job. As a means of testing his system, he applied it to simulated ATC problems that are used to train controllers at en route sites.

Air Route Traffic Control Center (ARTCC) training problems require controllers to handle a sector of airspace under a simulated traffic load for approximately one hour. The sector is a real-life sector within the region of the trainee's ATC facility. Although the traffic is simulated, the training problems include realistic situations for that sector. The training program for new controllers involves progressing through a sequence of these problems. The initial problems have relatively light traffic; the later ones create a workload heavier than what is expected under the most demanding real-world conditions.

Wesson was able to implement enough of the controller's functions to handle successfully the basic aspects of several of the problems. Our work attempts to reproduce Wesson's results (with different scenarios and a somewhat different planning technique) and tries to provide a basis for developing a completely automated air traffic controller.

Wesson later worked on a Rand Corporation team that used ATC to study distributed expert systems for planning and control [2, 3]. Since the team focused on alternatives to the current geographically distributed control system, much of the research is not applicable to our work because practical considerations constrain the existing ATC system to evolve gradually from its current state.

The Rand team also analyzed the proposed Automated En Route ATC (AERA) system [4] for the Federal Aviation Administration (FAA). From that work, a paper by Wesson [5] resulted. Using both analysis and dramatization, the paper provides a persuasive vision of how the current ATC system could evolve into the largely automated system that is planned as the last phase of AERA.

Following Wesson's work, S.E. Cross [6] applied techniques from the qualitative-physics area of AI research to represent the understanding a controller might have of the constraints that aerodynamics places on ATC actions. Another interesting aspect of Cross's thesis is a technique for decomposing a multiple-conflict scenario into a fundamental or minimal set of conflicts that need to be resolved. Like Wesson, Cross also applied his system to solving ATC training-scenario problems, but he focused only on conflict resolution. Furthermore, Cross dealt with static situations rather than dynamic simulations. Similarly, the Advisor for the Intelligent Resolution of Predicted Aircraft Conflicts (AIRPAC) system of C.A. Shively and K. Schwamb [7, 8] demonstrates a rule system that handles just static-conflict situations. More recently, J.D. Reierson [9] reported on the use of a rule-based system with dynamically simulated traffic to support the development of the third and most autonomous phase of AERA.

Solving static problems typically involves setting up a conflict between two or more aircraft whose initial locations, velocities, and flight plans are given. The computer program must then determine the best way to resolve the conflict. The aircraft, however, are never moved or given the resulting ATC clearances, nor do other aircraft appear and cause subsequent conflicts at later times, nor are other ATC considerations typically taken into account. Although the static method clearly captures some aspects of ATC problem solving, it has the same artificial relation to ATC as chess problems do to a game of chess.

Note, too, that there has been extensive AI research in general problem-independent planning techniques but it appears that this type of research is not applicable to the ATC problem. (See the box "Applying Artificial Intelligence Techniques to General Planning.")

The work reported in this article was influenced by earlier work at Lincoln Laboratory on the Electronic Flight Rules (EFR) system [10]. The system, which investigated the automation of some aspects of ATC, allowed pilots the freedom of visual flight rule (VFR) navigation but provided automated conflict resolution on an as-needed basis. Although the EFR conflict-resolution method was not an AI system per se, the method's successful use of a cost function to evaluate proposed alternatives influenced our research.

References

1. R.B. Wesson, *Problem-Solving with Simulation in the World of an Air Traffic Controller*, Ph.D. thesis, Dept. of Computer Sciences, University of Texas (Austin, TX, 1977).
2. P. Thorndyke, D. McArthur, and S. Cammarata, "Autopilot: A Distributed Planner for Air Fleet Control," *Rand Corporation Report N-1731-ARPA* (Santa Monica, CA, July 1981).
3. R. Steeb, D.J. McArthur, S.J. Commarata, S. Narain, and W.D. Giarla, "Distributed Problem Solving for Air Fleet Control: Framework and Implementation," in *Expert Systems: Techniques, Tools, and Applications*, P. Klahr and D. Waterman, eds., (Addison-Wesley, Reading, MA, 1986), pp. 391–432.
4. R.B. Wesson, K. Solomon, R. Steeb, P. Thorndyke, and K. Wescourt, "Scenarios for Evolution of Air Traffic Control," *Rand Corporation Report R-2698-FAA* (Santa Monica, CA, November 1981).
5. R.B. Wesson, "An Alternative Scenario for Air Traffic Control: Shared Control," *Rand Corporation Report P-6703* (Santa Monica, CA, Aug. 1981).
6. S.E. Cross, *Qualitative Reasoning in an Expert System Framework*, Ph.D. thesis, Coordinated Science Laboratory, University of Illinois Report T-124 (Urbana-Champaign, IL, May 1983).
7. C. Shively and K. Schwamb, "AIRPAC: Advisor for the Intelligent Resolution of Predicted Aircraft Conflicts," *The MITRE Corporation Report MT-84W164*, (McLean, VA, 1984).
8. C.A. Shively, "AIRPAC: Advisor for Intelligent Resolution of Predicted Aircraft Conflicts," in *Transportation Research Circular No. 310, Artificial Intelligence and the Air Traffic Control System* (Washington, DC, Dec. 1986). Report of a workshop held in October 1985, under the sponsorship of the Committee on Airfield and Airspace Capacity and Delay of the Transportation Research Board.
9. J.D. Reierson, presentation to the workshop on Artificial Intelligence and Air Traffic Control, NASA Ames Research Center, Moffett Field, CA, 9–10 February 1989. Published in *Advanced Computer Technology for NAS*, p. A-69, by the Advanced ATC Concepts Branch (ADS-120) of the Federal Aviation Administration.
10. J.W. Andrews and W.M. Hollister, "Electronic Flight Rules: An Alternative Separation Assurance Concept," *Project Report ATC-93*, Lincoln Laboratory (31 Dec. 1980), FAA-RD-80-2.

human decision maker.

Our project required the development of a working model of ATC planning and decision making. This article presents the model along with justifications for choosing its particular structure, and mentions some of the problems that result from using a fairly typical rule-based programming system in an ATC environment. Then we discuss better alternatives that might be taken. Finally, a simple example extracted from one of the training scenarios with slight modifications is given to illustrate the operation of the planner.

The work reported in this article addresses the demonstration of automated problem solving in an ATC context. Two important subjects—the role of such an automated decision maker within the overall ATC system, and how such a system will communicate with controllers and pilots—were not studied.

An Approach to Automated ATC

What Controllers Do

A detailed account of the tasks and responsibilities of air traffic controllers is beyond the scope of this article. For readers seeking such information, Ref. 5 gives a very basic introduction to the operation of the ATC system; the ATC handbook [6] contains the official description of the controller's responsibilities; Ref. 7 provides a more engineering-oriented description; and the *Airman's Information Manual* [8] presents ATC procedures from a pilot's point of view. A qualitative (if somewhat sensationalized) account of what a controller's job is like can be found in Ref. 9.

The following description focuses on those aspects of the controllers' job which must be handled by the automated system during the two training test problems. Specifically, the following functions will be covered:
(1) coordinating with other sectors,
(2) navigating aircraft,
(3) issuing altitude clearances, and
(4) maintaining aircraft separation.

Coordinating with other sectors. Each controller handles traffic in a volume of airspace called a sector. Working together, controllers function as a geographically distributed control system. The sectors may be adjacent either horizontally across the sector boundaries or vertically at specified floor and ceiling altitudes. A given aircraft is handed off from sector to sector as it flies along its route. This handoff of control responsibility is the basic coordination function between sectors. A controller must not allow an aircraft to leave his sector until the controller in the next sector accepts responsibility for the vehicle.

Navigating aircraft. A pilot must file a flight plan as a prerequisite to entering the ATC system under Instrument Flight Rules (IFR). (Under IFR, controllers are responsible for assuring safe separation between aircraft. It is also possible to fly under Visual Flight Rules [VFR], where the pilot assumes the responsibility for maintaining separation from other aircraft. Under VFR, the pilot is not required to have extensive interaction with the ATC system.) As part of an IFR flight plan, the pilot must precisely indicate what route of flight will be followed. This procedure consists of chronologically listing either the specific navigational fixes over which the aircraft will fly, or the segments of airways that will be used. (Airways are standard flight routes that appear on aeronautical charts).

Before a flight commences, the ATC system must approve the route and issue a clearance to the pilot. The pilot is expected to navigate the aircraft along the agreed-upon route. Controllers are not responsible for navigating aircraft, unless they elect to vector one off its route onto a specified magnetic heading. In such a situation, controllers are responsible for the detailed navigation of the aircraft until it is put back onto its original route or onto an alternative route that the aircraft is capable of navigating on its own. Vectoring is permitted only when the controller can observe the aircraft on radar and thus can determine its precise location.

