Term Project Report

Mathai Paul

Gx3081

Individual

# Problem Description

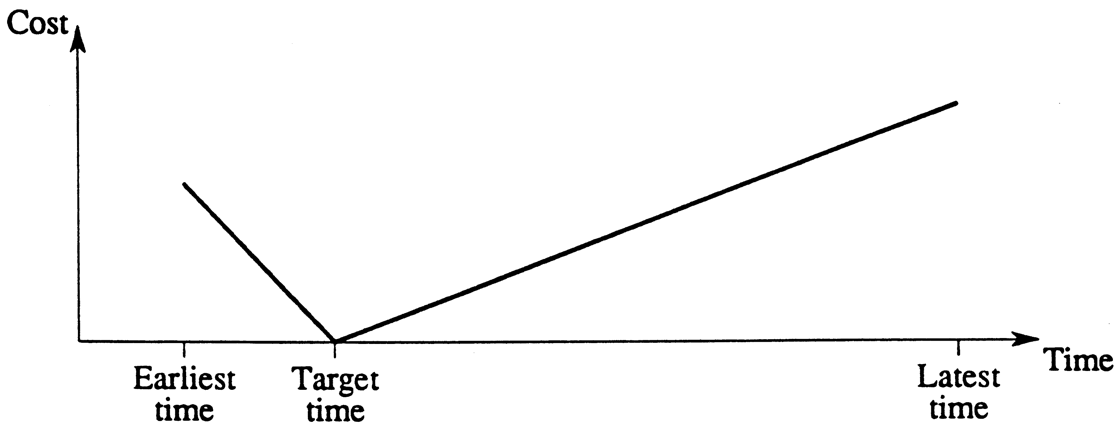
The problem I choose is that of scheduling aircraft (plane) landings at an airport. This problem is one of deciding a landing time for each plane such that each plane lands within a predetermined time window and that separation criteria between the landing of a plane and the landing of all successive planes are respected.

Upon entering within the radar range (radar horizon) of air traffic control (ATC) at an airport, a plane requires ATC to assign it a landing time, sometimes known as the broadcast time, and also, if more than one runway is in use, assign it a runway on which to land. The landing time must lie within a specified time window, bounded by an earliest time and a latest time, these times being different for different planes

The Operation restrictions for the above problem are Latest Landing Times and Runway Allocations. The latest landing time is set (as indicated above) based on fuel considerations. Assuming that this latest time is sufficiently large as to be of no consequence is unrealistic. Runway allocation deals with assigning an appropriate runway to a plane, especially for cases of more than one runway. For example, London Heathrow has two runways that, as a matter of policy, usually operate in segregated mode. London Gatwick, by contrast, has a single runway that (obviously) operates in mixed-mode

# Literature Review

The paper begins by introducing the problem and all the various variables that have to be looked into. It advances to look upon the complexities. This includes the control condition, explaining the variable separation time as the minimum required before another plance can land. To set mandatory minimum separation times, the appropriate aviation authorities classify planes into a small (e.g., three or four) number of classes and specify the separation that must apply between each class. The latest landing time is set (as indicated above) based on fuel considerations. Runway allocation deals with assigning an appro- priate runway to a plane . In this paper, they assume that they are minimizing total cost, where the cost for any plane is linearly related to deviation from its target time.

