NASA Technical Memorandum 100062

A Traveling-Salesman-Based Approach to Aircraft Scheduling in the Terminal Area

Robert A. Luenberger

March 1988

(NASA-TM-100062) A TRAVELING-SALESMAN-BASED APPROACH TO AIRCRAFT SCHEDULING IN THE TERMINAL AREA (NASA) 24 p CSCL 01C

N88-19424

Unclas
G3/03 0131797

National Aeronautics and Space Administration

NASA Technical Memorandum 100062

A Traveling-Salesman-Based Approach to Aircraft Scheduling in the Terminal Area

Robert A. Luenberger, Ames Research Center, Moffett Field, California

March 1988

NASA
National Aeronautics and Space Administration

Ames Research Center
Moffett Field, California 94035

SUMMARY

This paper presents an efficient algorithm, based on a well-known algorithm for the traveling salesman problem, for scheduling aircraft arrivals into major terminal areas. The algorithm permits, but strictly limits, reassigning an aircraft from its initial position in the landing order. This limitation is needed so that no aircraft or aircraft category is unduly penalized. Results indicate, for the mix of arrivals investigated, a potential increase in capacity in the 3-5% range. Furthermore, it is shown that the computation time for the algorithm grows only linearly with problem size.

INTRODUCTION

Airport capacity is not keeping pace with rapidly growing airline traffic, particularly at major terminal areas. Current air traffic control (ATC) procedures try to sequence arrivals in first-come-first-serve (FCFS) order. One potential means of achieving capacity gains is to modify the FCFS sequence, since the minimum separation between two aircraft is not a constant, but rather a function of the weight categories of the aircraft pair. From a scheduling standpoint, the ideal situation would be to bunch all aircraft of the same type; however, this is not realizable in actual controller operations. Hence the scheduling problem discussed here is concerned with only a limited deviation from the FCFS order.

This paper presents an efficient algorithm for aircraft scheduling, based on a well-known algorithm for the traveling salesman problem. The algorithm permits, but strictly limits, reassigning an aircraft from its FCFS position. The algorithm has been implemented on a VAX computer for use in simulation studies and has consistently produced the best-known solution to a variety of test cases in a small amount of time.

PROBLEM STATEMENT

In 1976, Roger Dear introduced the concept of Constrained Position Shifting (CPS) wherein aircraft must be scheduled within maximum position shift (MPS) of their original FCFS order (ref. 1). The point of this restriction is to ensure that the new schedule is implementable. In general, it is not feasible to reassign an aircraft to an arbitrary landing time far from its initially scheduled time.

The FAA has created specific regulations regarding the separation distance of aircraft based on their weight category. The corresponding time-separation matrix, T, used in this study is given in table 1.

TABLE 1.- MINIMUM TIME-SEPARATION MATRIX (sec)

		Second to land		
		Small	Large	Heavy
First	Small	110	78	78
to	Large	160	78	78
land	Heavy	220	125	104

Definition: The aircraft scheduling problem is,

Given an initial ordering of N aircraft $A_1, A_2, \ldots, A_N$, with corresponding weight categories $W(A_1), W(A_2), \ldots, W(A_N)$, following approach routes $R(A_1), R(A_2), \ldots, R(A_N)$, and the time separation matrix T, find a procedure which will produce a one-to-one mapping function, S, such that

$$S(A_1) = A_1 \tag{1a}$$

$$|[A_i - S(A_i)]| <= MPS \qquad \text{all } i \tag{1b}$$

$$\text{If } R(A_j) = R(A_k) \text{ and } j > k \quad \text{then } S(A_j) > S(A_k) \tag{1c}$$

where the objective is to minimize

$$\sum T\{S[W(A_i)], S[W(A_{i+1})]\} \quad \text{for } i = 1, N - 1 \tag{2}$$

In other words, the new schedule must satisfy:

1. The first aircraft must be scheduled first in the revised schedule.

2. All aircraft must be scheduled within MPS slots of their original FCFS order.

3. Planes following the same incoming route may not pass one another.

where the objective is to minimize the time at which the *N*th aircraft is landed.

The solution is then the vector $S(A_i)$, an ordering of the N aircraft. Thus a solution is a list of the aircraft in their new landing order. Further, the solution must be produced in a reasonable amount of time.

This is a combinatorial optimization problem with a large number of constraints. Figure 1 indicates a number of possible solution techniques. The middle