Issuing altitude clearances. Controllers are much more responsible for determining aircraft altitudes. Although pilots indicate preferred cruise altitudes on their flight plans, they do not specify their vertical profiles as a function of their positions along the routes. Before they can change from one assigned altitude to another, pilots must wait for altitude clearances from controllers. In principle, this procedure is no different from that of receiving route clearances. An aircraft, however, will typically receive only one route clearance for the entire en route portion of a flight, whereas several altitude clearances are usually required. In general, pilots prefer an immediate clearance to their filed cruise altitude at departure and another clearance to the airport approach altitude at a convenient distance from the destination airport. However, controllers often issue a series of clearances so that aircraft must climb and descend in stages.

Alternatively, a controller might issue what amounts to an altitude profile in the form of a clearance with altitude restrictions that tell pilots to be at, at or above, or at or below certain altitudes at certain points on their specified routes. A series of such restrictions may be issued in one clearance that constrains the vertical profile over some portion of the route.

It is important to note that although the controller specifies the time at which an aircraft may change its altitude and the altitudes that are allowable, the controller does not control the rate of climb or descent. Because the climb and descent rates depend on the performance characteristics of a particular aircraft, the pilot must control the rates. A controller can exercise some control by means of altitude restrictions that force certain minimum rates, but a pilot is free to reject such clearances if they are deemed unreasonable. From FAA tables [6], controllers can gain some knowledge of the likely climb and descent rates for most types of aircraft.

Maintaining aircraft separation. Safe distances between aircraft can be maintained either in a horizontal or vertical fashion. Horizontal-separation standards may be specified in terms of distances or times; vertical separations are always in terms of altitudes. The horizontal-separation standards that apply to a given situation are a function of a large number of variables [6]. In the training scenarios used to test our computer program, the required horizontal separation was 5 nautical miles (nmi). Vertical-separation standards are a function of altitude. For aircraft flying below Flight Level 290 (FL 290, approximately 29,000 ft), the minimum separation is 1,000 ft; for aircraft above FL 290, the minimum is 2,000 ft because of the lower accuracy of altimeters at high altitudes.

To maintain the required minimum separations, controllers use a variety of methods. Under radar surveillance, aircraft may be vectored so that the vehicles remain horizontally separated. Under both radar and nonradar conditions, controllers may assign altitudes, speeds, and revised routes of flight to aircraft, or an aircraft might be delayed by holding it at a particular point. To hold an aircraft, the controller instructs the pilot to perform either a standard 360° turn or a racetrack-shaped pattern at some point to which the pilot can reliably return by using the aircraft's navigational equipment. For a situation in which many aircraft must be held at the same point, the controller separates the vehicles vertically. This procedure results in a holding stack of aircraft. Another delaying tactic, which can be performed only when an aircraft is under radar surveillance, is to vector the aircraft through path-stretching maneuvers such as S-turns.

In general, vectors or altitude changes are the preferred methods of maintaining aircraft separation. Because speed adjustments are greatly restricted by aircraft performance capabilities, such adjustments are used in en route ATC

Applying Artificial Intelligence Techniques to General Planning

Reference 1 reviews the large body of research on applying artificial intelligence (AI) techniques to general planning, i.e., planning that is independent of specific tasks such as air traffic control (ATC). The report places the work in a common analytical framework. By and large, the approaches seem to have little applicability to real-life problems. The approaches focus on achieving some well-defined end point or goal by determining the subgoals necessary to accomplish that goal. Steps are taken to ensure that the methods used to achieve the subgoals do not interfere with one another. The state of the system being controlled is represented by logical assertions about the relationships of various objects within the system, and changes occur as discrete state changes. For example, in the simple toy-block stacking problems often used to illustrate the behavior of these planners, a block is represented as being on a table's surface at one step, and then on top of another block at the next step. There is no representation for the continuous process of moving the block between the two states.

One difficulty in applying this method to ATC problems is that in ATC there exist no particular end states that need to be achieved. That is, in general a large number of possible future situations are acceptable. Another difficulty is that the use of logical assertions does not capture the continuous behavior of physical systems such as aircraft in flight, and it also introduces a number of artificial logical problems to the system.

Instead of using the above approaches as a foundation, the planning methods described in our AI research derive more from the techniques used in game-playing algorithms such as chess programs. These games have the common characteristic that they cannot be thoroughly analyzed by working backwards from a desired end point. This characteristic is due, in part, to the existence of both a large number of acceptable end points and an even larger number of possible intermediate states leading to those end points. Thus, instead of working backwards, chess programs explore forward from the current situation. The analysis is done by the simulation of various possible moves and countermoves and the evaluation of their consequences. Numerous techniques that streamline the search process have been developed, and much work has investigated the accurate evaluation of the hypothesized board positions. Although chess and most other games are represented by discrete board states, the same principles can be applied to continuous systems by using the appropriate simulations. This approach appears to avoid all of the problems of time and change representation that plague those systems which use logical assertions or assumptions of discrete-state behavior.

References

1. D. Chapman, "Planning for Conjunctive Goals," *Artificial Intelligence Laboratory Technical Report 802*, MIT (Nov. 1985).

mainly for maintaining the desired separations between aircraft in a sequence. Holding is a powerful technique, but because of the large delay involved (aircraft need at least two minutes to execute one 360° turn and longer for a racetrack-shaped pattern), it is not used except when other methods will not work.

A particularly difficult situation occurs near airports where aircraft arriving along several merging routes must be formed into a single stream of traffic. At such a location, organizing the arriving aircraft into a final landing sequence is the major function of airport-approach controllers. The goal is to arrange the aircraft in a sequence with exactly the minimum allowable separation, thus achieving the maximum landing rate possible. Sequencing may also be needed for en route aircraft in areas where heavily traveled airways merge.

Why Is ATC Automation Difficult?

In automating the ATC decision-making process, the central problem is that many factors might bear on a particular decision, and there are usually a large number of actions or sequences of actions that could potentially be taken to resolve any problem. The factors vary from sector to sector (depending on the local geography and traffic flows) and from situation to situation (depending on the exact configuration of aircraft in the sector).

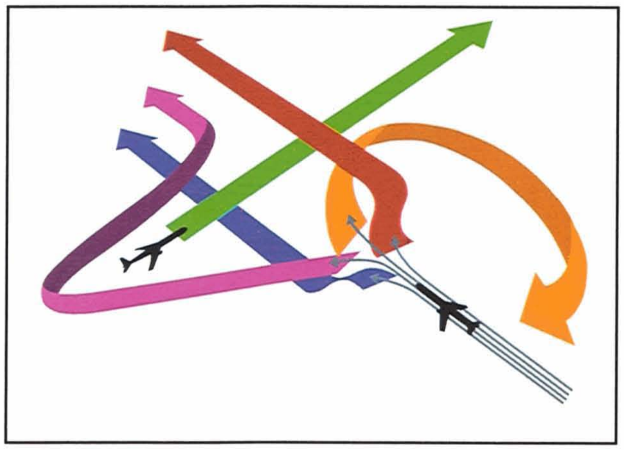


Fig. 1—Alternatives for resolving aircraft conflicts.