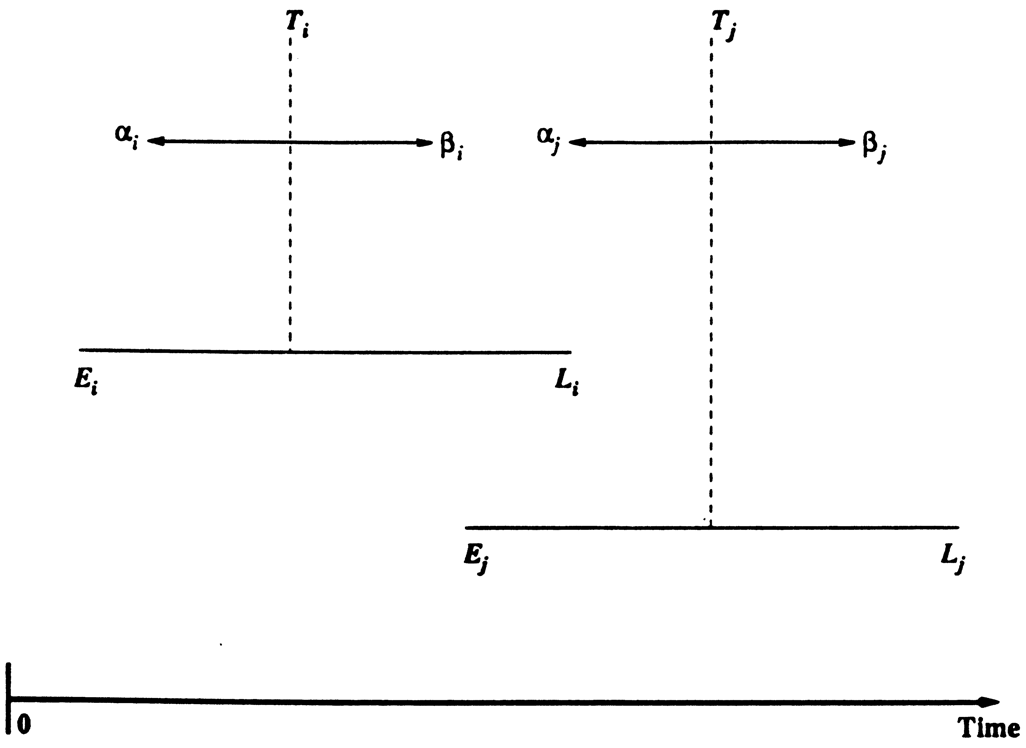


Although the cost function shown in Figure is nonlinear, the fact that it is composed of two linear portions enables us to linearize it and formulate the problem with a linear objective function.

The aircraft landing problem is a mixed-integer problem, hence only by adopting a linearizable objective can such a model be solved numerically to optimality.

The paper continuous to formulate a model for single runway. an initial mixed-integer zero–one formulation of the static single runway aircraft landing problem.

The paper also to clarify some of the constraints that will be given in this section, we provide a diagram. which depicts overlapping time windows for planes i and j.



If we can find an upper bound ZUB on the optimal solution to the problem, then we can tighten the time windows for each plane as indicated above In addition, such an upper bound can be used to curtail the LP-based tree search. THE ALGORITHM PRESENTED in this paper was pro- grammed in FORTRAN and run on a DEC 3000/700 (200 Mhz Alpha chip) for a number of test problems involving up to 50 planes. To solve the mixed-integer zero–one formulations of the problem to optimality using LP-based tree search, we used the CPLEX software package (CPLEX, 1994). Each test problem was solved with an increasing number of runways until the optimal solution value dropped to zero (indicating that we had sufficient runways to enable all planes to land on target).

A close up of text on a white background

Description automatically generated

The above table was used to put out the results.

# Preprocessing:

A lot of pre-processing had to go into this problem for the data was provide in a raw unstructured format. I used c++ to format the data first in structured format. Each of the files labeled ne\_airland\*\*.txt is converted from airland\*\*.txt. Which was then converted to CPLEX arrays since CPLEX on the mac does not support SheetConnection. I have included the three c++ programs and the converted data.

The Program is included in the submission as “cplex\_array\_and\_set\_creation.cpp” and “txt\_to\_sturct\_csv.cpp”. which can be run on any C++ compiler.

Note: the data file path has to be set each time based on OS.

# Model Description:

Separate Handwritten Document named “ModelDescription.pdf” included in the submission.

# Data:

Data is acquired from <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/airlandinfo.html>.

These data files are the test problems used in the papers "Scheduling aircraft landings - the static case" by J.E. Beasley, M. Krishnamoorthy, Y.M. Sharaiha and D. Abramson, Transportation Science, vol.34, 2000, pp180-197; and "Displacement problem and dynamically scheduling aircraft landings" by J.E. Beasley, M. Krishnamoorthy, Y.M. Sharaiha and D. Abramson, Journal of the Operational Research Society, vol.55, 2004, pp54-64.