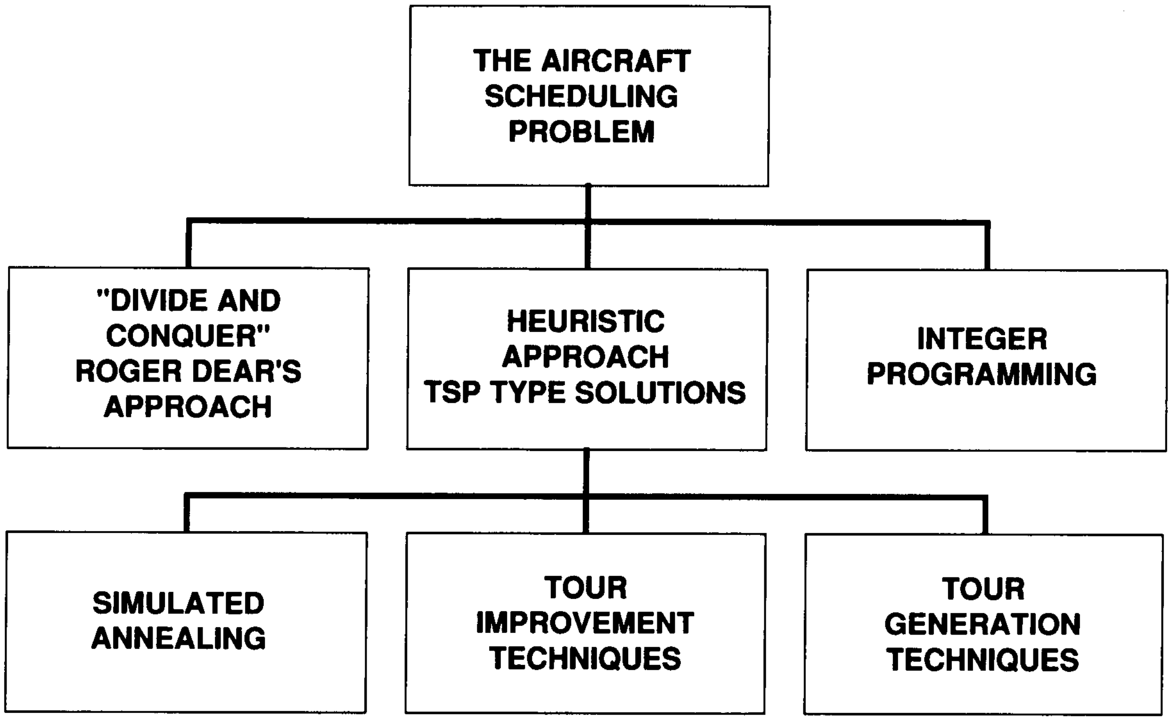
THE AIRCRAFT SCHEDULING PROBLEM
"DIVIDE AND CONQUER" ROGER DEAR'S APPROACH
HEURISTIC APPROACH TSP TYPE SOLUTIONS
INTEGER PROGRAMMING
SIMULATED ANNEALING
TOUR IMPROVEMENT TECHNIQUES
TOUR GENERATION TECHNIQUES

Figure 1.- Different approaches to TSP.

row represents three approaches to the problem and the bottom row indicates three subcategories of the heuristic approach. Roger Dear's method will be described briefly later. The algorithm described by this paper falls under the category of tour improvement techniques.

THE TRAVELING SALESMAN PROBLEM

A close analogy to this scheduling problem is the well-known Traveling Salesman Problem (TSP). The TSP is often stated as, "A salesman is required to visit each of N given cities once and only once, starting from any city and returning to the city of origin. What tour should he choose to minimize the total distance traveled?" In the aircraft scheduling problem, intercity distances are replaced with the time separation matrix shown earlier, and the restriction of finishing the tour where it was started is removed. With these changes we can lean on the wealth of existing work done on the TSP (ref. 2). Though techniques which guarantee optimal solutions do exist (e.g., branch and bound) they generally require prohibitive amounts of computer time and memory for even moderately sized problems. It has been shown, in fact, that the TSP is NP-Complete (i.e., there is no way to ensure finding the optimal solution in polynomial time) (ref. 3). One natural approach, then, is a modification of these techniques which sacrifices some accuracy for a savings in computation time.

Several computational studies have shown that certain heuristic approaches can provide optimal or nearly optimal solutions very quickly. Memory requirements are generally minimal and computational time usually grows only as a polynomial function of problem size (as opposed to exponentially as is the case for exact techniques).

The particular algorithm described here stems largely from the work of Lin and Kernighan who developed what is probably the most effective heuristic for the TSP (refs. 4 and 5). They describe a heuristic procedure for this type of problem as a four-step process:

1. Generate a pseudorandom feasible solution; i.e., a set (actually a vector) T that satisfies some criteria C.

2. Attempt to find an improved feasible solution T' by some transformation of T.

3. If an improved solution is found; e.g., $f(T') < f(T)$, then replace T by T' and repeat from step 2.

4. If no improved solution can be found, T is a locally optimal solution. Repeat from step 1 until computation time runs out, or the answers are satisfactory.

Extensions of their work to the asymmetric case were suggested by Kanellakis and Papadimitriou (ref. 6).

The algorithm presented here uses the aforementioned conditions to produce precisely characterized locally optimal solutions very quickly. Before describing the algorithm, consider the concept of λ-optimality.

Definition: A sequence of N aircraft is said to be λ-optimal (or simply λ-opt) if no one aircraft in the sequence can be moved r slots, $r <= \lambda$, in such a way as to improve the schedule. (This is a modification of the original definition given by Lin (ref. 4).)

Note that all schedules are 0-opt (i.e., $\lambda = 0$) and that a schedule which is λ-opt is also $\lambda' =$ opt for $\lambda' < \lambda$. Setting $\lambda =$ MPS, the algorithm produces a large number of MPS-opt solutions very quickly, while always storing the best solution to date. For the actual implementation of the algorithm, see the code in appendix A.

