EXPLORATORY RESEARCH

sesar JOINT UNDERTAKING

D5.2 - ATM Performance Metamodels - Final Release

Deliverable ID:	**D5.2**
Dissemination Level:	**PU**
Project Acronym:	**NOSTROMO**
Grant:	**892517**
Call:	**H2020-SESAR-2019-2**
Topic:	**SESAR-ER4-26-2019**
Consortium Coordinator:	**CRIDA**
Edition date:	**04 October 2022**
Edition:	**00.01.00**
Template Edition:	**02.00.05**

EUROPEAN PARTNERSHIP

Co-funded by the European Union

Authoring & Approval

Authors of the document

Name / Beneficiary	Position / Title	Date
Francisco Antunes / Technical University of Denmark	Project member	3 October 2022
Christoffer Riis / Technical University of Denmark	Project member	3 October 2022
Francisco Camara Pereira / Technical University of Denmark	Project member	3 October 2022
Gérald Gurtner / University of Westminster	Project member	3 October 2022
Majid Soolaki / University of Westminster	Project member	3 October 2022
Tatjana Bolic / University of Westminster	Project member	3 October 2022
Zak Tibichte / ISA	Project member	11 July 2022

Reviewers internal to the project

Name / Beneficiary	Position / Title	Date
Mayte Cano Rincón	Project Coordinator	4 October 2022

Reviewers external to the project

Name / Beneficiary	Position / Title	Date
N/A		

Approved for submission to the SJU By - Representatives of all beneficiaries involved in the project

Name / Beneficiary	Position / Title	Date
Mayte Cano / CRIDA	Project coordinator	4 October 2022
Francisco Antunes / DTU	Project member	4 October 2022
Gerarld Gurtner / UoW	Project member	4 October 2022

Rejected By - Representatives of beneficiaries involved in the project

Name and/or Beneficiary	Position / Title	Date
N/A		

Document History

Edition	Date	Status	Name / Beneficiary	Justification
00.00.01	28/06/2022	Draft	UoW	Initial version in Confluence platform
00.00.02	03/10/2022	Draft	UoW	Version for review
00.01.00	04/10/2022	Final Draft	UoW	Version for SJU delivery

EUROPEAN PARTNERSHIP

Co-funded by the European Union

EUROPEAN PARTNERSHIP

Co-funded by the European Union

NOSTROMO

NEXT-GENERATION OPEN SOURCE TOOLS FOR ATM PERFORMANCE MODELLING AND OPTIMISATION

This deliverable is part of a project that has received funding from the SESAR Joint Undertaking under grant agreement No 892517 under European Union's Horizon 2020 research and innovation programme.

Abstract

This deliverable presents the third iteration of the development of the two micromodels Flitan and Mercury and the results obtained with them with the active learning process, as described in the deliverables D3.X. In this iteration, Flitan implemented concepts from PJ08.01 and PJ02.08, and Mercury implemented a module related to PJ07.02. Mercury also developed an additional module related to PJ01.01, which description is presented in Annex only, since no results could be produced in time with it for this deliverable.

The development is presented in two different chapters for each simulator, with general descriptions referred to from D5.1. The modules related to each SESAR solution are described separately.

The latest version of the meta-modelling process is described briefly, followed by the results obtained with the two simulators, in distinct sections. This chapter shows the performance of the meta-model with respect to approximating micro simulators.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

Table of Contents

Abstract 4
1 Introduction 7
2 Micromodelling process 8
2.1 FLITAN development 8
2.2 Mercury development 12
3 Metamodelling process 17
4 Metamodelling results 20
4.1 FLITAN 20
4.2 Mercury 25
5 References 27
6 Acronyms 28
Appendix A Mercury Simulator 29
Appendix B Flight arrival coordinator 33

List of Tables

Table 1: Main principles. 13
Table 2: input variables. 16
Table 3: output variables. 16
Table 4: Example from the Josep Tarradellas Barcelona-El Prat Airport, East Air Traffic Control, relating the name of the configuration, configured sectors and elementary sectors. 21
Table 5: An example of a runway configuration data log produced by Flitan. 22
Table 6: Illustrative sample used by the metamodels integrating the encoded data from PJ.08-01 and PJ02-08, using control until LECPCTA and the corresponding airport LEPA. 23
Table 7: Total results of basic model. 35
Table 8: Total cost of flights regarding the first and the last available slots. 42

List of Figures

Figure 1: NOSTROMO experiment regions. 10
Figure 2: Active learning-based metamodelling process overview for ATM performance assessment. 18

EUROPEAN PARTNERSHIP
Co-funded by the European Union

Figure 3: Gaussian Process as metamodel starting with 10 data points..... 24

Figure 4: Gaussian Process as metamodel starting with 50 data points and using only the date variable. 25

Figure 5: XGBoost as metamodel starting with 100 data points, and Greedy Sampling in the input space (GSx) using query batches of size 10. 25

Figure 6: Active learning metamodelling performance for Absolute Equity. 25

Figure 7: Active Learning metamodelling performance for Saved Real Ratio Cost. 26

EUROPEAN PARTNERSHIP

Co-funded by the European Union

1 Introduction

This deliverable, which can be considered as an extension or an update on deliverable D5.1, is focused on the developments of the models that allow a meta-model to approximate the impact of some new concepts in the ATM system, with the help of microsimulators. In this third iteration of NOSTROMO, we implemented new concepts from three SESAR solutions, PJ07.02, PJ08-01, and PJ02.08, in the two microsimulators Flitan and Mercury.

As with previous deliverable D5.1, we describe how we implemented the concepts in the microsimulators, in section 2. Compared to the previous deliverable, we implemented new advanced concepts inspired from these solutions, that may be considered as more futuristic than the previous ones.

The metamodelling process is again explained in section Metamodelling process3, with the latest changes summarised there (while being fully reflected in D3.4.

Section 4 presents results similar to what was presented in D5.1, i.e. focused on the efficiency of the meta-modelling approach and not on the domain interpretation per se (i.e. the impact of the concepts on the ATM system).

In annex A, we copied the general description of the Mercury model, as presented in D5.1, which of course is still valid for this deliverable, since only new modules have been developed.

Finally, in Annex B, we described a module developed for Mercury for PJ01.01, whose results could unfortunately not be included in the present deliverable due to time constraint. This module is almost operational at the time of the delivery of this report, and will be used in future scientific articles.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

2 Micromodelling process

2.1 FLITAN development

2.1.1 Flitan Simulation Engine

Flitan is a network-wide performance assessment model based on the concept of node and connector. Initially developed by ISA Software to support the turned-around analysis for a EU FP7 project called TITAN. Subsequently, Flitan has been adapted to investigate the FlightPath 2050's 4-hours door-to-door target.

At its core, Flitan abstracts the ATM system as a network of nodes and connectors where flights travel safely and efficiently. At the highest level of abstraction, each node represents an ATM entity, be it an airport, sector, etc. A node is primarily characterised by its occupancy time distribution function, which defines the nominal time for the process it represents to be accomplished (e.g. the mean time to use a node) and the possible variance in that time. A Flitan connector is responsible for routing flights between adjacent nodes.

A flight starts its journey at an origin node (e.g. departure airport) and passes through a series of nodes and connectors until it reaches the final node (e.g. arrival airport). At each node, its time distribution function is sampled to determine the time needed for the flight to use the node. The time distribution is defined for a city pair and a specific aircraft model. For example, all the flights with the same aircraft model travelling between the same city pair will share the same distribution functions.

Since the implementation of SESAR solution PJ08-01, Flitan has used a full 4D trajectory flight model. Consequently, the time distribution function of a sector is precisely computed from the flight's sector crossings profile. So, the time a flight will need to use a sector is the difference between the sector exit time and the sector entry time.

The underlying Flitan simulation engine is based on a generic Business Process Modelling (BPM) capability and is seeded using a set of flight trajectories, airports and airspace data. Once active, each flight progresses through the network from node to the next node via connectors until it reaches its final destination.

The simulation engine is designed using a discrete event simulation engine- a type of simulation engine where events within the simulation are scheduled to occur at a particular time in the future, and each event is tied to a piece of code to be executed once that time arrives. It is comprised of a *clock*, an *event list* and the *event scheduling and dispatch* system.

The *scheduling-dispatch* system acts upon the head of the *event list,* which is always equal to the current time. Each piece of code tied to the event being processed can create new events on the schedule either at the current time or at some time in the future. Once all events at the current time are exhausted, the clock steps to the next discrete point in time (the next position in the list) and events that are scheduled at this new time in the simulation are

EUROPEAN PARTNERSHIP
Co-funded by the European Union

executed. A simulation continues to run until there are no events left in the system to dispatch, at which point the simulation is considered to be complete.

2.1.2 Flitan Enhancement in support of NOSTROMO project

In the framework of the NOSTROMO project, two SESAR Solutions have been selected to be implemented and used to feed the NOSTROMO metamodel: PJ-08.01 “Dynamic Airspace Configurations” and PJ02-08 “Traffic optimisation on single and multiple runway airports”.

In implementing the two SESAR solutions, Flitan targets three specific objectives:

1. Enhance Flitan simulation engine by adding modules that support both runway and airspace configurations
2. Enhance Flitan input and output data objects
3. Train the metamodel through the assessment of a large set of possible combinations of configurations

2.1.3 PJ08-01 SESAR Solution

The main objective of this solution is to allow Air Navigation Service Providers (ANSP) to organise, plan and manage airspace configuration in a flexible manner that increases capacity and reduces delays without impacting traffic trajectories. SESAR solution PJ08-01 is articulated around four key concepts:

1. ***Dynamically designed sectors*** tailored to specific flow patterns,
2. ***Dynamic sector configurations*** adapted to respond flexibly to available air traffic controller (ATCO) resources, changes to traffic demand, and performance objectives,
3. ***Dynamic mobile areas,*** which are temporarily reserved volumes of airspace designed to segregate military activities from civil air traffic,
4. ***Fully integrated CDM process*** between civil and military actors enabled by automated support tools.

The scope of the PJ08-01 implementation in Flitan is essentially focused on the second key concept, “Dynamic sector configurations”, and their assessment. Dynamic sector design and sector capacities computation are not part of the Flitan implementation of PJ.08-01.