The format of these data files is:

number of planes (p), freeze time

for each plane i (i=1,...,p):

appearance time, earliest landing time, target landing time,

latest landing time, penalty cost per unit of time for landing

before target, penalty cost per unit of time for landing

after target

for each plane j (j=1,...p): separation time required after

i lands before j can land

model characteristics ( # variables, constraints, run time  etc),

# Model Characteristics

Number of Variables:

107, where 100 is the separation matrix and the 7 are other metrics

Number of Planes:

100, no of planes in the solved problem

## Run time:-

Found incumbent of value 3370013.000000 after 0.02 sec. (6.54 ticks)

Tried aggregator 1 time.

MIP Presolve eliminated 10100 rows and 30200 columns.

Reduced MIP has 10500 rows, 11500 columns, and 32000 nonzeros.

Reduced MIP has 0 binaries, 300 generals, 0 SOSs, and 0 indicators.

Presolve time = 0.02 sec. (21.41 ticks)

Tried aggregator 1 time.

Reduced MIP has 10500 rows, 11500 columns, and 32000 nonzeros.

Reduced MIP has 0 binaries, 300 generals, 0 SOSs, and 0 indicators.

Presolve time = 0.02 sec. (15.74 ticks)

MIP emphasis: balance optimality and feasibility.

MIP search method: dynamic search.

Parallel mode: deterministic, using up to 12 threads.

Root relaxation solution time = 0.02 sec. (17.63 ticks)

Nodes Cuts/

Node Left Objective IInf Best Integer Best Bound ItCnt Gap

\* 0+ 0 3370013.0000 1710488.0000 49.24%

\* 0 0 integral 0 1710488.0000 1710488.0000 100 0.00%

Elapsed time = 0.11 sec. (75.51 ticks, tree = 0.00 MB)

Root node processing (before b&c):

Real time = 0.09 sec. (71.90 ticks)

Parallel b&c, 12 threads:

Real time = 0.00 sec. (0.00 ticks)

Sync time (average) = 0.00 sec.

Wait time (average) = 0.00 sec.

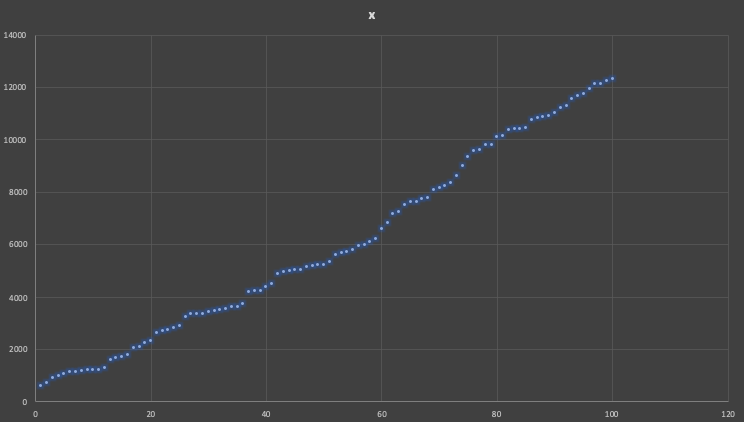
------------

Total (root+branch&cut) = 0.09 sec. (71.90 ticks)

# Results:

Separate Excel file named “Results.xlsx” included in the submission. It includes the graphs as well.

# Insights:



A picture containing object

Description automatically generated

A flat screen tv

Description automatically generated

# Meta-Heuristic (Simulated Annealing):

Simple Simulated Annealing was developed in c++ with the model solution as the first trial solution. The values were varied with the use of a random value and z value compared. Also ten rand number is used for comparing the probability of acceptance for if a new optimal solution is worse.

The Program is included in the submission as “simulated\_annealing.cpp”. which can be run on any C++ compiler.

Note: the data file path has to be set each time based on OS.

# Comparison of Integer Programming and Meta-Heuristic:

The result was nearly impossible to improve with met heuristic, firstly because the optimum solution that was use as trial was very close to the exact solution. Secondly the variation even the random, it wasn’t small enough at time to reach the exact solution. As an improvement to the current solution. One can add gradient decent to optimize the step size of variation at each iteration. The meta heuristic rand for 8 iterations and terminated, since that was the stopping condition hardcoded.