UNIVERSITY OF WESTMINSTER

June 2024

Mercury workshop

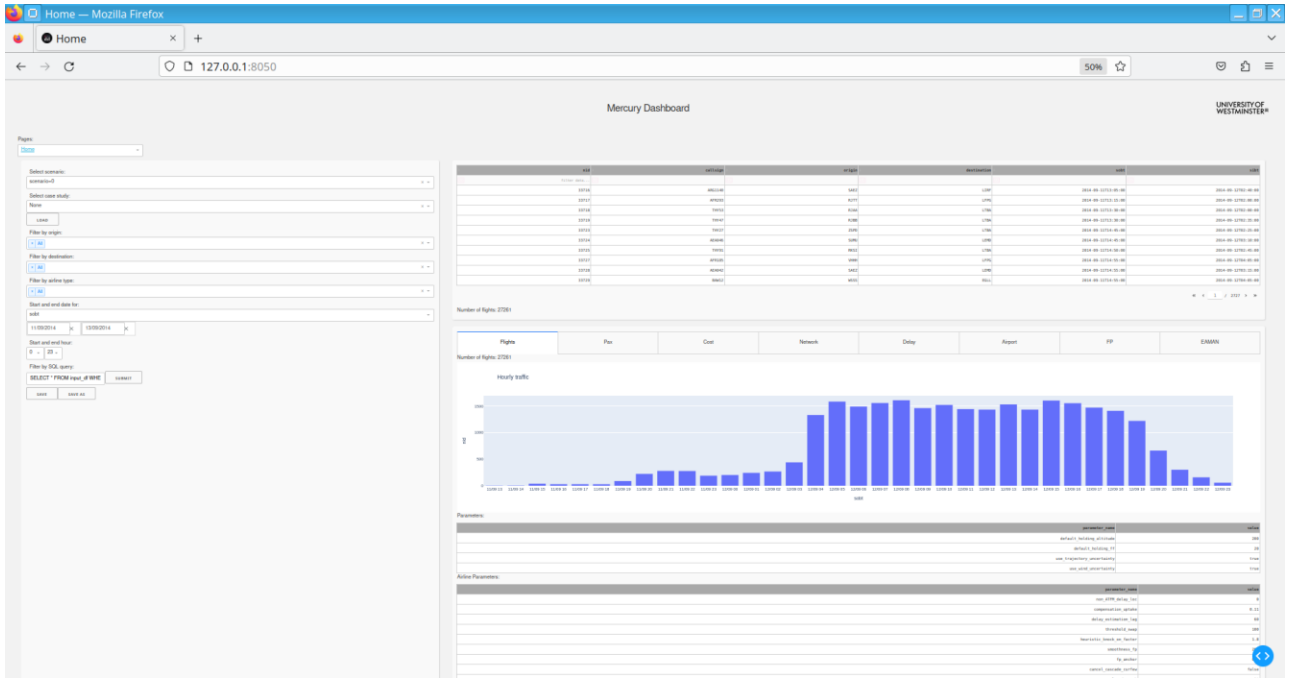
Gérald Gurtner, Michal Weiszer, Luis Delgado

Graphical user interface

Run by:

mercury_gui.py

- Web based GUI with Dash Plotly
- You can use a GUI to explore the data input and output structure, create new scenarios, case studies, etc.