2.1.4 PJ02-08 SESAR Solution

The main aim of this solution is to enhance airport throughput operations by ensuring that runways operate at their optimum capacity. SESAR solution PJ02-08 is focused on two key concepts:

1. An integrated runway sequence function to balance arrival flights and departure flights on a single runway, dependent runways or parallel runways
2. Use of a runway manager for airports with more than one runway to plan the optimal runway configuration

EUROPEAN PARTNERSHIP

Co-funded by the European Union

2.1.5 NOSTROMO SCENARIO

Opening Schemes

In support of SESAR Solution PJ08-01, twelve Flow Management Positions (FMP) have been selected to train the metamodel:

1. LECBCTAE (Barcelona East, Spain)
2. LECBCTAW (Barcelona West, Spain)
3. LECMCTAN (Madrid North, Spain)
4. LECMCTAS (Madrid South, Spain)
5. LECSCTA (Sevilla, Spain)
6. LECPCTA (Palma de Mallorca, Spain)
7. LFBBCTAE (Bordeaux East, France)
8. LFBBCTAN (Bordeaux North, France)
9. LFBBCTAS (Bordeaux South, France)
10. LFMMCTAE (Marseille East, France)
11. LFMMCTAW (Marseille West, France)
12. LPPCCTA (Lisbon, Portugal)

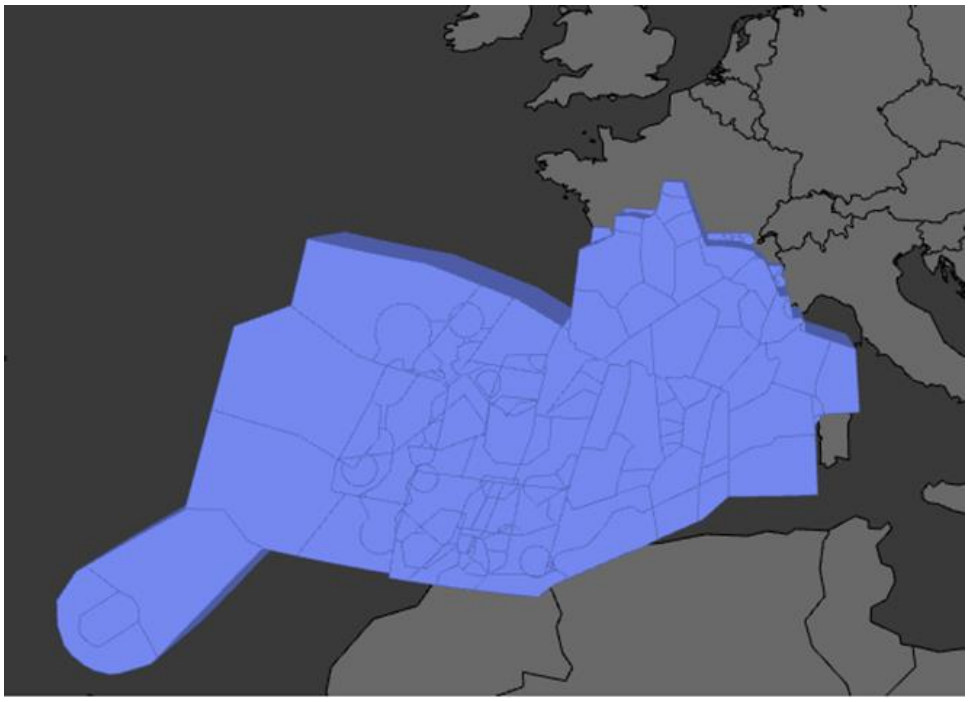


Figure 1: NOSTROMO experiment regions.

Runway Configurations

In support of PJ02-08, three Spanish airports have been chosen:

Co-funded by the European Union

1. Josep Tarradellas Barcelona-El Prat Airport: LEBL
2. Adolfo Suárez Madrid–Barajas Airport: LEMD
3. Palma de Mallorca Airport: LEPA

Traffic and environment data

Traffic and environment data is sourced from NM archives. The environment data uses the DDR2 format and is stored for each AIRAC cycle corresponding to a 28 days period.

Regarding air traffic, Flitan uses ALL-FT+ data format to construct its flight object model. For the sake of callsign uniqueness, Flitan constructs the callsign of each flight by concatenating the aircraft identifier (field #3) and IFPS identifier (field #7) using "-" as a separator. ALL-FT+ traffic data is sourced from NM archives. Traffic data spans from 2019-06-20 to 2019-07-17. The average daily traffic count is around 30000 flights.

2.1.6 Third iteration

In the previous simulation experiment set-up, we were assessing each solution at local levels

- Constructing metamodels at the level of local FMPs (solution PJ08-01)
- Constructing metamodels at the level of local airports (solution PJ02-08)

In the third iteration, we will assess both solutions globally at the level of the integrated Network Operation Plan (NOP) and Airport Operation Plan (AOP):

- AOP is composed of three airports
- NOP is composed of 12 control centres

In this objective, two directories have been created under the evaluated scenario as follows:

1. opening_schemes directory - This directory contains twelve sub-directories where each sub-directory corresponds to a specific FMP and is filled with their opening scheme files. For example, the sub-directory LECMCTAN contains the set of possible opening scheme files that may be deployed in a single day in Madrid North.
2. runway_activation_plans directory - This directory contains three sub-directories: LEBL, LEMD and LEPA. Each sub-directory contains all the possible runway activation plans for the corresponding airport.

Now that the simulation environment is set up, to run the third iteration, ISA Software developed a python script that automatically generates both airspace configuration opening schemes for the twelve FMPs and runway configurations for the three airports mentioned above, as well as extracts the adequate traffic data. The script will draw an opening scheme randomly from each FMP and merge them into a single opening scheme to be evaluated by Flitan. It does the same for the airport runway activation plans. Once the input is ready, the script runs Flitan automatically to assess both SESAR Solutions at the level of the combined FMPs and airports.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

2.2 Mercury development

In D5.1 we gave a general description of the Mercury simulator, than we reproduce in annex XXX for convenience. The purpose of this section is thus to describe the additional development carried out for NOSTROMO.

We describe the module related to PJ07.02, where we implemented a new algorithm destined to improve the efficiency of ATFM hotspot resolution processes while taking into account specific information from the airspace users.

The project has also developed a new module related to PJ01.01, with both a baseline and a more advanced extended arrival manager, taking into account cost of airspace users and probability of arrival. However, this module could not be included in the last release of the binaries to DTU due to on-going integration issues. As a a results, no result with this module are included in this deliverable. Nevertheless, Annex XXX describes the algorithm behind the optimisation process. Some considerations related to this module will be included in deliverable D6.2, and the NOSTROMO team has already planned to publish some results later in the year using this module in Mercury.

2.2.1 General description of Mercury

For a general description of Mercury, please refer to D5.1, or to Annex XXX.

2.2.2 PJ07.02 - Airspace user prioritisation for hotspots

Hotspot solving

For a general description of the hotspot solving process, please refer to D5.1. Below we only summarise the main points.

The goal of PJ07.02 is to solve more efficiently ATFM hotspot (for instance due to congestion), mainly at arrival airports, by taking into account airspace users preferences linked to the type of flights that are involved in the regulation. These mechanisms can take several forms, but the core of them is to try to exchange slots across different airlines while adequately compensate the airlines that lose from the new allocation. In the next

Mechanisms tested in the third iteration of NOSTROMO

In the third iteration of NOSTROMO, we are considering the previous algorithms developed from the second iteration, plus an additional one, described thereafter:

- FPFS: serves as baseline to compare the other mechanisms.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

- UDPP: optimal allocation intra-airlines. It represents the efficiency in the current situation, if all airlines use the UDPP tools in an efficient way.
- ISTOP: developed in BEACON, it uses information akin to what is passed to UDPP to rebuild an approximated normalised cost function and suggests suitable slot swaps among airlines.
- NNBound: uses the true cost functions of the airlines, it finds the best allocation, given that no airline loses in terms of cost. This is used for benchmarking, since it represents the maximum efficiency if local 'fairness' is enforced (nobody loses).
- GlobalOptimum: uses the true cost functions from the airlines, it finds the best allocation overall. This is used for benchmarking, since it represents the maximum theoretical efficiency reachable by any mechanism.
- Credit Mechanism (NEW): developed in BEACON, this mechanism allows airspace user to buy approximated cost functions that are then used in a central optimiser in order to find the best overall solution. Cost functions have higher prices with they put more strain on the system, and are paid in virtual credits.

Abbreviation / acronym	Full expression	Main principles
FPFS	First planned, first served	Serves as baseline to compare the other mechanisms; current mechanism in use, minimises the total delay assigned to flights in a regulation
UDPP	User-Driven Prioritisation Process	Optimal allocation intra-airlines. It represents the efficiency in the current situation, if all airlines use the UDPP tools in an efficient way.
ISTOP		Developed in BEACON, it uses information akin to what is passed to UDPP to rebuild an approximated normalised cost function and suggests suitable slot swaps among airlines.
NNBound	Non-negative bounded optimum	Uses the true cost functions of the airlines, it finds the best allocation, given that no airline loses in terms of cost. This is used for benchmarking, since it represents the maximum efficiency if local “fairness” is enforced (nobody loses).
GlobalOptimum	Unbounded optimum	Uses the true cost functions from the airlines, it finds the best allocation overall. This is used for benchmarking, since it represents the maximum theoretical efficiency reachable by any mechanism.
Credit mechanism (NEW)		For each flight, the airline approximates the cost function with a step function. The parameters of the step function fix the price that the airline has to pay in virtual credits for this flight.

Table 1: Main principles.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

The new algorithm was designed to extend naturally the ISTOP mechanism while getting closer to the global optimum. Indeed, while in ISTOP airlines give normalised cost functions to the central optimiser (see D5.1 for more details), in this mechanism the airlines can provide absolute cost functions. To mitigate the issue of airlines inflating their cost in order to have a better allocation, airlines need to pay for these cost functions.

More specifically, each cost function is approximated by a step function, with two parameters: the position and the size of the step (in minutes and euros respectively). These parameters, if they deviate from default "free" parameters, may require the airline to spend some of its virtual credits, endowed at the beginning of the simulation to each airline. For instance, the higher the jump (i.e. the higher the cost of the flight if this flight is allocated in this time region), the higher the price is going to be for the airline.

The airlines can also gain some credits by "relaxing" their cost parameters, for instance reducing the size of the jump. This should allow in particular airlines with a small number of flights in the regulation to gain some credits from the mechanism, in order to convert it into an advantage in a future hotspot. This particular point is crucial and was flagged by the project PJ07.02 as a desirable future requirement for new mechanisms.

Note that this mechanism was also designed to be fairly close to ISTOP, which is turn signed to be close to 'current' mechanisms like Margins. Indeed, it was found to be quite intuitive for airlines to set time limits after which their cost explode, as well as the cost incurred.

2.2.3 PJ01.01 - Flight Arrival Coordination

For a description of the algorithm developed during the third iteration of NOSTROMO, but whose results are not included in the present deliverable, please see Annex XXX.