A solution which is λ-opt need not be the globally optimal solution. The definition of λ-opt, for example, does not consider changes in the schedule involving the repositioning of several aircraft simultaneously. This is accomplished by incorporating a "random feasible tour generator."

The actual mechanics of the algorithm are as follows: The FCFS order is taken as the initial schedule and the time required to land all of the aircraft is computed. Then, by randomly selecting aircraft which are adjacent in the FCFS schedule and switching their order, a new schedule, but one still feasible in the sense of the MPS constraint, is generated. This completes the random feasible tour generator procedure. The second part of the algorithm generates an MPS-opt schedule by considering all feasible repositionings of a given aircraft, starting with the first, and the associated time required to land the aircraft; the best of these is then chosen. This is done for each aircraft and is then repeated until no more improvement is possible. In view of the definition above, it should be clear that the schedule at this point is MPS-opt.

The entire process is repeated a number of times with the best result to date always being stored.

It has been found that any given MPS-opt solution has a nontrivial probability, P_s, of being the optimal solution for that set of constraints. Thus, for any given problem one can estimate P_s (appendix B) and produce K MPS-opt schedules, from random initial starts, choosing K such that $1 - (1 - P_s)^K$ is as close to unity as desired.

Up to this stage the scheduling problem has been essentially modeled as an open-loop system; that is, let the system run until a large number of outputs are collected and store the best of these. The system described so far is illustrated in figure 2. However, the following observations are made:

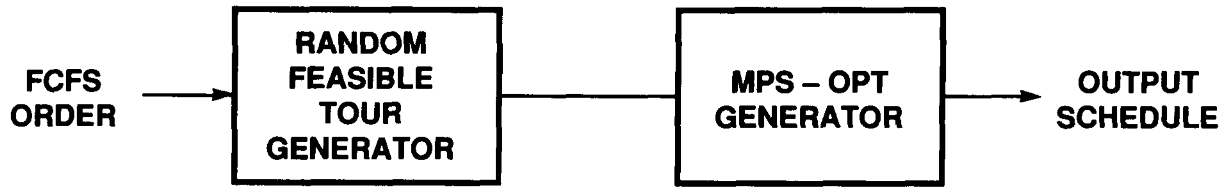

Figure 2.- The approach to the aircraft scheduling problem as studied in this paper.

1. Many of the locally optimal solutions are produced more than once.

2. After a number, q, of locally optimal solutions have been produced note that the set ℓ of elements common to all q of these solutions is, in general, not empty.

Based on these observations and incorporating a type of feedback, the following two procedures have the potential of improving the efficiency of the algorithm.

The Checkout Avoidance Procedure

At some stage, t, during the MPS-opt-producing algorithm a schedule S of cost C is produced to which no further improvement can be made. Denote by S a local optimum and the time from t until this determination is made is called the checkout time. If at a later stage we arrive at schedule S there is no need to go through the checkout procedure. This typically reduces run time by about 30%.

The Reduction Procedure

After producing q locally optimal solutions, consider the set ℓ of aircraft which maintain the same position in the schedule in each of them. Intuitively it seems that for large q any element of the set ℓ is very likely to be an element of the optimal solution. Thus the reduction procedure is as follows.

Produce a number of locally optimal solutions (approximately five solutions seems to be appropriate) and compute the set ℓ which is the intersection of these. Then, restrict all further considered schedules to contain ℓ. If an element of ℓ is also an element of the optimal solution then it is called proper reduction; otherwise it is referred to as improper reduction. To avoid improper reduction several independent runs using this procedure should be done.

This procedure is not currently incorporated into the algorithm for the following reasons. With small values of MPS the optimal solution has generally been found by the time five local optimums have been generated; hence, there is no need for it in this case. For large values of MPS, on the other hand, the routine is plagued by frequent improper reduction or no reduction at all.

Consider one final potential improvement. While searching for λ-opt solutions with λ = MPS, it is noticed that many schedules that are examined in fact violate the MPS constraint. For example, take some feasible schedule

1-2-4-3-5-6

and consider whether the schedule can be improved by moving the aircraft in position No. 3 (i.e., aircraft No. 4), assuming MPS = 1. Thus, the following schedules would be examined:

1-4-2-3-5-6

and

1-2-3-4-5-6

However, clearly the first of these violates the MPS constraint (aircraft No. 4 is in slot No. 2) and another possible legal schedule,

1-2-3-5-4-6

would not be considered. Hence the following definition of MPS-opt is more natural and was adopted.

Definition: A schedule is called MPS-opt if and only if moving any aircraft to another feasible position does not improve the schedule.

Finally, in the algorithm for producing MPS-opt schedules, two optimum-seeking methods are possible, a random descent or a steepest descent procedure. A random descent procedure makes the first improvement to the schedule that is found while the steepest descent procedure makes the best move possible at each step.

After collecting data on run time and quality of solution for each method, the random descent technique was adopted. Run time is usually reduced by over 50% while the likelihood of obtaining the optimal solution is not substantially affected.

