This folder groups a benchmark for Tail Assignment problem which consists in assigning a well-identified airplane to each flight leg of a given flight schedule, in order to minimize operational costs while complying with a number of operational constraints.

The main constraints to consider in the Tail Assignment problem are those specifying whether it is possible to successively operate two given flights with the same airplane. This implies compliance with (a) a minimum time condition; (b) an equipment continuity condition. Condition (a) aims at ensuring that a minimum time is left for cleaning, changing crew and passengers, and preparing aircraft for the next flight. Condition (b) enforces the arrival airport of the first flight to be the same as the departure airport for the second flight. Besides this, we have to take into account a set of "initial conditions" which impose that each airplane should start from an initial airport defined in advance.

For confidential reasons, a collection of test data sets has been generated using a data generation procedure producing test instances with controlled size and density has been designed. It makes use of an existing large scale real data set from a domestic low cost airline in Asia (composed of 1580 flight legs over 30 days with 37 airports and 50 airplanes involved).

The data-generation procedure takes as input parameters:

- a real parameter $d \in [0, 1]$ called density of the instance;
- the first and last day of the desired time period (between 1 and 30), defining the horizon h;
- the number of airplanes (np).

The procedure constructs a subset of flights forming the timetable of the generated instance as follows :

(a) First we solve the problem of determining np_{min}, the minimum number of airplanes necessary to cover all the flights in the set $\widetilde{F}$ of flights which have to be carried out in the specified horizon h. The initial positions of various airplanes are assumed to be the same as for the whole real

data set. This problem can easily be formulated as the search for a minimum flow in a network describing the possible connections between flights, with lower and upper capacity bounds on the arcs. It can be very efficiently solved by using linear programming or via a specialized network flow algorithm. We denote $\bar{P}$ the subset of airplanes used in the resulting optimum solution, $np_{min} = |\bar{P}|$, and for each airplane $j \in \bar{P}$, we denote $L_j \subset I$ the subset of flights operated by this airplane.

(b) After checking that $np \in \{1, ..., np_{min}\}$, which is the case for all the instances of the test set as soon as $np \leq 40$, we perform the following : a subset $\widetilde{P}$ of np district airplanes is randomly selected out of $\bar{P}$ to compose the fleet corresponding to the instances generated;

(c) A subset $\widehat{P}$ of $d \times np$ airplanes is randomly chosen out of the set $\widetilde{P}$, and the set of flights composing the desired instance is finally obtained as : $\bigcup_{j \in \widehat{P}} L_j$.

Thus the *density* of an instance is between the minimum number of airplanes necessary to cover all the flights of the timetable and the number of airplanes in the available fleet. Accordingly, a 100% dense instance is an instance for which the number of available airplanes is exactly equal to the minimum number of airplanes necessary to ensure feasibility. When the density is lower, more flexility is left to find feasible solutions and the problem may be expected to be easier to solve.

The various instances considered in our experiments have been obtained by varying the value of the three parameters:

- the density $d : 50\%, 70\%, 100\%$.
- the number of airplanes $np : 10, 20, 30, 40$;
- the time horizon $h : 7, 15, 21, 30$.

For each value of d,np, and h, 10 instance were generated, t=0..9.

This folder will contain three subfolders corresponding to the three values of density d. Under each subfolder there are other subfolders corresponding

to the values of np and h. Finally, found three files for each instance test t as follow :

1. **Data of each instance** :
   DataCplex_density="d"_p="p"_h="h"_test_"t".dat.
   The data format is compatible with a datafile of OPL Cplex. It contains the timetable, the assignment costs, the set of airports, the number of flights and airplanes, and the initial position of each airplane.
2. **Log file** from the Interactive Optimizer :
   Journal_Cplex_density="d"_p="p"_h="h"_test_"t".txt.
   The information provided in this file concern the linear relaxation of the original MIP at the root, the features of the nodes generated,....for more details about the information presented in this file, we recommend the documentation of CPLEX Studio IDE available on `http://www.ibm.com/support/knowledgecenter`
3. **Optimal assignment** :
   Optimal_Solution_density="d"_p="p"_h="h"_test_"t".txt.
   The file contains the sequence of flights assigned to each airplane. The list of flights are not ordered according to the departure time of flights. We just enumerate the list of flights assigned to each airplane.

The data-generation procedure and the model have been implemented with Python 2.7 and Cplex Studio 12.6.2. All computations have been performed on a workstation running Intel(R) Core(TM) i7-3740QM CPU @ 2.7 GHz.