Planning in the World of the Air Traffic Controller

Robert B. Wesson
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712

Abstract

An enroute air traffic control (ATC) simulation has provided the basis for research into the marriage of discrete simulation and artificial intelligence techniques. A program which simulates, using real world data, the movement of aircraft in an ATC environment forms a robot's world model. Using a production system to respond to events in the simulated world, the robot is able to look ahead and form a plan of instructions which guarantees safe, expedient aircraft transit. A distinction is made between the real world, where pilots can make mistakes, change their minds, etc., and an idealized plan-ahead world which the robot uses; the over-all simulation alternates between updating the real world and planning in the idealized one to investigate the robot's ability to plan in the face of uncertainty.

Introduction

Emphasis within problem solving research has recently been based on real-world problems and environments. Production systems incorporating an extensive base of expert knowledge (MYCIN [1], DENDRAL [2]) have performed well in specialized environments; other systems not usually regarded as problem solvers, such as Wilensky's natural language story understander [3], also show interesting problem solving abilities. Many problems any complex robot must face, however, have yet to be examined in depth. Fikes, Hart, and Nilsson [4] discussed a number of these, including (1) avoiding negative goals as well as achieving positive ones, (2) planning with constraints, and (3) operating in dynamic environments with multiple, independent processes not fully controlled and/or recognized by the robot. Of these three areas, (1) and (2) have been hardly touched, and (3) is the only area to have received significant attention. Hendrix proposed an event-driven process model [5] based on the STRIPS approach of add- and delete-sets. Howe, in his studies of cognitive development, uses techniques taken from simulation languages such as SIMULA and GPSS [6]. Both illustrate these reasonably general theories of problem representation using well-defined but stark subsets of a child's perceptual world. Other robot problem-solving systems have been surveyed by Siklossy[7].

The work reported here is an attempt to address several of these problem areas in a realistic, information-rich subset of the real world. The original project emphasis—the application of well-known problem solving techniques in a complex world—has given way to the more interesting task of creating a program which can, in the same complex world, create a plan based on its expectations, execute that plan, and modify it if unexpected events occur during execution. It must have the ability to recognize and predict the course of on-going processes, only a few of which it may control. It must selectively exercise that control to correct undesirable situations and improve others.

The Air Traffic Controller's World

The world chosen for this is that of a low altitude radar-assisted enroute air traffic controller. When an aircraft flies from one airport to another under instrument flight rules (as all airlines and military aircraft do), it passes through a number of controllers' jurisdictions: from ground control through tower and departure control to a succession of enroute controllers who track the flight on radar in level flight. The enroute controller's task is to keep the aircraft within his sector proceeding along their desired paths (according to individual flight plans filed before departure) subject to numerous constraints. Many of these constraints are generated by governmental rules (for example, two aircraft within five miles of each other must be assigned altitudes at least 1000 feet apart). The controller also has obvious constraints of expediency (an aircraft declaring an emergency must get priority over all other aircraft), physical limitations (aircraft cannot turn or climb instantaneously), and plain common sense (an aircraft must not be assigned an altitude too close to hazardous terrain or above its capabilities). Violation of these constraints causes a conflict which he must resolve.

Standardized methods are used to control the aircraft paths. The controller is in continual radio contact with all the aircraft in his sector. He can control the aircrafts' positions both vertically (climb to____feet, descend to_____feet) and horizontally (turn right/left to____heading, adjust speed to____knots). Pilots are required to obey these commands except under unusual circumstances. Thus, using his knowledge of the

airway structure and the aircraft intentions and capabilities, the controller must decide when to issue commands which "optimize" performance while satisfying all constraints.