RESULTS

When evaluating an algorithm for aircraft scheduling there are key factors--the net improvement in capacity over FCFS and the computation time required by the algorithm. To test the first of these an aircraft mix consisting of 15% small aircraft, with remainder equally distributed between large and heavy, was selected as a basis of comparison. Twenty different sample problems were used, each containing 40 aircraft. Table 2 provides a comparison of the percentage increase in airport throughput compared to FCFS for a method based on Roger Dear's methodology (DEAR) and MPS-opt. In DEAR, a window of 2 * MPS + 1 aircraft is considered and that portion of the schedule is optimized by complete enumeration, the first aircraft in this window is then pushed out and a new one enters at the bottom. This process continues until the end of the schedule is reached. A maximum position shift of three slots was determined to be the practical limit and is the greatest value examined in this paper. The computer program, however, does not require any modification for other values of MPS.

TABLE 2.- COMPARISON OF DEAR AND MPS-OPT METHODS

MPS	DEAR	One run MPS-opt	Best of 500 runs MPS-opt
1	1.81	2.36	2.84
2	2.98	3.88	4.65
3	4.20	4.24	5.70

The other important factor is computation time which is shown in figure 3 as a function of MPS and problem size. Computation time is in seconds of CPU time on the VAX for 500 runs.

The important feature of this graph is that the computation time grows approximately linearly with both problem size and MPS over the range studied.

Up to this point it has been assumed that aircraft following the same approach route would be prohibited from passing one another. This restriction, however, may not be entirely necessary in practice and increased throughput may be achieved by relaxing the constraints. Consider the following three alternative rules:

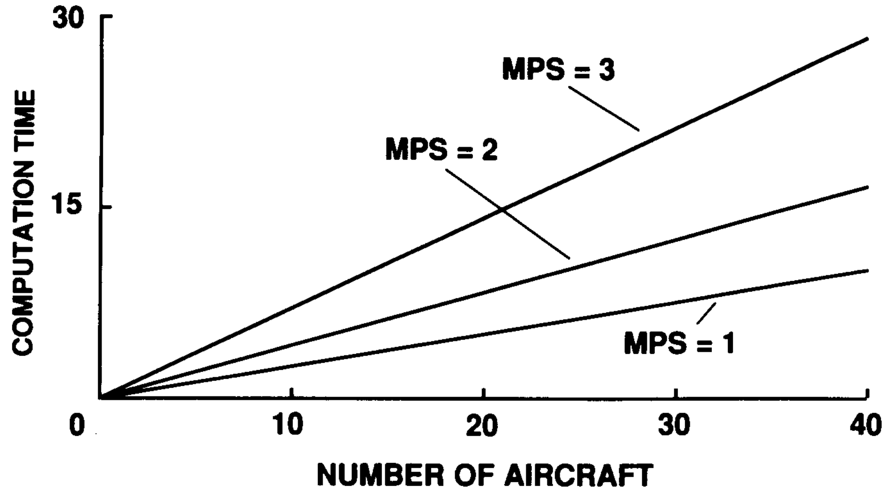

Figure 3.- Computation time for varying values of MPS.

1. The nominal condition in which aircraft on the same route may not pass one another.

2. The condition in which aircraft on the same route may switch positions with their immediate neighbors, but other switches are not permitted.

3. The condition in which no restriction is placed on aircraft switching based on their route.

These rules were applied to a route geometry based on arrival traffic to Denver's Stapleton airport, in order to understand the effect of various route passing constraints on capacity. The results are shown in figure 4. The figure plots increase in throughput as a function of MPS, for each rule. As seen from the figure, relaxing the passing constraint only slightly improves throughput.

ALTERNATIVE OBJECTIVE FUNCTIONS FOR FUTURE STUDY

There are a number of features that could be included into a scheduling algorithm which may greatly increase its usefulness to the ATC community. The two that come most readily to mind are a time varying objective function and a prioritizing system.

It is becoming common these days for airline companies to engage in a practice referred to as "hubbing" which has the effect of creating very nonuniform arrival rates throughout the day. Certain periods of the day may be very busy, while at other times the airport may be operating well below capacity. During the busy periods (i.e., when the incoming arrival rate exceeds airport capacity) the objective function of maximizing throughput should be used. During other times of the day, however, another objective function, such as minimizing total delay, may be more appropriate.

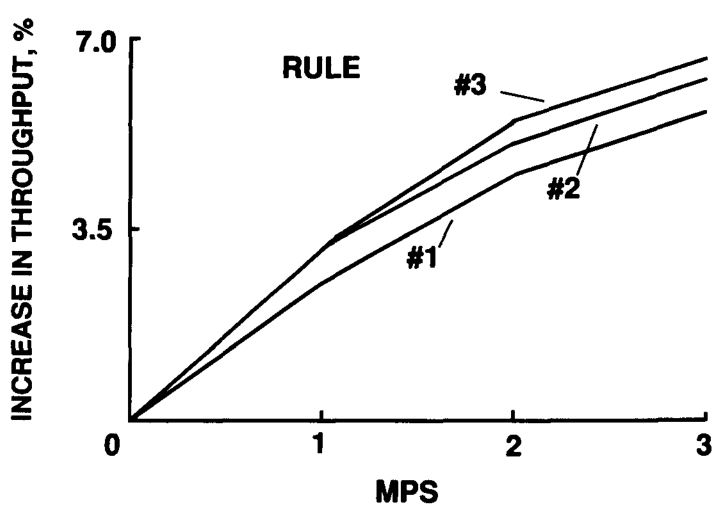

Figure 4.- Increase in throughput for different conditions and varying values of MPS.