For example, suppose two aircraft are flying on courses that cross each other. The vehicles are flying level at the same altitude, and their speeds and distances from the crossing point are such that the separation standards discussed earlier are likely to be violated (Fig. 1). If only this much information is available, possible resolutions to the conflict might be to turn one aircraft so that it travels behind the other, adjust the altitude of one or both aircraft so they are vertically separated at the crossing point, or delay one of the aircraft by either decreasing its speed or by implementing a delaying turn or holding maneuver. However, other considerations such as the following might invalidate some of the above options:

- There might be other aircraft in the vicinity and the proposed maneuvers might create conflicts with them. In some cases, it is desirable to resolve such secondary conflicts by maneuvering the secondary aircraft. It might be better in other instances to resolve the original conflict another way.
- Severe weather conditions might prohibit some of the proposed options. Also, an aircraft's proximity to the ground, mountains, or other physical obstructions might forbid certain maneuvers.
- If a maneuver forces an aircraft to cross or come close to a sector boundary, the controller is burdened with the additional work of having to coordinate the maneuver with the controller of the adjacent sector. This solution, however, might be acceptable if other options are even less desirable.
- Aircraft that are close to their maximum flying altitude will be unable to climb farther. Furthermore, aircraft without pressurized cabins or oxygen masks are forbidden to climb above 10,000 ft.
- Certain maneuvers, either by themselves or by the secondary conflicts they cause, might result in undue delays.
- Requiring a jet to fly long distances at low altitudes wastes fuel.
- It is inefficient to force a climbing aircraft to descend or a descending aircraft to climb.
- If the descent of an arriving aircraft is delayed, the vehicle might not have enough

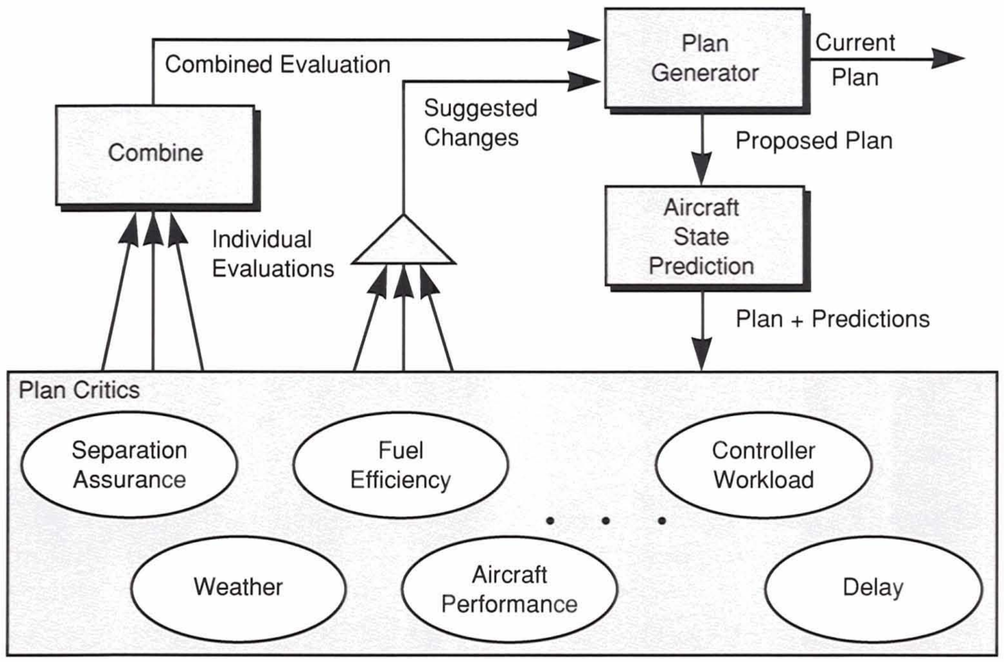


Fig. 2—Plan evaluation and modification flowchart.

time to reach the appropriate altitude required to approach the airport for landing purposes.

Hundreds of such considerations can affect the acceptability of a particular solution. Whether a particular consideration is applicable depends on the context in which the situation occurs, e.g., the geographic location and the existence of other aircraft in the vicinity.

It is difficult to devise a software structure that can accommodate all applicable considerations, evaluate all possible solutions, make the necessary trade-offs among the solutions, and, at the same time, be modular, maintainable, adaptable to local-site considerations, and fast enough to keep up with the dynamic nature of ATC.

Problems often arise when an automatic decision-making system is integrated with humans. Specifically, what recourse should be taken if human controllers disagree with the decisions of the expert system? One approach would be to make the automated system so intelligent that its plans and decisions were always understood by and usually acceptable to human controllers. A model for this interaction would be the current practice of having controllers work in teams of two or three to control a sector. In the team, one person is the primary controller and the others are responsible for relieving him of certain auxiliary functions. Communication among the team members depends on each one's comprehension of the ATC process at a very high level so that everyone understands everybody else's actions without any need for long explanations. However, for the automated system to achieve such a high level of competence, it would have to take into account *all* of the above-mentioned considerations. As stated earlier, the current clerical systems avoid this problem, as they merely supply additional data to controllers. That is, the current systems do not have to integrate the data into a final decision.

A Software Structure for ATC Problem Solving

Figure 2 diagrams a software structure that appears to address the above concerns. The structure also seems capable of solving problems in a way that is intuitive and understandable to human controllers. In the figure, a plan generator produces plans that detail how the automated controller will handle the current traffic situation for the specified sector. Each of the sector plans contains a set of individual aircraft plans that specify the routes of flight for the aircraft and the ATC clearances that are planned at particular points along these routes. Clearances might be altitude changes, speed changes, vectors, or holding commands. From the above information, it is possible to project within an approximate range the future horizontal positions and altitudes of the aircraft.

Given a plan and projections of the future positions and altitudes of aircraft, it is possible to write *plan critics*. A plan critic is an independent software module that is responsible for looking for a particular type of undesirable feature or consequence of a plan. Each plan critic produces a score that represents the module's evaluation of the plan from the module's particular point of view. In our system, the higher the score, the more severe are the problems with the given plan.

The individual scores are then weighted and combined by a simple summation into an overall score for the plan. The resulting score is fed back to the plan generator, which uses the score to rank the given plan against other possible plans. It is this combining function that allows the system to make trade-offs among the various considerations represented by the individual critics [10]. The individual scores are weighted to give the correct trade-offs among the various problems so that the system ranks plans in a desired order. In the current implementation of our system, the weights are adjusted by empirical analysis. The weight-adjusting process (which reflects learning) could perhaps be automated with techniques that neural network researchers are currently exploring. It might also be possible to determine the weights in an analytic way, e.g., by assigning to each type of problem an estimated cost in dollars.

The overall effect of the scoring process is the creation of a function that takes a plan as its argument and returns a number. Given that higher scores denote less acceptable plans, the plan generator's goal is to find a plan with as small a score as possible. Unfortunately, analytic techniques cannot be applied because the function is highly nonlinear. Also, the exact function will change as the system evolves and new plan critics are added. Thus an efficient search process is needed. Undirected search could become very expensive, as could search for the true optimum plan.

AI systems typically require the derivation of an efficient search algorithm. A common approach is to apply knowledge of the task domain (in our case, air traffic control) to restrict the search process so that good, albeit suboptimal, solutions are found quickly. In the approach proposed in this article, the same plan critics that score the plans can often provide suggestions as to how a plan could be improved from a local point of view. However, a particular plan critic is not able to determine the effect of its suggestions on the overall score because the critic purposely has no knowledge of other critics. Thus critics merely feed their suggestions to the plan generator, which generates new plans corresponding to the suggestions and passes the plans back to the critics for evaluation.

In searching for a solution, the plan generator explores some portion of a search tree in which the tree's root is the original plan. In the tree, child nodes represent plans that have been derived from their parent node's plan. The plan generator implements a particular search strategy to find the branch of the search tree that should be pursued next. The search strategy selects some already evaluated plan and its corresponding suggestions, and generates modified versions of the plan according to the suggestions. The search strategy may then elect to pursue one of the new branches further or to follow some other older branch that appears more promising. A stopping criterion that determines when a plan is acceptable must be defined. Acceptable plans are passed to that part of the system which implements the current plan. The current plan remains in effect until the occurrence of any event that increases the plan's score. An example would be the appearance of a new aircraft whose individual plan conflicts with the individual plans of some of the aircraft that are already known to the system. This type of situation initiates a new round of searching.

Limitations of the Current System Implementation

Our expert system is implemented in a combination of LISP and YAPS (Yet Another Production System) [11, 12]. As its name implies, YAPS is a fairly standard rule-based system that uses forward chaining [13]. YAPS allows arbitrary LISP functions to be used both on the right-hand side of rules (the *then* part of an *if-then* rule) and as tests on the left-hand side (the *if* part). This flexibility contrasts with many other rule-based systems whose computational capabilities are restricted to the functionality provided by their language designers. The implementation of YAPS used in our expert system was originally converted from Franz LISP to Zetalisp at MIT. Subsequent modifications at Lincoln Laboratory improved the implementation's execution speed and debugging facilities.

The general architecture described above appears to have the power and flexibility needed for automation of the ATC decision-making process. The current implementation, however, is limited in several respects and much further work is required to develop it fully. But even with the limitations, the expert system was able to handle the first two training problems of the Boston Air Route Traffic Control Center (ARTCC).

The current system has a very simple search procedure that searches only one level deep in the tree. The one-level restriction results in the following limitation. When the current plan has a problem, the plan generator will look only at those alternatives which are immediately derived from that plan. If some of the alternatives also have problems (such as secondary conflicts caused by maneuvers that resolved the initial problem), then the alternatives will receive high scores and will most likely be rejected. Note that if the system were to search deeper to generate resolutions to secondary problems, a plan that was better than any of the first-level plans might be discovered.