This mini-world seems to be a good vehicle for AI investigation. It is highly structured—state changes in the world are well-defined over time and space. Controller activities are limited to the issuance of aircraft commands drawn from a very small standardized set. Optimal performance is simple to judge in terms of aircraft transit times, number and kind of commands issued, and the like. Yet it offers a significantly richer environment than the AI worlds of the past. It is dynamic, so that the modelling of time becomes a paramount issue. It is non-precise—the simple descriptive techniques used for blocks worlds are inadequate because of real world uncertainty about object positions and the effects of actions (commands). The robot (controller program) is neither omniscient nor omnipotent. Expected events may not occur (an aircraft may fly at a different speed, heading, or altitude than expected) while unexpected ones may (new aircraft may arrive in the sector).

Controller Simulation Describe

ATC World Simulation

Since one of the main objectives of this work is the application of problem solving techniques, the simulation must closely duplicate the real world, making as few idealizations as possible. The information normally present on a controller's radar scope had to be stored internally, updated periodically, and presented interactively on a graphics terminal.

The first phase of the project began with the development of such a world model. Information about airport locations, radio beacon locations, airway structure (including minimum altitudes), control jurisdictions, and the like was taken from current aeronautical charts, reduced to typical x-y coordinates, and stored on data files. This data structure is designed to be easily modifiable for modular use by any controller station world-wide and for ease of maintenance.

A flightplan list "drives" the simulation. It contains everything an updating routine needs to determine an aircraft's movement as the simulation progresses. The most important dynamic data structure is the world, an instantaneous snapshot including time, all the aircraft active with their assigned and current speeds, altitudes, and headings, and any commands and/or events currently occurring. It also includes a link to past worlds recorded at strategic points, so that updating can be complemented by backdating to any time. These structures are declarative and serve as data for procedures which perform the mechanics of the simulation.

The Controller as a Problem Solver

A human can perform actions in order to change his future in a desirable way. Many actions are reactions to current situations, manifest in the form of learned responses to specific stored patterns. These can become highly complex and can result in high levels of performance. Habits and memorized behavior (such as playing the piano) are examples of this. Not normally regarded as problem solving, it nevertheless is responsible for much of human expertise. It is implausible, however, that high-level cognitive tasks can be accomplished solely by reacting to current situations. By augmenting this production system approach with a look-ahead capability, a clearer picture of time-oriented problem solving behavior emerges. It can be described as follows: using a model of the world as it is currently perceived and learned rules for describing the changes which may occur in that world, a human constructs a continuous simulation of the world. Within this simulation, he is able to note events which will occur without his intervention. He can hypothetically perform actions, observing how these actions affect the original event, whether new events are created, and how the action fits into his global strategy. A person's learning, in part, consists of continually reducing the number of surprises he faces day to day by adding more and more rules to this simulation model of the world. These informal rules enable him to correctly predict the natural world and how his, and others', actions perturb it.

In the world of air traffic, a controller quickly learns to sense aircraft rates of speed, altitude change, and heading change. This knowledge, procedural in nature, enables him to speed up time in his mind, looking ahead to predict potential collisions and events which require his intervention. In this future world, he can try out his various options and select the commands most likely to produce the best results. It is this behavior which we attempt to model.

Problem solving with Simulation The controller clearly has an intern
model which he uses to try out his proposed solutions. This model, derived from the real world but quite separate from it, gives rise to the dual simulation integral to this controller modelling program: A simulation of the real world (henceforth, the real world or RW) allows the programmer to construct realistic scenarios with which the controller program, or problem solver, interacts; the problem solver itself has its own independent and somewhat idealized simulation of the world (the problem solver's world or PW) based on a snapshot of the RW, the aircraft flight plans, and its own expectations of their behavior. It uses this idealized world to look ahead in time to spot impending trouble areas and investigate the results of its actions.