Another feature that would not be difficult to implement is one involving a system of priorities. For instance, it may be more desirable for small aircraft to absorb delays than for large, commercial traffic. This could be handled by creating a new objective function, g, to minimize total delay:

$$g = P_S D_S + P_L D_L + P_H D_H \quad \begin{aligned} P_S &= \text{Small-plane priority} \\ P_L &= \text{Large-plane priority} \\ P_H &= \text{Heavy-plane priority} \\ D_S &= \text{Small-plane delay} \\ D_L &= \text{Large-plane delay} \\ D_H &= \text{Heavy-plane delay} \end{aligned} \tag{3}$$

This equation could also be extended to the point where each aircraft is assigned its own priority level.

The disadvantage of this approach is that minimizing total delay is not the same as maximizing airport throughput. Hence, one may wish to consider simply assigning different values of MPS to different aircraft types. Also, there is no reason that the maximum forward shift allowed must be equal to the largest permitted rearward move.

CONCLUSIONS

After examining a number of possible algorithms for aircraft scheduling in the near terminal area the Lin-Kernighan algorithm was selected as being exceptionally well suited to the scheduling problem. Among the reasons for this selection is the natural way in which the MPS constraint can be incorporated.

A modified version of the algorithm was implemented in Fortran and it was tested on over two hundred sample problems. It was found that repeated runs on a problem from random initial starts leads to a high probability of finding the optimal solution among the locally optimal solutions generated. Furthermore, the algorithm is robust in the sense that in none of the test cases did it ever fail to generate a solution nor did it ever generate an infeasible solution.

Another advantage of this type of heuristic approach is that it is easy to trade off the quality of solution (in terms of best solution found or probability of obtaining the optimal solution) for run time.

APPENDIX A

PROGRAM OPERATION

The algorithm described was coded for the VAX in Fortran. It is written as a subroutine for Branch (a program written by Monica Alcabin, NASA Ames Research Center, that generates random schedules) and may be called anytime after the "schedule" subroutine. The only additional input required by the program is the value of MPS. The output consists of the percent increase in landing rate for the first run-through of the procedure as well as for the best of 500 runs.

Also displayed are the first occurrences of this result and how often it occurred. While the program is running it displays the current iteration number and the cost of the associated schedule, a blank line indicates the previous solution is the best located to date. Iterations which are not followed all the way through (see the section on checkout avoidance) are not displayed. The final schedule is stored in Eff.dat.