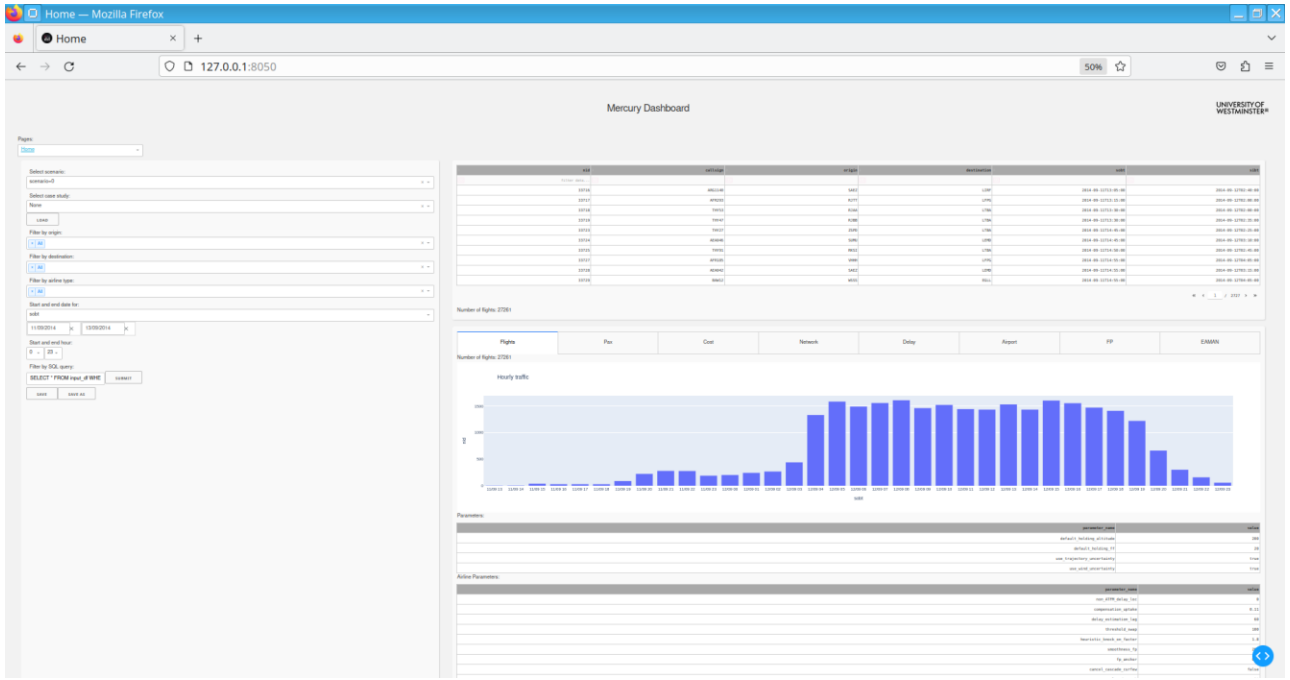
Graphical user interface

Panel on the left is for:

- Filtering and Queries,
- Loading and saving case studies

Tabs to explore and change parameters:

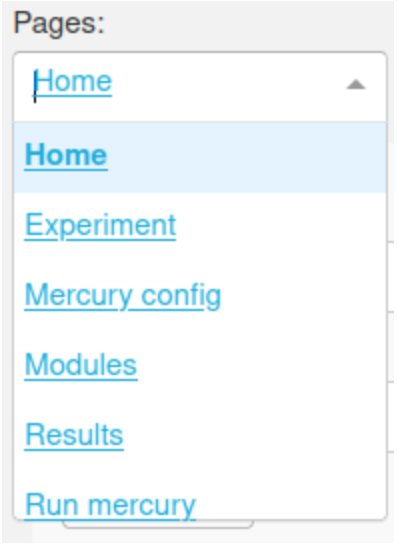
- Flights related parameters
- Pax - Itineraries
- Costs related parameters
- Network - regulations
- Delay
- Airports
- Flight plans: e.g. LEMD-LTBA for A330-301
- EAMAN



Graphical user interface

Pages:

- Home (Main)
- Experiment - enables to create experiments
- Mercury config - modifies config toml
- Modules - modifies module tomls
- Results - reads from results folder
- Run - runs Mercury from command line



- Some of the pages and functionalities are still work in progress

Data particularities (Input)

Flight schedules schedules/flight_schedule.parquet

- nid (*35874*)
- airline (*NWS*)
- airline_type (*REG*)
- origin (*LTAI*)
- destination (*LTBS*)
- sobt (*2014-09-12 02:00:00*)
- sibt (*2014-09-12 02:33:00*)
- registration (*VPBOW*)
- prev_flight_nid (*34243*)
- …