The result of this look-ahead is a plan of actions it proposes for the RW—commands to be given paired with issuance times. A monitor in the RW issues these commands at the appropriate times and checks for circumstances which would render the plan invalid. Note that the forward time threshold of the PW must always be significantly beyond the current RW time, because unforeseen events may require alteration of the plan. Suppose, for example, that the planner looks ahead for 20 minutes. Suppose, further, that a collision is imminent at minute 21 which requires commands to be given at minutes 14 and 17 to avert it. Clearly the plan is not a 20-minute plan at all but is valid for perhaps 10 minutes of real time at most. (These figures of 20 minutes for the look-ahead and 10 minutes for the expectation were chosen empirically for the serial plan-ahead..execute.. sequence simulation; an actual implementation would be a time-shared planner/RW monitor program with the planner almost continuously active.) This procedure of alternately planning ahead a lot, then updating the RW a little, leaving a large portion of the plan un-executed, can be best expresssed by the following PASCAL-like program:

```
SIMULATOR:
begin
  input initial RW description and flightplans;
  initialize time, aircraft positions;
  repeat
    PW <— RW;
    PLAN-AHEAD using PW for next 20 minutes,
               generating the plan of
               command-time pairs;
    repeat
      UPDATE world one time step;
      ISSUE commands from plan for this time;
    until  have updated 10 minutes (normal
           termination) or something unexpected
           occurs (an aircraft rejects a command;
           a new aircraft enters the sector; etc.)
           or all aircraft terminated
  until all aircraft terminated
end.
```

Emphasis here is on creating a realistic simulation with which the planner can work. A typical mix of controllable and uncontrollable, fast and slow, high and low air traffic is used for all exercises. Aircraft movement in the RW is randomized by the updating routine to reflect the output characteristics of a digitized radar system. This structure is a prototype which can allow further development for actual ATC implementation by extending its knowledge base along established lines and replacing the RW simulator above with a real-time monitor.

Looking Ahead

Within this structure is the capability for fully exercising the planner, where the "intelligence" of the system lies. This routine's task is to create a minimal sequence of aircraft commands which guarantees a specified time period free from conflicts based on the current world state and known aircraft intentions. It uses a look-ahead technique which is again incremental, updating the world step-by-step through time. Actions taken in the planner are caused by the activation of one or more event-response pairs. After every update, pre-defined events are checked for, such as a violation of the 1000 feet or 5 mile separation rule, or an aircraft descending too low. If one of these events occurs, it is placed on an event list with parameters describing the specifics of the event. After all the events have been noted for a specific world state, responses are made which ultimately generate command actions to be inserted into the plan. This planning behavior is best described as follows:

```
PLAN-AHEAD:
repeat
  CHECK-EVENTS and note occurrences on eventlist;
  if eventlist is non-null then
  begin
    pre-process events as required;
    TRY-OUT all responses to highest-priority
      event using lookahead with evaluation
      and select "best";
    if time of response < current time then
      BACKDATE world to time of response;
    post-process events as required
  end;
  ISSUE commands as required from plan;
  UPDATE world one time step
until looked ahead 20 min.
```

This scheme is similar to a production system in that actions are generated by the event-response pairs: if an aircraft approaching an airport is higher than the designated approach altitude, then a descend command will be generated. New knowledge can be easily added by inserting new event-response pairs. The traditional production approach of implementing knowledge in manageable kernels has been used quite successfully.

However, the planner departs from the usual production system technique in a number of significant areas. Where most production systems have employed a pattern-matcher operating with static, descriptive condition-action rules, the event-response rules in this system are implemented procedurally. While this technique has its drawbacks, such as requiring re-compilation whenever new rules are added, its advantages are numerous. Note that all the events are checked for with every update. Compiled event descriptions allow event recognition to be much faster. Both event and response procedures become more powerful with the full generality of a programming language behind them and thus fewer are needed. Highly structured data and procedure construction techniques are followed, so that the resulting event-response rules are quite transparent and easy to construct.