2.2.4 Simulations

We use the 12th of September 2014 to calibrate the model. Just like for the second iteration, we consider only the flights to and from Charles De Gaulle airport – with all corresponding information on passenger connections and turnaround times.

In order to answer the research questions and find interesting relationship between variables – harder to model and thus more interesting from the metamodelling point of view – we selected the input variables shown in Table 2 to be studied by the metamodelling process. The table includes the possible values, practical range, and default values of the parameters, all used during the metamodelling process.

Note that some of these parameters are calibration parameters, i.e. that they should be fixed at some point before a potential user runs the simulations. We included them to assess how active learning could help calibrating this sort of parameters in the model.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

Variable	Name in model	Description	Theoretical range	Practical range	Default
Mechanism for hotspot solving	hotspot_solver	Type of mechanism in the hotspot	['UDPP', 'UDPP+ISTOP', 'UDPP+ISTOP_TRUE','CM', 'NNOUND_TRUE', 'NNOUND_APPROX', 'GLOBAL_TRUE', 'GLOBAL_APPROX', 'CM']	NA	'UDPP'
Initial number of credits	initial_credits	Initial numbers of credits for CM	[0, Infty]	[0, 2000]	200
Price jump	price_jump	Price of one euro of jump in CM	[0, infty]	[0, 0.1]	0.01
Price margin	price_margin	Price of one minute of margin in CM	[0, infty]	[0, 10]	1
Default margin	default_margin	Free margin in CM	[0, infty]	[0 - 60]	15
Default jump	default_jump	Free jump in CM	[0, infty]	[0 - 20000]	3000
Reinjection	reinjection	Ratio of credits reinjecting into CM on top of honest parameters	[0, 1]	[0, 1]	0.1
Fuel price	fuel_price	Price of one kg of fuel - Continuous	[0, infty)	[0, 5]	1
Turn-around time scale	alpha_tat_mean	Scaler of mean of the distribution of turn-around times	[0, infty)	[0, 10]	1
Minimum connecting time scale	alpha_mct	Scaler of mean of the distribution of passenger minimum connecting times	[0, infty)	[0, 10]	1
Claim rate	claim_rate	Proportion of passengers	[0, 1]	[0, 1]	0.14

Co-funded by the European Union

		claiming compensation			

Table 2: input variables.

The performance indicators used for the third iteration are shown in Table 3. They are very much focused on the performance of the mechanism with regard to the hotspot resolution.

Variable	Name in model	Comment	Priority
Saved REAL cost in regulation w.r.t. FPFS allocation	ratio_cost	Average of ratio	1
Saved DECLARED cost in regulation w.r.t. FPFS allocation	ratio_cost_approx		3
Absolute equity: 1 - average of pair-differences in cost saved w.r.t to FPFS	equity		1
Relative equity: 1 - average of pair-differences in ratio cost saved w.r.t to FPFS/cost in FPFS	equity 2		2
Number of credits in average at the end of the simulation	credits		2
Gini coefficient for the distribution of credits at the end of the simulation	credits_gini		2

Table 3: output variables.

Co-funded by the European Union

3 Metamodelling process

This section briefly reviews the metamodelling process underlying NOSTROMO's methodology for ATM performance assessment, including the core concepts and features essential to its understanding. We collect revised material from the previous deliverables, including D3.1 Preliminary Metamodeling Methodology [3], D3.2 Metamodels Requirements Specification [4], D3.3 NOSTROMO Framework API + Associated Documentation [5], and D5.1 ATM Performance Metamodels - Preliminary Release [1].

Due to their overwhelming complexity, most ATM systems can only be modelled and studied through simulation. Despite the well-known advantages and features in modelling complex and stochastic real-world systems and their evolution over time, simulation-based solutions are not exempt from shortcomings.

As pointed out in [6], besides being traditionally highly costly to develop (depending, of course, on the degree of detail, realism and objectives), for every computer run, stochastic simulators only generate output estimates for a given set of input/parameter values. It is often the case where several independent runs are required to obtain a more reliable description of the full output distribution. Thus, different combinations of input/parameter values inevitably and accordingly lead to the systematic repetition of the entire experimental process so that statistical significance can be attained. Furthermore, suppose the simulation average computation time per single run is non-negligible (e.g. several minutes). In that case, we can trivially see that the exploration process of the entire input/parameter space, followed by observing the corresponding output behaviour, can be easily hindered. Even within magnitudes of several minutes, a simple study can be extremely time-consuming if not, in some extreme cases, virtually impossible to conduct in a timely manner.

To tackle the challenges mentioned above, simulation metamodels [7,8,9] can be employed to functionally approximate simulation models and then used as fast modelling proxies for the latter. Often characterized by mathematical simplicity and computational speed, metamodels allow for a reasonable number of expensive and/or redundant computer experiments to be bypassed, thereby leading to considerable time savings during the simulation exploration process. Additionally, the challenges generally posed by expensive simulation runs are akin to those where labelled data is not readily available and expensive to acquire. In such scenarios, active learning [10,11] emerges as a powerful learning paradigm aiming at attaining high prediction performance with as few data points as possible. Within it, a tuple encompassing the simulation input values and the output result (label) can be regarded as a single labelled data point. For simplicity, even though we only address metamodelling from a regression point of view, we decided to maintain the same terms traditionally used in the active learning community, which mostly focus on class label classification problems [12].

In NOSTROMO, we adopt an integrated approach that employs active learning strategies on top of simulation metamodelling processes to reduce the overall computational burden often associated with time-consuming and systematic simulation analysis. Eventually, our ultimate goal is to provide ATM researchers and practitioners with a parsimonious auxiliary tool to explore the output behaviour of simulation models in a more insightful and computationally efficient manner. For more details on this methodology and its elementary constituents, please refer to the deliverables D3.1 [3] and D3.4 [2]. Note, however, that we do not seek, nor it would be wise, to completely dismiss the simulation model after the corresponding metamodel is obtained. On the contrary, our ultimate goal is to deploy active learning metamodelling along with the simulator as a bundled modelling framework, as both models play a crucial and complementary role within this integrated approach.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

Figure 2 depicts an overview of the metamodelling process underlying the results presented in the following sections.

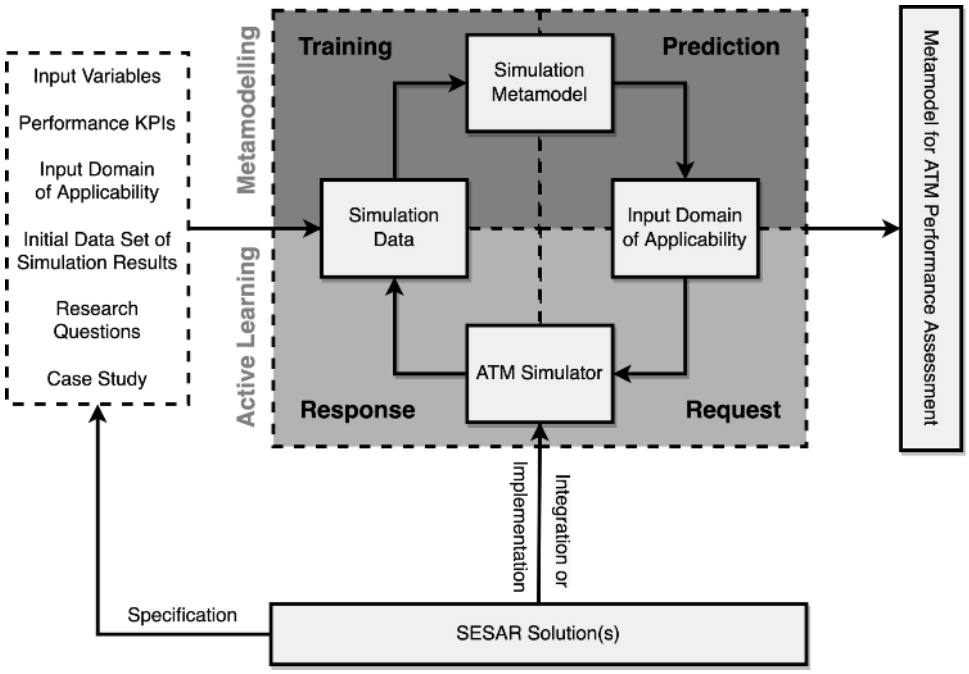


Figure 2: Active learning-based metamodelling process overview for ATM performance assessment.

Several requirements should be satisfied before proceeding with this methodology. The first step is to define the research questions and the case study, which are inevitably linked to the SESAR Solutions we aim to assess. From a modelling perspective, this will lead to the specification of the simulation input variables and performance KPI (outputs) of interest and, consequently, the input domain of applicability in which we want to explore the simulator's behaviour. Additionally, due to the iterative nature of active learning, an initial training data set should be built, essentially comprising a small set of simulation results generated according to a certain sampling strategy (e.g. Latin hypercube [13]) and based on the set of simulation variables we choose to work with.

Despite not being highlighted in Figure 1, it is equally crucial to define what kind of metamodel we plan to use, as it represents the modelling core of our approach. The Gaussian Processes (GP) regression framework [14], also known as Kriging [15], is a common choice for metamodelling [9], ultimately being our own within the project.

As we can observe from Figure 1, the NOSTROMO methodology can be essentially split into two alternating phases, namely, active learning and metamodelling, and globally comprised of four essential and sequential steps as follows:

1. **Training**: the metamodel is fitted to the simulation data. In the first iteration, this data corresponds to what we previously deemed as the initial data set.
2. **Prediction**: the fitted metamodel is used to predict over the simulation input region of interest.
3. **Request**: based on some acquisition criteria (e.g. maximum predictive variance), new unlabelled input data points are selected to be run by the simulator. Remember that

EUROPEAN PARTNERSHIP
Co-funded by the European Union

these points are comprised of simulation input points for which the corresponding true output values are unknown.

4. **Response**: the simulator provides new simulation output results corresponding to the points from step 3, which are then added to the current training set.

Steps 1-4 are repeated cyclically until a stopping criterion is satisfied. This criterion can be defined, for example, as a function of the metamodel's performance, such as accuracy or error reduction, or simply the number of iterations to be performed with respect to the available time, budget and resources.

Remember that the approximative nature of the metamodel calls for careful handling of the trade-off between speed, accuracy and computational budget. It is important to recognize and identify the performance threshold from which the mere addition of new training points will not significantly improve the ability of the metamodel to approximate the simulation results. In those cases, and especially from a metamodelling perspective, requesting more simulation results might be a waste of computational resources.