Passengers pax/pax_itineraries.parquet

nid	pax	avg_fare	ticket_type	leg1	leg2	leg3	leg4
4065	10	128	flex	45739	None	None	None
3241	15	134	economy	45739	34212	None	None
…	…	…	…	…	…	…	…
…	…	…	…	…	…		

Data particularities (Input)

Flight plan pool

Pool of pre-computed FP

Used by AOC in dispatching process

flight_plans_pool/fp_pool.parquet

id	icao_orig	icao_dest	bada_code_ac_model	fp_distance_nm	crco_cost_EUR
68918	LEBL	EGLL	A320-214	707.34	978.24
68919	LEBL	EGLL	A320-214	691.14	978.24
68920	LEBL	EGLL	A320-214	720.84	978.24
73653	LEBL	EGLL	A319-114	707.34	931.51
…	…	…	…	…	…

flight_plans_pool/fp_pool_ac_icao_ac_model.parquet

ac_icao	ac_model
A320	A320-214
A310	A310-314
...	…

Data particularities (Input)

Flight plan pool

Pool of pre-computed FP

Used by AOC in dispatching process

flight_plans_pool/fp_pool_point.parquet

fp_pool_id	sequence	name	alt_fp	time_min	dist_from_orig_nm	dist_to_dest_nm	wind	ansp
68918	0	takeoff	0.000	0.000000	0.00	707.34	0.0	LE
68918	1	(42.77, 0.997)	341.487	19.361446	129.59	577.75	0.0	LF
68918	2	TOC	390	22.112	148	559.34	0	LF
68918	3	(50.0, -1.691)	390	80.561417	583.69	123.65	0	EG
…	…	…	…	…	…	…	…	…

weight	fuel	planned_avg_speed_kt	max_speed_kt	min_speed_kt	mrc_speed_kt	lat	lon
60748.0	0.000	0.000	NaN	NaN	NaN	41.296944	2.078333
59374.9	1373.2	401.592	NaN	NaN	NaN	42.774722	0.997222
59179.8	1568.3	401.592	NaN	NaN	NaN	43.068778	0.901078
57177.8	3570.2	447.248	468.28	400.228	433.603	50	-1.691389
…	…	…	…	…	…	…	…

Extending Mercury

Module development

New module can be developed to modify the behaviours of some agents in Mercury.

Example: Modify taxi out behaviour of aircraft, simulating an issue on taxiway that extends the taxi out time by a changeable parameter.

General workflow:

1. Identify the agent in agents/: airport_operating_centre.py:AirportOperatingCentre
2. Identify the role within agent: TaxiOutProvider