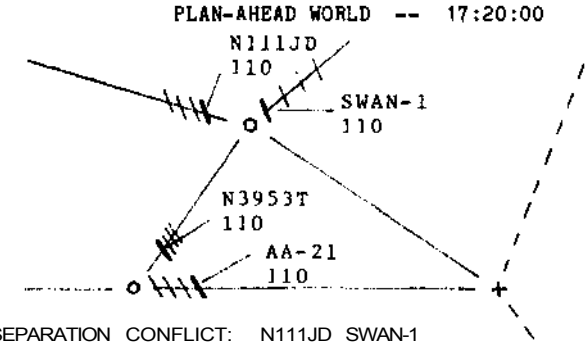
The main advantage of these procedural rules, however, comes in the way responses are handled. Many production systems disallow more than a single response for each event pattern; if allowed, multiple responses generally result in blind depth-first searches. In the ATC planner, multiple responses are not only allowed but encouraged. These responses may be proposed for any world state—past, present, or future. Some events may be conflicts in the present, requiring back-up and responses in the past for resolution; other events may be expected future conflicts, requiring responses in the present or near future. A response selection procedure "tries out" each possible specific response by performing that response on a world copy of the appropriate time and updating in the usual manner. This updating continues until another event occurs, forming a "level" or branch in the developing search tree. Events occurring this way are resolved by recursive calls to the response selection procedure until a pre-set level has been reached, whereupon an evaluation is performed. This evaluation is based upon the current world state (time, distance aircraft are from their goals, number and severity of current conflicts, etc.) and the path to it (number and type of commands issued in the past). Note that levels are separated by time, that states of a given level can have widely different times and that in fact states with a higher level number do not necessarily have later times. For example, a response at level 2 might require a back-up to a time before level 0 and cause new events to occur immediately. This would result in a level 3 time before the level 0 time and illustrates an attempted solution destroying a previously successful part of the plan.

This approach to response selection from a number of candidates by using a "mini-look-ahead" works fairly well. It seems to effectively capture the human notion of immediate or local knowledge so effective in pruning search trees. A controller can immediately choose to vector an aircraft to the left when the right side of its flight path is littered with traffic; this planner can make the same decision for the same reasons. Even in complex situations, the response procedure almost always picks the best response, resulting in very little wasted search and very efficient plan generation.

Aside from the knowledge within the productions, the planner also uses directly coded heuristic "common sense". For example, geographical locations of the events enable the planner to divide the world into subsets of aircraft which are then examined separately by the response procedures. This is something humans do only in a limited manner: directed attention is definitely necesssary, but our ability to remember complex solutions from one area, compute one for another, and merge the two solutions in time is dismal. The planner also has rules for preventing circular solution paths or deadlocks. All of this knowledge, in addition to event priorities and bookkeeping tasks, is implemented as event pre- and post-processing in the planner.

A Simple Example

The RW simulation and plan execution is straight-forward enough, but plan creation might be better understood by examining a simplified excerpt from a typical problem the program can solve. Suppose, while looking ahead within the planner, the situation depicted below occurs:



SEPARATION CONFLICT: N111JD SWAN-1
SEPARATION CONFLICT: N3953T AA-21
AA-21 TOO HIGH FOR APPROACH HAND-OFF
EXPECTED AT 17:30

The notation is that used in air traffic control displays. The hexagons and plus depict radio fixes; the solid lines connecting them are airways; the dotted line separates approach control's airspace. N111JD is a Lear Jet heading southeast at about 400 knots; SWAN-1 is a military jet heading southwest then west along the airways at 600 knots; N3953T is a single-engine Piper heading the same direction at 200 knots; and AA-21 is an American Airlines 707 heading east at 400 knots. All aircraft are at 11000 feet. The real world time is 17:10; the plan-ahead world time is 17:20 and two conflicts are occurring while a third is expected to occur at 17:30. The event- response pairs responsible for noting/solving conflicts in this scenario are informally stated as follows:

```
event 1:
  forall aircraft pairs i and j:
    if distance(i,j) < 5 miles and
      abs(alt[i]-alt[j]) < 1000
    then add "separation conflict: i and j"
      to eventlist

responses:
  for both i and j:
  1.  try climbing 1000 feet
  2.  try descending 1000 feet
  3.  try turning right ("vectoring" right)
  4.  try vectoring left
  5.  try slowing down
  6.  try holding (have aircraft fly in a circle)
```