Another limitation of the current system is its meager number of plan critics. The implemented critics identify conflicts between aircraft, detect when an incoming aircraft will not reach the appropriate terminal altitude in time, and complain if an overflight is cleared too far from its requested cruise altitude. For the current system to evolve into a competent air traffic controller, we must develop all of the needed critics and adjust their scores so that they reflect rankings that human controllers would consider appropriate. In the architecture of our system, the plan critics and the scoring function are the primary repositories of ATC knowledge.

The current system is also constrained by its planning and execution functions, which have a very limited repertoire of actions that are available to resolve problems. Thus most conflicts are resolved by altitude changes. The only exceptions are situations in which the routes of two aircraft merge and a conflict occurs either at the merge point or at the time when one of the aircraft overtakes the other. These situations are resolved by directing one of the aircraft to make a 360° right turn. Another weakness of the current system is that it does not check to make sure that the airspace in which this turn will be made is free of conflicts.

Because of the limited number of options and the limited depth of search, the plan generator can try all of the suggested solutions to a given conflict in a reasonable amount of time. Ultimately the plan generator will have to use a more selective strategy that searches only the most promising branches of a search tree. In such a strategy, the best plan (as defined by the system's evaluation function) might be missed under certain circumstances. The nature of ATC, however, is such that there are typically many acceptable solutions to any one problem, and finding the best solution is usually unnecessary. It is also possible in the future that those implementations which are optimized for speed and/or implementations that use special hardware might be able to conduct a full search in real time.

A Development Environment for the ATC Expert System

To test the automated controller with differ-

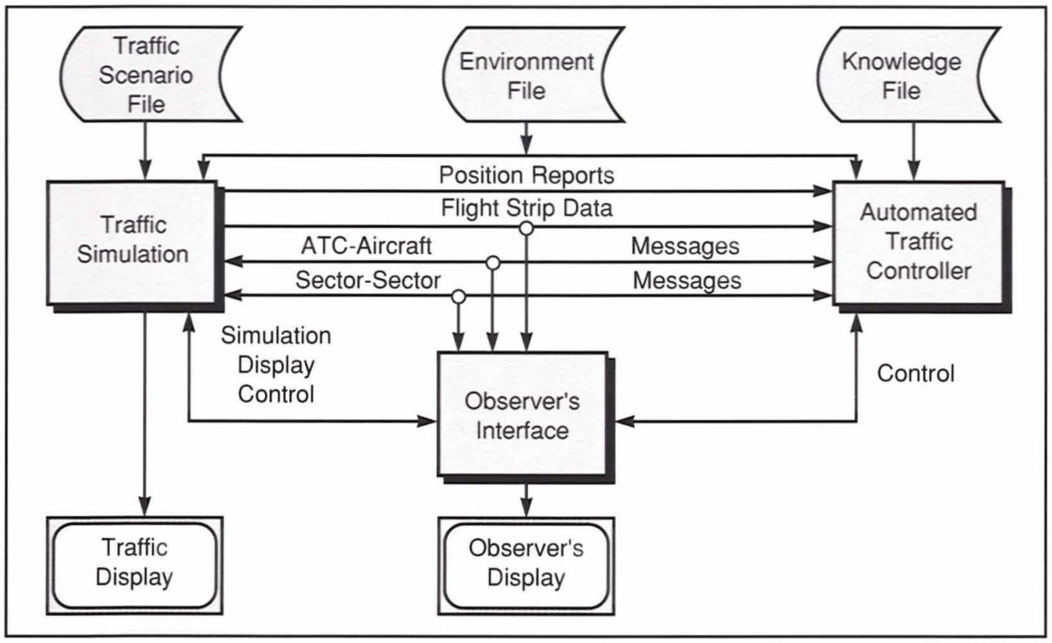
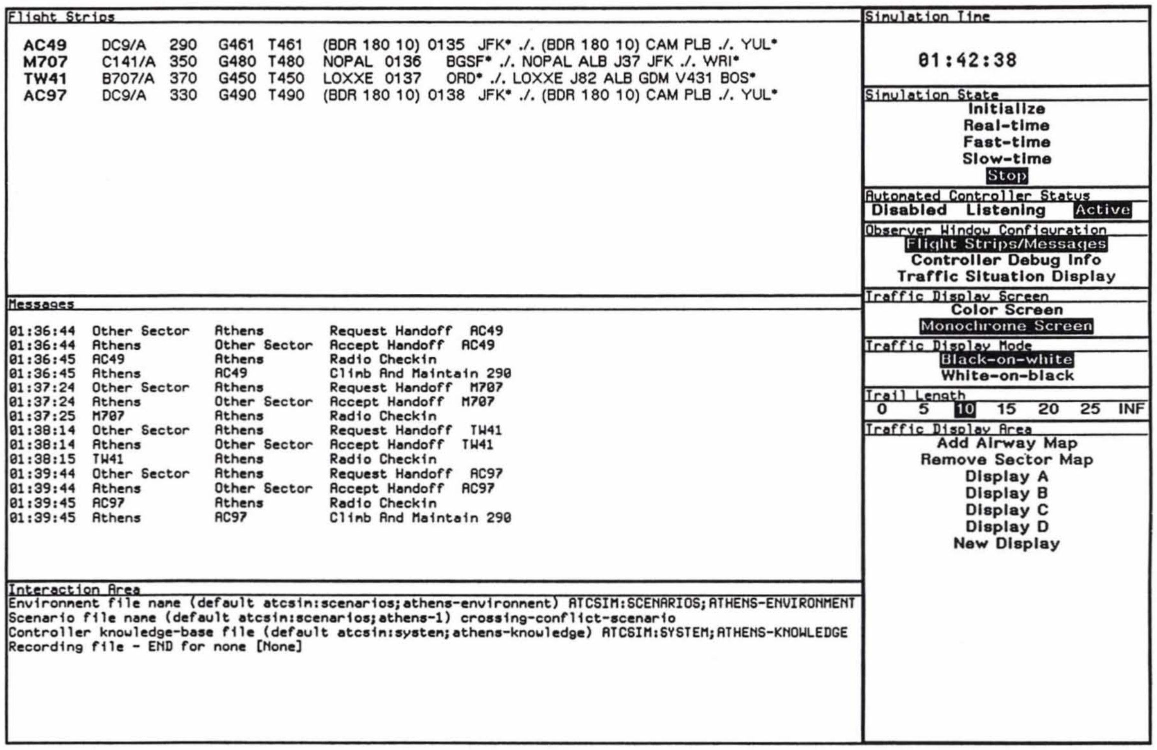


Fig. 3—Development environment of the ATC expert system.

ent traffic scenarios, we implemented a development and testing environment (Fig. 3) on a Symbolics 3670 computer. A set of interfaces provides the expert system with the same information that a real controller receives. Such information includes radar position reports;



```
Flight Strips

AC49    DC9/A   290  G461  T461   (BDR 180 10) 0135  JFK* ./. (BDR 180 10) CAM PLB ./. YUL*
M707    C141/A  350  G480  T480   NOPAL  0136    BGSF* ./. NOPAL ALB J37 JFK ./. WRI*
TW41    B707/A  370  G450  T450   LOXXE  0137    ORD* ./. LOXXE J82 ALB GDM V431 BOS*
AC97    DC9/A   330  G490  T490   (BDR 180 10) 0138  JFK* ./. (BDR 180 10) CAM PLB ./. YUL*

Messages

01:36:44  Other Sector  Athens        Request Handoff  AC49
01:36:44  Athens        Other Sector  Accept Handoff  AC49
01:36:45  AC49          Athens        Radio Checkin
01:36:45  Athens        AC49          Climb And Maintain 290
01:37:24  Other Sector  Athens        Request Handoff  M707
01:37:24  Athens        Other Sector  Accept Handoff  M707
01:37:25  M707          Athens        Radio Checkin
01:38:14  Other Sector  Athens        Request Handoff  TW41
01:38:14  Athens        Other Sector  Accept Handoff  TW41
01:38:15  TW41          Athens        Radio Checkin
01:39:44  Other Sector  Athens        Request Handoff  AC97
01:39:44  Athens        Other Sector  Accept Handoff  AC97
01:39:45  AC97          Athens        Radio Checkin
01:39:45  Athens        AC97          Climb And Maintain 290

Interaction Area
Environment file name (default atcsim:scenarios;athens-environment) ATCSIM:SCENARIOS;ATHENS-ENVIRONMENT
Scenario file name (default atcsim:scenarios;athens-1) crossing-conflict-scenario
Controller knowledge-base file (default atcsim:system;athens-knowledge) ATCSIM:SYSTEM;ATHENS-KNOWLEDGE
Recording file - END for none [None]

Simulation Time
      01:42:38

Simulation State
      Initialize
      Real-time
      Fast-time
      Slow-time
      Stop

Automated Controller Status
 Disabled   Listening   Active

Observer Window Configuration
      Flight Strips/Messages
      Controller Debug Info
      Traffic Situation Display

Traffic Display Screen
      Color Screen
      Monochrome Screen

Traffic Display Mode
      Black-on-white
      White-on-black

Trail Length
 0   5   10   15   20   25   INF

Traffic Display Area
      Add Airway Map
      Remove Sector Map
      Display A
      Display B
      Display C
      Display D
      New Display
```