The program is named Eff.for and is fairly heavily documented. There is also a program called Dear.for which operates in an analogous fashion and implements the methodology proposed by Roger Dear.

```
      SUBROUTINE OPT
C
C     This subroutine implements an algorithm for improving aircraft
C     landing schedules in terms of aircraft throughput.  This subroutine
C     should be called from BRANCH, a program written by Monica Alcabin
C     and available on her account.  This program reads in the list of
C     aircraft, their respective routes and weights and then produces
C     a new schedule which is stored in Eff.dat.
C     This work was done at NASA Ames Research Center, summer 1987, by
C     Robert Luenberger.
C
C     The five weight classes are:  1  -  Heavy equipped
C                                   2  -  Large equipped
C                                   3  -  Heavy unequipped
C                                   4  -  Large unequipped
C                                   7  -  Small unequipped
C


      COMMON NAC,IACTP(100),IACST(100),IFFX(100),KTABLE(100)
      COMMON ICORD(100),IWT(100),IBSTORD(100),ICL(100)
      COMMON IRW(100),RND(100),DSEED,IEVER(100)

      DIMENSION ICOST(7,7),JSTORE(6,100),JSCORE(6),ICHECK(400)

      OPEN (UNIT=4,NAME= 'EFF.DAT',TYPE= 'NEW')

C     Set up the cost matrix

      DATA (ICOST(1,J),J=1,7)/104,125,2*0,114,135,230/
      DATA (ICOST(2,J),J=1,7)/2*78,2*0,2*88,170/
      DATA (ICOST(5,J),J=1,7)/114,135,2*0,124,145,240/
      DATA (ICOST(6,J),J=1,7)/2*88,2*0,2*98,180/
      DATA (ICOST(7,J),J=1,7)/2*88,2*0,2*98,130/

C     Initially, all the aircraft are in FCFS-RW order.

1     TYPE *,'WHAT IS THE VALUE OF MPS?'
      ACCEPT *,MPS
      IF (MPS.EQ.0) THEN
         GO TO 300
      END IF

      ITIME=0

      DO K=1,400
        ICHECK(K)=0
      END DO

C     Now we set it such that IWT contains the weight class and
C     IRW contains the approach path information.
```

```
      DO I=9,NAC+8
         J=I-8
         IWT(J)=IACTP(I)
         IRW(J)=IFFX(I)
         ICL(J)=IACST(I)
      END DO


      INIT=0
      DO I=1,NAC-1
         INIT=INIT+ICOST(IWT(I),IWT(I+1))
      END DO

3     DO I=1,NAC
         ICORD(I)=I
         IBSTORD(I)=I
      END DO

      ITIME=ITIME+1

      IF (ITIME.EQ.1) THEN
        GO TO 4
      END IF

      IF (ITIME.EQ.501) THEN
         GO TO 250
      END IF
C
C    Produce a somewhat randomized version of the initial tour
C
      DO J=1,MPS
         DSEED=(DSEED+1.5)/3
         CALL GGUBS(DSEED,NAC,RND)
         K=3
         IF (REAL(ITIME+J)/2.EQ.(ITIME+J)/2) THEN
            K=2
         END IF
         DO I=K,NAC-1,2
            IF (IRW(ICORD(I)).NE.IRW(ICORD(I+1))) THEN
               IF (ICL(ICORD(I)).NE.ICL(ICORD(I+1))) THEN
               IF (RND(I).LT.0.5) THEN
                  IDUM=ICORD(I)
                  ICORD(I)=ICORD(I+1)
                  IBSTORD(I)=ICORD(I+1)
                  ICORD(I+1)=IDUM
                  IBSTORD(I+1)=IDUM
              END IF
               END IF
            END IF
         END DO
      END DO
```

```
C    ****************************************************************
C    *                                                              *
C    *          Now the MPS-opt improvement stuff starts           *
C    *                                                              *
C    ****************************************************************

4     IMARK=0
      IB=2

5     DO I=IB,NAC-1

      IBACK=ICORD(I)-I

C     First we must find the initial cost of this block

         ISTART=0
         DO K=I-1-MPS+IBACK,I+MPS+1+IBACK
              ISTART=ISTART+ICOST(IWT(ICORD(K)),IWT(ICORD(K+1)))
         END DO
C
C ISTEP will cover the range of possible locations for the
C current aircraft.
C
         DO ISTEP=IBACK-MPS,IBACK+MPS
            IF (((I+ISTEP).GT.NAC).OR.((I+ISTEP).LT.2)) THEN
               GO TO 10
            END IF
C
C  Do the 'throw' - foward moves
C
               IF (ISTEP.GT.0) THEN
                 ISTORE=ICORD(I)
                 DO K=I,I+ISTEP-1
                    ICORD(K)=ICORD(K+1)
                 END DO
                 ICORD(I+ISTEP)=ISTORE
               END IF
C
C  Do the 'throw' - rearward moves
C
               IF (ISTEP.LT.0) THEN
                 ISTORE=ICORD(I)
                 DO K=I,I+ISTEP+1,-1
                    ICORD(K)=ICORD(K-1)
                 END DO
                 ICORD(I+ISTEP)=ISTORE
               END IF
C
C  Evaluate this arrangement
C

               ICST=0
            DO K=I-1-MPS+IBACK,I+MPS+1+IBACK
               ICST=ICST+ICOST(IWT(ICORD(K)),IWT(ICORD(K+1)))
            END DO
            IDIFF=ICST-ISTART
            IF (IDIFF.GE.0) THEN
               GO TO 8
            END IF
C
C  Check the legality of the arrangement--MPS constraint
C
            LEGAL=1
            DO K=I-1-MPS+IBACK,I+MPS+1+IBACK
               IF (ABS(K-ICORD(K)).GT.MPS) THEN
                  LEGAL=0
                  GO TO 8
               END IF
            END DO
C
C  Check the legality of the arrangement--Runway constraint
C  and fairness to aircraft of the same type
C
            DO J=I-1-MPS+IBACK,I+MPS+1+IBACK
               DO K=J+1,I+MPS+2
                  IF (IRW(ICORD(J)).EQ.IRW(ICORD(K))) THEN
                     IF (ICORD(J).GT.ICORD(K)) THEN
                        LEGAL=0
                        GO TO 8
                    END IF
                  END IF
                  IF (ICL(ICORD(J)).EQ.ICL(ICORD(K))) THEN
                     IF (ICORD(J).GT.ICORD(K)) THEN
                        LEGAL=0
                        GO TO 8
                     END IF
                  END IF
               END DO
            END DO
C
C  If the move improves the schedule and is legal -- do it.
C

            IF ((IDIFF.LT.0).AND.(LEGAL.EQ.1)) THEN
               IF (ISTEP.GT.0) THEN
                  ISTORE=IBSTORD(I)
                  DO K=I,I+ISTEP-1
                     IBSTORD(K)=IBSTORD(K+1)
                  END DO
                  IBSTORD(I+ISTEP)=ISTORE
                END IF

                IF (ISTEP.LT.0) THEN
                   ISTORE=IBSTORD(I)
                   DO K=I,I+ISTEP+1,-1
```

```
                              IBSTORD(K)=IBSTORD(K-1)
                           END DO
                           IBSTORD(I+ISTEP)=ISTORE
                         END IF
C
C Store the current best order
C
                         IBSTORD(1)=1
                         DO K=I-1-MPS+IBACK,I+MPS+1+IBACK
                            ICORD(K)=IBSTORD(K)
                         END DO
                         IMARK=1
                         IB=I
                         GO TO 5
                     END IF
C
C If we are at this stage the move that was just evaluated will
C not be made and hence we need to undo it
C
8                    DO K=I+IBACK-MPS,I+IBACK+MPS
                        ICORD(K)=IBSTORD(K)
                     END DO

10          END DO
15      END DO

C
C  If no more good moves were made on the last pass through the
C  schedule then we have reached a local optimum and will stop
C
        IF (IMARK.EQ.0) THEN
           GO TO 100
        END IF
C
C Here we see if the current schedule has been evaluated in an
C earlier run through.  If so, we do not waste time on further
C evaluation.
C
        IFC=0
C        IBSTORD(1)=1
        DO K=1,NAC-1
           IFC=IFC+ICOST(IWT(IBSTORD(K)),IWT(IBSTORD(K+1)))
        END DO
        IF ((ICHECK(INIT-IFC)).EQ.1) THEN
           GO TO 150
        END IF
C
C  If a move was made during the last pass then we will go through
C  the schedule again, from the beginning
C

        IB=2
        IMARK=0
```

```
      GO TO 5
C
C  Here we compute the final cost for this solution
C

100   IFC=0
      IBSTORD(1)=1
      DO K=1,NAC-1
         IFC=IFC+ICOST(IWT(IBSTORD(K)),IWT(IBSTORD(K+1)))
      END DO
      TYPE *,ITIME,IFC
      ICHECK(INIT-IFC)=1
C
C  If this is the best solution to date then we store it
C
      IF ((IFC.LT.IBEVER).OR.(ITIME.EQ.1)) THEN
         IBEVER=IFC
         IFOIS=ITIME
         TYPE *,IFOIS
         ZHAP=0
         DO K=1,NAC
            IEVER(K)=IBSTORD(K)
         END DO
      END IF
C
C  Keep track of how many times this solution has occured
C
150   IF (IFC.EQ.IBEVER) THEN
         ZHAP=ZHAP+1
      END IF
C
C  Print out the results
C
      IF (ITIME.EQ.1) THEN
         TYPE *,'ONE RUN'
         TYPE *,'INITIAL COST: ',INIT
         TYPE *,'FINAL COST :',IBEVER
         TYPE *,' '
         TYPE *,'% SAVINGS: ',100*(REAL(INIT)-REAL(IBEVER))/REAL(IBEVER)
      END IF
      GO TO 3
250   TYPE *,' '
      TYPE *,'BEST ORDER: '
      IEVER(1)=1
      DO K=1,NAC
         TYPE *,IEVER(K)
         WRITE (4,251) IEVER(K)
      END DO
      WRITE (4,252) INIT,IBEVER
251   FORMAT (2X,I2)
252   FORMAT (2X,I4,4X,I4)
      TYPE *,'WEIGHTS:'
      DO K=1,NAC
```

```
          TYPE *,IWT(IEVER(K))
       END DO

       TYPE *,' '
       TYPE *,'BEST OF 500 RUNS'
       TYPE *,'INITIAL COST: ',INIT
       TYPE *,'FINAL COST: ',IBEVER
       TYPE *,' '
C
C Note that the formula below for 'savings' has been converted to reflect
C airport throughput savings, not time savings.
C
       TYPE *,'% SAVINGS: ',100*(REAL(INIT)-REAL(IBEVER))/REAL(IBEVER)
       TYPE *,' '
       TYPE *,'IT TOOK :',IFOIS
       TYPE *,'% OF TIME IT OCCURED: ',ZHAP/5
       GO TO 1

300    END
```