event 2
  forall aircraft i:
    if next fix is approach control
    and assigned altitude > 6000 feet
    then add "i too high for approach hand-off"
      to eventlist

response:
  1. descend aircraft i to 6000 feet or below

event 3
  forall aircraft i:
    if distanced, flightplan track) > 4 miles
    then add "i off track" to eventlist

responses:
  1. try "turn i back toward track"
  2. il distanced,next radio fix) < 40 miles
     then try "direct i to next radio fix"
  3. try doing nothing

Notice that the first and third events have multiple possible responses while the second has only one. As each response is tried, its specific parameters (time to give command, specific altitude, heading, speed, etc.) are computed from the nature of the event. A solution to this scenario proceeds as follows:

Event pre-processing would separate the eventlist into three subsets, each containing a single event, based on geographical separation. It would then give the RESPOND procedure world copies with only the geographically relevant aircraft activated. The first event would cause the twelve possible responses to a separation conflict to be tried out. A search tree of two levels (generally) would be generated and the terminal nodes evaluated. The response chosen would be "descend N111JD to 10000 feet at 17:19" because it prevents the conflict at 17:20 and the evaluation parameters, in simulating controllers' preferences, favor descend commands given to slower aircraft. The second event would then be processed similarly, but the result would be "turn N3953T right 45 degrees at 17:17" because the program would correctly see during look-ahead that such a radar vector would allow N3953T to cut across the dog-leg formed by the radio beacon and proceed along a shorter path toward its destination. The final event would cause no look-ahead, merely a "descend AA-21 to 6000 feet at 17:25" command to be inserted into the plan with the other two. The search tree generated while processing the second event is partially presented in Figure 1. Vertical lines indicate normal updating without conflicts. Horizontal lines indicate alternate worlds with the listed commands issued. Circles indicate worlds where the numbered conflicts occurred and backdating was necessary, or where evaluations took place. The diagram indicates normal updating until 17:20, when a type 1 (separation) conflict occurred. The best computed solution path is darkened. Only the initial command of the path is recorded in the plan.

The total plan-ahead world would then be back-dated to 17:17 (the time of the earliest command) and then updated normally, with these commands being given at 17:17, 17:19, and 17:25. With every update after 17:26, the event "N3953T off track" would trigger an attempt to turn it back to the airway, but SWAN-1 will have caught up with N3953T by then and prevent that choice of action. Only after SWAN-1 has safely passed will the command "turn N3953T 45 degrees left; intercept the airway and proceed on course" be inserted into the plan. At 17:23, when N111JD has the approach control radio fix as its next one, the event "N111JD too high for approach hand-off—expected at 17:30" would insert "descend N111JD to 6000 at 17:25". Continuing, at 17:25 both N111JD and AA-21 would be given descents to 6000, only to conflict at 17:29. The best resolution, "descend AA-21 to 5000 feet at 17:24", would be added to the plan and the "17:25 descend AA-21 to 6000" command deleted by event post-processing. With this final addition, updating proceeds past 17:30 and PLAN-AHEAD returns the following plan to the real world:

17:17	turn N3953T right 45 degrees
17:19	descend N111JD to 10000
17:24	descend AA-21 to 5000
17:25	descend N111JD to 6000
17:28	turn N3953T left 45 degrees to intercept airway

The plan can be improved by descending N111JD to 6000 at 17:19, thus saving a command, but this behavior accurately reflects the controller's tendency to stair-step aircraft down, solving conflicts on a local basis.