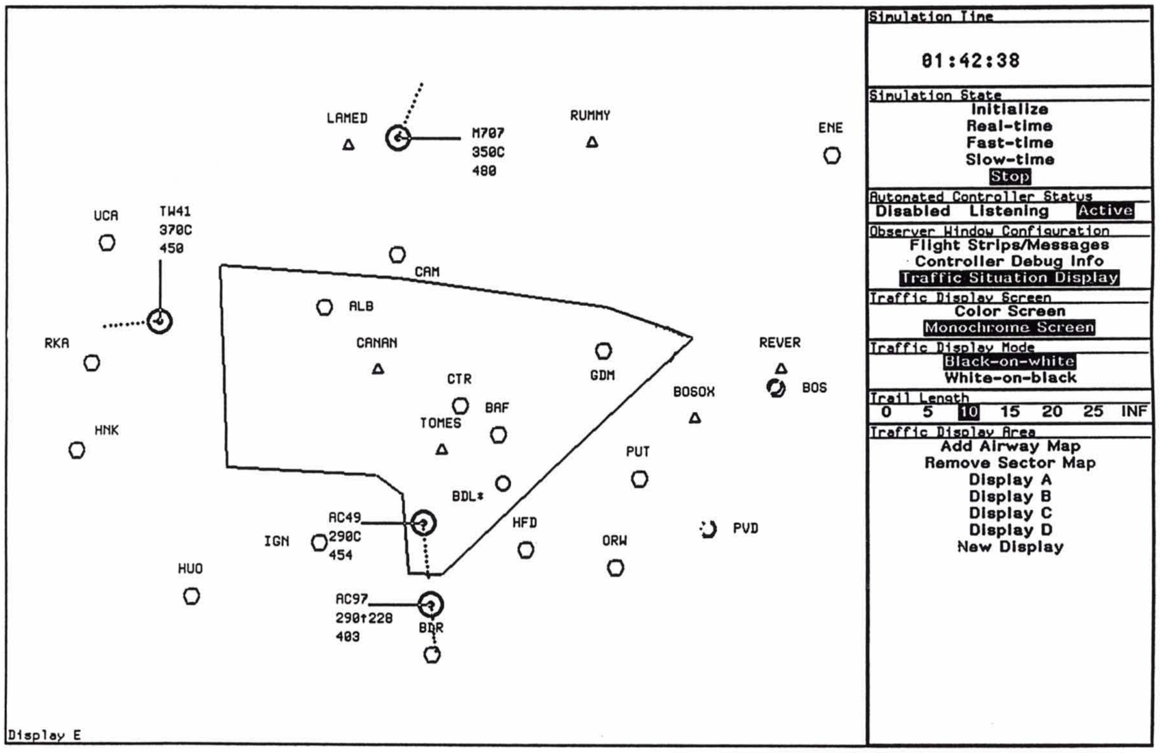
Fig. 4—Observer's display.

flight-strip data, which give the routes of flights and their desired cruise altitudes and speeds; radio messages to and from aircraft; and, for coordination purposes, messages to and from adjacent sector controllers. Except for position reports, information flowing across the interfaces is displayed on a monochrome screen, which allows observation of the system's performance. The display (Fig. 4), which provides menus that allow an operator to control the system's operation, contains three windows: one for displaying flight-strip information, another for displaying messages between the system and pilots or adjacent sector controllers, and a third for entering input parameters from the keyboard.

A color screen displays aircraft positions along with airways, navigational aids, airports, and sector boundaries. Figure 5 shows a monochrome version of this display. In the figure, aircraft positions are represented by dots surrounded by 5-nmi-diameter circles, which provide distance references. The screen contains track histories (information detailing the previous positions of aircraft), whose time lengths are controlled from the observer's display. Each aircraft symbol is accompanied by a data tag similar to those on standard ATC displays. A data tag gives an aircraft's flight identity, cleared altitude, actual altitude (if different from the cleared altitude), and ground speed.

The flight-strip information is mouse-sensitive; i.e., flight strips act as menu items. When the flight strip of an aircraft is selected, a first-level menu appears and allows the operator to do several things: issue ATC commands to the aircraft, change the position of the data tag relative to the aircraft symbol, perform certain actions as if piloting the aircraft, or perform actions relating to intersector handoff of aircraft responsibility. When the ATC-command or pilot-action mode is selected, a second-level menu appears and shows the clearances or actions accepted by the simulated aircraft.

A major component of this development environment is a traffic simulator based on a system developed by Antonio Elias and John Pararas at the MIT Flight Transportation Laboratory. The



Simulation Time
01:42:38

Simulation State
Initialize
Real-time
Fast-time
Slow-time
[✓ Stop]

Automated Controller Status
Disabled Listening [✓ Active]

Observer Window Configuration
Flight Strips/Messages
Controller Debug Info
[✓ Traffic Situation Display]

Traffic Display Screen
Color Screen
[✓ Monochrome Screen]

Traffic Display Mode
[✓ Black-on-white]
White-on-black

Trail Length
0 5 [✓ 10] 15 20 25 INF

Traffic Display Area
Add Airway Map
Remove Sector Map
Display A
Display B
Display C
Display D
New Display

Fig. 5—Traffic-situation display.

resulting simulator is driven by a traffic scenario file that specifies the same information for each aircraft as would be found on a flight plan, specifically, information regarding the type of aircraft and the vehicle's flight route, cruise altitude, and cruise speed. Other initialization information such as the altitude of the aircraft can be provided, and there is also a mechanism for specifying the time when certain ATC actions are scheduled to occur. Such pre-scripted actions include issuing altitude clearances that aircraft would receive in adjacent sectors, and initiating handoffs. This capability mimics a similar time-based function in the simulators currently in use at en route centers. In fact, the entire scenario file is a fairly direct translation of the scenario card decks used at the centers. Data were obtained for the first 10 of the 18 problems that the Boston en route center was using in late 1983. The two simplest problems were turned into scenario files for use by the expert system.

In addition to scenario files, the simulator and the expert system use an environment file, which provides map information about the region of interest. The map information includes the positions of navigational aids and airports, the routes of all airways used in the scenarios, and the boundaries of ATC sectors.

Our development environment serves two purposes. As indicated above, it allows the testing of the expert system on lifelike problems during which developers and human expert air traffic controllers can observe the system's behavior. It is also possible to disconnect the development environment from the expert system and allow the observer to control the traffic through the menus mentioned above. In this way it is possible to see how a controller would handle a certain situation, or to allow developers to try out certain control approaches manually before programming them into the expert system.

An Example

In this section we present a simple example that demonstrates some of the system's concepts and limitations discussed earlier. The example is abstracted from the first of the 18 Boston ARTCC training problems. All of the problems were set in a high-altitude (i.e., above 18,000 ft) sector called the Athens sector. Figure 6 shows the initial traffic situation. The only change made to the original scenario was to adjust the timing of the aircraft labeled M707 so that it would conflict with TW41 over Albany (ALB), N.Y. Flight M707 is an overflight; TW41 is an arrival for Boston (BOS); and AC49, a departure from John F. Kennedy International Airport in New York City, is headed for Montreal.

The upper portion of Fig. 7 shows the altitude of TW41 as a function of the distance traveled along the aircraft's path. TW41 enters from the left and flies level at its cruise altitude. Just before Albany, however, the planner intends to issue a descent clearance. From that point on, TW41's altitude is shown as a band that represents the planner's uncertainty about the aircraft's descent rate. The upper line of the band indicates the minimum descent rate, the lower line the maximum rate, and the middle line the nominal rate. The locations of the rectangular boxes labeled M707 and AC49 show where the two aircraft will cross TW41's path. The boxes indicate a 2,000-ft altitude separation and 5-nmi horizontal separation on both sides of the planned positions of these aircraft. Thus potential conflicts between the aircraft are denoted by areas where TW41's path intersects the boxes. If TW41 is predicted to be in a box at the same time that the box's aircraft is expected to be there, a conflict is declared.