APPENDIX B

PROBABILITY OF OPTIMAL SOLUTION

A HEURISTIC ESTIMATE OF THE PROBABILITY OF OBTAINING THE OPTIMAL SOLUTION

Unfortunately, as opposed to the TSP, there are no classic test cases for the aircraft scheduling problem that one can use to make comparisons among various techniques. To try to compensate for this, several test problems were devised and run on a variety of different algorithms which were available. The best of these was then conjectured to be the optimal solution and will be treated as such in the discussion that follows. Often, a number of different algorithms produced the same best solution, thereby providing at least circumstantial evidence that it is indeed the optimal solution. Further, five problems for each MPS of one, two, and three, and their associated conjectured optimal solutions will be presented with the hope that other authors will either improve upon them or verify their optimality. Eventually, these problems will be able to serve as benchmark cases. For each of these, we set the percentage of small aircraft at 15% and assume that the rest of the traffic is evenly distributed between large and heavy aircraft; each set of problems consists of 40 aircraft.

Here some empirical data on the probability that an MPS-opt solution is in fact optimal (the best-known solution) is presented. Over 200 problems were run, divided equally among MPS values of one, two, and three, and involving various numbers of aircraft. The algorithm was run 500 times for each case and it failed to obtain the best-known solution in less than 3% of the cases. To emphasize again, however, these best-known solutions may not in fact be optimal.