As a final note, it is always important to reiterate that is it not our objective to discard the underlying simulation model under study but instead to complement it with an additional auxiliary tool that is able to efficiently explore its behaviour across its input-output space and easily identify and highlighting important relationships between the involved variables.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

4 Metamodelling results

4.1 FLITAN

In this section, we detail the translation layer for Flitan's categorical variables and present some results regarding the integration of SESAR Solutions PJ.08-01 and PJ02-08. It is worthwhile mentioning that the metamodel has no connection to the combination of both solutions, whose implementation was independently conducted within Flitan prior to the metamodelling process. From a modelling perspective, the integration of these Solutions ends up, in practice, corresponding to a new version of Flitan, i.e., to a new simulator with additional variables.

4.1.1 Data Encoding

As previously mentioned and discussed in D5.1 [15], most machine learning metamodels are not adequate to model Flitan's input data due to its categorical (string-based) type. Furthermore, the data used by Flitan is scattered across multiple log files. For these reasons, important extra steps encompassing data collection, conversion, and merging are required prior to the metamodelling itself. We can see these steps as part of the design of "translation layer" mentioned in D3.3 [3] and D3.4 [15].

Although both mentioned SESAR Solutions have now been combined and integrated, their data conversion processes, especially concerning their corresponding input data, are similar and can be addressed separately.

PJ.08-01 Solution

This Solution regards the implementation of Dynamic Airspace Configuration (DAC). At the core of this Solution is the concept of the opening schemes - daily schedules of airspace sector configurations - and their corresponding flight delays as an output performance indicator. For a single operation day and fixed traffic demand, the idea was to establish a relationship between the opening scheme and configuration delay via metamodel and eventually predict the latter and allow a more efficient exploration of the simulator's behaviour while minimizing the corresponding computational burden.

An arbitrary airspace configuration can be broken down into a set of combined sectors, each of which is further composed of elementary sectors, as presented in Table 4.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

Center	Unit	Configuation Name	Configured Sectors	Elementary Sectors
LECB	LECBCTAE	1A	LECBBKE	LECBCCL+LECBCCU+LECBMMS+LECBMNL+LECBMNU+LECBVNI+LECBVVS
LECB	LECBCTAE	2A	LECBCVN LECBMVS	LECBCCL+LECBCCU+LECBVNI LECBMMS+LECBMNL+LECBMNU+LECBVVS
LECB	LECBCTAE	3A	LECBCCC LECBMMI LECBVVI	LECBCCL+LECBCCU LECBMMS+LECBMNL+LECBMNU LECBVNI+LECBVVS
LECB	LECBCTAE	3B	LECBCCC LECBMNI LECBVMS	LECBCCL+LECBCCU LECBMNL+LECBMNU LECBMMS+LECBVNI+LECBVVS

Table 4: Example from the Josep Tarradellas Barcelona-El Prat Airport, East Air Traffic Control, relating the name of the configuration, configured sectors and elementary sectors.

The basic idea behind PJ.08-01's encoding within Flitan is that we assume each configuration can be mapped into a set of possible configured sectors. This set does not comprise all the theoretically possible combinations of elementary sectors but those most frequently used within the historical data acquired explicitly for this project. As expected, each cntrol unit is associated with a distinct set of configured sectors and, consequently, we propose each be separately modelled by a different metamodel. Additionally, each unit has its own pre-defined activation time periods across which the different available configurations can be permutated.

The conversion process from categorical to dummy variables, regarded as a feature engineering step, has been described previously in D5.1 [15]. For more details, please refer to this deliverable.

PJ02-08 Solution

To implement this Solution, Flitan introduced a runway configuration manager and runway-dependent operations functions into the simulation engine for optimal runway configuration planning while enhancing its sequencing functions to balance arrival flights and departure flights on single, dependent or parallel runways. The delay is also used as a performance metric as in the previous Solution.

Table 5 presents a data log sample produced by Flitan at the runway configuration level for airports LEBL, LEMD and LEPA. The different configurations available per airport are named after the main cardinal directions. Within the same airport, only one configuration can be active for a given time period. The departure and arrival delays, and their sum, are used as throughput performance metrics.

Co-funded by the European Union

Airport	Configuration	Start time	End time	Depart. delay	Arrival delay	Sum delay
LEBL	WEST	2019-06-23T05:05	2019-06-23T22:46:59	43136	65635	108771
LEBL	NORTH	2019-06-23T22:47	2019-06-23T23:59:59	0	7609	7609
LEBL	NORTH	2019-06-23T00:00	2019-06-23T05:04:59	4409	0	4409
LEMD	NORTH	2019-06-23T00:00	2019-06-23T08:56:59	1331	0	1331
LEMD	SOUTH	2019-06-23T08:57	2019-06-23T23:59:59	11535	12686	24221
LEPA	WEST	2019-06-23T14:16	2019-06-23T18:06:59	10641	14622	25263
LEPA	EAST	2019-06-23T18:07	2019-06-23T23:59:59	2246	6732	8978
LEPA	EAST	2019-06-23T00:00	2019-06-23T14:15:59	29243	21480	50723

Table 5: An example of a runway configuration data log produced by Flitan.

Combination of PJ.08-01 and PJ02-08 Solutions

The metamodel is agnostic to how the solutions were combined and implemented within the simulator. From the metamodelling perspective, this new version of Flitan (3dr iteration), integrating both PJ.08-1 and PJ02-08, is regarded as a different simulator with a refreshed set of input and output variables. The new input data is a combination between the encoded opening schemes at the FMP level and the runway configurations at the airport level. Because time is an important component of Flitan's input data, we concluded that, from a metamodelling perspective, it would make sense to use the same activation timeslots for both DAC and runway configurations. By default, Flitan's input data was using different timeslots per control centre and airport runways. Moreover, we have concluded that, from a metamodelling perspective, it would not make sense to integrate control centres with airports that are not contained within them (for example, combining runway configuration plans from Lisbon airport with Barcelona's DAC). To this end, we forced the control centers' DAC timeslots, which are independently fixed, into the corresponding airport runway configuration activation plans. Therefore, we ended up focusing on the following combination of centres and airports:

- LECPCTA (Palma de Mallorca, Spain) + LEPA (Palma de Mallorca Airport)

EUROPEAN PARTNERSHIP

Co-funded by the European Union

- LECMCTAN (Madrid North, Spain) + LEMD (Adolfo Suárez Madrid–Barajas Airport)
- LECMCTAS (Madrid South, Spain) + LEMD (Adolfo Suárez Madrid–Barajas Airport)
- LECBCTAE (Barcelona East, Spain) + LEBL (Josep Tarradellas Barcelona-El Prat Airport)
- LECBCTAW (Barcelona West, Spain) + LEBL (Josep Tarradellas Barcelona-El Prat Airport)

This yields five different metamodels, each modelling a different control centre-airport pair. Table 6 shows an illustrative sample of the integrated data used by the metamodels.

Center /Unit	Air Space Conf.	Start time	End Time	LECP GIX	LECP F2X	LECP L2E	LECP APR	…	DAC Delay	RW Config	Arriv Delay	Depart Delay	Sum Delay
LECPCTA	2A	09/07/2019 03:05	09/07/2019 03:59	0	0	0	1		166	EAST	0	0	0
LECPCTA	6A	09/07/2019 04:00	09/07/2019 04:59	0	1	0	0		339	WEST	1393	0	1393

Table 6: Illustrative sample used by the metamodels integrating the encoded data from PJ.08-01 and PJ02-08, using control until LECPCTA and the corresponding airport LEPA.

Remember that, as seen in D5.1 [15], the airspace configurations were not processed via one-hot encoding. Instead, the used process translates the idea that an arbitrary air space configuration, for a given centre, is defined as a fixed and unique combination of active configured sectors (LECPGIX, LECPF2X, etc.). Those configured sectors that are active (or open) are represented by “1“, and “0” is trivially assigned to those that are inactive (or closed). Since a certain air space configuration is described as a combination of at least one active configured sector, the sum of all “1”s is always equal to or greater than 1. Similarly, the runway configuration must also be converted to numerical values. One-hot or ordinal encoding can be easily adopted as conversion methods.

Note that this integrated encoding presented herein is not unique and merely pertains to a proposal within this project. Other encodings could have been considered and should be explored in the future. Ultimately, any adopted encoding will be conditioned by the nature of the simulation data and the underlying simulators, as well as on the objectives of the simulation study.

Co-funded by the European Union

4.1.2 Results

This section focuses on the results obtained for LECPCTA (Palma de Mallorca, Spain) + LEPA (Palma de Mallorca Airport). Within this experimental setup, each simulation day is, in terms of input data, comprised of 17 fixed and predefined timeslots, 25 DAC-related variables (total number of configured sectors), two variables encoding the runway configuration (East and West), and the date itself. In our setting, an individual data point is defined as an entire simulation day with 17 timeslots, encompassing both the opening scheme and the runway configuration daily plans, along with the corresponding delays. The simulation time window spans from 20/06/2019 to 17/07/2019. The resulting search input space is quite large and sparse due to the one-hot representations. Figure 1, Figure 4, and Figure 5 depict some results originating from three different experimental settings.

Contrary to the case of Mercury, the results obtained for Flitan were not as clear so far. Although overall, there is a clear increasing tendency as the number of active learning iterations increases, both random and variance-based strategies behaved similarly. We believe that one of the causes for this is indeed the sparsity of the (encoded) input space associated with high variance in the output space. Another reason might lie in the “artificially” generated output metric (DAC Delay + Sum Delay) inspired by the integration of both Solutions. Nevertheless, further investigation is required for future work.

Note, however, the difference in the RRSE scale when only the date is used (see Figure 4). This only strengthens the potential importance of including the total number of daily flights (demand) as an additional input feature.