Implementation

This problem-solving simulation has been working as described at the University of Texas for about a year now. The controller's display was simulated on an IMLAC PDS-1d graphics terminal using locally-written software; the simulation and planner both run on a DEC-10 interfaced with the IMLAC. The language chosen for implementation, PASCAL, seems the best compromise between concerns of efficiency and generality. With all the knowledge discussed so far implemented, the program runs in 40K words of core. Typical simulations of up to twenty aircraft, with up to ten active at any one time, involving fifteen to twenty commands required over about one hour of simulated time, can be run in 30 CPU seconds. These situations correspond roughly to a 100$ sector load factor, as defined by ATC centers for training purposes. Contrived situations involving multi-aircraft separation conflicts (all aircraft at the same plaoe at the same time at the same altitude) take considerably longer: a five aircraft conflict resolution takes two minutes; a seven aircraft one takes three.

This performance degradation is caused by the sharp increase in the number of possible responses examined: a separation conflict has the most general response procedure with six potential responses for each aircraft involved. As more experience is gained with the system, perhaps this single event can be broken down into more specific ' events requiring less search to produoe responses. In any case, separation conflicts involving over two or three aircraft are extremely rare and are difficult for humans as well.

In addition to these "absolute" performance tests, the program has also been tested against an expert human controller. Data was taken from a near-by ATC center during controller training sessions. Photographs of the actual radar scope, recordings of pilot-controller communications, and flightplan strips enabled a complete re-construction of several hour-long secnarios and afforded the means to directly compare the program's performance to that of a highly-trained controller. The program was able to handle virtually all of the conflict situations recorded, although multiple conflicts hard for the human proved to require extensive back-up and execution time in the program as well. Using fairly common criteria such as number of commands required, aircraft transit times, and the like, the program was able to do as well as or better than the controller on all of the scenarios. In one particularly tough scenario, the program issued 15 commands to the controller's 40, and in 16 out of 19 individual aircraft cases equalled or exceeded controller performance.

Program performance so far indicates that practical implementation could be considered. The magnitude of the problem is tremendous indeed, yet the marked success of this one-man prototype system with its small size and run times indicates that the essentials are present upon which to expand. If a commitment were made to deploy such a system, its input would be currently available digitized radar data, and its output (the commands) could be relayed to the pilots by a human controller, a CRT or data link installed in the aircraft, or a computer-generated voice. The program would require an ability to recognize problems it cannot solve in a reasonable time and ask the-human controller for help; its knowledge of aircraft and environment specifics would have to be increased substantially; it would have to be the epitome of reliability. These problems appear to be manageable. The most difficult problems any implementor would face will probably be in the realm of engineering and management: interfacing the planning computers with the flight plan processing and data reduction ones, maintaining the data bases required, designing a fail-safe man-machine interface and back-up system. In addition, the psychological problems of pilot, controller, and public acceptance appear to be the most potent of all.

Conclusions and Directions

The air traffic controller system described here is another example of a problem solver which performs with a high level of competence in a very limited environment. Its actual capabilities have been difficult to judge to date; finding scenarios which truly "challenge" it is just one of the problems here. There should be benefits from this research, however, extending beyond the world of air traffic control. The concepts this system embodies are general enough that their applicability in more traditional robot worlds should be explored. Constraint satisfaction is a significant part of the planner's task. It differs from more traditional work in this area [8] in that both the degree of freedom and the desire to minimize the number of "steps" in the solution path are much greater.

The success of this planner concerning time-dependent problems is notable as well. The idea of making decisions based on local knowledge, then simulating forward in time to observe how the results conform to a global strategy is quite intuitive. Before now, the problem of time has compounded the more general frame problem of robot problem solvers [9]; this discrete simulation approach has the decided advantage of simplicity and appears to be psychologically reasonable as well. It could be applied to any world requiring planning and execution under uncertainty and complexity.

Acknowlegments

The work reported here is part of the author's ongoing research towards the PhD degree. Discussions with Daniel Chester, Robert F. Simmons, and Laurent Siklossy were particularly helpful in establishing the direction of this research and this paper.

Bibliography

1. Shortliffe, E. H. "MYCIN: A Rule-Based Computer Program for Advising Physicians Regarding Antimicrobial Therapy Selection". Stanford AI Lab Memo AIM-251, 1974.