The lower portion of Fig. 7 shows the corresponding state of the planner's search tree. The planner's initial state is shown at a time prior to TW41's appearance. Thus no problem has been detected yet and the initial state has a low score of zero. The planner goes from that state to a state in which TW41 appears and the conflict with M707 is discovered. The second state, therefore, has a high (i.e., an undesirable) score. The planner then tries to reduce the score. Note that only TW41's conflict with M707 has been detected at this point.

We designed our rule-based system so that the discovery of a conflict leads immediately to an investigation that tries to resolve the conflict.

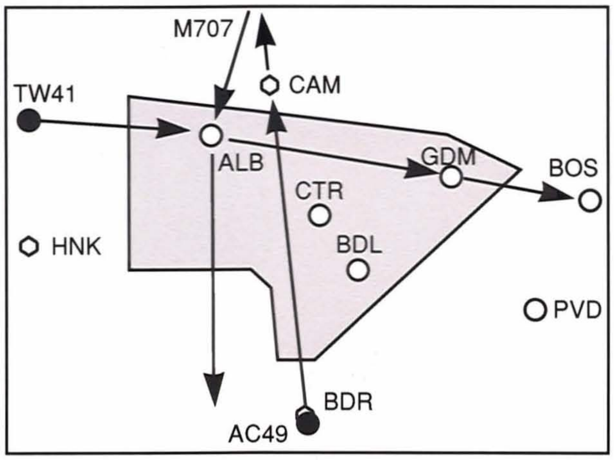


Fig. 6—Initial-conflict situation.

Thus, in the example, the other conflict with AC49 will not be discovered until after the M707 conflict is resolved. A better implementation would be to find all of a given plan's conflicts first, and then prioritize the conflicts so that they can be solved in order of their importance. Alternatively, a sophisticated system might find cases in which several conflicts could be resolved by a single action. These types of advanced algorithms are difficult to implement in a rule-based system like YAPS.

In the example, the critic that detects aircraft-crossing conflicts proposes four possible solutions: TW41 can be restricted to pass either below or above M707, or M707 can be restricted to pass either below or above TW41. For the first case in which TW41 is constrained below M707, Fig. 8 shows the planner's search state and the corresponding altitude plan. The heavy black line below M707 represents the restriction that TW41 must be at or below 33,000 ft over the portion of path that is denoted by the heavy line's length. Upon evaluating this new plan, the system finds a subsequent conflict between TW41 and AC49. Consequently, this plan receives a high score.

The example demonstrates the search limitation of our current system. Because the system searches only one level deep in the search tree, the secondary conflict between TW41 and AC49 could not be solved, for example, by starting TW41's descent earlier so that the aircraft would

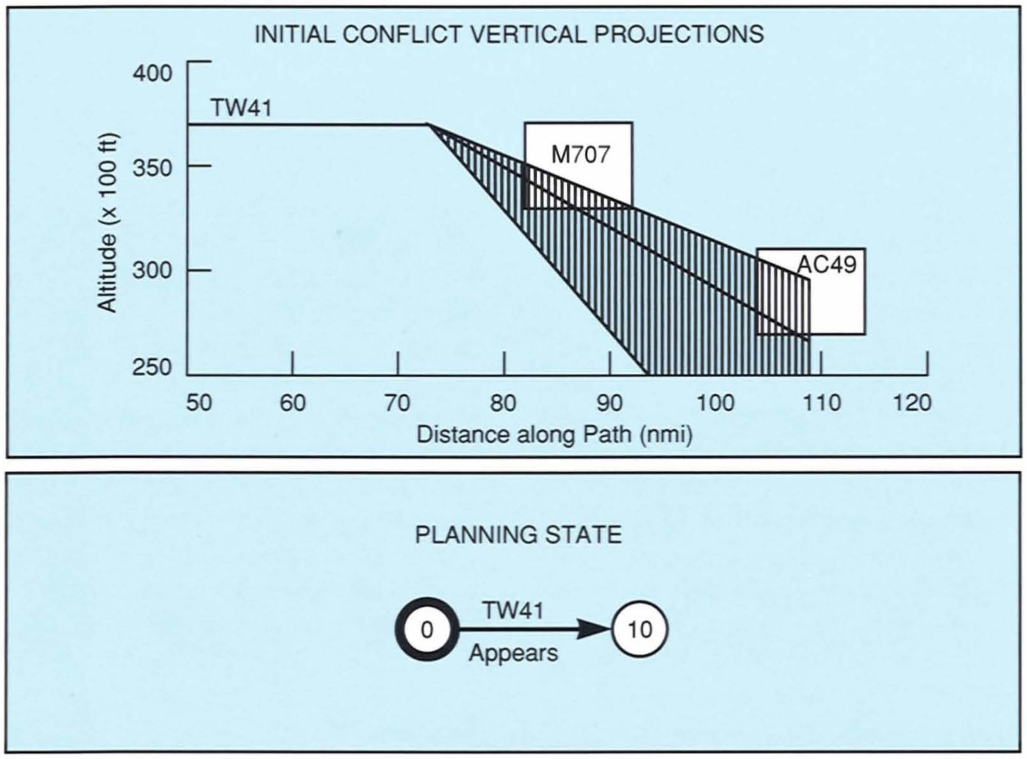


Fig. 7—Initial-conflict vertical projection (top figure) and planning state (bottom figure).

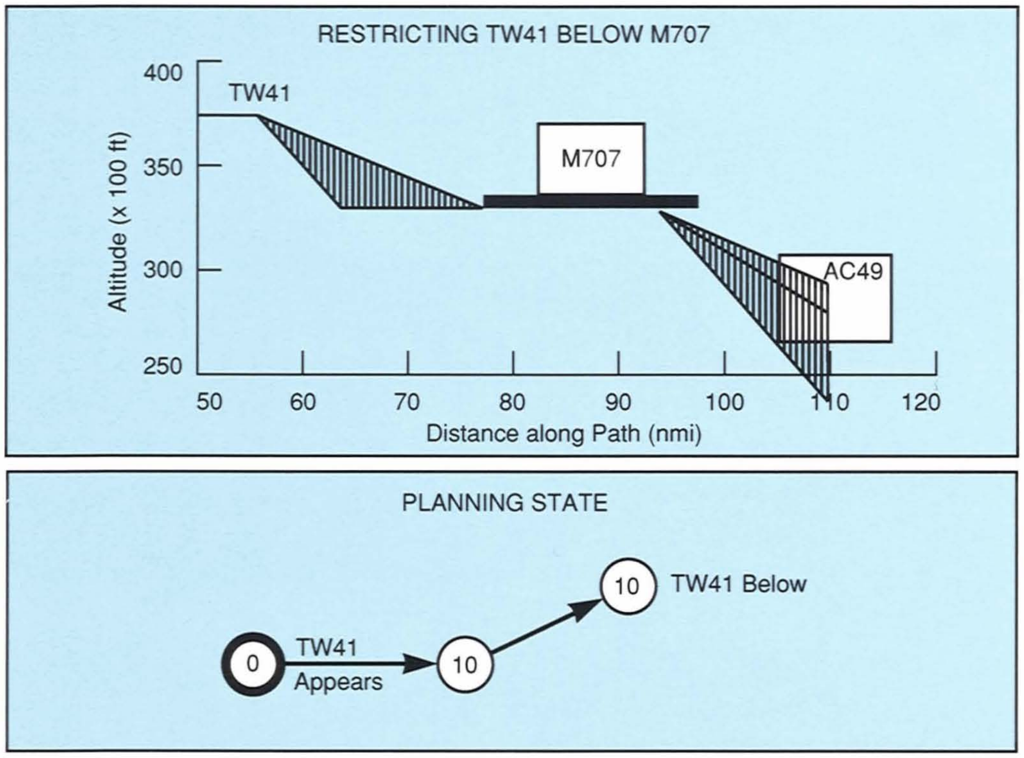


Fig. 8—Examining the consequences of restricting TW41 below M707.

fly below both M707 and AC49. Instead, the system altogether rejected the alternative of restricting TW41 below M707.