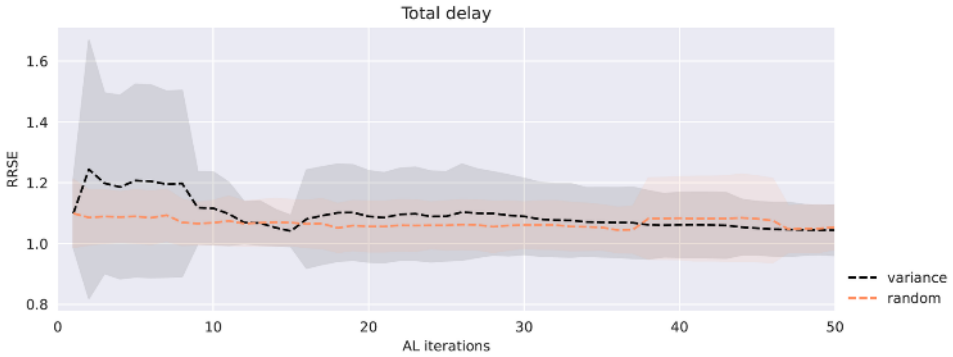
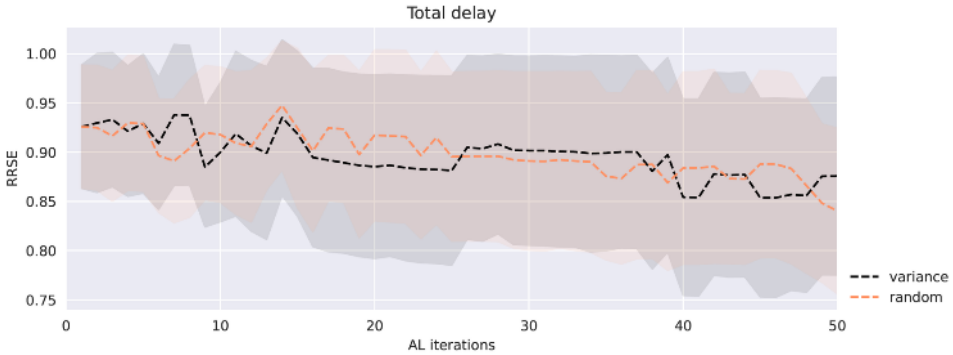


Figure 3: Gaussian Process as metamodel starting with 10 data points.



Co-funded by the European Union

Figure 4: Gaussian Process as metamodel starting with 50 data points and using only the date variable.

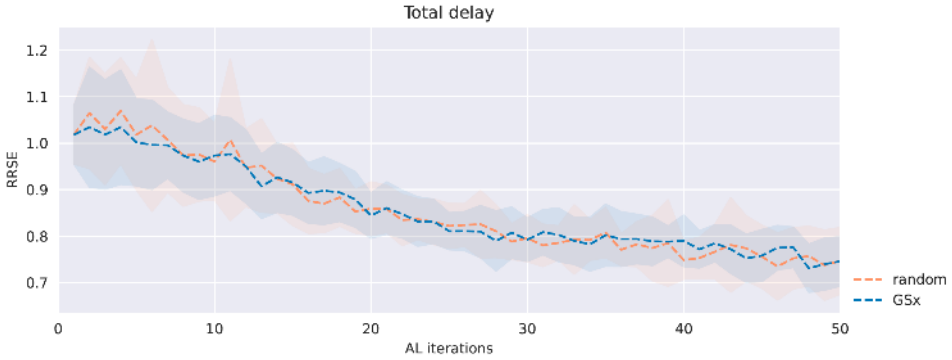


Figure 5: XGBoost as metamodel starting with 100 data points, and Greedy Sampling in the input space (GSx) using query batches of size 10.

4.2 Mercury

In this section, we report the active learning metamodelling results focused on two output metrics, namely, Absolute Equity and Saved Real Cost in regulation w.r.t. FPFS allocation. For regularization ad scaling purposes, both output variables were log-transformed by the function log(x+1). The used input variables are briefly described in Section 2.2. Figure 6 and Figure 7 summarize the evolution of metamodelling as the number of iterations increases. Within the two reported experimental settings, both random and variance-based active learning strategies were employed. We can conclude that, as expected, the metamodels' performance increases as more points are added to the training set. The greatest difference between the two strategies is twofold. The variance-based active learning metamodelling yields a much more significant and faster performance improvement in the first few iterations when compared to the random approach. Moreover, the performance variances associated with the metamodel within each iteration are clearly greater in the case of the latter. Note that individual metamodels were developed for each output, although both share the same input space comprised of 11 input variables.

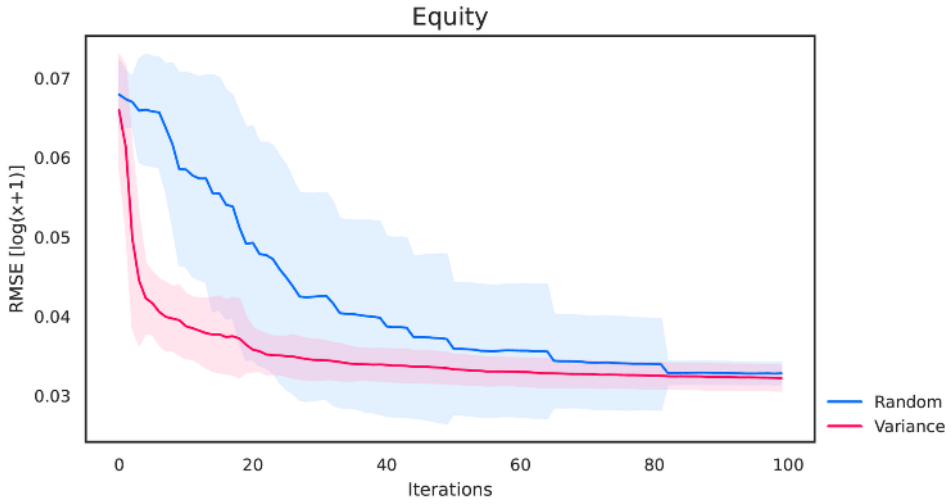


Figure 6: Active learning metamodelling performance for Absolute Equity.

Co-funded by the European Union

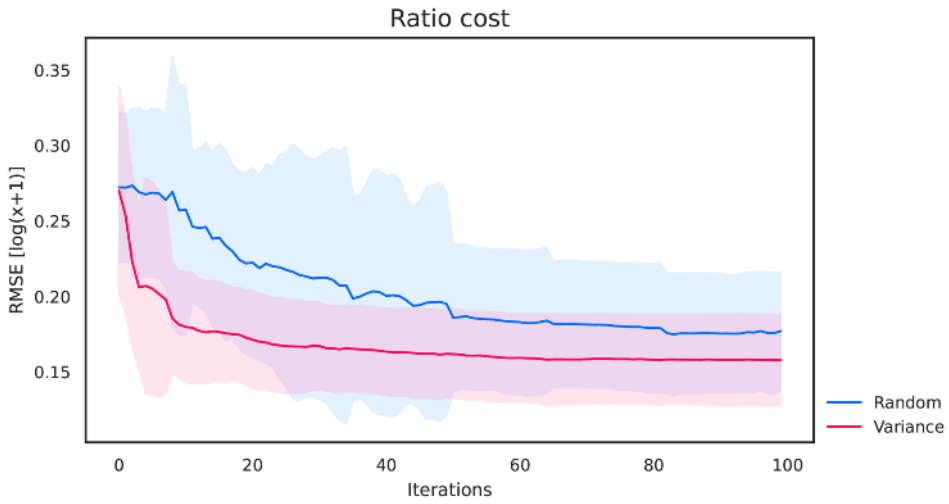


Figure 7: Active Learning metamodelling performance for Saved Real Ratio Cost.

Co-funded by the European Union

5 References

[1] Nostromo: Next-generation Open-Source Tools for ATM performance Modelling and Optimisation (Q1 2020). Preliminary Metamodelling Methodology, Deliverable D3.1, WP3.

[2] Nostromo: Next-generation Open-Source Tools for ATM performance Modelling and Optimisation (Q2 2020). Metamodels Requirements Specification, Deliverable D3.2, WP3.

[3] Nostromo: Next-generation Open-Source Tools for ATM performance Modelling and Optimisation (Q4 2020). NOSTROMO Framework API + Associated documentation, Deliverable D3.3, WP3.

[4] Law, A. M. (2015). Simulation modeling and analysis (5th Ed.). New York: Mcgraw-hill.

[5] L. W. Friedman, The simulation metamodel. Springer Science & Business Media, 2012.

[6] Jack PC Kleijnen and Robert G Sargent. A methodology for fitting and validating metamodels in simulation. European Journal of Operational Research, 120(1):14–29, 2000.

[7] Gramacy, R. B. (2020). Surrogates: Gaussian process modeling, design, and optimization for the applied sciences. Chapman and Hall/CRC.

[8] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[9] Xizhao Wang and Junhai Zhai. Learning with Uncertainty. CRC Press, 2016.

[10] Kumar, P., & Gupta, A. (2020). Active learning query strategies for classification, regression, and clustering: a survey. Journal of Computer Science and Technology, 35(4), 913-945.

[11] Nostromo: Next-generation Open-Source Tools for ATM performance Modelling and Optimisation (Q3 2022). NOSTROMO Final Metamodelling Methodology, Deliverable D3.4, WP3, to be submitted.

[12] Viana, F. A. (2016). A tutorial on Latin hypercube design of experiments. Quality and reliability engineering international, 32(5), 1975-1985.

[13] Williams, C. K., & Rasmussen, C. E. (2006). Gaussian Processes for Machine Learning. Cambridge, MA: MIT Press.

[14] Chilès, J. P., & Desassis, N. (2018). Fifty years of kriging. In Handbook of mathematical geosciences (pp. 589-612). Springer, Cham.

[15] Nostromo: Next-generation Open-Source Tools for ATM performance Modelling and Optimisation (Q2 2022). D5.1 ATM Performance Metamodels - Preliminary Release, Deliverable D5.1, WP5.

6 Acronyms

Acronym	Definition
ATM	Air Traffic Management
BPM	Business Process Modelling
ANSP	Air Navigation Service Providers
ATCO	Available Air Traffic Controller
FMP	Flow Management Positions
AOP	Airport Operation Plan
NM	
DDR	Demand Data Repository
AIRAC	Aeronautical Information Regulation and Control
ATFM	Air Traffic Flow Managament
IFPS	Integrated Initial Flight Plan Processing System
KPI	Key Performance Indicator
FMP	Flow Management Positions
NOP	Network Operation Plan
GP	Gaussian Process

EUROPEAN PARTNERSHIP

Co-funded by the European Union

Appendix A Mercury Simulator

Mercury is a simulator developed over several years which is able to produce detailed network-wide performance assessment, in particular regarding passenger mobility in Europe.

Mercury is implemented as an event-driven simulator. The underlying model can be seen as a Monte Carlo simulation, sampling distributions (delays, missed connections, etc.) based on causal rules representing actual processes of the air transportation system (e.g. if passenger delay is bigger than a given threshold, an airline incurs costs for compensation and assistance to the passenger). It can also be seen as an agent-based model, and this paradigm has been followed closely when it was designed. A series of agent instances sends messages back and forth and reacts to events. Their memory is private, i.e. they have attributes that cannot be accessed by other agents. This opens the way to modelling imperfect knowledge and eases the implementation of rules of thumb and approximated decision-making processes, more realistic than full 'hyper-rational' agents in general.

