Getting Started with the ELSA Air Traffic Simulator

0.1 Using the Tactical layer

The Tactical layer aims at transforming planned trajectories into “actual” trajectories by deconflicting them with a simulated super-controller. The standard input is an “M1 file” whereas the output is an “M3 file”, similarly to what exists in DDR data.

The code of the Tactical ABM is stored in src folder. In the config folder there is the config.cfg that contains the values of most of the parameters.

You can directly compile the source code in the following way:

```
cd src/

LC_ALL=C gcc -O3 -c *.c && gcc *.o -o ../ElsaABM.so -lm

cd ../
```

Once the binary `ElsaABM.so` is produced it is possible to execute the simulations:

```
./ElsaABM.so M1_file M3_file Config_file seed
```

where `M1_file` is the file containing the trajectories which will serve as input for the model and `M3_file` is the file that will contain the output, i.e. the trajectories deconflicted by the controller.

Using the files provided as examples in the tests/example/ folder, the user can run:

```
./ElsaABM.so example/M1_example.dat example/M3_example.dat
 config/config.cfg 3497
```

which should produce the `M3_example_0.dat` file with deconflicted trajectories.

0.1.1 Input-Output File Format

Either M1 input file and M3 output files have the same format. The first line must contain the number of flights (trajectories contained in the files):

```
1475\tNflight
```

the following lines are the trajectory of each aircraft. In particular the first two positions, separated by a $\backslash t$, represent respectively the ID, a unique int number that identify the flight, and the number of NVP in the related route.

The following positions are the nvps of the route. They consist in 5 values, separated by a comma: the horizontal position (latitude and longitude), the Flight Level, the time in the format ‘2010-05-06 10:20:32’, and an integer number that refers to the sector to which the navpoint belongs.

0.1.2 Configuration File

Most of the parameters are stored in the configuration file. Changing these values do not require to re-compile the code.

An example of a config file can be found in `config/`. Each line starting with # is a comment, whereas all other lines are values. The name of the variable it refers to needs to be written after a '`\t#`'. For instance:

```
#This is a comment
24\t#t_w\n
```

gives $t_w = 24$. The position of the values in the configuration file does not matter.

Note that the ABM does not perform any consistency checks on the variables. For example if one wants to increase the lookahead t_w, one needs to fix the product $t_w \times t_r \times t_i$ which represents the time-step[1]. These three values are the most crucial ones for the simulations and should be chosen with care. The user can find their descriptions in the deliverable D2.4 included in the doc/ folder.

Most of the values of the configuration file can be changed without too much trouble. The most simple features are tunable with the parameters `nsim` (number of simulations), and `Nm_shock`, (number of shocks), `radius` (radius of the shocks in nautical miles). Using and modifying other parameters requires some more deeper understanding of the models, which can be achieving by reading D2.4, included in the doc/ folder. See also the caveats section 0.2.

0.1.3 Additional files

Other files are required to run the simulations, which are called respectively `temp_nvp.dat`, `shock_tmp.dat`, `sector_capacities.dat`, and `bound_latlon.dat`. Some examples are stored in the config folder `config/`. Their paths have to be specified in the config file, in the corresponding entries.

The `temp_nvp.dat` consists in a two columns text file with latitude and longitude of the temporary points used for rerouting. The number of temporary nvps used in the simulation is defined via a #define variable called `NTMP` in mSector.h. The user must be careful because the code:

- does not check if the points are within the ACC.
- does not check the number of points matches `NTMP`.

The `shock_tmp.dat` has the same format of `temp_nvp.dat` and it contains the centers of the shocks used in the simulations. This also has to be created via an external script, which is not included at the moment in the repository.

[1]The smaller is this product, the better works the ABM by the way. See deliverable D2.4 included in the doc/ folder, however a small time-step implies a high computational effort. A good choice for the time-step is about 3min.

The `sector_capacities.dat` has two columns: the first one stores the ID of the sector, the second the related capacity. These labels need to be consistent with the ones present in the M1 file. Note that the user can use the '-1' for navpoints which do not have to be under control. This is more specifically used for the first and the last points of the trajectories (see caveats 0.2).

Finally, the file `bound_latlon.dat` contains the boundaries of the airspace as a list of latitudes/longitudes, like the `temp_nvp.dat` file.

0.1.4 Define

Other parameters are defined in the header files. Every time you want to change these parameters you need to recompile the code. For example in mABM.h is defined LS that is the minimum improvement of the trajectory for a direct:

```
#define LS 1000
```

Another useful #define is DTMP_P that is the Maximum distance in meters of the selected temporary points for the rerouting operations.

Others #define do not have associated values. For example:

```
#define SINGLE_TOUCH
```

If you comment this #define the ABM can perform a multiple modification of the route of the same aircraft in the same time-step if it does not find any solution.

0.2 Caveats

Here are some potential traps for new users concerning the tactical layer:

- The time step t_s is the one the most important parameter of the model. It is equal to $t_s = t_i \times t_w \times t_r$:

  1. The time increment t_i is the time resolution for the trajectories. It should be very small (around 8 seconds),
  2. The time window t_w represents the time horizon of the controller on which it will compute the future potential conflicts.
  3. The time roll t_r, which is a fraction of the time window, represents the time after which the controller updates trajectories of the flights.

  In particular, a user who would like to increase the time horizon of the controller would need to keep the time-step $t_s = t_i \times t_w \times t_r$ fixed by reducing the time-roll, otherwise different effects will mix up.

- The life duration of shocks is computed with respect to the time step. Hence changing t_i, t_w, or t_r will change the duration of the flights.

- The shocks appear only on 10 flight levels, not on a whole column of air. The variables `shock_f_lvl_min` and `shock_f_lvl_max` do not change this fact. Instead, they are fixing the possible interval of flight levels of apparition of the shocks.

- The starting and ending dates have to be informed in the config file at the corresponding lines. They need to be consistent with the trajectories provided.