Figure 9 shows the remainder of the search tree for the first conflict and the selected altitude profile. The selected plan holds TW41 high until the aircraft has passed over M707. The system has determined that TW41 still has enough time to descend into Boston, albeit with less margin than in the original plan. The conflict-detection critic ignores the corner where TW41's maximum descent rate might cause it to enter AC49's protected airspace. As currently implemented, the critic is more concerned with the nominal projection than with the maximum and minimum projections. However, a more advanced planner might treat this situation as a secondary conflict that needs to be resolved by restricting TW41's altitude to be above that of AC49.

In the example, the other two options (M707 restricted above TW41 and M707 restricted below TW41) for resolving the original problem receive scores equal to the option of restricting TW41 above M707. The planner randomly selects one solution. However, forcing this choice would be better because, in general, it is preferable not to disturb an overflight. This additional criterion could be implemented by the creation of a critic that penalizes those plans which interfere with overflights.

Now that the planner has solved the first conflict, another aircraft appears (Fig. 10). AC97 has the same route as AC49 but the aircraft has filed for a slightly higher cruise altitude. Figure 11 shows the resulting altitude plan and planning state. The newly proposed plan for TW41 now conflicts with the initial plan for AC97. The planner tries to resolve this conflict by using the same four options as before: TW41 above, TW41 below, AC97 above, and AC97 below. The basis for these options is the current plan, which is the result of resolving the first conflict described above, and the addition of the plan for AC97.

If TW41 is held above AC97, TW41 will not have enough time to descend to 8,000 ft for entry into the Boston terminal area. Thus that particular alternative receives a high score. When the planner evaluates the effect of trying to restrict TW41 below AC97, the state-prediction

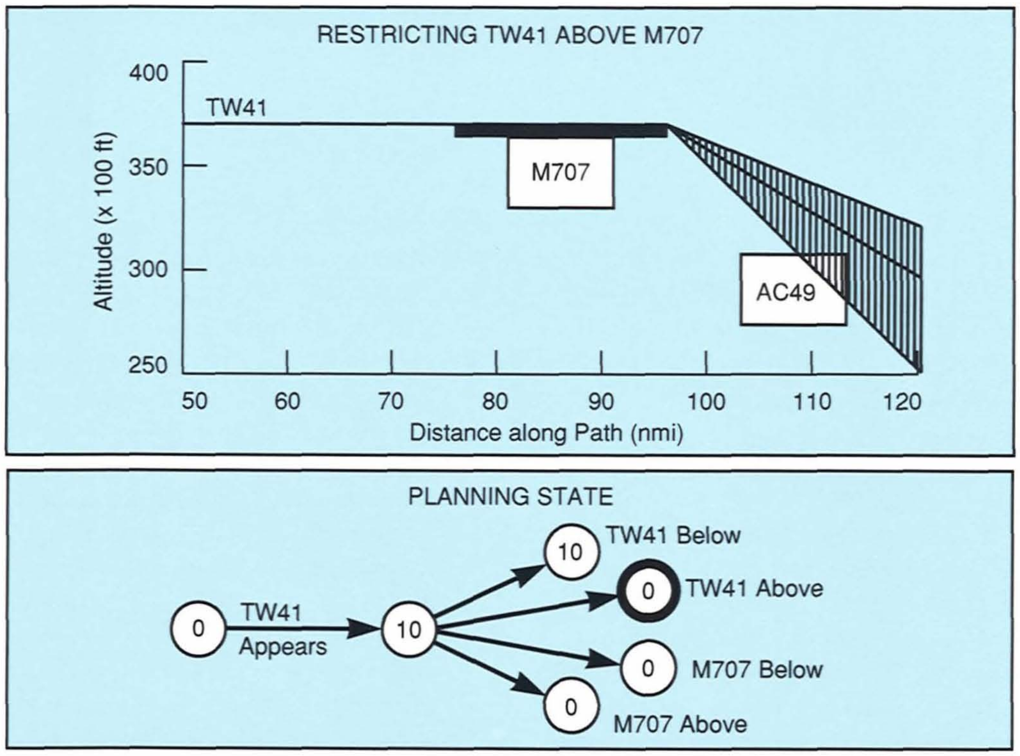


Fig. 9—Resolution of the initial conflict.

routines determine that because TW41's descent rate is limited, the aircraft cannot go both above M707 and below AC97. Therefore, the conflict is not resolved and this option also receives a high score.

Figure 12 shows the final planning state and the selected altitude plan. Note that the score due to TW41's inability to reach the proper altitude for terminal-area entry is not as bad as the score for an option that contains a conflict. The planner chooses the plan that restricts AC97 below TW41 until AC97 is past the point of conflict, and then clears AC97 to its desired cruise altitude. This alternative has the same score as the alternative of restricting AC97 above TW41 and, once again, the final solution is arbitrarily chosen. As in the previous case, there are reasons that the chosen alternative is superior; additional critics could be written to represent these criteria in order to force the superior solution.

Figures 4 and 5 show the development system's displays approximately 2 min after the second conflict is resolved. Note that AC97 has been cleared to 29,000 ft rather than its requested cruise altitude of 33,000 ft.

Conclusion

The system described in this article was capable of successfully performing basic ATC and conflict-resolution tasks for the easiest two

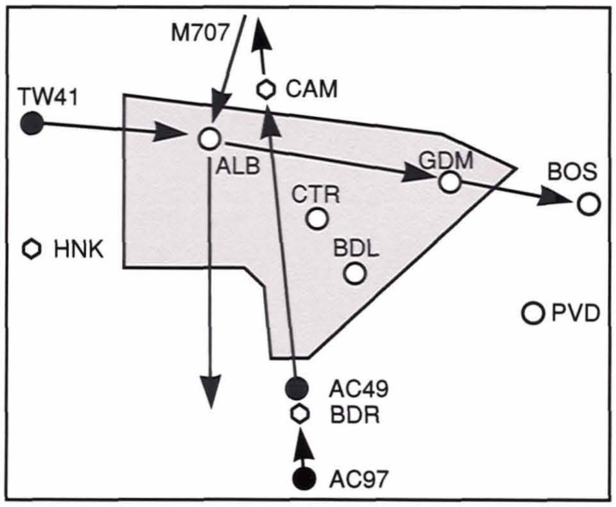


Fig. 10—Second-conflict situation.

of the Boston ARTCC training scenarios. The system, which operated in real time for the first scenario, used a Symbolics 3670 computer that also ran the simulation and display programs. The system's limited knowledge base restricted the available number of problem-resolution actions and kept the system's behavior from being robust, or truly expert. However, for these two scenarios and some smaller scenarios derived from them, the system maintained proper aircraft separation, cleared aircraft to their proper altitudes, and coordinated handoffs with adjacent sectors.

The major conclusions of this study are the following:

- The system's overall planning architecture appears capable of performing automated ATC. The architecture satisfies the basic functional requirement of being able to incorporate ATC knowledge in a modular fashion. This modularity allows the system to make trade-offs among the possibly conflicting recommendations of the knowledge sources. The architecture also enables the search for good (but not necessarily optimal) solutions in a reasonable amount of time.
- Further work is needed to expand the system's search process and evaluation mechanism. In particular, multilevel search and a well-defined method for developing the evaluation weights appear necessary. The similarity of the basic planning mechanisms of our system to those used in automated chess programs implies that useful guidance might be obtained from recent advances in that area. The structural similarity of the evaluation mechanism to neural networks suggests an adaptive learning approach to the weight-setting process.
- An extensive amount of work is needed, both to capture the necessary ATC knowledge in the plan critics, and to test the system on a large enough sample of problems to ensure error-free performance. One way of accomplishing the above while obtaining some benefits during the devel-