The scope of the simulator is to model individual aircraft throughout one day of operation, including turnaround processes, tracking the passengers on board, as well as passenger connections. The passengers in Mercury are modelled through passenger processes, simulating for instance connecting times for individual passengers, connecting options, etc. The flight is described in terms of times it takes to complete different processes (taxi, take-off, cruise, etc). The simulation of the en-route phase is approximated using actual flight plans and delay distributions. The fuel consumption can be assessed with quite a good precision, thanks to BADA models, but a full trajectory optimiser is not included in this version. Hence, the simulator relies heavily on historical flight data to sample navigation times FOOTXXX. FOOTXXX: a trajectory optimiser is in general required when one wants to modify an aircraft trajectory. For instance, modifying the speed of the aircraft in general changes the descent profile. In NOSTROMO we are relying on past data instead (for this second iteration), using similar aircraft that underwent similar changes in order to rebuild trajectories. Note that in the third iteration of the model, another process will be used, based on BADA4 model sampling.

Different types of agents are present in the system, sometimes instantiated multiple times (e.g. airline operating centre), sometimes once (e.g. network manager). We describe the most important ones succinctly in the following.

Airline Operating Centres (AOC)

This agent is the most important in the simulation and the most advanced. It is tasked with following its flights and passengers, making decisions when disruptions hit, providing information to the Network Manager agent if needed, etc.

AOC decision making-process is based on usage of detailed cost functions, estimated using [13] and representing the best guess we have on the costs to airlines. The costs include fuel,

Co-funded by the European Union

explicit passenger direct costs (compensation/duty of care), implicit maintenance and crew costs, curfew costs, etc. The cost function of a flight (the cost as a function of the arrival time for instance) is deterministic and features typically various 'jumps', corresponding to passengers missing their next connections or curfew infringement (potentially on the next rotations).

The cost function is used for different processes, like the two mechanisms described thereafter.

Flight

The flight agent is tasked with estimating different phases of flights, drawing randomly delays where needed, based on historical distributions. It is mainly communicating with its AOC. For instance, the AOC may provide to the flight its cost function to be able to make informed decision on speed in advanced mechanisms.

Ground Airport

The ground airport takes mostly care of two processes. First, it manages the connecting passengers, drawing random values for their connecting times, sampled from a distribution calibrated on past data. This connecting time is compared to the minimum connecting times, available for each airport and type of connection (national-international, international-international, national-national). Second, it does the same for the aircraft, drawing at random values for their turnaround times. This distribution is also calibrated on past data. The departure delay is then applied to the next flight if the turn-around time is too high.

DMAN and AMAN

These agents track explicitly the departures and arrivals at their airport and ask for ATFM regulations if the traffic needs to be regulated. Depending on their nature (see new mechanisms thereafter), they may communicate to the flights en-route to ask them to change their speed.

Network Manager

The network manager is responsible for considering flight plan submissions from the airlines and managing ATFM regulations. It checks with DMAN and AMAN if the estimated traffic would surpass the capacity in their respective airports and arranges for flights to be delayed, according to different rules (see mechanisms thereafter). Note that on top of regulations for traffic, regulations can be randomly sampled from historical data, to simulate loss of capacity due to other events, like weather.

Other considerations

Passengers are bundled in groups that share common itineraries, and these groups are handled by the other agents, like the AOC and the ground airport. These groups are dynamic, i.e. in case of missed connections they can be split if their respective passengers need to board different planes. Note that in this simulator the passengers do not have to make any decision, and are thus not considered as agents.

Finally, we highlight two important limitations of the simulator:

The crew is not explicitly considered by the model. In other words, the airline does not take into account crew rotations. It only considers an approximated cost for the airline to operate with delay, which takes into account the average cost of dealing with disrupted crew processes. This is due to the difficulty to get historical crew rosters from airlines.

Apart from the runways themselves, there is no explicit model of the airspace. In other words:

the model does not track through which sectors a flight is going, and thus cannot issue regulations based on traffic on these pieces of airspace. En-route ATFM regulations are thus simulated as random delays.

no tactical traffic control takes place. We do not model controllers, and the trajectory is not modified based on their potential actions. Instead, we used distribution of “delays”, which modify the flight times between navigation points, extracted from historical data.

As highlighted above, the simulator relies heavily on historical data, that it uses in particular to create and then sample various distributions. The results in this deliverable have been obtained with a calibration of the model on the 12th of September 2014, for which we have all the required data, and the data sources are shown in table XXXXX.

Data source	Main usage	Reference
DDR2	Used to get the set of flights, origin-destination, routes, aircraft type, estimated cruise wind, distributions on climb and descent profiles, requested nominal cruise speeds and flight levels, companies, alliances, airspace structure, ATFM regulations	[12]
Cost of delay report	Used to compute cost of delay function	[13]
IATA Summer Season 2010 from CODA	taxi times	[14]
DDR2	minimum turnaround times, minimum connecting times	[15]
CODA	non-ATFM delays	[16]
Paxis, GDS	For passenger itineraries, including fares and class	[15]

Co-funded by the European Union

Data source	Main usage	Reference
Innovata (Cirium)	Flight schedules, for scheduled times of arrival and departure	-

For the third iteration of the model (presented in D5.2), the project will use the same kind of data, updated to match the 14th of September 2018, thus provided a more up to date estimation of the KPIs.

More details on the Mercury simulator can be found in [10].

Co-funded by the European Union

Appendix B Flight arrival coordinator

This section develops new mathematical models for the fight arrival coordinator problem. The objective is trying to start arrival traffic sequencing earlier than is the case by AMAN in the concept of E-AMAN. The E-AMAN extends the horizon to have smoother traffic management near the airport area. This process would reduce the delay, operational costs, CO2 emission, and smooth delivery of arrival traffic to the runways. We suggest dividing the space near the airport into three different horizons as follows (see Fig 1):

- Data Horizon (DH) = 600 NM
- Command Horizon (CH) = 500 NM
- Tactical Horizon (TH) = 100 NM

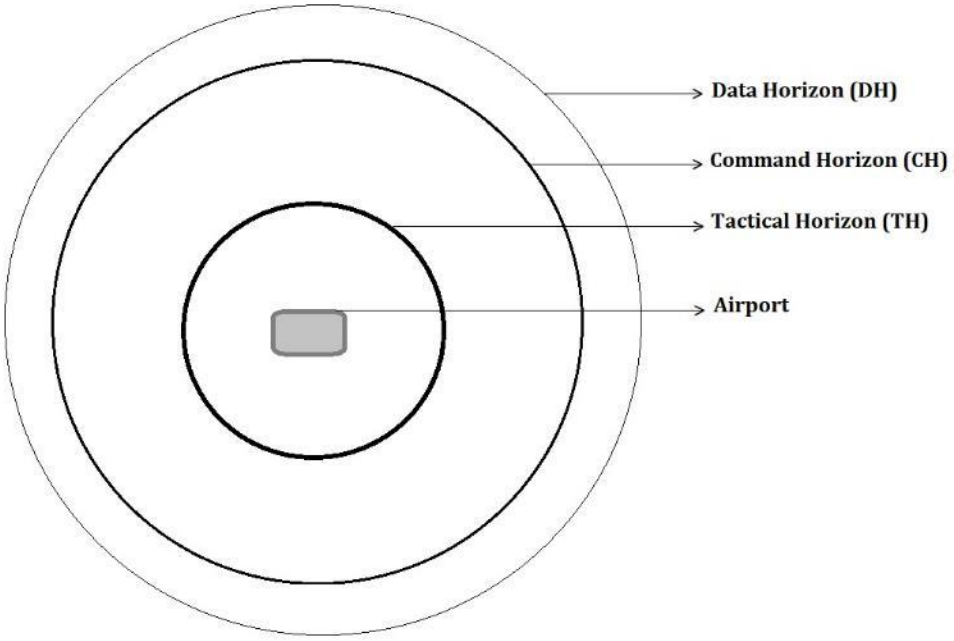


Fig 1: Three different horizons near an airport

Here we propose an E-AMAN model that extends to 500NM for CH. When the flights hit Data Horizon (DH), those flights' intentions are sent to the E-AMAN. Once the target flight enters Command Horizon (CH), the mathematical model (or optimiser) is run to find an optimal slot for that flight at the runway. Here, we consider the other flights within DH in optimising. Compared to previous optimisation processes, the E-AMAN takes into account the cost of delay and fuel for the airline instead of delay alone and uses information on the distribution time of arrival to manage uncertainty (e.g., due to wind). Based on the optimal slot assigned, the E-AMAN issues a command to the flight to maintain “Initial Speed”, “Speed Up”, or “Slow Down”. It also assigns minutes of holding if the delay cannot be absorbed during the cruise phase.

Co-funded by the European Union

Basic model description

In the basic model, we suppose that each slot's size equals two minutes. This part aims to demonstrate the trade-off between two different objectives that could be considered in the objective function. The first one is "Delay", and the second one is "Imbalance", which is the sum of flight probabilities that arrive in a specific slot. We use the information on different flights via the website of http://www.flightradar24.com

. We record the arrivals to London Heathrow Airport on several days. Then select 395 flights and 538 slots as an instance. We record the probability distributions of different flights.

We develop the basic models below to solve the problem:

Min "Delay"	Min "Imbalance"
$Min\ Objective\ Function = \sum_{i=0}^{Flights} \sum_{j=E_i}^{Slots} (j - E_i) \times X_i^j$	$Min\ Objective\ Function = \sum_{j=0}^{Slots} Z_j$
$\sum_{j=0}^{Slots} X_i^j = 1 \qquad \forall i \in Flights$	$\sum_{j=0}^{Slots} X_i^j = 1 \qquad \forall i \in Flights$
$\left(\sum_{i=0}^{Flights} Y_i^j - certain\ value\right) \leq Threshold\ value\ \ for\ Imbalance \quad \forall j \in slots$	$\sum_{i=0}^{Flights} \sum_{j=E_i}^{Slots} (j - E_i) \times X_i^j <= Threshold\ value\ \ for\ delay$
$Y_i^{j+t} \geq (X_i^j - 1) + A_i^{E_i+t} \quad \forall i, t, and\ j \geq E_i$	$Y_i^{j+t} \geq (X_i^j - 1) + A_i^{E_i+t} \quad \forall i, t, and\ j \geq E_i$
$X_i^j \in \{0,1\}, Y_i^j, Z_j \geq 0,$	$\left(\sum_{i=0}^{Flights} Y_i^j - certain\ value\right) < Z_j \quad \forall j \in slots$

Here the parameter E(i) is the number of the first slot where the probability distribution of flight i starts. We assume that each flight can take one slot at the start of its distribution. Also, in the right-hand model, the sum of delays must be lower than a threshold value for the delay, while in the left-hand model, the imbalance factor must be lower than the threshold value, and we assume that a certain value is equal to one here.