2. Buchanan, Sutherland, and Feigenbaum. "HEURISTIC DENDRAL: A program for Generating Explanatory Hypotheses in Organic Chemistry". Machine Intelligence it, American Elsevier, New York, 1969.

3. Wilensky, R. "Using Plans to Understand Natural Language". Proceedings ACM '76, 46-50.

4. Fikes, R., Hart, P. and Nilsson, N. "Some New Directions in Robot Problem Solving". Machine Intelligence I, American Elsevier, New York, 1972.

5. Hendrix, G. G. "Modelling Simultaneous Actions and Continuous Processes". Artificial Intelligence 4, 145-180, 1973-

6. Howe, J. A. M. and Young, R. M. Progress in Cognitive Development. D. A. I. Research Report No. 17, University of Edinburgh, 1976.

7. Siklossy, L. "Some Issues in Problem-Solving in Modelled Worlds". Proceedings Third International Congress of Cybernetics and Systems, Bucharest, Romania, 1977.

8. Fikes, R. E. "REF-ARF: A System for Solving Problems Stated as Procedures". Artificial Intelligence 1/1-2, 27-121, 1970.

9. Raphael, B. "The Frame Problem in Problem-Solving Systems". In Findler, N. and Meltzer, B. (Eds.) Artificial Intelligence and. Heuristic Programming, American Elsevier, New York, 1971.

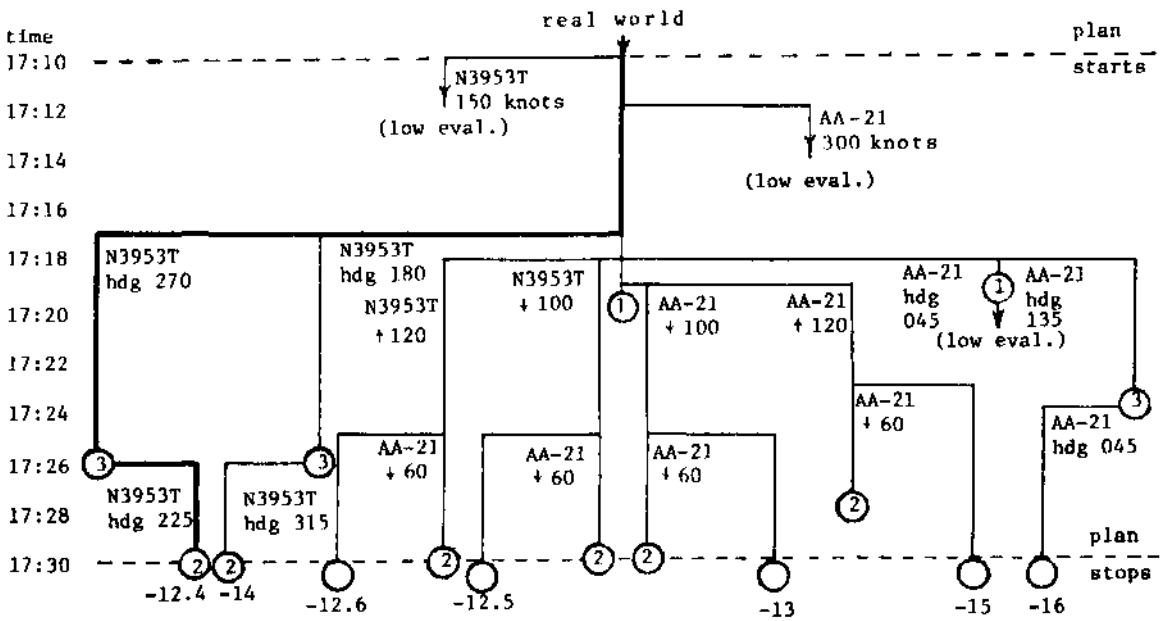


Figure 1