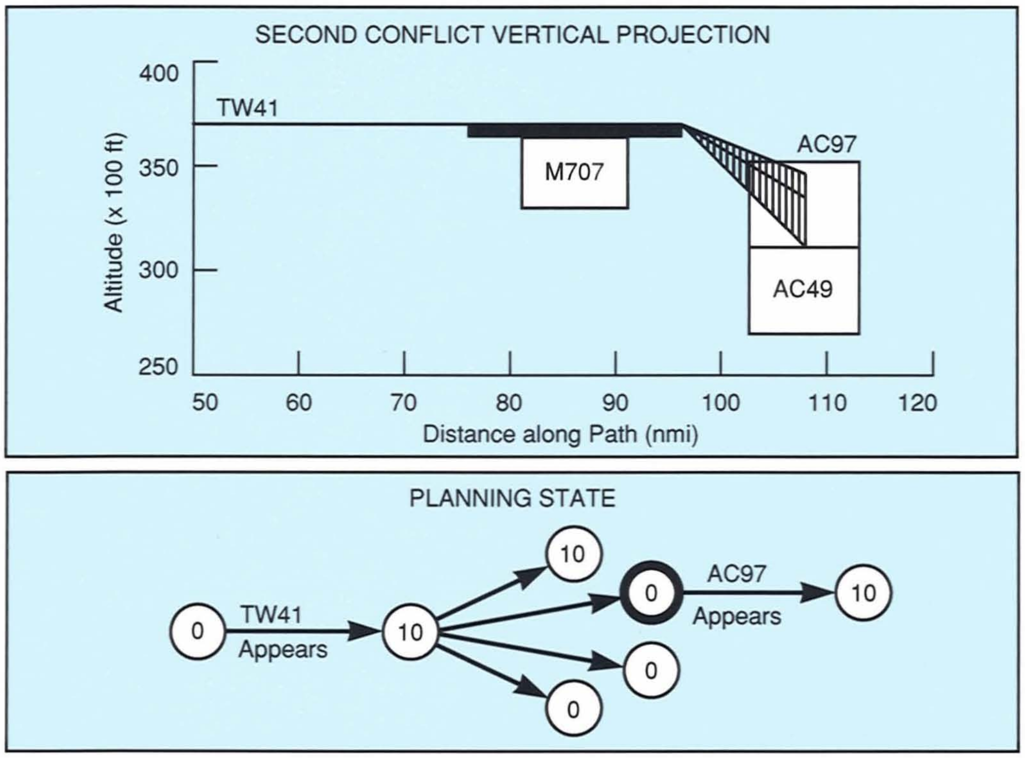


Fig. 11—Second-conflict vertical projection (top figure) and planning state (bottom figure).

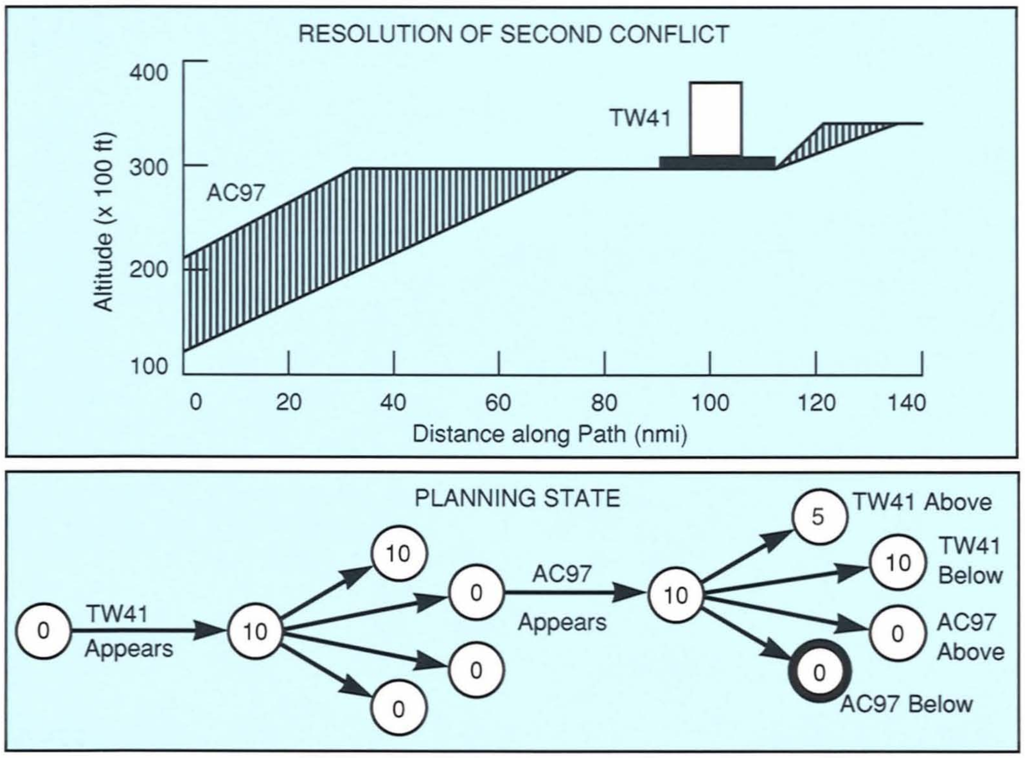


Fig. 12—Resolution of the second conflict.

opment period would be to use the system as a training aid. By doing so, we could use the large number of already developed training scenarios.

- Simple forward-chaining rule-based mechanisms are not suitable for systems with both nonstatic data and the need to represent mutually exclusive alternatives while searching for solutions. In such systems, two types of mechanisms, one for deleting derived facts that have become invalid and the other for keeping alternative plans separate, must be explicitly programmed. This requirement makes the rules more complex. Assumption-based truth-maintenance systems [14] appear to provide these types of mechanisms as part of their basic design while, at the same time, allowing for the modular representation of knowledge through the use of rules. Such systems need to be tested in an ATC context to determine if their expected benefits can be achieved while real-time performance is maintained.
- When a problem involves large numbers of similar facts, the standard ways of implementing rule-based systems are subject to combinatorial explosions of rule-and-fact matches in the system's working memory. The repercussions of such explosions can be ameliorated with coding tricks, but this solution places a great burden on programmers and makes the coding less comprehensible. The use of coding tricks also leads to system behavior in which a small change in a rule or the addition of a new rule greatly affects performance. This effect brings into question the suitability of a rule-based approach for situations in which real-time performance is required. In rule-based approaches, there appears to be a trade-off between the potential modularity and compactness of a system, and the system's speed of execution. However, this trade-off may not be fundamental, and improved automated optimization

techniques may result in a system that is modular, compact, and capable of real-time operation.

Acknowledgments

The author thanks Don Underwood, who did much of the programming to develop the system described in this article; and Steve Olson, who did most of the YAPS start-up and modification work. The author thanks Steve Campbell, who also was the source of many useful ideas and comments.

This work was sponsored by the Federal Aviation Administration.

References

1. Special issue on the FAA's Advanced Automation System (AAS), *Computer* **20** (Feb. 1987).
2. A.H. Gisch and B.C. Zimmerman, "AERA—Toward Greater En Route Air Traffic Control Automation," *The MITRE Corporation Report MP-86W29* (McLean, VA, Oct. 1986).
3. J.A. Kingsbury, "An AERA for This Century," *The MITRE Corporation Report MP-86W28* (McLean, VA, Oct. 1986).
4. D.A. Spencer, J.W. Andrews, and J.D. Welch, "An Experimental Examination of the Benefits of Improved Terminal Air Traffic Control Planning," *Lincoln Laboratory Journal* **2,** 527 (1989).
5. G.A. Gilbert, *Air Traffic Control: The Uncrowded Sky* (Smithsonian Institution Press, Washington, DC, 1973).
6. *Air Traffic Control, Order 7110.65,* U.S. Department of Transportation, Federal Aviation Administration (Washington, DC, updated periodically).
7. H. Ammerman, C. Fligg Jr., W. Pieser, G. Jones, K. Tischer, and G. Kloster, "En Route/Terminal ATC Operations Concept," *Computer Technology Associates, Inc., Report* (Englewood, CO, 28 Oct. 1983) DOT/FAA/AP-83-16.
8. *Airman's Information Manual,* U.S. Department of Transportation, Federal Aviation Administration (Washington, DC, updated periodically).
9. D. Biggs, *Pressure Cooker* (W.W. Norton, New York, 1979).
10. J.W. Andrews and W.M. Hollister, "Electronic Flight Rules: An Alternative Separation Assurance Concept," *Project Report ATC-93,* Lincoln Laboratory (31 Dec. 1980), FAA-RD-80-2.
11. E.M. Allen, "YAPS: Yet Another Production System," Department of Computer Science, University of Maryland, *Report TR-1146* (Feb. 1982).
12. E.M. Allen, "YAPS: A Production Rule System Meets Objects," *Proc. of the Ntl. Conf. on Artificial Intelligence AAAI-83, Washington, DC, 22–26 Aug. 1983,* p. 5.
13. P.H. Winston, *Artificial Intelligence,* 2nd ed. (Addison-Wesley, Reading, MA, 1984).
14. J. de Kleer, "An Assumption-Based TMS," *Artificial Intelligence* **28,** 127 (1986).

DAVID A. SPENCER is a senior staff member of the System Design and Evaluation Group at Lincoln Laboratory. His focus of research has been in computer software systems related to air traffic control. Dave received a B.S.E. in electrical engineering from Princeton University, and an E.E. and S.M. in electrical engineering and computer science from MIT, where he completed his master's thesis at the university's Artificial Intelligence Laboratory.