The decision variables, Y(i,j), determine the optimal probability distribution for flight i in slot j. The parameter is equal to the initial probability of flight i arriving at slot j, which we calculated through http://flightradar24.com

. We run the model on a laptop with the specification of the AMD Ryzen 7, 3700U, 2.30 GHz CPU, 16 GB RAM, and Python 3.

Co-funded by the European Union

Scenario	Threshold value	Objective 1: Delay	Objective 2: Imbalance	Run time (s)
1	0	0	18.73	336
2	5	5	17.43	360
3	8	8	16.76	368
4	12	12	16.03	375
5	20	20	14.9	395
6	30	30	13.83	399
7	40	40	12.96	943
8	50	50	12.13	1707
9	140	140	7.4	-
10	300	300	2.89	-

Table 7: Total results of basic model

We define ten scenarios to demonstrate that we face two conflicting objective functions: "Delay" and "Imbalance" indicators. In the curves below, the results are shown:

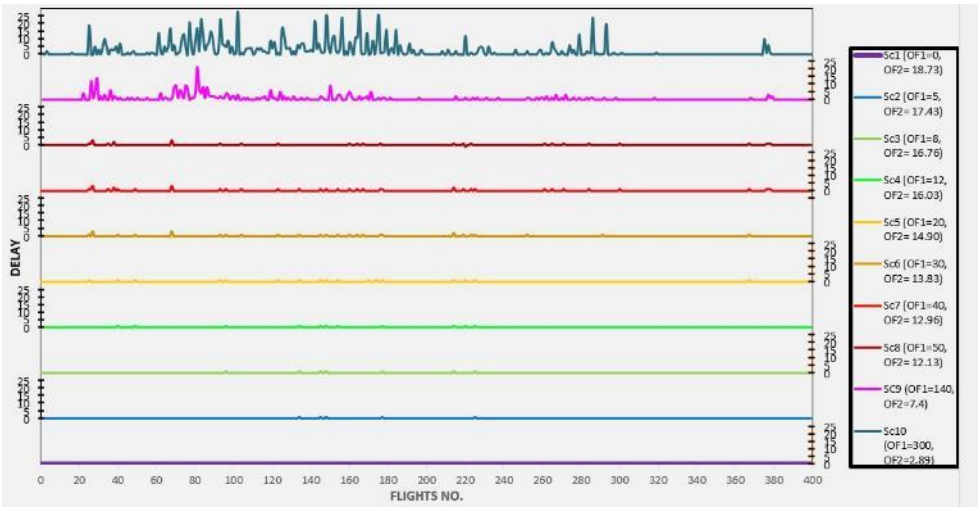


Fig 2: First objective function (Delay) under ten scenarios

Co-funded by the European Union

Fig 2 shows the results of the basic model under the objective function of delay. From scenarios 1 to 10, we are increasing the threshold values. From the perspective of "Delay", the best scenario is the first one, and the objective function is equal to 0, but the worse scenario is the last one with 300 slots of delay. In Fig 3, we analyse the model under the second objective function, the "Imbalance" indicator:

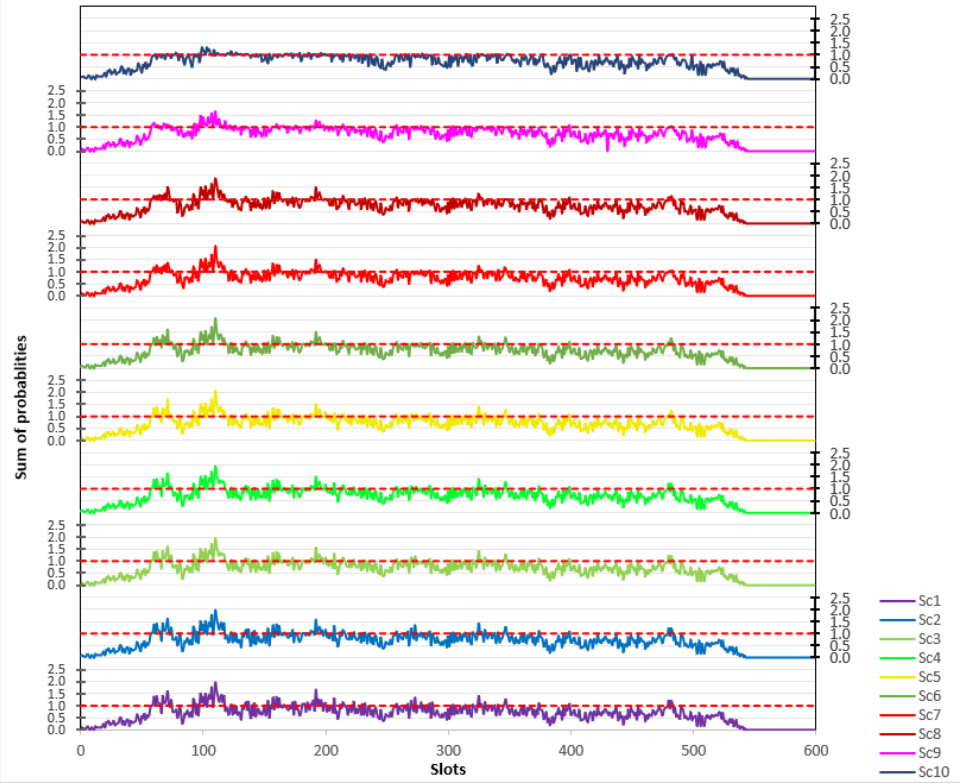


Fig 3: Second objective function (Imbalance) under ten scenarios

Here we assume the certain value is equal to one, which means that in the second objective function, Imbalance, we count the sum of probabilities of flights if that is greater than one in each slot. As mentioned in Fig 3, the best scenario is scenario ten, and the worse scenario is the first one we face, the more flights in each slot. These results demonstrate that the two objectives of "Delay" and "Imbalance" are conflicting objectives.

In the next part, we explain how we model the advanced model for the problem.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

Formulation of Advanced model

In this section, we develop new mathematical models for the fight arrival coordinator problem, explain how the mathematical model is developed, and optimiser works. Firstly, the sets, parameters and decision variables are introduced as follows:

Sets:

$i \in FNCD$	Index set of flights which are existing between DH and H
j $\in \{0,1,2,..,J\}$	Index set of available slots

Parameters:

$first\_slot_i$	First available slot for flight i to arrive if it takes maximum speed
$last\_slot_i$	Last available slot for flight i to arrive if it takes minimum speed without holding
$last\_slot\_hol_i$	Last available slot for flight i to arrive considering minimum speed and holding
max_hol	Maximum allowed holding for each flight
$demand_j$	The demand of slot j which can be computed based on the output of the previous run of the optimiser
$prob\_matrix_{i,j}$	The probability of that flight i arrives at slot j
$a_{i,h}$	The normalised probability of flight i arrive at time h
$w_{i,j,t}$	The probability of that flight i arrives at slot j+t for t $\in \{0,1,2,\dots,20\}$
$cost\_matrix_{i,j}$	The cost of slot j for the flight i
$lb_j, ub_j, size_j$	Lower bound, upper bound and size of slot j
$threshold\_value$	Threshold value for sum of probabilities for each slot
big_m	Large integer value

Decision Variable:

$X_{i,j}$	Binary decision variable takes one if flight i takes slot j as the start of its distribution, otherwise zero
$Y_{i,j}$	The probability of that flight i arrives at slot j after issuing a command
H_i	Holding required for flight i
$Cost\_Function_i$	The cost function of flight i

The parameters of the model a(i,h) and w(i,j,t), are computed based on the parameter prob_matrix(i,j). We define these two parameters because when a target flight is issued

Co-funded by the European Union

commands such as “Speed Up” or “Slow Down”, the distribution will be changed if the size of different slots is not equal. Therefore, firstly we normalised the probabilities based on the unit of minutes (matrix a) and then found a new matrix (matrix w) that could be used in the optimiser (see Fig. 4).

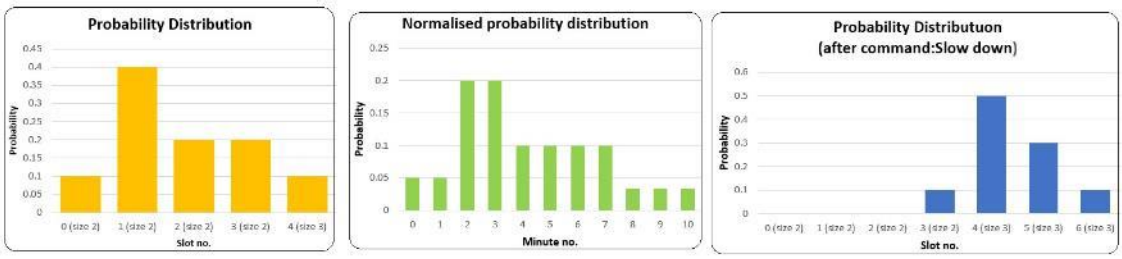


Fig 4: Calculate the probability distribution of Flight 1 after issuing the command “Slow Down”

In Fig. 4, the process of computing the matrix w is shown, and we supposed that flight i takes optimal slot 3 as the start of its distribution and is issued the command “Slow Down”. As the slot numbers between {0-4} and {3-6} are different in terms of size, the probability distribution is changed as input of the optimiser. Therefore, in the optimiser, we work with matrix w instead of parameter prob_matrix(i,j).

In following the new mathematical model for the flight arrival coordinator problem is presented:

EUROPEAN PARTNERSHIP

Co-funded by the European Union

$$Min \sum_{i \in FNCD} Cost\_Function_i \quad (1)$$

$$Cost\_Function_i = \sum_{j=first\_slot_i}^{J} cost\_matrix_{i,j} \times Y_{i,j} \qquad \forall i \in FNCD \quad (2)$$

$$\sum_{j=first\_slot_i}^{last\_slot\_hol_i} X_{i,j} = 1 \qquad \forall i \in FNCD \quad (3)$$

$$\sum_{j=0}^{J} Y_{i,j} = 1 \qquad \forall i \in FNCD \quad (4)$$

$$Y_{i,j+t} \geq big_m \times (X_{i,j} - 1) + w_{i,j,t} \qquad \forall i \in FNCD, j \in [first\_slot_i, last\_slot\_hol_i], j + t \leq J \quad (5)$$

$$Y_{i,j+t} \leq big_m \times (1 - X_{i,j}) + w_{i,j,t} \qquad \forall i \in FNCD, j \in [first\_slot_i, last\_slot\_hol_i], j + t \leq J \quad (6)$$

$$\sum_{j'=last\_slot_i+1}^{j} size_{j'} + big_m \times (X_{i,j} - 1) \leq H_i \qquad \forall i \in FNCD, j \in J \quad (7)$$

$$\sum_{i \in FNCD} Y_{i,j} + demand_j \leq threshold\_value \qquad \forall j, j \geq first_{slot_i} \ for\ all\ flights \in FNCD \quad (8)$$

$$X_{i,j} \in \{0,1\},\ Y_{i,j},\ H_i,\ Cost\_Function_i \geq 0 \qquad \forall i \in FNCD, j \in J \quad (9)$$

In equation (1), the sum of the cost function of flights between DH and CH is minimised. The cost function is defined as the flight's cost matrix at each slot multiplied by the probability of arriving at that slot in equation (2). In equation (3), we guarantee that each flight takes one slot as the start of its distribution, and the sum of probabilities must be equal to one for each flight between DH and CH regarding equation (4). The probability distribution for each flight is computed in equations (5-6). Based on these two equations, when the start of the distribution is determined, the probabilities are computed, taking their values from matrix w. equation (7) defines finding out the holding for each flight which happened when that flight could not take the slot only with the "Slow Down" command. Therefore, we consider holding as waiting for that flight. Regarding equation (8), we guarantee that sum of probabilities for each slot must be lower than the threshold value. At the same time, we consider those flights is issued in the optimiser's previous run as parameter demand. Finally, the decision variables are mentioned in the last equation, equation (9).