The probability that a particular run of the algorithm will produce the best-known solution, P_S, varies greatly from problem to problem, but is loosely a function of MPS and problem size. Empirical data suggest that

$$P_S \sim 0.8(0.5)^{MPS}$$

and that it is largely independent of problem size over the range studied.

To help provide a standard for comparison among various scheduling algorithms the following five conjectured optimal solutions are presented. These test cases can be generated by running Branch (written by Monica Alcabin, NASA Ames Research Center) as given in table B-1.

Table B-1 indicates the percentage increase in landing rate for each test case. In each case, the schedule which obtains this solution can be found by running the program described in appendix A. These solutions all adhere to the original runway constraint condition given in the problem statement.

TABLE B-1.- PERCENT INCREASE IN THROUGHPUT FOR FIVE TEST CASES WITH VARYING VALUES OF MPS

Number of aircraft:	40
Number of routes:	5
Percent small:	15
Percent large:	42.5
Percent heavy:	42.5
Random number seed:	1-5

	MPS		
SEED	1	2	3
1	2.44	5.78	5.78
2	3.25	4.53	4.53
3	3.30	6.02	6.52
4	1.85	2.80	3.10
5	5.69	6.00	7.17

ON COMPARING HEURISTIC ALGORITHMS

Consider the situation where we can devote T seconds of computer time to a combinatorial optimization problem. If a given algorithm cannot provide a solution within this time it is deemed unacceptable and is removed from consideration. Now, assume that one wishes to compare two heuristic algorithms A_1 and A_2 and that each produces a number of locally optimal solutions in time T. Algorithm A_1 produces solutions which are optimal with probability P_1 in time t_1 while algorithm A_2 requires t_2 to produce solutions which are optimal with probability P_2. To get a good estimate of the P_i's and t_i's the algorithms must be run on a large number of sample problems. The probability of A_1 producing an optimal solution in time T is:

$$P_1^{\sim} = 1 - (1 - P_1)^{k_1}, \qquad k_1 = T/t_1$$

Similarly,

$$P_2^{\sim} = 1 - (1 - P_2)^{k_2}, \qquad k_2 = T/t_2$$

Thus, A_1 is considered to be superior to A_2 if $P_1^{\sim} > P_2^{\sim}$. That is,

$$A_1 > A_2 \text{ if } (1 - P_2)^{T/t_2} > (1 - P_1)^{T/t_1}$$

REFERENCES

1. Dear, R. G.: The Dynamic Scheduling of Aircraft in the Near Terminal Area. MIT Flight Transportation Laboratory Report R76-9, September 1976.

2. Lawler, E. L.; Lenstra, J. K.; Rinnooy Kan, A. H. G.; and Shmoys, D. B.: The Traveling Salesman Problem. John Wiley and Sons, Chichester, Great Britain, 1985.

3. Papadimitriou, C. H.: The Euclidian Traveling Salesman Problem is NP-Complete. Theoretical Computer Science, vol. 4, June 1977, pp. 237-244.

4. Lin, S.: Computer Solutions of the Traveling Salesman Problem. Bell System Technical Journal, vol. 44, December 1965, pp. 2245-2269.

5. Lin, S.; and Kernighan, W.: An Effective Heuristic for the Traveling Salesman Problem. Operations Research, vol. 21, March-April 1973, pp. 498-516.

6. Kanellakis, P. C.; and Papadimitriou, C. H.: Local Search for the Asymmetric Traveling Salesman Problem. Operations Research, vol. 28, no. 5, September-October 1980, pp. 1086-1099.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA TM-100062	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle A Traveling-Salesman-Based Approach to Aircraft Scheduling in the Terminal Area	5. Report Date March 1988
	6. Performing Organization Code
7. Author(s) Robert A. Luenberger (California Institute of Technology, Pasadena, CA; working with San Jose State University summer program)	8. Performing Organization Report No. A-88074
	10. Work Unit No. 505-66-11
9. Performing Organization Name and Address Ames Research Center Moffett Field, CA 94035	11. Contract or Grant No.
	13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546-0001	14. Sponsoring Agency Code

15. Supplementary Notes

Point of Contact: Monica Alcabin, Ames Research Center, MS 210-9
Moffett Field, CA 94035 (415) 694-5451 or FTS 464-5451

16. Abstract

This paper presents an efficient algorithm, based on a well-known algorithm for the traveling salesman problem, for scheduling aircraft arrivals into major terminal areas. The algorithm permits, but strictly limits, reassigning an aircraft from its initial position in the landing order. This limitation is needed so that no aircraft or aircraft category is unduly penalized. Results indicate, for the mix of arrivals investigated, a potential increase in capacity in the 3-5% range. Furthermore, it is shown that the computation time for the algorithm grows only linearly with problem size.

17. Key Words (Suggested by Author(s)) Traveling salesman Scheduling Terminal area	18. Distribution Statement Unclassified-Unlimited Subject Category - 03

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of pages 22	22. Price A02

NASA FORM 1626 OCT 86

For sale by the National Technical Information Service, Springfield, Virginia 22161