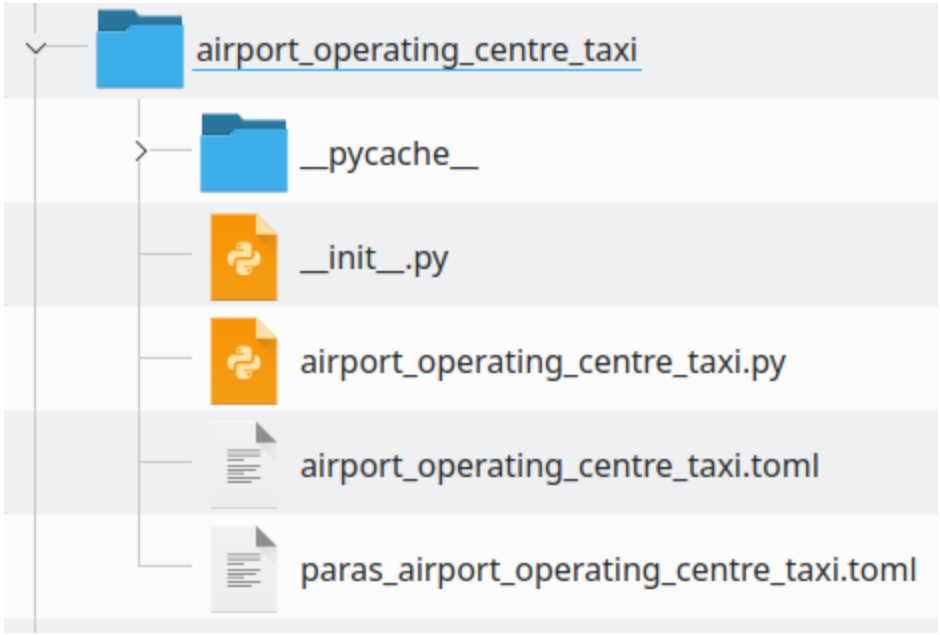
Extending Mercury

Module development

Example: airport_operating_centre_taxi

3. Create module files

- create a folder in the modules folder.
- in this folder, you can create all the files you need. The file structure should be like this:
- Empty init file
- MODULE_NAME.py
- paras_MODULE_NAME.toml
- MODULE_NAME.toml



Extending Mercury

Module development

paras_MODULE_NAME.toml

- Contains new module parameters (to be used in CLI)

MODULE_NAME.toml

- Contains module specs in 3 categories:
  1. Info: name, metrics, requirements, incompatibilities
  2. Agent level settings
  3. Role level settings

Mostly contain what is being replaced in the original agent

[agent_modif.AGENT_NAME.ROLE_NAME]

Old method = NEW method

Example: airport_operating_centre_taxi

```
[paras]
taxi_const = 0
```

```
[info]
name = "airport_operating_centre_taxi"
description = "taxi out modification for workshop demo"
incompatibilities = []
requirements = []
get_metric = "None"

[agent_modif.AirportOperatingCentre]
on_init = "on_init_agent"
new_parameters = [
        "taxi_const",
    ]

[agent_modif.AirportOperatingCentre.TaxiOutProvider]
on_init = "on_init"
compute_taxi_out_time = "compute_taxi_out_time_NEW"


receive = "receive_new_messages"
```

Extending Mercury

Module development

- MODULE_NAME.py

- Contains module code in 3 categories:
  - Agent level: on init agent
  - Role level: on init, new methods
  - New messages (we don't have any new here, just for example)

- Copy the relevant code from the original agent into module and then change.
- Make sure that the new names match with *.toml files

Extending Mercury

Module development

Example: airport_operating_centre_taxi.py

```
from Mercury.core.delivery_system import Letter

# ================== Agent Modification ================== #
# These functions should be created for each modified agent

# ----------------- airport_operating_centre ----------------- #
def on_init_agent(self):
    self.taxi_const = self.airport_operating_centre_taxi__taxi_const

# ProvideConnectingTime

def on_init(self):
    self.requests = {}

def compute_taxi_out_time_NEW(self, ac_icao, ao_type, taxi_out_time_estimation):
    """
    Sample the taxi-out time from the distribution
    """
    # Note that taxi-out does not depend on aircraft type (ac_icao) nor airline operator type (ao_type) for now
    taxi_out_time = taxi_out_time_estimation + self.agent.taxi_time_add_dists.rvs(random_state=self.agent.rs)+self.agent.taxi_const
    print('new taxi time is ',taxi_out_time)
    return max(self.agent.min_tt, taxi_out_time)

def receive_new_messages(self, msg):
    #print (self, 'RECEIVES A MESSAGE OF TYPE:', msg['type'])
    if msg['type'] == 'flight_potential_delay_recover_information':
        self.app.wait_for_flight_potential_delay_recover_information(msg)
        return True
    else:
        return False
```

Extending Mercury

Module development

Example: airport_operating_centre_taxi

- The name should match the NAME OF THE FOLDER in which the files live.
- The module can then be used by modifying the modules_to_load parameter in case_study_config.toml

```
[paras]
    [paras.airlines]
    fuel_price = 0.1

    [paras.modules]
    modules_to_load = ["airport_operating_centre_taxi"]
```

- We can then use the new parameter in the cli of the Mercury object like any other parameter:
- python3 ./mercury.py -id -1 -cs -1 -
  -airport_operating_centre_taxi__taxi_const 10
- We can check the results in results folder, output_flights.csv / axot

Extending Mercury

Module development

Not covered (check the docs):

- Get_metric: a custom method (get_metric) to gather important metrics for final analysis.
- Module flavours: Sometimes one may write several versions of a module that share a lot in common.

UNIVERSITY OF
WESTMINSTER