Here we present some examples of how the optimiser works.

Example 1) The probability distributions of two flights are shown here. We compute the sum of probabilities of two flights as below:

Co-funded by the European Union

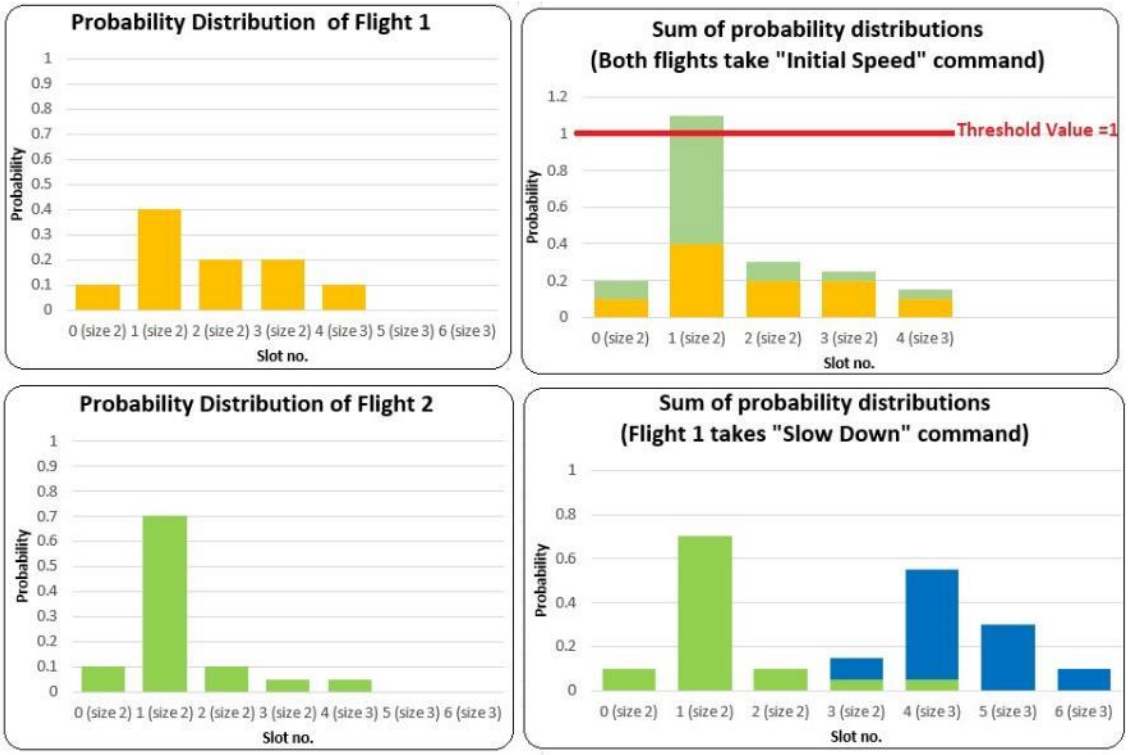


Fig 5: Sum of probabilities of two flights and the threshold value

With their probability distributions, two flight numbers, 1 and 2, are shown on the left side of Fig 5. If we issue the “Initial Speed” and suppose the threshold value is equal to one, the sum of probabilities would be greater than the threshold value in slot 1 (Fig 5, right up). However, if we issue the “Slow Down” command to flight 1 like Fig 4, the sum of probabilities would be lower than the threshold value (Fig 5, right down).

In the proposed model, we minimise the total cost considering the sum of probabilities of flights in each slot is lower than the threshold value.

Example 2) Suppose seven flights exist between DH and CH and Flight 6 (target flight) hits the CH. We run the optimiser to find out the optimal slots for these flights. In Fig 6, their probability distributions of them are indicated.

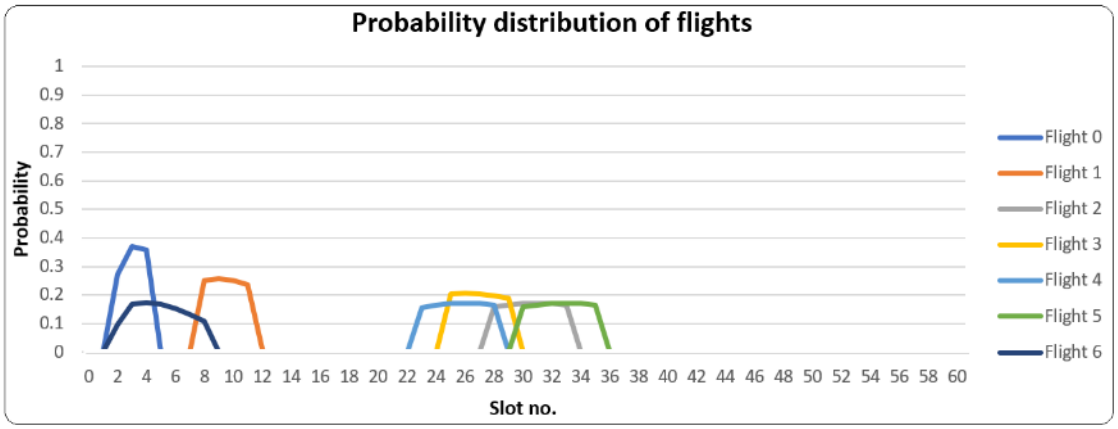


Fig 6: Probability distributions of seven flights between DH & CH

Here we suppose that the maximum allowed holding is 10 minutes, and the cost matrix for the first available and last available (with holding) slots are as follows:

Slot number	Flight 0	Flight 1	Flight 2	Flight 3	Flight 4	Flight 5	Flight 6
	Flight 0	Flight 1	Flight 2	Flight 3	Flight 4	Flight 5	Flight 6
0	843.3146						577.0506
1	79.40954						624.9372
2	24.62053						656.5097
3	27.86043						596.644
4	28.9612						588.321
5	30.07312	5.950961					595.6829
6	31.45583	6.97452					607.8146
7	32.60369	7.930028					623.6344
8	33.76431	8.889472					639.4523
9	34.93806	9.853					655.267
10	36.1253	10.82076					671.0772
11		11.79288					686.8819
12		12.76952					702.6798

Co-funded by the European Union

13		13.75078					718.4699
14		14.73681					
15		15.84731					
16		16.84827					
17		17.85439					
18							
19							
20					5.841493		
21					6.803682		
22				21.32351	7.726176		
23				22.52733	8.650965		
24				23.74359	9.578138		
25			25.41603	24.97236	10.50778		
26			44.90242	26.59115	11.43996		
27			64.39418	27.85732	12.37478	1249.689	
28			83.89168	29.13576	13.3123	1249.689	
29			103.3952	30.42627	14.25259	1249.689	
30			123.0437	31.7286	15.2655	653.8528	
31			142.5669	33.04244	16.21451	276.8866	
32			162.0971	34.36742	17.16653	99.31297	
33			181.6347	35.70311	18.1216	28.20443	
34			201.1797	37.04901		5.382184	
35			220.7325	38.89292		4.033688	
36			240.2932			4.991423	
37			259.862			5.953183	
38						7.013563	
39						7.98888	

Table 8: Total cost of flights regarding the first and the last available slots

Co-funded by the European Union

We use these inputs as a parameter in the mathematical model. We run the model in a laptop with the specification of the AMD Ryzen 7, 3700U, 2.30 GHz CPU, 16 GB RAM, and Python3 is used for running the optimiser. The result of the optimiser is indicated in Fig 7:

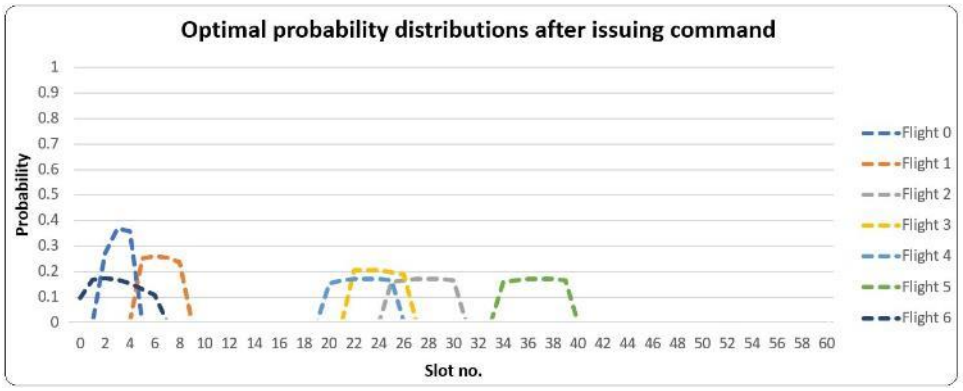


Fig 7: optimal slot assignment to seven flights and their new distribution

The result of the optimiser shows that the target flight (no. 6) must start from slot 0 instead of starting its distribution from slot 2. Therefore, the “Speed Up” command has been issued, taking the new speed of 7.464 instead of the previous speed of 7.157. The new speed is selected in the range of acceptable minimum and maximum speeds. The cost function for the target flight is equal to 609.7. Although we calculate the optimal slot assignment for all flights between DH and CH in each run of the optimiser, we issue the command only to the target flight. The reason behind that is the possibility that in the next run of the optimiser (when a new target flight hits CH), the previous optimal slot for a flight is not optimal anymore. Therefore, we count the flights between DH and CH in each optimiser run and issue the command to target flights.

Co-funded by the European Union

Co-funded by
the European Union